

UNIVERSIDAD DE HUANUCO
FACULTAD DE INGENIERIA
PROGRAMA ACADÉMICO DE INGENIERÍA DE SISTEMAS E
INFORMÁTICA



TESIS

**“ELABORACIÓN DEL FRAMEWORK “XRL8” BAJO LA
TECNOLOGÍA CSS PARA AGILIZAR EL DESARROLLO FRONT-
END”**

**PARA OPTAR EL TÍTULO PROFESIONAL DE INGENIERO
DE SISTEMAS E INFORMÁTICA**

AUTOR: Herrera Condezo, Phool Antony

ASESOR: Rodríguez Meléndez, Fabio

HUÁNUCO – PERÚ

2022

U

TIPO DEL TRABAJO DE INVESTIGACIÓN:

- Tesis (X)
- Trabajo de Suficiencia Profesional ()
- Trabajo de Investigación ()
- Trabajo Académico ()

LÍNEAS DE INVESTIGACIÓN: Tecnologías de la información y comunicación

AÑO DE LA LÍNEA DE INVESTIGACIÓN (2020)

CAMPO DE CONOCIMIENTO OCDE:

Área: Ingeniería, Tecnología

Sub área: Ingeniería eléctrica, Ingeniería electrónica

Disciplina: Sistemas de automatización, Sistemas de control

DATOS DEL PROGRAMA:

Nombre del Grado/Título a recibir: Título

Profesional de Ingeniero de sistemas e informática

Código del Programa: P06

Tipo de Financiamiento:

- Propio (X)
- UDH ()
- Fondos Concursables ()

DATOS DEL AUTOR:

Documento Nacional de Identidad (DNI): 72225661

DATOS DEL ASESOR:

Documento Nacional de Identidad (DNI): 42883191

Grado/Título: Maestro en ingeniería de sistemas, mención en tecnologías de información y comunicación

Código ORCID: 0000-0003-4533-5595

DATOS DE LOS JURADOS:

N°	APELLIDOS Y NOMBRES	GRADO	DNI	Código ORCID
1	Solis Jara, Paolo Edver	Ingeniero de sistemas e informática	41656218	0000-0002-6936-1985
2	Vigilio Arratea, Freddy Claydermam	Maestro en ingeniería de sistemas e informática con mención en gerencia de sistemas y tecnologías de información	43691515	0000-0002-3982-6518
3	López De La Cruz, Edgardo Cristiam Iván	Magister en ciencias de la educación mención: educación ambiental y desarrollo sostenible	40394603	0000-0001-9815-7708

D

H



UNIVERSIDAD DE HUÁNUCO

Facultad de Ingeniería

P.A DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

ACTA DE SUSTENTACIÓN DE TESIS PARA OPTAR EL TÍTULO PROFESIONAL DE INGENIERO (A) DE SISTEMAS E INFORMÁTICA

En la ciudad de Huánuco, siendo las 16:41 horas del día 01 del mes de setiembre del año 2022, en el Auditorio de la Facultad de Ingeniería, en cumplimiento de lo señalado en el Reglamento de Grados y Títulos de la Universidad de Huánuco, se reunieron el **Jurado Calificador** integrado por los docentes:

- Ing. Paolo Edver Solis Jara, PRESIDENTE
- Mg. Freddy Claydermam Vigilio Arratea SECRETARIO
- Mg. Edgardo Cristiam Iván López De La Cruz VOCAL

Nombrados mediante la Resolución N° **RESOLUCIÓN No 1673-2022-D-FI-UDH** para evaluar la **Tesis** intitulada:

"
Elaboración del framework "XRLS" bajo la Tecnología CSS para agilizar el desarrollo Front-End.
"

.....", presentado por el (la) **Bachiller: Phool Antony HERRERA CONDEZO**, para optar el Título Profesional de Ingeniero (a) de Sistemas e Informática.

Dicho acto de sustentación se desarrolló en dos etapas: exposición y absolución de preguntas: procediéndose luego a la evaluación por parte de los miembros del Jurado.

Habiendo absuelto las objeciones que le fueron formuladas por los miembros del Jurado y de conformidad con las respectivas disposiciones reglamentarias, procedieron a deliberar y calificar, declarándolo (a) aprobado por unanimidad con el calificativo cuantitativo de 17 y cualitativo de Muy bueno. (Art. 47)

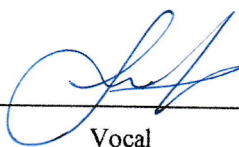
Siendo las 17:40 horas del día 01 del mes de setiembre del año 2022, los miembros del Jurado Calificador firman la presente Acta en señal de conformidad.



Presidente



Secretario



Vocal

DEDICATORIA

La presente tesis está dedicada a Dios, por iluminarme en el camino correcto.

A mis padres por su cariño interminable y ánimos para seguir adelante, frente a los retos y adversidades de la vida.

A la Ing. Bertha Campos Ríos por ser como una madre más, siempre dispuesta a ayudar y apoyar incondicionalmente.

AGRADECIMIENTOS

Quiero agradecer a nuestra alma mater,
Universidad de Huánuco por los
conocimientos brindados a lo largo de mi
estadía.

A mis profesores que me nutrieron de
conocimientos y valores a lo largo de mi
travesía por las aulas, fomentándome los
valores y disciplina.

ÍNDICE

DEDICATORIA	II
AGRADECIMIENTOS.....	III
ÍNDICE.....	IV
ÍNDICE DE FIGURAS.....	VI
RESUMEN	XVI
ABSTRACT.....	XVII
INTRODUCCIÓN.....	XVIII
CAPÍTULO I.....	19
1. PLANTEAMIENTO DEL ESTUDIO	19
1.1. Línea de investigación.....	19
1.2. Descripción del problema.....	19
1.3. Formulación del Problema	20
1.4. Justificación de la Propuesta.....	20
1.5. Propuesta de solución y alcance.....	22
1.6. Objetivos	23
1.6.1. Objetivo general	23
1.6.2. Objetivos específicos	23
CAPÍTULO II.....	24
2. MARCO TEÓRICO.....	24
2.1. Antecedentes de la investigación.....	24
2.1.1. Antecedentes Internacionales	24
2.1.2. Antecedentes Nacionales.....	25
2.1.3. Antecedentes locales	27
2.2. Descripciones Conceptuales.....	27
CAPÍTULO III.....	31
3. METODOLOGIA.....	31
3.1. Metodología	31
3.1.1. Etapas a desarrollar	33
3.1.2. Uso de la metodología OOCSS.....	35
3.2. Herramientas.....	35
CAPÍTULO IV.....	38
4. DESARROLLO E IMPLEMENTACIÓN	38
4.1 Desarrollo e Implementación.....	38

CONCLUSIONES	405
REFERENCIAS	406
ANEXO	407

ÍNDICE DE FIGURAS

Figura 1 Código de la barra de navegacion	42
Figura 2 Leyenda de los estados del elemento	42
Figura 3 Compatibilidad web propiedad width	44
Figura 4 Compatibilidad web propiedad height	45
Figura 5 Compatibilidad web propiedad background-color	45
Figura 6 Compatibilidad web propiedad fixed	47
Figura 7 Código repetido del componente barra de navegacion	50
Figura 8 Creación del objeto header_footer en función mixin	51
Figura 9 Metodología OOCSS en el componente barra de navegacion	52
Figura 10 Variables Sass del componente barra de navegacion parte I	52
Figura 11 Variables Sass del componente barra de navegacion parte II	53
Figura 12 Comando git para agregar contenido al índice de trabajo	54
Figura 13 Comando git para mostrar el estado del proyecto	54
Figura 14 Comando git para empaquetar los datos con un nombre de referencia	54
Figura 15 Comando git para subir los cambios a github	55
Figura 16 Visualización en github el componente barra de navegacion	55
Figura 17 Código html5 final del componente barra de navegacion	56
Figura 18 Código de pie de pagina	61
Figura 19 Compatibilidad web, propiedad text-align.	63
Figura 20 Compatibilidad web, propiedad padding.	64
Figura 21 Compatibilidad web, propiedad color	64
Figura 22 Código repetido del componente pie de pagina.....	67
Figura 23 Metodología OOCSS en el componente pie de pagina	68
Figura 24 Variables Sass del componente pie de página parte I	69
Figura 25 Variables Sass del componente pie de página parte II	69
Figura 26 Visualización en github el componente pie de pagina	71
Figura 27 Código html5 final del componente pie de pagina	72
Figura 28 Código de input formulario parte I.....	77
Figura 29 Código de input formulario parte II.....	78
Figura 30 Código de input formulario parte III.....	78
Figura 31 Código de input formulario parte IV	79
Figura 32 Código de input formulario parte V	79
Figura 33 Código de input formulario parte VI	80
Figura 34 Código de input formulario parte VII	80
Figura 35 Código de input formulario parte VIII	81
Figura 36 Código de input formulario parte IX	81
Figura 37 Código de input formulario parte X	82
Figura 38 Compatibilidad web, propiedad border	84
Figura 39 Compatibilidad web, propiedad margin.....	85
Figura 40 Compatibilidad web, propiedad border-radius	85
Figura 41 Compatibilidad web, propiedad font-weight	86
Figura 42 Compatibilidad web, propiedad font-size	87
Figura 43 Compatibilidad web, propiedad line-height	87

Figura 44	Compatibilidad web, propiedad display:flex	88
Figura 45	Compatibilidad web, propiedad align-items	89
Figura 46	Compatibilidad web, propiedad justify-content.....	90
Figura 47	Gráfico del valor ease-in-out en la propiedad transition.....	90
Figura 48	Compatibilidad web, propiedad transition	91
Figura 49	Compatibilidad web, propiedad outline	91
Figura 50	Compatibilidad web, propiedad box-shadow	92
Figura 51	Código repetido del componente input formulario parte I	96
Figura 52	Código repetido del componente input formulario parte II	96
Figura 53	Código repetido del componente input formulario parte III	96
Figura 54	Código repetido del componente input formulario parte IV	97
Figura 55	Código repetido del componente input formulario parte V	97
Figura 56	Código repetido del componente input formulario parte VI	97
Figura 57	Código repetido del componente input formulario parte VII	97
Figura 58	Código repetido del componente input formulario parte VIII	98
Figura 59	Código repetido del componente input formulario parte IX	98
Figura 60	Código repetido del componente input formulario parte X.....	98
Figura 61	Creación del objeto general_fiel_and_desing en función mixin .	101
Figura 62	Creación del objeto h2_fiel_and_desing en función mixin	101
Figura 63	Creación del objeto input_fiel_and_desing en función mixin	101
Figura 64	Creación del objeto label_fiel_and_desing en función mixin.....	102
Figura 65	Creación del objeto condicional_label_fiel_and_desing en función mixin	102
Figura 66	Metodología OOCSS en el componente input formulario parte I	102
Figura 67	Metodología OOCSS en el componente input formulario parte II	103
Figura 68	Metodología OOCSS en el componente input formulario parte III	103
Figura 69	Metodología OOCSS en el componente input formulario parte IV	104
Figura 70	Metodología OOCSS en el componente input formulario parte V	104
Figura 71	Metodología OOCSS en el componente input formulario parte VI	104
Figura 72	Visualización en github del componente input formulario	106
Figura 73	Código html5 final del componente input formulario parte I	107
Figura 74	Código html5 final del componente input formulario parte II	107
Figura 75	Código html5 final del componente input formulario parte III	108
Figura 76	Código de responsive desing parte I	112
Figura 77	Código de responsive desing parte II	113
Figura 78	Código de responsive desing parte III.....	113
Figura 79	Código de responsive desing parte IV	114
Figura 80	Código repetido del componente responsive desing parte I	123
Figura 81	Código repetido del componente responsive desing parte II	123
Figura 82	Código repetido del componente responsive desing parte III	123
Figura 83	Código repetido del componente responsive desing parte IV	123
Figura 84	Código repetido del componente responsive desing parte V	123

Figura 85 Código repetido del componente responsive desing parte VI....	123
Figura 86 Creación del objeto ancho_col en la función	125
Figura 87 Creación del objeto en un ciclo for para la clase col-chiki-n	125
Figura 88 Creación del objeto usando puntos de corte dentro de un ciclo each.....	125
Figura 89 Metodología OOCSS en el componente responsive desing parte I	126
Figura 90 Metodología OOCSS en el componente responsive desing parte II	126
Figura 91 Variables Sass del componente responsive desing.....	127
Figura 92 Visualización en github del componente responsive desing	128
Figura 93 Código html5 final del componente responsive desing.....	129
Figura 94 Código de libro 3d parte I.....	132
Figura 95 Código de libro 3d parte II.....	133
Figura 96 Código de libro 3d parte III.....	133
Figura 97 Código de libro 3d parte IV	134
Figura 98 Código de libro 3d parte V	134
Figura 99 Código de libro 3d parte VI	135
Figura 100 Compatibilidad web, propiedad transform-origin.....	138
Figura 101 Compatibilidad web, propiedad perspective	139
Figura 102 Compatibilidad web, propiedad transform-style	140
Figura 103 Compatibilidad web, propiedad transforms.....	140
Figura 104 Uso de media query en el componente libro 3d.....	142
Figura 105 Código repetido del componente libro 3d parte I	144
Figura 106 Código repetido del componente libro 3d parte II	144
Figura 107 Código repetido del componente libro 3d parte III	144
Figura 108 Código repetido del componente libro 3d parte IV	145
Figura 109 Código repetido del componente libro 3d parte V	145
Figura 110 Código repetido del componente libro 3d parte VI.....	145
Figura 111 Código repetido del componente libro 3d parte VII	145
Figura 112 Código repetido del componente libro 3d parte VIII	146
Figura 113 Código repetido del componente libro 3d parte IX.....	146
Figura 114 Código repetido del componente libro 3d parte X.....	146
Figura 115 Creación del objeto btn_hoja_navega en función mixin.....	148
Figura 116 Creación del objeto hojas_pag_libro en función mixin	149
Figura 117 Creación del objeto porta_and_tomo en función mixin	149
Figura 118 Creación del objeto en un ciclo para la clase tomo_libro_3d ...	149
Figura 119 Creación del objeto en un ciclo para el identificador pagina	149
Figura 120 Creación del objeto dimencion_caja_and_tareas en función mixin	149
Figura 121 Creación del objeto adelante_and_tareas en función mixin.....	150
Figura 122 Creación del objeto en un ciclo para la clase carta.....	150
Figura 123 Metodología OOCSS en el componente libro 3d parte I	150
Figura 124 Metodología OOCSS en el componente libro 3d parte II	150
Figura 125 Metodología OOCSS en el componente libro 3d parte III	150
Figura 126 Metodología OOCSS en el componente libro 3d parte IV.....	151
Figura 127 Metodología OOCSS en el componente libro 3d parte V.....	151

Figura 128 Metodología OOCSS en el componente libro 3d parte VI.....	151
Figura 129 Metodología OOCSS en el componente libro 3d parte VII.....	151
Figura 130 Metodología OOCSS en el componente libro 3d parte VIII.....	151
Figura 131 Metodología OOCSS en el componente libro 3d parte IX.....	152
Figura 132 Metodología OOCSS en el componente libro 3d parte X.....	152
Figura 133 Variables Sass del componente libro 3d.....	152
Figura 134 Visualización en github del componente libro 3d	154
Figura 135 Código html5 final del componente libro 3d	155
Figura 136 Código html5 final del submodulo carta 3d	155
Figura 137 Código de pre-carga parte I.....	158
Figura 138 Código de pre-carga parte II	159
Figura 139 Código de pre-carga parte III	159
Figura 140 Código de pre-carga parte IV.....	160
Figura 141 Código de pre-carga parte V.....	160
Figura 142 Código de pre-carga parte VI.....	161
Figura 143 Código de pre-carga parte VII.....	161
Figura 144 Código de pre-carga parte VIII.....	162
Figura 145 Código de pre-carga parte IX.....	162
Figura 146 Compatibilidad web, propiedad background	164
Figura 147 Compatibilidad web, propiedad top.....	164
Figura 148 Compatibilidad web, propiedad left.....	165
Figura 149 Compatibilidad web, propiedad content.....	165
Figura 150 Compatibilidad web, propiedad animation	167
Figura 151 Compatibilidad web, propiedad animation-delay	168
Figura 152 Compatibilidad web, propiedad filter	168
Figura 153 Uso de media query en el componente pre carga	171
Figura 154 Código repetido del componente pre-carga parte I.....	173
Figura 155 Código repetido del componente pre-carga parte II.....	173
Figura 156 Código repetido del componente pre-carga parte III.....	173
Figura 157 Código repetido del componente pre-carga parte IV	174
Figura 158 Código repetido del componente pre-carga parte V	174
Figura 159 Código repetido del componente pre-carga parte VI	174
Figura 160 Código repetido del componente pre-carga parte VII	174
Figura 161 Código repetido del componente pre-carga parte VIII	175
Figura 162 Código repetido del componente pre-carga parte IX	175
Figura 163 Código repetido del componente pre-carga parte X	175
Figura 164 Creación del objeto precarga1_be_af en función mixin	177
Figura 165 Creación del objeto pre_carga_base en función mixin	177
Figura 166 Creación del objeto en un ciclo para la clase obj.....	177
Figura 167 Creación del objeto pre_carga_3_base en función mixin	178
Figura 168 Creación del objeto pre_carga_centrado en función mixin	178
Figura 169 Metodología OOCSS en el componente pre-carga parte I	178
Figura 170 Metodología OOCSS en el componente pre-carga parte II	178
Figura 171 Metodología OOCSS en el componente pre-carga parte III	179
Figura 172 Metodología OOCSS en el componente pre-carga parte IV	179
Figura 173 Metodología OOCSS en el componente pre-carga parte V	179
Figura 174 Metodología OOCSS en el componente pre-carga parte VI	179

Figura 175 Metodología OOCSS en el componente pre-carga parte VII ...	179
Figura 176 Variables Sass del componente pre-carga parte I	180
Figura 177 Variables Sass del componente pre-carga parte II	180
Figura 178 Variables Sass del componente pre-carga parte III	180
Figura 179 Variables Sass del componente pre-carga parte IV	181
Figura 180 Visualización en github del componente pre-carga	182
Figura 181 Código html5 final del componente pre-carga parte I	183
Figura 182 Código html5 final del componente pre-carga parte II	183
Figura 183 Código html5 final del componente pre-carga parte III	184
Figura 184 Código html5 final del componente pre-carga parte IV	184
Figura 185 Código de slider automatico parte I	186
Figura 186 Código de slider automatico parte II	186
Figura 187 Compatibilidad web, propiedad overflow	188
Figura 188 Compatibilidad web, propiedad list-style	189
Figura 189 Estructura de un keyframes parte I	189
Figura 190 Estructura de un keyframes parte II	189
Figura 191 Compatibilidad web propiedad keframes	190
Figura 192 Ejemplo de uso de la propiedad animation	190
Figura 193 Código repetido del componente slider automático	193
Figura 194 Creación del objeto slider_base en función mixin	194
Figura 195 Metodología OOCSS en el componente slider automático	194
Figura 196 Variables sass del componente slider automático	195
Figura 197 Visualización en github del componente slider automático	197
Figura 198 Código html5 final del componente slider automático	198
Figura 199 Código de paginacion parte I	201
Figura 200 Código de paginacion parte II	201
Figura 201 Compatibilidad web, propiedad border-top-left-radius	203
Figura 202 Código repetido del componente paginación parte I	206
Figura 203 Código repetido del componente paginación parte II	206
Figura 204 Creación del objeto paginacion_retrocesos en función mixin ..	207
Figura 205 Metodología OOCSS en el componente paginación parte I	207
Figura 206 Metodología OOCSS en el componente paginación parte II ...	207
Figura 207 Variables sass del componente paginación	207
Figura 208 Visualización en github del componente paginación	209
Figura 209 Código html5 final del componente paginación	210
Figura 210 Código de texturas parte I	212
Figura 211 Código de texturas parte II	213
Figura 212 Código de texturas parte III	213
Figura 213 Código de texturas parte IV	214
Figura 214 Compatibilidad web, propiedad box-sizing	215
Figura 215 Código repetido del componente texturas.	218
Figura 216 Creación del objeto texturas_base en función mixin	219
Figura 217 Creación del objeto en un ciclo for para la clase puntero-n	219
Figura 218 Metodología OOCSS en el componente texturas	219
Figura 219 Variables sass del componente texturas	219
Figura 220 Visualización en github del componente texturas.	221
Figura 221 Código html5 final del componente texturas parte I	222

Figura 222 Código html5 final del componente texturas parte II	222
Figura 223 Código html5 final del componente texturas parte III	222
Figura 224 Código html5 final del componente texturas parte IV	222
Figura 225 Código html5 final del componente texturas parte V	222
Figura 226 Código html5 final del componente texturas parte VI	222
Figura 227 Código html5 final del componente texturas parte VII	222
Figura 228 Código html5 final del componente texturas parte VIII	222
Figura 229 Código html5 final del componente texturas parte IX	223
Figura 230 Código html5 final del componente texturas parte X	223
Figura 231 Código html5 final del componente texturas parte XI	223
Figura 232 Código html5 final del componente texturas parte XII	223
Figura 233 Código html5 final del componente texturas parte XIII	223
Figura 234 Código html5 final del componente texturas parte XIV	223
Figura 235 Código html5 final del componente texturas parte XV	223
Figura 236 Código parallax parte I	226
Figura 237 Código parallax parte I	227
Figura 238 Código parallax parte II	227
Figura 239 Código parallax parte III	228
Figura 240 Compatibilidad web propiedad background-repeat	229
Figura 241 Compatibilidad web propiedad background-attachment	229
Figura 242 Uso de media query en el componente parrallax parte I	231
Figura 243 Uso de media query en el componente parrallax parte II	232
Figura 244 Uso de media query en el componente parrallax parte III	232
Figura 245 Código repetido del componente parallax parte I	234
Figura 246 Código repetido del componente parallax parte II	234
Figura 247 Creación del objeto parallax_sin_before_base en función mixin	235
Figura 248 Creación del objeto sin_margen_espacios en función mixin ...	236
Figura 249 Metodología OOCSS en el componente parallax parte I	236
Figura 250 Metodología OOCSS en el componente parallax parte II	236
Figura 251 Metodología OOCSS en el componente parallax parte III	236
Figura 252 Variables sass del componente parallax	237
Figura 253 Visualización en github del componente parallax	238
Figura 254 Código html5 final del componente parallax parte I	239
Figura 255 Código html5 final del componente parallax parte II	239
Figura 256 Código tablas parte I	243
Figura 257 Código tablas parte II	243
Figura 258 Código tablas parte III	244
Figura 259 Compatibilidad web propiedad border-collapse	246
Figura 260 Código repetido del componente tablas parte I	249
Figura 261 Código repetido del componente tablas parte II	249
Figura 262 Código repetido del componente tablas parte III	249
Figura 263 Código repetido del componente tablas parte IV	249
Figura 264 Código repetido del componente tablas parte V	250
Figura 265 Código repetido del componente tablas parte VI	250
Figura 266 Código repetido del componente tablas parte VII	250

Figura 267 Creación del objeto tabla_simple_and_compleja_base en función mixin	251
Figura 268 Metodología OOCSS en el componente tablas parte I	252
Figura 269 Metodología OOCSS en el componente tablas parte II	252
Figura 270 Metodología OOCSS en el componente tablas parte III	252
Figura 271 Metodología OOCSS en el componente tablas parte IV	252
Figura 272 Variables sass del componente tablas parte I	253
Figura 273 Variables sass del componente tablas parte II	253
Figura 274 Visualización en github del componente tablas	254
Figura 275 Código html5 final del componente tablas parte I	256
Figura 276 Código html5 final del componente tablas parte II	257
Figura 277 Código de imagen redonda parte I	260
Figura 278 Código de imagen redonda parte II	261
Figura 279 Compatibilidad web propiedad max-width	262
Figura 280 Compatibilidad web propiedad min-width	263
Figura 281 Compatibilidad web propiedad max-height	263
Figura 282 Compatibilidad web propiedad min-height	264
Figura 283 Código repetido del componente imagen redonda parte I	267
Figura 284 Código repetido del componente imagen redonda parte II	267
Figura 285 Creación del objeto img_redonda_base en función mixin.....	268
Figura 286 Metodología OOCSS en el componente imagen redonda	269
Figura 287 Variables sass del componente imagen redonda parte I	269
Figura 288 Variables sass del componente imagen redonda parte II	269
Figura 289 Visualización en github del componente imagen redonda	270
Figura 290 Código html5 final del componente imagen redonda	271
Figura 291 Código de botones parte I.....	275
Figura 292 Código de botones parte II.....	276
Figura 293 Código de botones parte III.....	276
Figura 294 Código de botones parte IV	277
Figura 295 Código de botones parte V	277
Figura 296 Código de botones parte VI	278
Figura 297 Código de botones parte VII	278
Figura 298 Código de botones parte VIII	279
Figura 299 Código de botones parte IX	279
Figura 300 Código de botones parte X	280
Figura 301 Código de botones parte XI	280
Figura 302 Uso de media query en el componente botones.....	286
Figura 303 Código repetido del componente botones parte I	288
Figura 304 Código repetido del componente botones parte II	288
Figura 305 Código repetido del componente botones parte III	289
Figura 306 Código repetido del componente botones parte IV	289
Figura 307 Código repetido del componente botones parte V	289
Figura 308 Código repetido del componente botones parte VI	289
Figura 309 Código repetido del componente botones parte VII.....	290
Figura 310 Código repetido del componente botones parte VIII	290
Figura 311 Código repetido del componente botones parte IX.....	290
Figura 312 Creación del objeto botón_clasico_base en función mixin.....	292

Figura 313 Creación del objeto botón_luminoso_base en función mixin ...	292
Figura 314 Creación del objeto botón_luminoso_plasma en función mixin	292
Figura 315 Creación del objeto botón_espejo_base en función mixin	293
Figura 316 Creación del objeto boton_animado_termita en función mixin.	293
Figura 317 Metodología OOCSS en el componente botones parte I	294
Figura 318 Metodología OOCSS en el componente botones parte II	294
Figura 319 Metodología OOCSS en el componente botones parte III	294
Figura 320 Metodología OOCSS en el componente botones parte IV.....	294
Figura 321 Metodología OOCSS en el componente botones parte V.....	294
Figura 322 Metodología OOCSS en el componente botones parte VI.....	295
Figura 323 Metodología OOCSS en el componente botones parte VII.....	295
Figura 324 Metodología OOCSS en el componente botones parte VIII.....	295
Figura 325 Metodología OOCSS en el componente botones parte IX.....	295
Figura 326 Metodología OOCSS en el componente botones parte X.....	295
Figura 327 Metodología OOCSS en el componente botones parte XI.....	296
Figura 328 Metodología OOCSS en el componente botones parte XII.....	296
Figura 329 Variables sass del componente botones	296
Figura 330 Visualización en github del componente botones	298
Figura 331 Código html5 final del componente botones parte I.....	299
Figura 332 Código html5 final del componente botones parte II	299
Figura 333 Código html5 final del componente botones parte III	299
Figura 334 Código html5 final del componente botones parte IV.....	299
Figura 335 Código html5 final del componente botones parte V	299
Figura 336 Código de formularios parte I.....	302
Figura 337 Código de formularios parte II.....	303
Figura 338 Código de formularios parte III.....	303
Figura 339 Código de formularios parte IV	304
Figura 340 Código de formularios parte V	304
Figura 341 Código de formularios parte VI	305
Figura 342 Gráfico del valor ease en la propiedad transition	308
Figura 343 Uso de media query en el componente formularios parte I.....	310
Figura 344 Uso de media query en el componente formularios parte II.....	311
Figura 345 Uso de media query en el componente formularios parte III....	311
Figura 346 Uso de media query en el componente formularios parte IV ...	311
Figura 347 Uso de media query en el componente formularios parte V ...	311
Figura 348 Uso de media query en el componente formularios parte VI ...	311
Figura 349 Código repetido del componente formularios parte I	313
Figura 350 Código repetido del componente formularios parte II	313
Figura 351 Código repetido del componente formularios parte III	314
Figura 352 Código repetido del componente formularios parte IV	314
Figura 353 Código repetido del componente formularios parte V	314
Figura 354 Código repetido del componente formularios parte VI	314
Figura 355 Creación del objeto base_formulario_trans en función mixin...	315
Figura 356 Metodología OOCSS en el componente formularios parte I	316
Figura 357 Metodología OOCSS en el componente formularios parte II ...	317
Figura 358 Metodología OOCSS en el componente formularios parte III ..	317
Figura 359 Metodología OOCSS en el componente formularios parte IV..	317

Figura 360 Variables sass del componente formularios parte I	317
Figura 361 Variables sass del componente formularios parte II	318
Figura 362 Visualización en github del componente formularios	319
Figura 363 Código html5 final del componente formularios parte I.....	320
Figura 364 Código html5 final del componente formularios parte II	320
Figura 365 Grafico de barras de Google trends sobre como centrar un div	321
Figura 366 Código de centrado horizontal y vertical parte I.....	324
Figura 367 Código de centrado horizontal y vertical parte II.....	324
Figura 368 Compatibilidad web, propiedad position	327
Figura 369 Código repetido del componente centrado horizontal y vertical parte I.....	330
Figura 370 Código repetido del componente centrado horizontal y vertical parte II.....	331
Figura 371 Creación del objeto centrador_pro_base en función mixin	332
Figura 372 Metodología OOCSS en el componente centrado horizontal y vertical parte I	332
Figura 373 Metodología OOCSS en el componente centrado horizontal y vertical parte II	332
Figura 374 Variables sass del componente centrado vertical y horizontal.	332
Figura 375 Visualización en github del componente centrado horizontal y vertical	334
Figura 376 Código html5 final del componente centrado horizontal y vertical parte I.....	335
Figura 377 Código html5 final del componente centrado horizontal y vertical parte II.....	335
Figura 378 Gráfico de barras de Google trends sobre accordion menu	336
Figura 379 Código de menu acordeon parte I	339
Figura 380 Código de menu acordeon parte II	340
Figura 381 Código de menú acordeón parte III	340
Figura 382 Código repetido del componente menú acordeón	347
Figura 383 Creación del objeto menú_acordeon en función mixin	348
Figura 384 Metodología OOCSS en el componente menú acordeon	349
Figura 385 Variables sass del componente menú acordeón	349
Figura 386 Visualización en github del componente menú acordeón	351
Figura 387 Código html5 final del componente menú acordeón.....	352
Figura 388 Gráfico de barras de Google trends sobre animated background	353
Figura 389 Código de la imagen en movimiento.....	355
Figura 390 Compatibilidad web propiedad background-size	357
Figura 391 Código repetido del componente imagen en movimiento parte I	360
Figura 392 Código repetido del componente imagen en movimiento parte II	361
Figura 393 Creación del objeto imagen_movimiento_base en función mixin	362
Figura 394 Creación del objeto en un ciclo para la clase movimiento	362
Figura 395 Variables sass del componente imagen en movimiento	362

Figura 396 Visualización en github del componente imagen en movimiento	364
Figura 397 Código html5 final del componente imagen en movimiento.....	365
Figura 398 Gráfico de barras de Google trends sobre modal formularios .	365
Figura 399 Código de la modal formularios parte I	369
Figura 400 Código de la modal formularios parte II	370
Figura 401 Código de la modal formularios parte III	370
Figura 402 Código de la modal formularios parte IV	371
Figura 403 Código de la modal formularios parte V	371
Figura 404 Código de la modal formularios parte VI.....	372
Figura 405 Código de la modal formularios parte VII.....	372
Figura 406 Uso de media query en el componente modal formularios parte I	377
Figura 407 Uso de media query en el componente modal formularios parte II	378
Figura 408 Código repetido del componente modal formularios parte I.....	379
Figura 409 Código repetido del componente modal formularios parte II....	380
Figura 410 Código repetido del componente modal formularios parte III... 380	
Figura 411 Código repetido del componente modal formularios parte IV .. 380	
Figura 412 Código repetido del componente modal formularios parte V ... 381	
Figura 413 Creación del objeto modal_formulario_y_publicidad_base en función mixin.....	382
Figura 414 Metodología OOCSS en el componente modal formulario parte I	382
Figura 415 Metodología OOCSS en el componente modal formulario parte II	382
Figura 416 Metodología OOCSS en el componente modal formulario parte III	383
Figura 417 Metodología OOCSS en el componente modal formulario parte IV	383
Figura 418 Metodología OOCSS en el componente modal formulario parte V	383
Figura 419 Variables sass del componente modal formulario parte I	384
Figura 420 Variables sass del componente modal formulario parte II	384
Figura 421 Visualización en github del componente modal formulario	385
Figura 422 Código html5 final del componente modal formulario parte I ... 387	
Figura 423 Código html5 final del componente modal formulario parte II .. 388	
Figura 424 Código html5 final del componente modal formulario parte III . 389	
Figura 425 Código html5 final del componente modal formulario parte IV . 390	
Figura 426 Gráfico de barras de Google trends sobre efecto neón	391
Figura 427 Código de neón	394
Figura 428 Código repetido del componente neón	400
Figura 429 Creación del objeto neon_base en función mixin	401
Figura 430 Metodología OOCSS en el componente neón.....	401
Figura 431 Variables sass del componente neon	402
Figura 432 Visualización en github del componente neón.....	403
Figura 433 Código html5 final del componente neón.....	404

RESUMEN

La investigación tecnológica se centró en el Desarrollar un framework “hojas de estilo en cascada” para diseño web responsivo en español siendo este el objetivo principal de la investigación, para ello se crearon varios elementos propios del framework para su inclusión y disponibilidad al usuario al momento de realizar la maquetación de páginas web. La metodología aplicada estuvo basada en 6 fases: Investigación del efecto e importancia, codificación del efecto, adecuación al diseño responsivo, pruebas en relación a otros efectos, aplicación del OOCSS, y documentación. En cuanto al desarrollo de los elementos propios del framework, se construyeron teniendo en cuenta la similitud de las características de algunas tecnologías presentes en el mercado actual, solo modificando quitando algunas cosas innecesarias y agregando aspectos claves para su funcionamiento.

Los elementos más principales que se construyeron para el framework fueron: barra de navegación, botón, slider, paginación, pie de página, formulario, pre cargas, textura, parallax entre otros. Cada control tiene su propia descripción y explicación del para que sirve y como se emplea, un demo donde se da a conocer la aplicación del efecto, en el proceso de maquetación.

Para la evaluación del framework se procedido a realizarlo en cada componente como parte de una fase de la metodología empleada, la evaluación consistió en poner todas las clases en una sola página y recargar la página y ver el comportamiento de cada elemento, y ver si se generaba conflictos entre los elementos. Finalmente, última fase se realizó la documentación, en el sitio www.phooldx.com y también en la página de GitHub para las consultas y usos del framework.

Palabras clave: CSS, diseño responsivo, maquetación, metodología, framework

ABSTRACT

The technological research focused on the development of a framework "cascading style sheets" for responsive web design in Spanish, this being the main objective of the research, for which several elements of the framework were created for inclusion and availability to the user at the time of making the layout of web pages. The methodology applied was based on 6 phases: investigation of the effect and its importance, coding of the effect, adaptation to the responsive design, testing in relation to other effects, application of OOCSS, and documentation. As for the development of the framework elements, they were built taking into account the similarity of the characteristics of some technologies present in the current market, only modifying by removing some unnecessary things and adding key aspects for its operation.

The main elements that were built for the framework were: navigation bar, button, slider, pagination, footer, form, preloads, texture, parallax among others. Each control has its own description and explanation of what it is used for and how it is used, a demo where the application of the effect in the layout process is shown.

For the evaluation of the framework we proceeded to perform it in each component as part of a phase of the methodology used, the evaluation consisted of putting all the classes on a single page and reload the page and see the behavior of each element, and see if it generated conflicts between the elements. Finally, the last phase was the documentation, in the site www.phooldx.com and also in the GitHub page for queries and uses of the framework.

Keywords: CSS, responsive design, layout, methodology, framework, framework

INTRODUCCIÓN

La investigación de tipo tecnológica se centró en la línea de investigación desarrollo de software, el en Capítulo I se da a conocer el problema fundamental que consistió en como de desarrolla un framework CSS para diseño web responsivo en español, también se plantea el objetivo de desarrollar un framework CSS para diseño web responsivo en español, en cuanto a los específicos: se da nomenclatura al framework “XRL8” como una estructura básica para agilizar y optimizar su velocidad en el usuario final. También el de implementar Clases básicas en el desarrollo del framework para mejorar su compatibilidad en google Chrome por ser uno de los navegadores con mayor demanda en búsquedas.

En el capítulo II, se listan los antecedentes, de los cuales en los internacionales hay evidencias de creación de frameworks para CSS, en cambio en los nacionales solo aplicación de frameworks ya creado, de todas formas se consideran las investigaciones más pertinentes.

En el capítulo III, se da a conocer la metodología de desarrollo, el cual consistió en el propio desarrollo de los elementos de los efectos, y sus fases, el empleo de la metodología OOCSS y las herramientas empleadas para su desarrollo, en esta sección se da a conocer a más detalles cada elemento sus fases correspondientes, sus características y sus formas de evaluación.

En el capítulo IV se da a conocer el desarrollo detallado del framework, una lista de componentes incluidas en las 6 fases propias de la metodología, dando a conocer los atributos y funcionalidad de las clases, en cuanto a la evaluación de cada elemento, se incluye por cada elemento en su propia secuencia, finalmente en las conclusiones del estudio, se consolida el trabajo realizado y se menciona que el framework permitirá el desarrollo de la maquetación de forma más rápida y amigable usando el navegado Google Chrome.

CAPÍTULO I

1. PLANTEAMIENTO DEL ESTUDIO

1.1. Línea de investigación

Dentro del desarrollo de software, la línea de investigación, se centra en: Gestión y Desarrollo de Sistemas de Información.

Las referencias empleadas son:

- Desarrollar nuevas tecnologías enfocadas en la optimización de los recursos.
- Desarrollar e implementar un framework aplicable a empresas de acuerdo a su envergadura.

Un framework de CSS es una librería que albergará una gran lista de muchas y muchas líneas de código CSS que contendrán estilos de diseño y efectos web genéricos ya establecidos por el creador del framework, este código CSS estarán predefinidas todas las clases, que podrán ser usadas para implementar diseños web.

Aportan una serie de utilidades que pueden ser aprovechadas frecuentemente en los distintos diseños web.

1.2. Descripción del problema

El uso de los framework de desarrollo web han traído avances y facilidades al momento de la creación de sitios y aplicaciones web, pero no todos proveen las herramienta necesarias, y en otros caso el nivel de complejidad que tienen al emplearlos, así como lo menciona la página de Emprenderalia (2020), la primordial desventaja que se relaciona al uso de frameworks es la curva de aprendizaje. Si bien es cierto que al usar frameworks, la aplicación se desarrollaría en menos tiempo. Y es cierto, sin embargo, esto ocurre cuando se conoce a profundidad el framework.

El problema asociado a la creación del framework XRL8 es que la gran mayoría de los frameworks que ya existen en la actualidad como bootstrap, materialize, bulma, essence, semanticUI, foundation, cascade, simple, etc. No cuentan con efectos 3D en su estructura, ya que estas pueden ser usadas en formularios que giren o en paneles que tengan información detrás de sí mismas, pudiéndole emplearlas en empresas que brinden servicios de comida y bebida, etc.

Asimismo, una gran limitante para la comunidad hispanohablante es el hecho de que la gran totalidad de los frameworks se encuentran en el idioma inglés, tanto la documentación del mismo, como las clases empleadas en el CSS.

Los otros frameworks traen consigo extensiones de otras librerías, lo que los hace más pesados al momento de ser cargados en los navegadores.

Otra problemática es que los frameworks actuales carecen en su gran mayoría de animaciones netamente puras hechas en CSS3, por lo que recurren a la librería JQuery para hacerlas, cuando pueden emplear código CSS para hacer usando la propiedad @keyframes.

1.3. Formulación del Problema

¿Cómo se desarrolla un framework CSS para diseño web responsivo en español?

1.4. Justificación de la Propuesta

Se usa para simplificar tiempo de trabajo a la hora de maquetar y diseñar un sitio dinámico web. Se emplea para que la empresa informática que trabaja en el desarrollo web no contrate una mano de obra extra. Se dispone para darle un aspecto más vivo y dinámico a nuestras creaciones. Será un framework amigable e intuitivo ante el

usuario, con esto me refiero a que el usuario final aprenderá rápido, y no tendrá que memorizar las clases del CSS a la hora de llamar. Se empleará para mejorar los proyectos web, para darle un énfasis profesional de alto calibre, para que no se vean los proyectos carentes de estética.

Básicamente para minimizar el riesgo de errores a la hora de maquetar, para agilizar el desarrollo de los desarrollares o del usuario que utilizara el frameworks. Nos permitirá hacer las bases de los proyectos adaptables a dispositivos móviles, desde el principio de su uso. También permite reducir tiempo de escritura de código.

Su uso amplificará los efectos al construir una página web desde lo más básico, hasta algo más avanzado, los diseños serán amigables y agradables a la vista del usuario final, por su parte el framework podrá ser usado como material de apoyo en las universidades para el maquetado y diseño, para que se puedan guiar los alumnos.

Nos dará seguridad al abrirlo en todo tipo de navegadores como el opera mini, etc. Asimismo, con el framework se pretende que los usuarios contribuyan a la comunidad sus experiencias de usuario para ir desarrollando nuevas versiones del framework como soporte para las versiones, como sabemos la tecnología va avanzando a largos rasgos, por ello surgen nuevas necesidades a la hora de desarrollar tantas plantillas web para php, páginas web, aplicaciones web, sistemas web, etc. Mayormente todo este trabajo en grandes empresas de software recae sobre el encargado comúnmente llamado el desarrollador frontend con el framework para no ser muy redundante otro uso que se pretende dar es de que para optimizar el rendimiento de las páginas web realizadas con el framework para agilizarla al momento de cargarla y optimizar el rendimiento, el framework contara con clases específicas con el que el usuario deseara trabajar, para que solo descargue una parte del framework o la parte que más le agrade, explicándolo de otro punto de vista entre todo el servicio que brindara el framework "XRL8".

1.5. Propuesta de solución y alcance

Crear un framework de CSS, que tenga el nombre de las clases en español y que sea intuitivo para el usuario al llamarlas, para que no distinga las edades ni las experiencias del usuario, que no contengan palabras en inglés, que además el framework creado tenga efectos para que sea una experiencia más gustosa el desarrollar aplicativos webs, páginas web, etc.

El framework "XRL8" contará con efectos altamente notorio, esto será:

efecto 3D en los login tanto al registrarse como para el acceso usando las propiedades transition, traslate.. Así mismo también tendrán los efectos tradiciones como: efecto slider, efecto hover. Otro efecto muy usado es el efecto acordeón, para que se deslice un menú de opciones en las barras de navegación, botones con las propiedades active para que cambie de color los bordes al darles clic. Como en todo framework tendrá sus rejillas para que sea responsive las webs maquetadas el sistema usado para el responsive de las paginas será 3 espacios de dimensión en ordenadores, 2 de dimensión en tablets y 1 de dimensión para smartphones como dispositivos móviles.

Usaremos la propiedad de flex-box como modelo de diseño, para que sean responsive, los colores más usados de dicho framework serán. Blanco y negro que son colores neutrales más dos colores alternativos, botones negros con bordes blancos, formularios transparentes, para poder tener un fondo detrás de cada formulario.

Otra cosa que tendrá el framework será que incrementará su tamaño al darle clic esto será posible con la propiedad de CSS active. Otra novedad que tendrá para sus usuarios este framework será que para poder ver los reportes en las tablas con la propiedad hover se interceptaran las fijas con las columnas por donde pase el mouse para de esta manera tener una mejor visualización de los datos. No podía quedarse atrás el efecto parallax para darle un poco de profesionalismo

a nuestros diseños web, incluirá ventanas modales que funcionaran si uso de JavaScript. Para la compatibilidad web usaremos prefijos como web-kit para google Chrome.

1.6. Objetivos

1.6.1. Objetivo general

Desarrollar un framework CSS para diseño web responsivo en español

1.6.2. Objetivos específicos

- Diseñar el framework “XRL8” como una estructura básica para agilizar y optimizar su velocidad en el usuario final.
- Implementar clases básicas en el desarrollo del framework para mejorar su compatibilidad en google Chrome por ser uno de los navegadores con mayor demanda en búsquedas.

CAPÍTULO II

2. MARCO TEÓRICO

2.1. Antecedentes de la investigación

2.1.1. Antecedentes Internacionales

Balmaceda (2018). *Framework E-commerce. Universidad de Chile.*
Chile

El objetivo de la investigación fue de implementar un Framework e-Commerce utilizando tecnología free Open Source, bajo una metodología propia y basada en el lenguaje de programación JavaScript. Los resultados del estudio fueron: De los objetivos planteados inicialmente, no se cumplió el referente a la evaluación del uso del Framework para el desarrollo de una solución particular de servicio e-Commerce. El respecto de los demás objetivos planteados al comienzo, sí se cumplieron puesto que se logró desarrollar un Framework que, si bien es cierto, no constituye una solución equivalente en funcionalidades a los Framework descritos en un inicio dada la gran cantidad de características que estos tienen, si representa el core de una plataforma con características nuevas, útiles para el desarrollo de plataformas modernas.

Ledesma (2017). *Frameworks de arquitectura empresarial.* Universidad Nacional de la Plata. Argentina

En este sentido, se debe resaltar la importancia de especificar los requerimientos y realizar el modelado integrado que proporcione los métodos y la tecnología que se requerirá para el diseño e implementación de la arquitectura corporativa adecuada, para llegar a ello se debe definir claramente el diseño, el alcance, así como también el tipo de framework a utilizar entre otros aspectos de vital importancia.

Rincón (2017). *Diseño de un framework para el gobierno de información con base COBIT*. Universidad Nacional de Colombia. Colombia

El objetivo principal del estudio fue de realizar una revisión e investigación de la diversidad que existe sobre Framework de Arquitectura Empresarial. La metodología de la investigación se centró en el diseño metodológico documental, que consistió en la consulta y documentación de otros frameworks. Los resultados del estudio fueron: El desarrollo de un framework de gobierno de información tiene como objetivo presentar el valor que tiene la información en la organización como habilitadora de estrategias o como riesgo en caso de no ser tratada adecuadamente. La implementación de un marco de gobierno de información, implica conocer los activos de información y la organización necesita saber qué información tiene, cómo la tiene, dónde la tiene, quién la usa, quien la custodia y quien toma decisiones sobre ella para apalancar iniciativas que optimicen el valor de la información lo cual se podría traducir en ahorros o nuevos ingresos.

2.1.2. Antecedentes Nacionales

Briceño (2021). *Implementación de nueva tecnología front-end para mejorar el rendimiento de sitios web*. Universidad Tecnológica del Perú

El objetivo de la investigación fue de implementar una nueva tecnología front-end en sitios web desarrollados en WordPress que no sean óptimos y tengan una carga lenta para mejorarles su rendimiento. El desarrollo del estudio se basó en la metodología SCRUM, obteniendo los siguientes resultados: El diseño de las páginas web son de un estilo minimalista y adaptable a diferentes tamaños de dispositivos. Se cumplió con el diseño de las áreas del sitio web similar a los componentes

del framework css Bootstrap. El maquetado de las páginas web cumple con los estándares de la W3C y esta verificado con la herramienta Validator. Las advertencias y errores fueron eliminados. La programación en Next.js ha mejorado la carga del sitio web y se puede verificar con la herramienta de internet para análisis de velocidad Pingdom Tools.

Cruz (2018). *Implementación web site, full Responsive Design con HTML5.Para hoteles en la ciudad de Huaraz; 2015*.Universidad Católica los ángeles de Chimbote. Perú

El objetivo de la investigación fue de implementar un web site, full responsive design con html5, para hoteles en la ciudad de Huaraz. La metodología empleada se baso en una investigación no experimental de nivel descriptivo. Se llego a los siguientes resultados: Con el fin de dar a conocer servicios y bondades por medio de la web, la investigación se clasificó como una investigación de diseño no experimental de tipo descriptiva, la muestra estuvo constituido por 20 empleados de las 10 empresas Hoteleras involucradas, para la recolección de datos se utilizó el instrumento del cuestionario mediante la técnica de la encuesta.

Llatas (2017). *Análisis comparativo de Frameworks PHP para medir el rendimiento*. Universidad Señor de Sipan. Perú

El objetivo de la investigación fue de realizar un análisis comparativo para medir el rendimiento de frameworks PHP. La investigación de tipo tecnología y aplicada, bajo el diseño cuasi experimental, llego a los siguientes resultados:

Una comparación de Framework PHP es importante antes de realizar la programación, esto ayudara a permitir una elección confiable y segura, de lo contrario podemos perder tiempo en el

desarrollo como también tener problemas de rendimiento por no elegir el framework adecuado. En esta investigación se ha realizado la comparación de framework, en donde se han elegido dos framework en base a investigaciones previas relacionado a frameworks PHP, además de considerar indicadores del factor rendimiento que luego se medirán en un sistema desarrollado con cada framework seleccionado.

2.1.3. Antecedentes locales

No se encontraron datos locales ya que en la carrera de ingeniería de sistemas e informáticas somos una cantidad reducida que ingresamos, así como los que egresan, por otra parte, el único antecedente local de la universidad de Huánuco donde usan un framework como herramienta para ver el aprendizaje de alumnos, que dicho de paso encuentro irrelevante ya que la tesis que desarrollo es de tecnología web.

2.2. Descripciones Conceptuales

Framework

Garcia, X. (2015) señala que, framework es un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar.

Frameworks del lenguaje CSS

Valbuena, M. (2014) señala que Debido a que las estructuras CSS son complicadas surgen los frameworks que ayudan a facilitar el desarrollo de las aplicaciones. En la actualidad existen gran variedad de frameworks de CSS.

HTML5

Regatto, F. (2015) señala que HTML (Hyper Text Markup Language) lenguaje de marcas de hipertexto, es el lenguaje básico de marcas mediante el cual, se encuentran construidos todos los sitios web que se encuentran publicados en Internet, es una forma semántica de representar la estructura y contenido de una página web. Por otro lado HTML5 es la versión más reciente de este lenguaje de marcas que todavía se encuentra inconcluso. Este lenguaje está siendo desarrollado por el W3C (World Wide Web Consortium), consorcio internacional encargado de diseñar, desarrollar, validar, aprobar y recomendar estándares web

Front-End

Según Valdivia (2016) señala que: Dentro del contexto del desarrollo de aplicaciones web, implica el uso de las tecnologías con las que interactúa directamente el usuario. Normalmente estas tecnologías son desarrolladas en los lenguajes de HTML, CSS y Javascript; también se usan las herramientas de diseño gráfico como Photoshop o Fireworks. El objetivo es desarrollar la interfaz gráfica de usuario (GUI), buscando una experiencia de uso bien valorada por el usuario final, siendo en algunos casos necesario hacer investigación, estudios y pruebas para llegar a este fin. Además, dentro del desarrollo de las aplicaciones web es posible desarrollar el front-end de la aplicación sin contar con una aplicación back-end que interactúe con la base de datos.

Back-end

Valdivia, J. (2016) señala que En el contexto del desarrollo de aplicaciones están implicadas las actividades realizadas del lado del servidor; es decir, las tareas de base de datos y los servidores de aplicaciones que el usuario no puede visualizar en el explorador de Internet. Los lenguajes usados comúnmente son PHP, Java, Ruby,

.NET, Python, entre otros, los cuales son los encargados de interactuar con la base de datos (Kavourgias, 2015; Alvarado, 2012).

Responsive web design

Valdivia, J. (2016) señala que Este término es usado en referencia a la manera en que cada explorador de Internet responde a su ambiente, aportando así a la mejora de la experiencia del usuario en el uso del sitio, independientemente del tamaño de resolución y pantalla del dispositivo. Para lograr esto es imprescindible el uso de CSS3 acompañado del uso media queries dentro del mismo CSS (Fielding, 2014).

Document Object Model (DOM).

Sandoval, A. (2015) señala que Es una interfaz de programación para documentos HTML y XML. Establece una representación estructurada del documento. Proporciona interfaces para acceder y modificar la estructura, el contenido y la presentación del documento. Representa el documento como un conjunto de nodos estructurados con sus propiedades y métodos, Mediante las API de DOM, dicha representación puede modificarse desde cualquier lenguaje de programación.

Página Web

Sandoval, A. (2015) señala que Una página web es un documento que tiene formato HTML y será visualizado en un browser como Internet Explorer, Firefox, etc.

Menú

Sandoval, A. (2015) señala que Lista de las opciones que el usuario puede seleccionar. La tendencia es que estos menús aparezcan y

desaparezcan de la pantalla sin destruir lo que está debajo. Evitan tener que memorizar las opciones. A menudo se manejan por medio de un ratón.

Prototipo web

Grajales, C. (2016). Señala que Según la página ingeniería en sistemas 2009 un prototipo web “es un modelo del comportamiento del sistema que puede ser usado para entenderlo completamente o ciertos aspectos de él y así clarificar los requerimientos. Un prototipo es una representación de un sistema, aunque no es un sistema completo, posee las características del sistema final o parte de ellas”

CAPÍTULO III

3. METODOLOGIA

3.1. Metodología

La metodología OOCSS, fue creada por la Ing. Nicole Sullivan en 2008, se le denomina así porque es “orientada a objetos” lo correcto debería ser módulos pero el autor decidió llamarlo así con fines estéticos ya que en un lenguaje de programación orientado a objetos significa otro tipo de semántica. Esta metodología fue desarrollada con la finalidad de ordenar nuestro código CSS a medida que este vaya creciendo asimismo para que sea mantenible, administrable y sobre todo reutilizable ese es el propósito del uso de esta metodología. Estos módulos u “objetos” pueden ser botones, contenedores, div, etc. La condición para que sean llamados “objetos” es que se repita el código y este asimismo sea reutilizable en las etiquetas HTML. Al usar la metodología OOCSS debemos de encapsular ese código común de CSS repetitivo denominado skins (pieles) para reutilizar en otras estructuras u objetos de la web.

Este es código CSS sin aplicar OOCSS:

```
#modulo
{
width:300px;
height:300px;
overflow-y: scroll;
border: 1px solid #999;
background-color: rgba(135, 135, 135, 0.5);
color: #F00;
}

#boton
{
width:200px;
height:45px;
padding:5px 20px;
overflow:hidden;
border: 1px solid #999;
```

```
background-color: rgba(135, 135, 135, 0.5);
color: #F00;
}
```

#noticia

```
{
width:300px;
height:auto;
border: 1px solid #999;
background-color: rgba(135, 135, 135, 0.5);
color: #F00;
}
```

Éste es el concepto de OOCSS:

.modulo

```
{width:300px;
height:300px;
overflow-y: scroll;
}
```

```
.boton{
width:200px;
height:45px;
padding:5px 20px;
overflow:hidden;
}
```

```
.noticia{
width:300px;
height:auto;
}
```

```
.mi-nueva-skin{
border: 1px solid #999;
background-color: rgba(135, 135, 135, 0.5);
color: #F00;
}
```

En el HTML resultara:

```
a class=""boton"
div class="modulo"
div class=""noticia"
```

3.1.1. Etapas a desarrollar

Se siguieron las siguientes fases en la construcción del framework XRL8 cada fase está más especificada y detalla en el capítulo IV del presente documento a continuación se muestra cada etapa clasificada en fases de forma general.

Fase 1. Investigación del efecto y su importancia, elegir un efecto o crearlo:

En esta fase se empieza describiendo la funcionalidad de dicho componente la utilidad del mismo, luego nos centramos en buscar librerías, frameworks, repositorios que tengan o hallan tratado de solucionar o tengan las funcionalidades descritas al inicio. Así mismo se detalla el modo de trabajo, con esto quiero decir si se considerara como un módulo dicho componente y si este tendrá submódulos o se trabajara con módulos independientes dependiendo de la complejidad del mismo componente que mi persona considere o clasifique en base a mi experiencia.

Fase 2. Codificar el efecto:

En esta fase se empieza mostrando el código completo, así como las propiedades más significativas que se usó en la construcción y con ello la compatibilidad web de dicha propiedad en diversos navegadores.

También tenemos dentro de esta fase una sección denominada principales dificultades donde se detalla las dificultades con las que nos topamos en la construcción de dicho componente y como se solucionó. También se especifica que usamos para hacer las pruebas

Fase 3. Hacerlo responsive:

Para esta fase se describe las técnicas que se empleó para hacer cada componente esto puede variar en cada componente según las propiedades necesarias que se especifica en el capítulo IV del presente documento.

Fase 4. Testing asegurarse que no tenga conflicto con otros efectos:

En esta fase se probó que la maquetación no tuviera conflicto con las clases ya existentes, así como los errores hallados al hacer este teting. Además se escribió un registro de incidentes en un archivo llamado `_error_logs.scss` donde se daba prioridad según el nivel de conflicto detectado.

Fase 5. Aplicar la metodología OOCSS, llevándolo a sass:

En esta fase se aplicó el uso de la metodología OOCSS los cuales son:

- **Código css.** - Esta parte está desarrollada en la fase 2.
- **Código repetido.** - Se procedía a identificar las propiedades y código repetido para luego aislarlo por separado.
- **Crear un objeto.** - En esta parte se agarraba el código repetido se hacía un bosquejo en forma de función y se definía si recibirá parámetros de entrada y como sería los parámetros de salida. Esto en forma de bosquejo para tener una visión más exacta de las cosas y poder desarrollar una lógica más profunda donde se aplique buenas prácticas de desarrollo front-end.

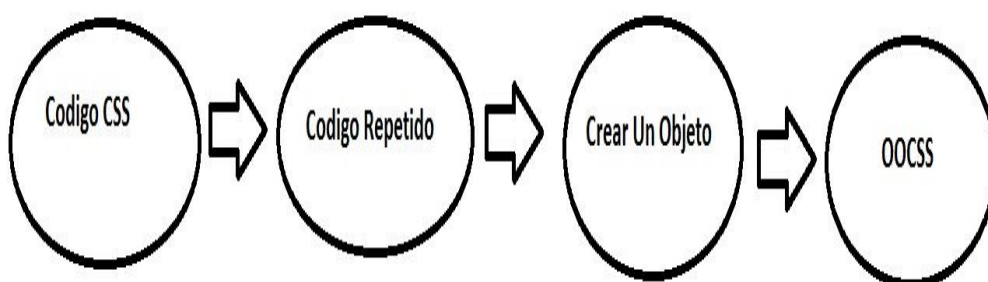
En esta sección se puede encontrar bucles como el `for`, `while` o simplemente una función simple las cuales serán usadas.

- **OOCSS.** - Finalmente se integra la metodología con el uso de la propiedad `@include` que trae el pre-procesador de css SASS. En esta sección también se muestra las variables creadas las cuales fueron usadas en desarrollo del componente.

Fase 6.- Documentar en la web site www.phooldx.com:

Luego de tener aplicada correctamente la metodología se procedió a guardar lo realizado en un repositorio en Github, también se publicó el código HTML5 para que los usuarios puedan implementar los componentes en sus proyectos.

3.1.2. Uso de la metodología OOCSS



Fuente: Elaboración propia. Basado en [Nicole Sullivan, 2009]

3.2. Herramientas

CSS: Es una hoja de estilos en cascada que sirve para dar diseño, efectos y estructura a las páginas web que fue creada por la W3C comúnmente llamado la red informática mundial.

Se usará la versión 3 de css para armar la estructura del framework, así como el diseño del mismo con sus respectivos efectos según sea el caso a emplear y usar algunas de sus características que trae:

- Bordos redondeados.
- Colores RGBA
- Sobras para texto
- Animaciones y transiciones
- Cajas con sobras

Dreamweaver: Es un software, pero más allá de ello es un editor de código que permite la construcción, diseño y edición de páginas web además trae una potente herramienta que hace posible el poder pre-visualizarlas en tiempo real los trabajos realizados sin compilarlos.

Usaremos todas las ventajas ya mencionadas anterior mente para escribir en este software el código que será posible la creación del framework.

Navegador-web: Es un software computacional que permite el acceso a la Web, su labor más importante es interpretar la información de distintos sitios web para que poder ser visualizados gráficamente, así como también realizar actividades en ella, enviar y recibir correo, ver videos, escuchar música.

Se aplicará para ver la compatibilidad en los principales navegadores como: mozilla Firefox y google Chrome. Además, permitirá la detección de errores para su posterior corrección.

Git: Es un software que sirve para el control de versiones hechas, que ayuda hacer el mantenimiento del proyecto asimismo se puede llevar un registro de todos los cambios que se harán a la elaboración del framework.

Pretendo usarlo organizar mejor mi código y tener un orden excelente en cuanto a su desarrollo, desde la creación del framework hasta cuando esté listo para su comercialización.

CodePen: Es una aplicación web que permite crear, editar y visualizar proyectos online sin la necesidad de un software. Asimismo, los usuarios comparten sus proyectos e interactúan entre ellos en la comunidad. Además, la plataforma es intuitiva y amigable con el usuario.

Se empleará para experimentar diseños, códigos, estilos nuevos antes de implementarlo al framework.

HTMLCompresor: Es un aplicativo web que permite comprimir código css en líneas corridas para reducir muchos bytes en almacenamiento. Otra funcionalidad que posee es la de validar, mostrar errores de código css, asimismo esta respaldada por la W3C que es un consorcio internacional que trabajan para desarrollar estándares Web.

Se aplicará al proyecto a la hora de comprimir código para que al usuario final le corra rápido la carga de páginas web en el servidor, asimismo el código css que escribiremos será puro y limpio al pasarlo por la validación W3C que trae dicha web en una de sus opciones de uso.

CAPÍTULO IV

4. DESARROLLO E IMPLEMENTACIÓN

4.1 Desarrollo e Implementación

BARRA DE NAVEGACIÓN

FASE 1. Investigación del efecto y su importancia, elegir un efecto o crearlo

Como primer paso lo primero que se hizo fue investigar e indagar sobre el efecto, en el caso de la barra de navegación es uno de los principales pilares para la construcción de una web. Mas que eso yo lo denominaría como una estructura principal e irrompible junto al pie de página como al cuerpo o contenido del sitio. Se le podría denominar uno de los pilares fundamentales un formato de construcción global para la web.

Otro beneficio es el posicionamiento web, ya que los principales motores de búsquedas del mundo, emplean la siguiente lógica para posicionar las páginas web como primeras.

- El uso de las etiquetas HTML5
- La estructura primaria o fundamental en los cuales son header, body, footer. Dicho de otra forma, la barra de navegación ya que dentro está el header como etiqueta principal
- La apertura y cierre correcto de las etiquetas HTML5

Como ya mencionamos uno de las características por el cual se escogió hacer este efecto fue por servir de estructura básica para todo proyecto construido en la web sea backend o frontend o una simple página web.

Otro del motivo el cual fue elegido el efecto para formar parte del framework XRL8 fue que el proyecto presente debe estar a la altura de los frameworks que están en la actualidad siendo usados en la industria del desarrollo web como son:

- **Bootstrap.** - El presente framework cuenta con la clase .navbar para poder crear barras de navegación, si desean saber más sobre ello puede visitar la web oficial de documentación de dicho framework mediante el siguiente enlace:

<https://getbootstrap.com/docs/4.1/components/navbar/>

- **Bulma.** - Este framework también cuenta con un modelo de barra de navegación denominado .navbar, para ver cómo se emplea y formas de uso revisar el siguiente enlace que los llevara a la documentación oficial de dicho framework:

<https://bulma.io/documentation/components/navbar/>

- **TailwindCSS.**- Quizá sea este el framework de css más usado en los últimos 3 años sobre pasando a los demás por sus formas de integración con NodeJS y react. Este framework dentro de su contenido también tiene una clase llamada .navbar que mezclada con otras clases se puede construir un menú de navegación. Gracias a sus componentes, estos a diferencia de los demás se caracteriza por no tener clases predefinidas para efectos sino todo lo contrario tiene un conjunto de clases en el cual el usuario puede elegir los efectos más simples para colocar a su diseño. Para revisar su documentación ingresar al siguiente enlace:

<https://v1.tailwindcss.com/components/navigation>

- **Foundation.**- Uno de los primeros frameworks en salir al mercado sin duda será foundation aunque con el pasar de los años a perdido popularidad este framework también dentro de sus clases tiene una sección para hacer header o menú de navegación usando la clase .menú, que por lo general este menú es un menú de navegación en el cual se suele usar y poner en la parte superior de página web que luego servirá de como una barra de navegación. Para ver el uso de este se puede recurrir al siguiente link:

<https://get.foundation/sites/docs/menu.html>

- **Materialize.**- Otro framework del medio es materialize el cual también cuenta con una sección llamada navbar el cual ayuda a construir menú de navegación. Lo podremos observar más detalladamente accediendo mediante el siguiente link:

<https://materializecss.com/navbar.html>

- **UIKIT.**- También este framework cuenta con una clase para hacer barra de navegación la clase empleada en este caso tiene por nombre .uk-navbar para ver más sobre este framework recomiendo revisar la documentación en el siguiente enlace:

<https://getuikit.com/docs/navbar>

- **Semantic UI.**- A continuación tenemos otro framework de css muy completo la verdad el cual tiene soporte y componentes para ser usado con react, la clase la cual ayuda a crear barra de navegación es denominada .menu , podemos ver una gran cantidad de ejemplos mediante el siguiente enlace:

<https://react.semantic-ui.com/collections/menu>

Modulo: Un módulo según definición propia es una sección de trabajo principal a comparación de un submódulo este puede ser denominado la estructura padre y los submódulos estructuras hijas. En este efecto en particular el módulo vendría a ser la siguiente clase:

- **Barra de navegación.** Para este efecto nos basamos en la sencillez de una barra de navegación, es algo clásico se le podría definir, tiene como color base el blanco y negro, así como un padding para mantenerlos exactamente al margen.

Submódulos: Considere llamarle Un submódulo, a los efectos o clases que están dentro de un módulo en este caso el módulo es barra de

navegación y footer porque comparten atributos similares. Un submódulo según denominación propia vendría hacer un efecto similar al original, pero con características diferentes, hagamos un ejemplo para comprender esto de una manera mejor.

- Barra de navegación estática. - Este sería el módulo original y tradicional donde la barra de navegación siempre está en la parte de arriba y como su nombre lo sugiere es estática. Este sería el módulo original
- Barra de navegación voladora. - Este sería un sub modulo ya que tendría propiedades compartidas con el módulo original, pero discreparían en funcionalidades mientras el módulo original es estático este sub modulo como hace referencia a su nombre se movería conforme al scroll que hace el usuario.
- Barra de navegación distribución. – En este caso también sería un sub modulo ya que como se mencionó anteriormente compartiría propiedades del módulo principal, pero se diferenciaría de los ya mencionados ya que podría tener las propiedades del módulo original, además tendría también las propiedades del submódulo barra de navegación voladora, adicionalmente su atributo principal seria distribuir el menú en 3 partes, una parte derecha para partes del menú como, inicio, trabajos. Una parte central con la clase .logo para poner el logo de la empresa, etc. Una parte izquierda para poner contáctenos y ubiquemos.

Estos submódulos son muy útiles ya que desde mi punto de vista ayuda a diversificar los efectos y las clases. Beneficiando al usuario final en la construcción de sitios web. Actualmente el framework XRL8 no tiene pensado sacar sub módulos para el efecto barra de navegación.

FASE 2.- Codificar el efecto

Antes de ver esta sección a fondo, aclararemos que en este efecto en particular la codificación de código no fue tan engorroso. Se usó propiedades básicas las cuales se procederá a detallar con lujo de detalle cómo fue su elaboración paso a paso.

Bien luego de tener claro que efecto queríamos construir y porque lo íbamos a construir como se vio en el capítulo anterior se procedió a escribir código como se muestra a continuación.

Código completo

Figura 1. Código de la barra de navegacion

```
1  .Barra-adm {  
2    width: 100%;  
3    height: 40px;  
4    background-color: black;  
5    position: fixed;  
6  }
```

Descripción de las propiedades más significativas:

A continuación, se procederá a detallar las propiedades más usadas y su aplicación. También se mostrará la compatibilidad web de cada propiedad para su mejor comprensión dejamos una leyenda en la cual se indica si tiene compatibilidad web, entre otros factores.

Figura 2. Leyenda de los estados del elemento



Propiedad width:100% :

Se empleó width 100% para el ancho de este efecto ya que si bien esta la propiedad width: tiene diversas unidades de medidas como:

- EM
- VH
- PIXELES
- REM
- PORCENTAJE

Se fue descartando una a una al momento de probarla, ya que la al usar la unidad EM este se distorsionaba por su propia naturaleza, básicamente esta propiedad usa el ancho de la pantalla del dispositivo donde estemos y las va ajustando así que fue descartado. Lo que se necesitaba era algo fijo así que probamos ahora con VH, si bien casi el 86% del código usado para la construcción del framework se usó esta unidad por los beneficios que este tiene al hacer responsive así como a la hora de mezclar propiedades reacciona bien, se optó por no usar ya que básicamente esta propiedad usa el alto del dispositivo para ajustarlo a medida que este valla variando no era lo que se buscaba ya que recapitulando un poco lo que se buscaba era que los valores no sean estáticos y sin importar el ancho o largo del dispositivo donde se encuentre el usuario este no debería distorsionarse. Así que se procedió a usar PIXELES en este caso si respondía bien pero no era el adecuado ya que al agregar contenido a la barra de navegación como texto u otros componentes este se distorcionarían. Solo quedaba 2 opciones probar con REM ,al hacerlo nos dimos cuenta que tampoco cubría las expectativas planteadas ya que se le podría denominar a REM que es pariente de EM ya que ambos funcionan de manera similar, ambos funcionan con el ancho del dispositivo donde se conecte el usuario así que al final usamos PORCENTAJE ya que este si cumplía las expectativas planteadas en cuanto al ancho no se distorsionaba sea cual sea la dimensión del dispositivo donde se conecte el usuario además de que viendo a futuro este no se distorsionaría sea cual sea el contenido que le agregáramos.

A continuación, vemos la compatibilidad web al usar esta propiedad en los principales navegadores.

Figura 3. Compatibilidad web propiedad width

Browser	Usage % of all users
IE	6-10
Edge	12-97
Firefox	2-96
Chrome	4-97
Safari	3.1-15.1
Opera	10-82
Safari on iOS	3.2-15.1
Opera Mini	all
Android Browser	2.1-4.4.4
Opera Mobile	12-12.1
Chrome for Android	98
Firefox for Android	96
UC Browser for Android	12.12
Samsung Internet	4-15.0
QQ Browser	16.0
Baidu Browser	10.4
KaiOS Browser	7.12
	2.5

Propiedad height:40px:

Luego de ver las unidades de medida en la propiedad anterior se decidió trabajar en pixeles, ya que se necesitaba un valor estático poniendo como un límite que no tenga en cuenta las dimensiones del dispositivo del usuario, en ese sentido se descartó las propiedades que toman el ancho del dispositivos como son REM y EM por otra parte también se descartó a los que toman el alto del dispositivo como son VH. Luego de reducir nuestro marco de trabajo se tenía PORCENTAJE o PIXELES así que descartamos a PORCENTAJE porque tampoco necesitábamos que sea ajustable el alto ya que normalmente para eso tendríamos que dejarlo en automático ya que por default el navegador le asigna ese valor.

Se empleó PIXELES en especial para el alto de la barra de navegación con la finalidad de poner un límite en cuanto al tema de la estética, ya que si sobrepasa esta dimensión se vería desproporcional ya que por lo general una barra de navegación solo lleva letras las cuales son palabras que redirigen a inicio, contáctenos, proyectos, ubíquenos, productos. En algunos casos llevan el logo de la compañía o la empresa. Dicho de paso estos logos son bien pequeños y difícilmente sobrepasan los 40px.

A continuación, vemos la compatibilidad web al usar esta propiedad en los principales navegadores.

Figura 4 Compatibilidad web propiedad height

IE	Edge	Firefox	Chrome	Safari	Opera	Safari on iOS	Opera Mini	Android Browser	Opera Mobile	Chrome for Android	Firefox for Android	UC Browser for Android	Samsung Internet	QQ Browser	Baidu Browser	KaiOS Browser
6-10	12-97	2-96	4-97	3.1-15.1	10-82	3.2-15.1		2.1-4.4.4	12-12.1				4-15.0			
11	98	97	98	15.3	83	15.3	all	98	64	98	96	12.12	16.0	10.4	7.12	2.5
		98-99	99-101	15.4-TP		15.4										

Propiedad background-color: black

Se le asigno este color por default a la clase ya que si bien puede ser cambiada. Se optó por dejarla ya que es un color elegante, así como un color primario.

A continuación, vemos la compatibilidad web al usar esta propiedad en los principales navegadores.

Figura 5 Compatibilidad web propiedad background-color

IE	Edge	Firefox	Chrome	Safari	Opera	Safari on iOS	Opera Mini	Android Browser	Opera Mobile	Chrome for Android	Firefox for Android	UC Browser for Android	Samsung Internet	QQ Browser	Baidu Browser	KaiOS Browser
6-10	12-97	2-96	4-97	3.1-15.1	10-82	3.2-15.1		2.1-4.4.4	12-12.1				4-15.0			
11	98	97	98	15.3	83	15.3	all	98	64	98	96	12.12	16.0	10.4	7.12	2.5
		98-99	99-101	15.4-TP		15.4										

Propiedad position: fixed

Esta propiedad en particular lo que hace independientemente del valor que seleccionemos es mantener la posición del elemento HTML5 en este caso, mantendrá la posición en nuestra barra de navegación. Para comprender un poco esta propiedad aremos analogía a un escuadro de soldados que van hacer un operativo. En este caso el escuadro de soldados seria la clase padre el cual dentro tendrá clase hijos que haciendo analogía seria cada uno de los soldados que forman parte del escuadrón. Continuando con la analogía un escuadrón mantendrá su posición cuando este en terreno desconocido y estén en fase de exploración, mantener la posición no es más que estar en un estado, este estado puede ser estático, como también puede ser estar en movimiento.

La propiedad position tiene varios valores los cuales son:

- STATIC
- RELATIVE
- ABSOLUTE
- STICKY
- FIXED

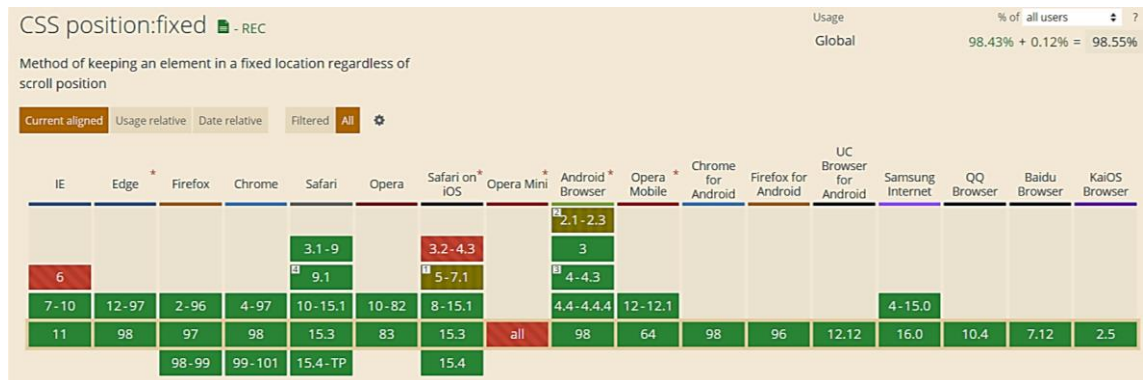
La posición STATIC, RELATIVE son más de conservadoras y tienen un comportamiento similar ya que ambas se combinan para que ocupe un espacio definido, dicho de otra forma ambas propiedades ocupan espacio en el DOM y no pueden sobreponerse uno encima del otro. Pero por otra parte la propiedad ABSOLUTE lo que hace es sobreponerse sobre cualquier elemento que tenga static o relative. Bien en la actualidad se considera mala práctica a la hora de maquetar usar la propiedad position para maquetar, ya que no es favorable para hacerlo responsive, en vez de eso ahora se usa flexbox, y en algunos casos grid. Dicho de paso ambas propiedades formaron parte en la actualización llamada css3.

Bien ahora conozcamos más a detalle que hacen los valores de STICKY y FIXED bien ambos valores comparten los mismos comportamientos una de las diferencias que tienen es que con FIXED es un valor estático pero absoluto a la vez, mientras que STICKY comparte las mismas propiedades que FIXED, pero la diferencia es que puede volver al lugar de origen y mientras esté haciendo scroll el usuario este se moverá como si fuera una propiedad FIXED.

La razón por la que uso FIXED en lugar de STICKY es porque quiero que la barra de navegación sea estática y absoluta a la vez para así poder jugar con los valores y efectos que le pongamos a un futuro, mientras que si eligiera STICKY este si bien volvería a su sitio de inicio y se vería mejor, lo cumpliría con la finalidad del framework ser escalable, ya que si por cualquier motivo quisiéramos que la barra de navegación cuente con más espacio o efectos este descuadraría la maqueta.

A continuación, vemos la compatibilidad web al usar esta propiedad en los principales navegadores.

Figura 6 Compatibilidad web propiedad fixed



Principales dificultades:

Para este efecto en concreto la principal dificultad fue organizar de distintas formas posibles, así como ver las posibles conjugaciones que pueden hacer con el efecto para que el usuario al final no tenga problemas al implementarlo. Esas dificultades surgieron a raíz de las siguientes preguntas:

- **¿Si el usuario quiere como podría hacer estático el efecto?**

Cambiando la propiedad `position:fixed` por una con `position:relative`, además una de las ventajas de escribir código css limpio es que se puede chancar el código de la clase reescribiendo el código y en últimas instancias usar de la propiedad `!important`.

- **¿El efecto debe ser lo más sencillo posible, como lo puedo hacer escalable?**

Se puede hacer escalable el proyecto agregando submódulos, estos pueden ser como los descritos al principio de esta página.

- **¿Qué pasa si el usuario quiere un menú vertical, serviría esta barra de navegación como base?**

No, ya que la construcción de un menú vertical es mucho más compleja ya que no depende de una propiedad, se define o se construye usando muchas propiedades y sobre todo probando cada una de ellas,

así como las conjugaciones que se tiene que emplear para que no choquen propiedades de una clase con otra.

- **¿Cómo hago para que este efecto no haga conflicto con otra clase?**

Para ello solo hay 2 caminos uno de ellos elimina el 60% de que esto ocurra y es usando la metodología OOCSS y el otro camino es probando y corrigiendo esos errores.

Otra dificultad fue saber cuál que valor tomar si el `position:fixed` o `position:sticky` ya que como mencionamos tienen similares usos, y para llegar a una decisión se tuvo que probar además, de que se pudo haber usado `position: absolute`, pero se descartó enseguida porque hace conflicto con las otras clases y con la forma del maquetado aunque se haya usado flex-box en la maquetación se observó que el problema persistiría así que se evitó en mayor medida usar la propiedad `position: absolute`.

¿Qué usamos para hacer las pruebas?

El navegador Google Chrome y opera mini. También se usa la siguiente página web. <http://www.responsinator.com>

Este con motivos de asegurarnos que funcione en dispositivos móviles y tabletas. Ya que en base a lo trabajado no es del todo fiable usar el navegador para verificar la correcta maquetación de los elementos, así como tampoco es fiable depender de un software web, por ello se decidió diversificar en cuanto a las herramientas para probar las clases y efectos trabajados en el framework XRL8.

También usamos como editor de código visual studio code y su extensión live server para visualizar en vivo los efectos realizados en el css en tiempo real, también usamos el autoguardado que trae dicho editor de código para no estar guardando el proyecto a cada rato. Así conseguimos un mejor desempeño y evitamos el tedioso guardar cambios a cada rato, también evitamos el que la memoria cache se llene mostrándonos los mismos resultados.

FASE 3.- Hacerlo responsive

Con este efecto en particular fue sencillo ya que la propiedad width estaba en % además de que no se necesitó una maquetación extra como veremos más adelante en el presente informe. En caso se hubiera dado la oportunidad de maquetar más el efecto para que tome forma y se moldee, lo que hubiéramos hecho en ese caso sería:

- Usar la propiedad flex-box y sus posibles conjugaciones con cada uno de sus valores según sea el caso.
- Hubiéramos trabajado con la propiedad @media query en las resoluciones de pantalla para celulares y tabletas. Siempre empezando a maquetar por la parte de celulares y luego tabletas.
- También hubiéramos usado las unidades de medida vh o rem para texto y contenedores ya que estas son imprescindibles a la hora de hacer responsive un sitio web.

FASE 4.- Testing asegurarse que no tenga conflicto con otros efectos.

En esta parte del desarrollo de la clase se dispuso a probar poniéndolo con las demás clases juntándolas. Para ello lo que se hizo fue colocar dentro de un archivo index.html todas las clases trabajadas y ver como el efecto de barra de navegación respondió. En algunos casos había conflicto con los demás efectos para corregir esto se procedió a implementar en el desarrollo dos cosas básicas la primera fue:

- Para detectar que clase era la que causaba el conflicto y ver en que archivo scss estaba. Se procedía a inspeccionar la página para detectar donde estaba el problema. En muchas ocasiones no se daba con el archivo causante del error entonces en ese caso aplicábamos la frase "Divide y vencerás". Lo que se procedió hacer fue comentar de uno en uno cada efecto en el archivo principal que los llamaba esto reducía las líneas de código de css, además como teníamos en tiempo real los cambios fue mucho más rápido detectar el problema. Ya que no se tenía que recargar a cada rato además otro punto a favor era que se auto

compilaba los cambios de sass automáticamente gracias a nodeJS el cual mediante la consola de comandos de Windows cuando detectaba un cambio en las líneas de código se auto compilaba y generando o reemplazando el archivo css en el cual hacíamos las pruebas, esto aumento la productividad para crear los efectos ya que al comentar de uno en uno cada archivo sass llegaba un momento donde desaparecía el error entonces revisábamos y nos dimos cuenta de que era uno o en algunos casos hasta tres efectos que hacían conflicto con la clase escrita.

- Se creó un archivo llamado `_error_logs.scss` para dentro de ese archivo escribir la clase que hace conflicto con la clase escrita, además de describir una nota breve en caso no se disponía de tiempo para solucionarlo luego, esto permitió tener un reporte más preciso de los incidentes que íbamos detectando.

FASE 5.- Aplicar la metodología OOCSS, llevándolo a SASS

- **Código css.** - Esta parte ya estaría completa por lo descrito anteriormente.
- **Código repetido.** - Identificación de código repetido

En este pequeño apartado luego de escribir y debugear el error se procedió a identificar el código css repetido para llevarlo a un archivo SCSS en sass, para su mayor optimización y aplicación de la metodología. En la presente clase descrita se observó que se repetía la clase de barra de navegación con la clase de footer. Así que se procedió a reciclar el siguiente código.

Figura 7 Código repetido del componente barra de navegación

```
1 width:100%;  
2 height: 40px;  
3 background-color:black;
```

- **Crear un objeto.** - crear un mixing y hacer la lógica de la función

Una vez identificado el código css repetido se procede hacer hace una función la cual se ejecute al llamado del código, mediante esta técnica nuestro código será más puro y limpio. Cabe recalcar que la barra de navegación, así como el pie de página comparte características similares ya que se consideró que fuesen simétricas, por ello en la función se consideró poner el nombre de los dos efectos que comparten el mismo código.

```
function header_footer(){  
  
width:100%;  
  
height:40px;  
  
}
```

Luego de tener la lógica se procede a llevarlo a SASS para ello tenemos dos opciones, la primera es usar @function ya que como su nombre ase referencia es para crear una función la otra opción es usar @mixin, pero ambas tienen diferencias. La primera retorna un valor, mientras @mixin solo llama a los valores en este caso no necesitamos que retorne un valor por ello usaremos @mixin. Entonces quedaría de la siguiente forma:

Figura 8 Creación del objeto header_footer en función mixin

```
1  @mixin header_footer{  
2    width:100%;  
3    height: 40px;  
4  }  
5
```

- **OOCSS.** – Luego de tener la función o la lógica para reciclar el código lo que se procede a

Integrar la metodología OOCSS para ello debemos llamar al mixin para ello solo necesitamos agregar @include “nombre del mixin”. Luego de integrarlo ya estará aplicada correctamente la metodología OOCSS

Figura 9 Metodología OOCSS en el componente barra de navegacion

```
1  .Barra-adm
2  {
3    @include header_footer;
4    background-color:$color-negro-1;
5    position:$barra-adm-position;
6  }
```

También muy importante se observa que al implementar la metodología queda fuera del mixin pero dentro de la clase .Barra-adm dos propiedades un background y position. Estas están acompañadas de variables dicho de paso SASS tiene la particularidad de que se puede asignar variables a las propiedades de css casi asemejándose a un lenguaje de programación.

Esto permite dos cosas:

- Usar condicionales dentro de SASS para acceder a clases de css, que veremos en el transcurso del proyecto.
- Hacer más personalizable el framework, ya para hacer cambio solo bastaría con cambiar la variable o añadir una en vez de revisar el código css ya compilado.

Bien volviendo al tema de las variables estas variables se encuentran en un archivo sass denominado variables valga la redundancia donde:

Figura 10 Variables Sass del componente barra de navegacion parte I

```
1  $barra-adm-position:fixed;
2  $barra-adm-fonts:$fuente_normal;
```

A su vez dentro del archivo variables están todas las variables usadas en la construcción del proyecto, así mismo como vemos en la imagen en la línea de código numero dos está llama a \$fuente_normal este a su vez es una variable de tipo de letra como se ve a continuación:

Figura 11 Variables Sass del componente barra de navegacion parte II

```
1  /***** Variables - Fuente_Tipografica *****/
2  /// Tipografia del framework
3  /// @group Tipografia
4  $fuente_comic:"Comic Sans MS", cursive, sans-serif;
5  /// @group Tipografia
6  $fuente_formal:Verdana, Geneva, sans-serif;
7  /// @group Tipografia
8  $fuente_periodico:Courier, monospace;
9  /// @group Tipografia
10 $fuente_soda:Impact, Charcoal, sans-serif;
```

Muy importante que si se usa una variable para llamar a otra variable como en un lenguaje de programación esta debe estar declarada de llamarla sino marca error en la consola.

Como se ha visto SASS es un preprocesador de css por ello se ejecuta antes del css tradicional esto permite añadir variables agregar condicionales, etc. Con la aplicación de la metodología como se observa se reutiliza el código css, además de que se hace escalable el proyecto. Ya que, si en un futuro queremos cambiar o añadir más propiedades css, solo cambiaríamos las variables o añadiríamos más propiedades en el mixin

FASE 6.- Documentar en la web site www.phooldx.com

Finalmente, si el efecto paso por todos los de más filtros se procede a su documentación respectiva, para documentar todos los efectos se creó una página web de documentación usando clases y elementos del mismo framework. A continuación, se describirá como se procede a la documentación de la misma. Siguiendo estos pasos:

- Se copia las etiquetas de HTML5 junto con la clase a un txt temporal para no perder la sintaxis
- Se guarda los cambios en el compilador de SASS
- Como es la parte final se procedió hacer copias de seguridad para no perder el avance, en archivos RAR. Estos archivos estaban por versiones

enumerados además se agregaba un pequeño texto describiendo el avance. Este guardado se hacía de manera local en un USB.

- Para una mayor seguridad del avance del código se procedió luego a copiar la versión del archivo RAR junto a la descripción del mismo.

Guardado de código en GitHub. - Luego de tener la copia del proyecto a nivel físico en un USB, se procedió a guardarlo en un repositorio de GitHub para lo cual se usó Git, para ello se procedió de la siguiente forma:

- Se guardaba todo el proyecto con el siguiente comando:

Figura 12 Comando git para agregar contenido al índice de trabajo

```
F:\ZONA TESIS\XRL8\XRL8>git add -A
warning: LF will be replaced by CRLF in .gitignore.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in .vscode/settings.json.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in BITACORA XRL8.txt.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in css/xrl8.css.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in dashboard.html.
```

- Luego se verificaba los cambios guardados con el siguiente comando:

Figura 13 Comando git para mostrar el estado del proyecto

```
Administrador: Símbolo del sistema
F:\ZONA TESIS\XRL8\XRL8>git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   css/xrl8.css
    modified:   css/xrl8.css.map
    modified:   documentacion/Documentacion Etiquetas.txt
    modified:   index.html
```

- Después se hacía el commit, es decir se le asignaba un nombre a los cambios realizados con el siguiente comando:

Figura 14. Comando git para empaquetar los datos con un nombre de referencia

```
Administrador: Símbolo del sistema
F:\ZONA TESIS\XRL8\XRL8>git commit -m "nombre de la version"
```


- Luego se procede a empaquetar los datos para que se suban al repositorio de GitHub con el siguiente código:

Figura 15. Comando git para subir los cambios a github

```
Administrador: Símbolo del sistema
F:\ZONA TESIS\XRL8\XRL8>git push origin master
Enumerating objects: 38, done.
Counting objects: 100% (38/38), done.
Delta compression using up to 8 threads
Compressing objects: 100% (21/21), done.
Writing objects: 100% (21/21), 50.64 KiB | 7.23 MiB/s, done.
Total 21 (delta 14), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (14/14), completed with 14 local objects.
To https://github.com/Phool-Frontend/XRL8
 26c1dbf..4fab8db master -> master
```

- Finalmente, luego de haber enviados el proyecto empaquetado hacia GitHub como se puede apreciar en la imagen, este queda guardado en dicho repositorio.

Figura 16. Visualización en github el componente barra de navegacion



Se guarda en una escala de tiempo, con todos los cambios hechos desde la creación del proyecto, esto ayuda a tener un mejor control de versiones. Asimismo, también ayuda a que más personas se sumen al proyecto y puedan clonar el proyecto y trabajar por separado el proyecto, entre otras muchas ventajas.

- Así mismo se manejó un archivo llamado bitácora.txt donde aquí se registró todos los incidentes ocurridos, así como los cambios hechos, se registraba también las metas cumplidas, también las tareas nuevas para

desarrollar al día siguiente. Al finalizar con un efecto por completo se quitaba de la lista de prioridades así dábamos pase al siguiente efecto o problema a resolver. Esto servía para tener un mejor orden y saber ejecutar prioridades.

- Luego de tener guardado el avance y de registrar el avanzase, se procedió a copiar del archivo Documentación Etiquetas.txt el fragmento de código que se copió anteriormente para luego pegarlo en www.nosetup.org esta página web se encargaba de convertir html en texto plano para poder visualizar html
- Antes de finalizar se procedió a crear un archivo llamado `modulo_barra_de_navegación.html` donde se dividía en tres partes:
 - En la primera se describía un poco al efecto que era y que hacia
 - En la segunda parte se procedía a copiar el código ya formateado obtenido de www.nosetup.com para luego pegarlo.
 - En la tercera parte lo que se hacía era mostrar un apartado llamado estructura donde se describía como se llamaban las variables para que sea modificado en SASS.

Por último, quedo de la siguiente forma lista para usarse en cualquier página web. Cabe recalcar que dentro del span usamos Font-awesome para tener el icono de menú hamburguesa.

Figura 17. Código html5 final del componente barra de navegación

```
<div class="Barra-adm">
  <span class="fa fa-align-justify">
</span>
</div>
```

PIE DE PAGINA

FASE 1. Investigación del efecto y su importancia, elegir un efecto o crearlo

Como primer paso lo primero que se hizo fue investigar e indagar sobre el efecto, en el caso del pie de página también llamado footer es uno de los principales pilares para la construcción de una web. Más que eso yo lo denominaría como una estructura principal e irrompible junto a la barra de navegación, como al cuerpo o contenido del sitio. Se le podría denominar uno de los pilares fundamentales un formato de construcción global para la web.

Otro beneficio es el posicionamiento web, ya que los principales motores de búsquedas del mundo, emplean la siguiente lógica para posicionar las páginas web como primeras.

- El uso de las etiquetas HTML5
- La estructura primaria o fundamental en los cuales son header, body, footer. Dicho de otra forma, la barra de navegación ya que dentro está el header como etiqueta principal
- La apertura y cierre correcto de las etiquetas HTML5

Como ya mencionamos uno de las características por el cual se escogió hacer este efecto fue por servir de estructura básica para todo proyecto construido en la web sea backend o frontend o una simple página web.

Otro del motivo el cual fue elegido el efecto para formar parte del framework XRL8 fue que el proyecto presente debe estar a la altura de los frameworks que están en la actualidad siendo usados en la industria del desarrollo web como son:

- **Bootstrap.** El presente framework si bien no cuenta con una clase especial para el footer en su documentación si cuenta con ejemplos denominados “snippets” donde muestran cómo se puede armar un pie de página usando otras clases del propio framework, en fin, observando este hecho se decidió crear un pie de página como clase además de que se encontró

que el pie de página que se encuentra en “snippets” no permanece siempre abajo si el contenido en el body es poco. Se encontró que si llevas la resolución de la pantalla a 1076px por 1938px a 2152px x 3876px esta no se mantiene siempre abajo y la principal función del footer es mantenerse siempre abajo sin importar el contenido que se encuentre. Si desean saber más sobre ello puede visitar la web oficial de documentación de dicho framework mediante el siguiente enlace:

<https://getbootstrap.com/docs/5.1/examples/#snippets>

- **Bulma.**- Este framework también cuenta con un modelo de pie de página denominado .footer, para ver cómo se emplea y formas de uso revisar el siguiente enlace que los llevara a la documentación oficial de dicho framework:

<https://bulma.io/documentation/layout/footer/>

- **TailwindCSS.**- Quiso sea este el framework de css más usado en los últimos 3 años sobre pasando a los demás por sus formas de integración con NodeJS y react. Este framework no cuenta con una clase específicamente para hacer el pie de página pero en cambio cuenta con componentes los cuales combinando muchas clase se puede lograr tener un pie de página. Para revisar su documentación ingresar al siguiente enlace:

https://tailwindui.com/components/marketing/page_section/footer

- **Foundation.**- Uno de los primeros frameworks en salir al mercado sin duda será foundation aunque con el pasar de los años a perdido popularidad este framework también dentro de sus clases tiene una sección para hacer footer o pie de página usando la clase .responsive-blog-footer. Para ver el uso de este se puede recurrir al siguiente link:

<https://get.foundation/building-blocks/blocks/responsive-blog-footer.html>

- **Materialize.-** Otro framework del medio es materialize el cual también cuenta con una sección llamada page-footer el cual ayuda a construir un pie de página. Lo podremos observar más detalladamente accediendo mediante el siguiente link:

<https://materializecss.com/footer.html>

- **UIKIT.-** También este framework cuenta con una clase para hacer el pie de página la clase empleada en este caso tiene por nombre .uk-card-footer para ver más sobre este framework recomiendo revisar la documentación en el siguiente enlace:

<https://getuikit.com/docs/card>

Modulo: Un módulo según definición propia es una sección de trabajo principal a comparación de un submódulo este puede ser denominado la estructura padre y los submódulos estructuras hijas. En este efecto en particular el módulo vendría a ser la siguiente clase:

- **Pie de página.** Para este efecto nos basamos en la sencillez de footer, es algo clásico se le podría definir, tiene como color base el blanco y negro, así como un padding para mantenerlos exactamente al margen. Se planea sacar a futuro nuevas versiones más robustas.

Submódulos: Considere llamarle Un submódulo, a los efectos o clases que están dentro de un módulo en este caso el módulo es barra de navegación y footer porque comparten atributos similares. Un submódulo según denominación propia vendría hacer un efecto similar al original, pero con características diferentes, hagamos un ejemplo para comprender esto de una manera mejor.

- **Pie de página.** Este sería el módulo original y tradicional donde el pie de página tiene las características de estar siempre abajo sin importar el contenido.

- **Pie de página social.** Este sería un sub módulo ya que tendría propiedades compartidas con el módulo original, pero tendría variantes que compartiría con el módulo principal el cual sería permanecer siempre abajo sin depender del contenido. La razón por la cual es un submódulo es por que comparte las características del módulo base llamado pie de página, pero la diferencia es que tiene en su estructura enlaces para las redes sociales.
- **Pie de página bidireccional.** En este caso también sería un submódulo ya que como se mencionó anteriormente comparte características del módulo principal que es pie de página, pero lo que le diferencia de los demás es que este submódulo, esta maquetado en su estructura interna para ser bidireccional como su mismo nombre lo dice dicho en pocas palabras estaría separado en dos lados uno derecho para poner las redes sociales como también poner enlaces, el lado izquierdo seria para poner el famoso copyright.

Estos submódulos son muy útiles ya que desde mi punto de vista ayuda a diversificar los efectos y las clases. Beneficiando al usuario final en la construcción de sitios web. Actualmente el framework XRL8 no tiene pensado sacar sub módulos para el efecto pie de página.

FASE 2.- Codificar el efecto

Antes de ver esta sección a fondo, debo aclarar que la codificación fue sencilla ya que se decidió que sería semejante a la de la barra de navegación, esto se hizo con temas de distribución de espacio para que sea simétrico y así evitar una abrupta desproporción, cabe señalar que esto puede cambiar ya que framework es moldeable sea por parte de un usuario que solo usara el framework como para un desarrollador front-end con conocimiento del preprocesador SASS.

Bien luego de tener claro que efecto queríamos construir y porque lo íbamos a construir como se vio en el capítulo anterior se procedió a escribir código como se muestra a continuación.

Código completo

Figura 18. Código de pie de pagina

```
1  .PieDePagina {  
2    width: 100%;  
3    height: 40px;  
4    text-align: center;  
5    padding: 10px;  
6    background: black;  
7    color: white;  
8  }
```

Descripción de las propiedades más significativas:

A continuación, se procederá a detallar las propiedades más usadas y su aplicación. También se mostrará la compatibilidad web de cada propiedad para su mejor comprensión esta se observa en la figura 2.

Propiedad width:100% :

Se empleó width 100% para el ancho de este efecto ya que si bien esta la propiedad width: tiene diversas unidades de medidas como:

- EM
- VH
- PIXELES
- REM
- PORCENTAJE

Se fue descartando una a una al momento de probarla, ya que la al usar la unidad EM este se distorsionaba por su propia naturaleza, básicamente esta propiedad usa el ancho de la pantalla del dispositivo donde estemos y las va ajustando así que fue descartado. Lo que se necesitaba era algo fijo así que probamos ahora con VH, si bien casi el 86% del código usado

para la construcción del framework se usó esta unidad por los beneficios que este tiene al hacer responsive así como a la hora de mezclar propiedades reacciona bien, se optó por no usar ya que básicamente esta propiedad usa el alto del dispositivo para ajustarlo a medida que este valla variando no era lo que se buscaba ya que recapitulando un poco lo que se buscaba era que los valores no sean estáticos y sin importar el ancho o largo del dispositivo donde se encuentre el usuario este no debería distorsionarse. Así que se procedió a usar PÍXELES en este caso si respondía bien pero no era el adecuado ya que al agregar contenido a la barra de navegación como texto u otros componentes estos se distorsionarían. Solo quedaba 2 opciones probar con REM ,al hacerlo nos dimos cuenta que tampoco cubría las expectativas planteadas ya que se le podría denominar a REM que es pariente de EM ya que ambos funcionan de manera similar, ambos funcionan con el ancho del dispositivo donde se conecte el usuario así que al final usamos PORCENTAJE ya que este si cumplía las expectativas planteadas en cuanto al ancho no se distorsionaba sea cual sea la dimensión del dispositivo donde se conecte el usuario además de que viendo a futuro este no se distorsionaría sea cual sea el contenido que le agregáramos.

A continuación, veamos la compatibilidad web al usar esta propiedad en la figura 3.

Propiedad height:40px:

Luego de ver las unidades de medida en la propiedad anterior se decidió trabajar en píxeles, ya que se necesitaba un valor estático poniendo como un límite que no tenga en cuenta las dimensiones del dispositivo del usuario, en ese sentido se descartó las propiedades que toman el ancho del dispositivos como son REM y EM por otra parte también se descartó a los que toman el alto del dispositivo como son VH .Luego de reducir nuestro marco de trabajo se tenía PORCENTAJE o PÍXELES así que descartamos a PORCENTAJE porque tampoco necesitábamos que sea ajustable el alto ya que normalmente para eso tendríamos que dejarlo en automático ya que por default el navegador le asigna ese valor.

Se empleó PÍXELES en especial para el alto de la barra de navegación con la finalidad de poner un límite en cuanto al tema de la estética, ya que si sobrepasa esta dimensión se vería desproporcional ya que por lo general una barra de navegación solo lleva letras las cuales son palabras que redirigen a inicio, contáctenos, proyectos, ubiquemos, productos. En algunos casos llevan el logo de la compañía o la empresa. Dicho de paso estos logos son bien pequeños y difícilmente sobrepasan los 40px.

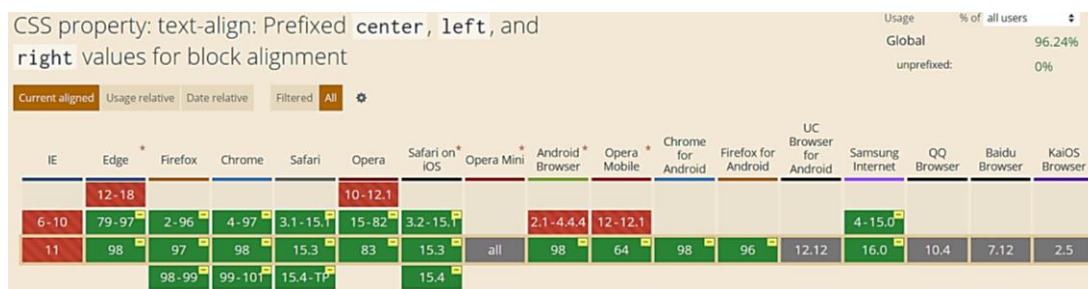
A continuación, veamos la compatibilidad web al usar esta propiedad en la figura 4.

Propiedad text-align: center

Se añadió esta propiedad ya que, al ser un pie de página sencillo, así como generalmente mostrar un mensaje de copyright como derechos de autor como también avisos legales estos deben estar centrados por ello se eligió ese valor.

A continuación, veamos la compatibilidad web al usar esta propiedad en los principales navegadores.

Figura 19. Compatibilidad web, propiedad text-align.

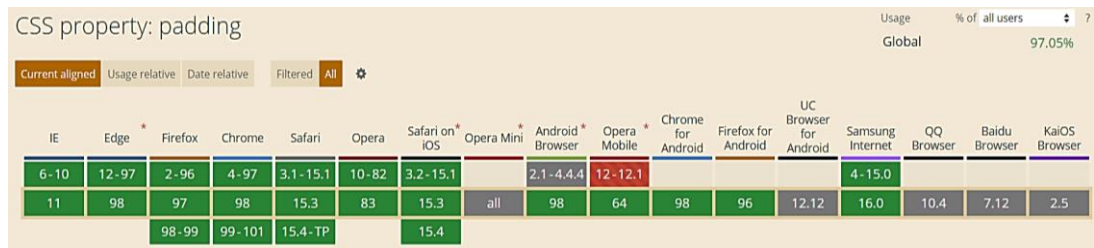


Propiedad padding:10px

Por ser un pie de página este debe pasar desapercibido en la mayoría de casos por ende no debe llamar mucho la atención como el contenido o la barra de navegación, así que se decidió optar por incluir la unidad de medida pixeles ya que iba más de acuerdo a la ocasión.

A continuación, veamos la compatibilidad web al usar esta propiedad en los principales navegadores.

Figura 20. Compatibilidad web, propiedad padding.



Propiedad background-color: black

Se le asignó este color por default a la clase ya que si bien puede ser cambiada. Se optó por dejarla ya que es un color elegante, así como un color primario.

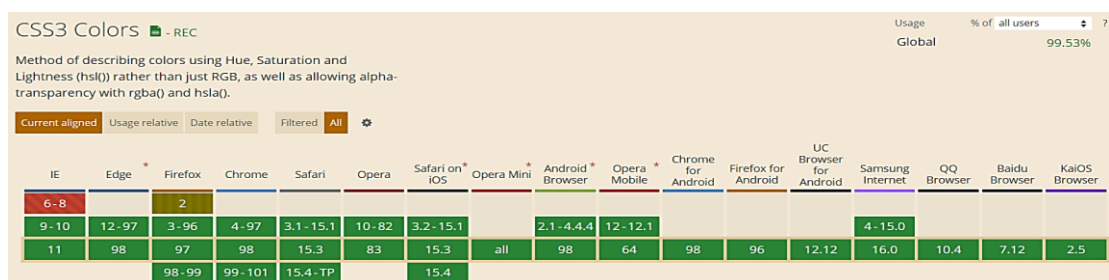
A continuación, vemos la compatibilidad web al usar esta propiedad en la figura 5.

Propiedad color: white

Como color por default se eligió el blanco ya que es combinaría con cualquier tonalidad sea oscura o clara salvo con el mismo color blanco claro está, además de esta puede ser modificada mediante otra clase que le puede agregar el usuario o el desarrollador web.

A continuación, vemos la compatibilidad web al usar esta propiedad en los principales navegadores.

Figura 21. Compatibilidad web, propiedad color



Principales dificultades:

Para este efecto en concreto la principal dificultad fue organizar de distintas formas posibles, así como ver las posibles conjugaciones que pueden hacer con el efecto para que el usuario al final no tenga problemas al implementarlo. Esas dificultades surgieron a raíz de las siguientes preguntas:

¿Cómo haga que siempre flote hacia abajo el footer?

Este fue el único inconveniente y el más difícil de solucionar ya que se llegó a la conclusión de que footer solo se iba para abajo cuando había bastante contenido y este dicho de una forma extravagante aplastaba al footer y solo ahí funcionaba el dicho de siempre abajo, se usó varias artimañas de css como el uso de position:absolute y botom:0 entre muchas otras pero ninguna satisfacía lo que se buscaba ya que solo funcionaba para determinados resoluciones de pantallas como por ejemplo el modo pc o el modo tabletas etc. Luego de un arduo trabajo se llegó a la conclusión para al fin responder la pregunta. La respuesta es usar un div como contenedor en la maquetación

¿Qué usamos para hacer las pruebas?

El navegador Google Chrome y opera mini. También se usa la siguiente página web. <http://www.responsinator.com>

Este con motivos de asegurarnos que funcione en dispositivos móviles y tabletas. Ya que en base a lo trabajado no es del todo fiable usar el navegador para verificar la correcta maquetación de los elementos, así como tampoco es fiable depender de un software web, por ello se decidió diversificar en cuanto a las herramientas para probar las clases y efectos trabajados en el framework XRL8.

También usamos como editar de código visual studio code y su extensión live server para visualizar en vivo los efectos realizados en el css en tiempo real, también usamos el autoguardado que trae dicho editor de código para no estar guardando el proyecto a cada rato. Así conseguimos un mejor desempeño y evitamos el tedioso guardar cambios a cada rato, también evitamos el que la memoria cache se llene mostrándonos los mismos resultados.

FASE 3.- Hacerlo responsive

Con este efecto en particular fue sencillo ya que la propiedad width estaba en % además de que no se necesitó una maquetación extra como veremos más adelante en el presente informe. En caso se hubiera dado la oportunidad de maquetar más el efecto para que tome forma y se moldee, lo que hubiéramos hecho en ese caso sería:

- Usar la propiedad flex-box y sus posibles conjugaciones con cada uno de sus valores según sea el caso.
- Hubiéramos trabajado con la propiedad @media query en las resoluciones de pantalla para celulares y tabletas. Siempre empezando a maquetar por la parte de celulares y luego tabletas.
- También hubiéramos usado las unidades de medida vh o rem para texto y contenedores ya que estas son imprescindibles a la hora de hacer responsive un sitio web.

FASE 4.- Testing asegurarse que no tenga conflicto con otros efectos.

En esta parte del desarrollo de la clase se dispuso a probar poniéndolo con las demás clases juntándolas. Para ello lo que se hizo fue colocar dentro de un archivo index.html todas las clases trabajadas y ver como el efecto pie de página responde. Una de los conflictos frecuentes fue en que el footer se posicionaba encima del contenido de la página web, otro conflicto fue con el sistema de rejillas, para corregir esto se procedió a implementar en el desarrollo dos cosas básicas la primera fue:

- Para detectar que clase era la que causaba el conflicto y ver en que archivo scss estaba. Se procedía a inspeccionar la página para detectar donde estaba el problema. En muchas ocasiones no se daba con el archivo causante del error entonces en ese caso aplicábamos la frase “Divide y vencerás”. Lo que se procedió hacer fue comentar de uno en uno cada efecto en el archivo principal que los llamaba esto reducía las líneas de código de css, además como teníamos en tiempo real los cambios fue mucho más rápido detectar el problema. Ya que no se tenía

que recargar a cada rato además otro punto a favor era que se auto compilaba los cambios de sass automáticamente gracias a nodeJS el cual mediante la consola de comandos de Windows cuando detectaba un cambio en las líneas de código se auto compilaba y generando o reemplazando el archivo css en el cual hacíamos las pruebas, esto aumento la productividad para crear los efectos ya que al comentar de uno en uno cada archivo sass llegaba un momento donde desaparecía el error entonces revisábamos y nos dimos cuenta de que era uno o en algunos casos hasta tres efectos que hacían conflicto con la clase escrita.

- Se procedió a escribir mediante comentarios en un archivo llamado `_error_logs.scss` la clase que hace conflicto con la clase escrita, además de describir una nota breve en caso no se disponía de tiempo para solucionarlo luego, esto permitió tener un reporte más preciso de los incidentes que íbamos detectando.

FASE 5.- Aplicar la metodología OOCSS, llevandolo a SASS

- **Código css.** - Esta parte ya estaría completa por lo descrito anteriormente.

- **Código repetido.** - Identificación de código repetido

En este pequeño apartado luego de escribir y debugear el error se procedió a identificar el código css repetido para llevarlo a un archivo SCSS en sass, para su mayor optimización y aplicación de la metodología. En la presente clase descrita se observó que se repetía la clase de footer con la clase de barra de la barra de navegación. Así que se procedió a reciclar el siguiente código.

Figura 22. Código repetido del componente pie de pagina

```
1 width:100%;  
2 height: 40px;  
3 background-color:black;
```

- **Crear un objeto.** - crear un mixing y hacer la lógica de la función

Una vez identificado el código css repetido se procede hacer hace una función la cual se ejecute al llamado del código, mediante esta técnica nuestro código será más puro y limpio. Cabe recalcar que el pie de página, así como la barra de navegación comparte características similares ya que se consideró que fuesen simétricas, por ello en la función se consideró poner el nombre de los dos efectos que comparten el mismo código.

```
function header_footer(){  
  
width:100%;  
  
height:40px;  
  
}
```

Luego de tener la lógica se procede a llevarlo a SASS para ello tenemos dos opciones, la primera es usar @function ya que como su nombre ase referencia es para crear una función la otra opción es usar @mixin, pero ambas tienen diferencias. La primera retorna un valor, mientras @mixin solo llama a los valores en este caso no necesitamos que retorne un valor por ello usaremos @mixin esto se puede apreciar en la figura 8.

- **OOCSS.** – Luego de tener la función o la lógica para reciclar el código lo que se procede a

Integrar la metodología OOCSS para ello debemos llamar al mixin para ello solo necesitamos agregar @include “nombre del mixin”. Luego de integrarlo ya estará aplicada correctamente la metodología OOCSS

Figura 23. Metodología OOCSS en el componente pie de pagina

```
1  .PieDePagina  
2  {  
3      @include header_footer;  
4      text-align:$pie-pagina-texto;  
5      padding:$pie-pagina-padding;  
6      background:$color-negro-1;  
7      color:$color-blanco-1;  
8  }
```

También muy importante se observa que al implementar la metodología queda fuera del mixin pero dentro de la clase .PieDePagina cuatro propiedades un text-align, padding, background, color. Estas están acompañadas de variables ya que sass permite declarar variables a las propiedades de css casi asemejándose a un lenguaje de programación.

Esto permite dos cosas:

- Usar condicionales dentro de SASS para acceder a clases de css, que veremos en el transcurso del proyecto.
- Hacer más personalizable el framework, ya para hacer cambio solo bastaría con cambiar la variable o añadir una en vez de revisar el código css ya compilado.

Bien volviendo al tema de las variables estas variables se encuentran en un archivo sass denominado variables valga la redundancia donde:

Figura 24. Variables Sass del componente pie de página parte I

```
$pie-pagina-texto:center;  
$pie-pagina-padding:10px;
```

Estos son variables enfocados exclusivamente al efecto pie de página.

Figura 25. Variables Sass del componente pie de página parte II

```
/// Color primario blanco  
$color-blanco-1:white;  
  
/// Color negro N°01  
$color-negro-1:black;
```

Estas variables son globales las cuales se pueden usar en cualquier efecto

Muy importante que si se usa una variable para llamar a otra variable como en un lenguaje de programación esta debe estar declarada de llamarla sino marca error en la consola.

Como se ha visto SASS es un preprocesador de css por ello se ejecuta antes del css tradicional esto permite añadir variables agregar condicionales, etc. Con la aplicación de la metodología como se observa se reutiliza el código css, además de que se hace escalable el proyecto. Ya que, si en un futuro queremos cambiar o añadir más propiedades css, solo cambiaríamos las variables o añadiríamos más propiedades en el mixin

FASE 6.- Documentar en la web site www.phooldx.com

Finalmente, si el efecto paso por todos los de más filtros se procede a su documentación respectiva, para documentar todos los efectos se creó una página web de documentación usando clases y elementos del mismo framework. A continuación, se describirá como se procede a la documentación de la misma. Siguiendo estos pasos:

- Se copia las etiquetas de HTML5 junto con la clase a un txt temporal para no perder la sintaxis
- Se guarda los cambios en el compilador de SASS
- Como es la parte final se procedió hacer copias de seguridad para no perder el avance, en archivos RAR. Estos archivos estaban por versiones enumerados además se agregaba un pequeño texto describiendo el avance. Este guardado se hacía de manera local en un USB.
- Para una mayor seguridad del avance del código se procedió luego a copiar la versión del archivo RAR junto a la descripción del mismo.

Guardado de código en GitHub. - Luego de tener la copia del proyecto a nivel físico en un USB, se procedió a guardarlo en un repositorio de GitHub para lo cual se usó Git, para ello se procedió de la siguiente forma:

- Se guardaba todo el proyecto con el siguiente comando, que se puede apreciar en la figura 12.
- Luego se verificaba los cambios guardados con el siguiente comando, que se puede apreciar en la figura 13.
- Después se hacía el commit, es decir se le asignaba un nombre a los cambios realizados en la figura 14 se aprecia dicho comando.

- Luego se procede a empaquetar los datos para que se suban al repositorio de GitHub como se aprecia en la figura 15.
- Finalmente, luego de haber enviados el proyecto empaquetado hacia GitHub como se puede apreciar en la imagen, este queda guardado en dicho repositorio.

Figura 26. Visualización en github el componente pie de pagina



Se guarda en una escala de tiempo, con todos los cambios hechos desde la creación del proyecto, esto ayuda a tener un mejor control de versiones. Asimismo, también ayuda a que más personas se sumen al proyecto y puedan clonar el proyecto y trabajar por separado el proyecto, entre otras muchas ventajas.

- Así mismo se manejó un archivo llamado bitácora.txt donde aquí se registró todos los incidentes ocurridos, así como los cambios hechos, se registraba también las metas cumplidas, también las tareas nuevas para desarrollar al día siguiente. Al finalizar con un efecto por completo se quitaba de la lista de prioridades así dábamos pase al siguiente efecto o problema ha resolver. Esto servía para tener un mejor orden y saber ejecutar prioridades.
- Luego de tener guardado el avance y de registrar el avanzase, se procedió a copiar del archivo Documentación Etiquetas.txt el fragmento de código que se copió anteriormente para luego pegarlo en www.nosetup.org esta página web se encargaba de convertir html en texto plano para poder visualizar html
- Antes de finalizar se procedió a crear un archivo llamado modulo_pie_de_pagina.html donde se dividía en tres partes:

- En la primera se describía un poco al efecto que era y que hacia
- En la segunda parte se procedía a copiar el código ya formateado obtenido de www.nosetup.com para luego pegarlo.
- En la tercera parte lo que se hacía era mostrar un apartado llamado estructura donde se describía como se llamaban las variables para que sea modificado en SASS.

Figura 27. Código html5 final del componente pie de pagina

```
<div class="PieDePagina">  
  copy@Phool Herrera Condezo  
</div>
```

INPUT FORMULARIO

FASE 1.- Investigación del efecto y su importancia

Como primer paso lo primero que se iso fue investigar e indagar sobre el efecto, al hacer esto nos dimos cuenta que estrictamente es una parte fundamental sobre todo en el desarrollo front-end para la recepción de datos mediante los inputs que luego de ser tratados se envía al backend.

Cabe recalcar que muchos autores consideran dentro de sus formularios a los inputs, dicho de otra forma, los inputs y formularios están en una sola clase, pero la mayoría de frameworks como es el caso de XRL8 lo consideramos por separado debido a que no necesariamente debe haber un formulario de por medio para usar un input tipo text o tipo email para simplemente recibir uno o dos datos de por medio.

Otro punto importante cabe recalcar es que cada framework que está en producción tienen este efecto para los inputs debido a la importancia en el desarrollo web ya que estos formularios permiten recopilar y mostrar información al usuario. Algunos de esos frameworks son:

- **Bootstrap.** - El presente framework cuenta con la clase `.form-control` el cual sirve para como contenedor poder dar estilos a los distintos inputs

como tipo fecha, texto, área, etc. Si desean saber más sobre ello puede visitar la web oficial de documentación de dicho framework mediante el siguiente enlace:

<https://getbootstrap.com/docs/4.0/components/forms/>

- **Bulma.**- Este framework también cuenta con un diseño para los inputs los cuales la clase principal es denominada .field, para ver cómo se emplea y formas de uso revisar el siguiente enlace que los llevara a la documentación oficial de dicho framework:

<https://bulma.io/documentation/form/general/>

- **TailwindCSS.** - Quizá sea este el framework de css más usado en los últimos 3 años sobre pasando a los demás por sus formas de integración con NodeJS y react. Este framework dentro de su contenido también tiene una clase llamada. w-full que combinadas con otras clases se puede construir un formulario con inputs muy llamativos. Gracias a sus componentes, estos a diferencia de los demás se caracteriza por no tener clases predefinidas para efectos sino todo lo contrario tiene un conjunto de clases en el cual el usuario puede elegir los efectos más simples para colocar a su diseño. Para revisar su documentación ingresar al siguiente enlace:

<https://v1.tailwindcss.com/components/forms>

- **Foundation.**- Uno de los primeros frameworks en salir al mercado sin duda será foundation aunque con el pasar de los años a perdido popularidad este framework también dentro de sus clases tiene una sección para hacer formularios pero en particular no se enfoca mucho en los inputs que van dentro de los formularios como si lo hace XRL8 o los demás frameworks ya mencionados anteriormente, sino que los trabaja por separado por lo general van dentro de la clase .row donde este es el contenedor. Para ver el uso de este se puede recurrir al siguiente link:

<https://get.foundation/sites/docs-v5/components/forms.html>

- **Materialize.**- Otro framework del medio es materialize el cual también cuenta con una sección llamada formularios el cual ayuda a construir formularios más rápidamente, su modo de uso para la mayoría de efectos es abrir una clase `.row` donde este será el contenedor luego dentro colocar la clase `.input-field` con ello ya se puede tener un formulario con su input texto personalizado además de otras clases más que debe de tener ya que solo mencione lo más importante. Lo podremos observar más detalladamente accediendo mediante el siguiente link:

<https://materializecss.com/text-inputs.html>

- **UIKIT.**- También este framework cuenta con una clase para dar diseño y buen aspecto a los formularios, la clase más usada es la clase `.uk-input` el cual se usa con el input tipo texto, dicho de paso está bien elaborada la página de documentación la cual pueden visitar, ya que tiene un efecto para cada tipo de input sea text, email, date, select, etc. Lo pueden revisar con mayor profundidad ingresando al siguiente enlace:

<https://getuikit.com/docs/form>

- **Semantic UI.**- A continuación tenemos otro framework de css muy completo la verdad el cual tiene soporte y componentes para ser usado con react, la clase la cual ayuda a crear formularios y a personalizar inputs la clase se llama `.ui form` se usa como contenedor para darle las propiedades generales, pero dentro para darle estilos al input se usa la clase `.field` podemos ver una gran cantidad de ejemplos mediante el siguiente enlace:

<https://semantic-ui.com/collections/form.html>

Modulo: Un módulo según definición propia es una sección de trabajo principal a comparación de un submódulo este puede ser denominado la estructura padre y los submódulos estructuras hijas. En este efecto en particular el módulo vendría a ser la siguiente clase:

- **Input clásico.** - Para este efecto nos basamos en los inputs tradicionales sin mucha decoración, tiene espacios internos con la propiedad padding,

también se la unidad vh para hacerlo receptivo a distintos tipos de pantalla. Ya que la unidad vh toma el alto de la pantalla y como los dispositivos son proporcionales a la altura fue la mejor opción para optar.

Una de las mayores dificultades en este módulo fue el probar las distintas unidades de medidas para que fueran adaptativas a distintos tipos de pantalla o encontrar al que se viera bien en todas, ya que son 2 cosas distintas el hacerlas responsive el maquetado y otra muy distinta hacer responsive los formularios.

Adicionalmente para probar que el efecto lograra alcanzar un buen estándar de calidad se probó físicamente y virtualmente. Ya que no me convencía del todo hacer las pruebas en el navegador o usando páginas webs externas.

Para las pruebas responsive en una pc física hablando literalmente se probó en un PC de escritorio con las siguientes características de pantalla.

1920x1080

Luego se probó en una laptop con las siguientes dimensiones de pantalla:

1366 x 768

Posteriormente se encontró una forma más segura y confiable de hacer el responsive para tabletas y celulares. Él fue usar la siguiente página web:

<http://www.responsinator.com>

Los formatos de inputs que soportan la clase son:

- Input text
- Input number
- Input email
- Input password

Submódulos: Considere llamarle submódulo, a los efectos o clases que están dentro de un módulo en este caso el módulo es input formulario. Un submódulo según denominación propia vendría hacer un efecto o clase similar al original caso contrario estos comparten los mismos rasgos, pero tienen características que lo hacen diferente, veamos a continuación cuales son los submódulos del módulo input formulario:

- **Input volador.** - Este módulo fue pensado en diversificar nuestras clases en los formularios, consiste en que el placeholder salga de su estado en reposo para volar al margen superior del input dando un efecto de 3D para luego poder escribir en el input.

Los formatos de inputs que soportan la clase son:

- Input text
 - Input number
 - Input email
 - Input password
-
- **Input diseñador.** - Este submódulo fue creado con la finalidad de distinguimos de la competencia, proporciona solides al usarlo. Ya que se ven extraordinariamente bien y junto a sus predecesores tiene un efecto 3D ya que el placeholder se eleva a la parte de arriba.

Pero este tiene la cualidad de tener una barra de carga debajo muy sutil la verdad que se dispara al hacer clic en el input.

Cada submódulo tiene 3 tamaños pequeño, mediano y grande. Esto lo que hace es facilita la visualización al momento de hacerlo responsive. Estos submódulos son muy útiles ya que desde mi punto de vista ayuda a diversificar los efectos y las clases. Beneficiando al usuario final en la construcción de sitios web.

FASE 2.- Codificar el efecto

Antes de ver esta sección a fondo quisiera hacer énfasis en lo tedioso que fue codificar cada uno de los submódulos ya que estos necesitaban una posición diferente, así como también fue muy ajetreado el identificar que unidad de medida usar en los submódulos ya que hacía conflicto o quedaban descuadrados. Bien luego de tener claro que efecto queríamos construir y porque lo íbamos a construir como se vio en el capítulo anterior se procedió a escribir código como se muestra a continuación.

Código completo

❖ Código del módulo input clásico

[parte 01 – input clásico]

Figura 28. Código de input formulario parte I

```
1  .campo_form {
2    padding: 10px 20px;
3  }
4  .campo_form input {
5    height: 5vh;
6  }
7
8  .form_base {
9    width: 80%;
10   height: auto;
11   border: solid 4.3px #f7f7f9;
12   margin: 0 auto;
13   border-radius: 3px;
14   background-color: white;
15 }
16 .form_base label {
17   color: #363636;
18   font-weight: 700;
19   font-size: 0.91rem;
20 }
21
22 .campo_form input[type=text] {
23   line-height: 200px;
24 }
25 .campo_form input[type=email] {
26   line-height: 200px;
27 }
28 .campo_form .buscador {
29   display: flex;
30   padding: 0 0 10px 0;
31   align-items: center;
32   justify-content: center;
33 }
34 .campo_form .buscador .fa {
35   color: black;
36   border: 1px solid silver;
37   border-radius: 0.3rem 0 0 0.3rem !important;
38   border-right: none;
39   height: 5vh;
40   line-height: 5vh;
41   padding: 0 1vh;
42 }
```

[parte 02 – input clásico]

Figura 29. Código de input formulario parte II

```
43 .campo_form .buscador input[type=search] {
44   font-weight: 100;
45   width: 100%;
46   border-radius: 0 0.3rem 0.3rem 0 !important;
47 }
48 .campo_form .max_buscador 1 {
49   height: 5.9vh !important;
50   line-height: 5.9vh !important;
51 }
52 .campo_form .max_buscador input[type=search] {
53   height: 5.9vh;
54 }
55 .campo_form .min_buscador 1 {
56   height: 4vh !important;
57   line-height: 4vh !important;
58 }
59 .campo_form .min_buscador input[type=search] {
60   height: 4vh;
61 }
62 .campo_form .min_clasi {
63   height: 4vh;
64 }
65 .campo_form .max_clasi {
66   height: 5.9vh;
67 }
68
69 .form_base input, .form_base textarea {
70   padding-left: 10px;
71   font-size: 15px;
72   border: 1px solid silver;
73   border-radius: 0.3rem;
74   width: 100%;
75 }
76 .form_base input:focus, .form_base textarea:focus {
77   transition: border-color 0.3s ease-in-out, box-shadow 0.3s ease-in-out;
78   outline: 0;
79   box-shadow: 0 0 0 3px rgba(13, 110, 253, 0.04);
80   border-color: #86b7fe;
81 }
```

❖ Código del submódulo input volador

[parte 01 – input volador]

Figura 30. Código de input formulario parte III

```
1
2 .contenedor_form_volador {
3   margin: auto;
4   max-width: 80%;
5   padding: 2vw;
6   box-sizing: border-box;
7   border: 1px solid #dadce0;
8   -webkit-border-radius: 8px;
9   border-radius: 8px;
10  background-color: white;
11 }
12 .contenedor_form_volador h2 {
13   padding-bottom: 30px;
14   color: #fff;
15   text-align: center;
16   color: #202124;
17   font-family: "Google Sans", "Noto Sans Myanmar UI", arial, sans-serif;
18   font-size: 24px;
19   font-weight: 400;
20 }
21 .contenedor_form_volador input[type=submit] {
22   border: none;
23   outline: none;
24   color: #fff;
25   background-color: #1a73e8;
26   padding: 0.625rem 1.25rem;
27   cursor: pointer;
28   border-radius: 0.312rem;
29   font-size: 1rem;
30   margin: 0 auto;
31 }
```


[parte 02 – input volador]

Figura 31. Código de input formulario parte IV

```
32
33 .contenedor_form_volador .campo_bord_volador input {
34   width: 100%;
35   background: transparent;
36   margin-bottom: 1.875rem;
37   border-radius: 4px;
38   font-size: larger;
39   border: 1px solid #ccc;
40   padding: 0.5rem 0.7rem;
41 }
42 .contenedor_form_volador .campo_bord_volador label {
43   position: absolute;
44   top: 0;
45   pointer-events: none;
46   transition: 0.5s;
47   padding: 0.5rem;
48   color: grey;
49   left: 30px;
50   font-size: 0.9rem;
51 }
52 .contenedor_form_volador .campo_bord_volador input[type=number] {
53   padding: 0.55rem 0.7rem;
54   padding-bottom: 0.5rem;
55 }
56 .contenedor_form_volador .campo_bord_volador input[type=date] {
57   color: #ffffff;
58   padding: 0.31rem 0.7rem;
59 }
60 .contenedor_form_volador .campo_bord_volador input[type=date]::before {
61   color: #999999;
62 }
```

[parte 03 – input volador]

Figura 32. Código de input formulario parte V

```
63 .contenedor_form_volador .campo_bord_volador input[type=date]:focus,
64 .contenedor_form_volador .campo_bord_volador input[type=date]:valid {
65   color: black;
66 }
67 .contenedor_form_volador .campo_bord_volador input[type=date]:focus::before,
68 .contenedor_form_volador .campo_bord_volador input[type=date]:valid::before {
69   content: "" !important;
70 }
71 .contenedor_form_volador .campo_bord_volador input[type=date]::before {
72   color: #999999;
73 }
74 .contenedor_form_volador .campo_bord_volador input[type=date] {
75   color: #ffffff;
76 }
77 .contenedor_form_volador .campo_bord_volador input[type=date]:focus,
78 .contenedor_form_volador .campo_bord_volador input[type=date]:valid {
79   color: black;
80 }
81 .contenedor_form_volador .campo_bord_volador input[type=date]:focus::before,
82 .contenedor_form_volador .campo_bord_volador input[type=date]:valid::before {
83   content: "" !important;
84 }
85 .contenedor_form_volador .campo_bord_volador .buscador {
86   display: flex;
87   flex-direction: row-reverse;
88   align-items: center;
89 }
90 .contenedor_form_volador .campo_bord_volador .buscador .fa {
91   margin-bottom: 1.676rem;
92   position: absolute;
93   padding: 0px 9px;
94 }
95 .contenedor_form_volador .campo_bord_volador .buscador input:focus {
96   outline: none;
97   border: 2px solid #1a73e8;
98 }
99 .contenedor_form_volador .campo_bord_volador .buscador input[type=search] {
100   padding-right: 25px;
101 }
102
```

[parte 04 – input volador]

Figura 33. Código de input formulario parte VI

```
103 .contenedor_form_volador .campo_bord_volador input:focus ~ label,
104 .contenedor_form_volador .campo_bord_volador input:valid ~ label,
105 .contenedor_form_volador .campo_bord_volador input:not([value=""]) ~ label {
106   font-size: 0.75rem;
107   top: -0.6rem;
108   height: 18px;
109   padding: 0px 5px !important;
110   text-align: center;
111   left: 30px;
112   color: #1a73e8;
113   background-color: white;
114 }
115
116 .contenedor_form_volador .campo_bord_volador, .contenedor_form_progre .campo_bord_progre {
117   position: relative;
118   padding: 0 20px;
119 }
120 .contenedor_form_volador .campo_bord_volador .contenedor_input_select, .contenedor_form_progre .campo_bord_progre .contenedor_input_select {
121   display: contents;
122   pointer-events: all !important;
123 }
124
125 .contenedor_form_volador input[type=submit]:hover {
126   background-color: #287ae6;
127   box-shadow: 0 1px 1px 0 rgba(66, 133, 244, 0.45), 0 1px 3px 1px rgba(66, 133, 244, 0.3);
128 }
129 .contenedor_form_volador .campo_bord_volador input:focus {
130   outline: none;
131   border: 2px solid #1a73e8;
132 }
133 .contenedor_form_volador .campo_bord_volador .contenedor_input_select svg {
134   right: 30px;
135   top: 21.5%;
136 }
137 .contenedor_form_volador .campo_bord_volador .contenedor_input_select select {
138   margin-bottom: 1.875rem;
139   padding: 0.559rem 0.7rem;
140   /*Este es de input field y desing*/
141 }
142
143
```

❖ Código del submódulo input diseñador

[parte 01 – input diseñador]

Figura 34. Código de input formulario parte VII

```
1 .contenedor_form_progre {
2   margin: auto;
3   max-width: 80%;
4   padding: 2vw;
5   box-sizing: border-box;
6   border: 1px solid #dadce0;
7   -webkit-border-radius: 8px;
8   border-radius: 8px;
9   background-color: white;
10 }
11 .contenedor_form_progre h2 {
12   padding-bottom: 30px;
13   color: #fff;
14   text-align: center;
15   color: #202124;
16   font-family: "Google Sans", "Noto Sans Myanmar UI", arial, sans-serif;
17   font-size: 24px;
18   font-weight: 400;
19 }
20 .contenedor_form_progre .campo_bord_progre input {
21   width: 100%;
22   background: transparent;
23   width: 100%;
24   background: transparent;
25   padding: 0.5rem 0.29rem;
26   border: none;
27   padding-bottom: 0.3rem;
28   padding-top: 0.6rem;
29   border-bottom: 1px solid #c6c6c6;
30   font-size: 18px;
31 }
```

[parte 02 – input diseñador]

Figura 35. Código de input formulario parte VIII

```
32 .contenedor_form_progre .campo_bord_progre input:focus {
33   outline: none;
34   border: 2px solid #1a73e8;
35   border: none !important;
36 }
37 .contenedor_form_progre .campo_bord_progre label {
38   position: absolute;
39   top: 0;
40   pointer-events: none;
41   transition: 0.5s;
42   padding: 0.5rem;
43   color: #c6c6c6;
44   font-size: 1rem;
45 }
46 .contenedor_form_progre .campo_bord_progre input[type=date] {
47   padding-bottom: 0.13rem;
48 }
49 .contenedor_form_progre .campo_bord_progre .contenedor_input_select select {
50   color: black;
51   font-weight: 500;
52   font-size: 1.155rem;
53   -webkit-appearance: none;
54   padding: 0.47rem 0.7rem;
55   width: 100%;
56   border-radius: unset;
57   background: #fff;
58   border-bottom: 1px solid #ccc !important;
59   cursor: pointer;
60   font-family: inherit;
61   transition: all 1s ease;
62   border: none;
63 }
64 .contenedor_form_progre .campo_bord_progre .contenedor_input_select select:hover {
65   outline: none;
66   box-shadow: 0 0 0px rgba(0, 119, 255, 0.2);
67   transition: all 0.3s;
68   color: #2196f3;
69 }
70 .contenedor_form_progre .campo_bord_progre .contenedor_input_select select:focus-visible {
71   outline: none;
72 }
73 .contenedor_form_progre .campo_bord_progre .contenedor_input_select svg {
74   right: 30px;
75   top: 21.5%;
76 }
77 .contenedor_form_progre .campo_bord_progre .contenedor_input_select select {
78   margin-bottom: 1.875rem;
79   padding: 0.559rem 0.7rem;
80 }
81 .contenedor_form_progre .campo_bord_progre input[type=date]:before {
82   color: #999999;
83 }
84 .contenedor_form_progre .campo_bord_progre input[type=date] {
85   color: #ffffff;
86 }
87 .contenedor_form_progre .campo_bord_progre input[type=date]:focus,
88 .contenedor_form_progre .campo_bord_progre input[type=date]:valid {
89   color: black;
90 }
91 .contenedor_form_progre .campo_bord_progre input[type=date]:focus:before,
92 .contenedor_form_progre .campo_bord_progre input[type=date]:valid:before {
93   content: "" !important;
94 }
95 .contenedor_form_progre .campo_bord_progre .buscador_i {
96   right: 1.5rem;
97   cursor: pointer;
98   position: absolute;
99   top: 0;
100  line-height: 3;
101 }
```

[parte 03 – input diseñador]

Figura 36. Código de input formulario parte IX

```
102 .contenedor_form_progre .campo_bord_progre input[type=search] {
103   padding-right: 25px;
104 }
105
106 .contenedor_form_progre .campo_bord_progre input:focus ~ label,
107 .contenedor_form_progre .campo_bord_progre input:valid ~ label,
108 .contenedor_form_progre .campo_bord_progre input:not([value=""]) ~ label {
109   font-size: 0.75rem;
110   top: -0.6rem;
111   height: 18px;
112   padding: 0px 5px !important;
113   text-align: center;
114   color: #2196f3;
115 }
116
117 .barra {
118   position: relative;
119   display: block;
120   width: 100%;
121   margin-bottom: 1.875rem;
122 }
```

[parte 04 – input diseñador]

Figura 37. Código de input formulario parte X

```
123
124 input:focus ~ .barra::before, textarea:focus ~ .barra::before {
125   width: 100%;
126 }
127
128 .barra::before {
129   content: "";
130   height: 2px;
131   width: 0;
132   bottom: 0;
133   position: absolute;
134   background: linear-gradient(to right, #6A82FB, #FC5C7D);
135   transition: 0.3s ease all;
136   left: 0%;
137 }
138
139 .min_progre {
140   /**** Input buscador ****/
141 }
142 .min_progre label {
143   padding: 0.2rem !important;
144 }
145 .min_progre input {
146   padding-top: 0.3rem !important;
147   padding-bottom: 0px !important;
148 }
149 .min_progre input[type=number] {
150   padding-bottom: 0px !important;
151 }
152 .min_progre input[type=date] {
153   padding-top: 1.3px !important;
154 }
155 .min_progre .buscador i {
156   line-height: 2.5 !important;
157 }
158
159 .max_progre label {
160   padding: 0.56rem !important;
161   top: 6px !important;
162 }
163 .max_progre input {
164   padding: 0.7rem 0.3rem !important;
165   padding-top: 0.9rem !important;
166 }
167 .max_progre input[type=number] {
168   padding-bottom: 0.37rem !important;
169 }
170 .max_progre input[type=password] {
171   padding-bottom: 0.37rem !important;
172 }
173 .max_progre input[type=date] {
174   padding-bottom: 0.2rem !important;
175 }
176 .max_progre input[type=email] {
177   padding-bottom: 0.47rem !important;
178   padding-top: 0.8rem !important;
179 }
180 .max_progre input[type=text] {
181   padding-bottom: 0.57rem !important;
182   padding-top: 0.7rem !important;
183 }
184 .max_progre .buscador label {
185   padding-top: 0.65rem !important;
186 }
187 .max_progre .buscador i {
188   line-height: 4.3 !important;
189 }
190 .max_progre .buscador input[type=search] {
191   padding-bottom: 0.44rem !important;
192 }
193
194 .max_progre input:focus ~ label,
195 .max_progre input:valid ~ label, .max_progre input:not([value=""]) ~ label {
196   top: -0.6rem !important;
197 }
198
```

Descripción de las propiedades más significativas:

A continuación, se procederá a detallar las propiedades más usadas y su aplicación. También se mostrará la compatibilidad web de cada propiedad para su mejor comprensión dejamos una leyenda en la cual se indica si tiene compatibilidad web, entre otros factores. Todo lo mencionado se observa en la figura 2.

Propiedad padding:10px 20px:

El padding es un relleno interno, en esta clase se usó la unidad de medida pixeles ya que se necesitaba un valor fijo además que no se distorsione en los distintos dispositivos. Dicho de paso se usó 10px para darle relleno hacia arriba y hacia abajo así como se le otorgo 20px para los rellenos de la izquierda y derecha, están de forma abreviada por eso solo se colocó 2 valores.

A continuación, veamos la compatibilidad web al usar esta propiedad en los principales navegadores como se puede apreciar en la figura 20.

Propiedad height: 5vh

Se usó la propiedad height para delimitar el alto de los inputs, se optó por usar la unidad de medida vh ya que este está condicionado al alto del dispositivo desde donde se conecten sea pc, tablets o celulares. Esta propiedad lo que hace es que siempre se vea de una dimensión proporcionada.

A continuación, veamos la compatibilidad web al usar esta propiedad en los principales navegadores como se observa en la figura 4.

Propiedad width:80% :

Se empleo width 80% para el ancho de este efecto ya que si bien esta la propiedad width: tiene diversas unidades de medidas como:

- EM
- VH
- PÍXELES
- REM
- PORCENTAJE

Se fue descartando una a una al momento de probarla, ya que la al usar la unidad EM este se distorsionaba por su propia naturaleza, básicamente esta propiedad usa el ancho de la pantalla del dispositivo donde estemos y las va ajustando así que fue descartado.

A continuación, veamos la compatibilidad web al usar esta propiedad en los principales navegadores se puede apreciar lo mencionado en la figura 3.

Propiedad height:auto

La función de esta propiedad es no tener definido un alto ya que se le asignara uno por default el navegador esto garantiza que sea moldeable el alto del contenedor así mientras más contenido halla dentro este seguirá creciendo y en caso contrario halla poco contenido dentro este tendrá una altura lo suficiente para cubrir los elementos que se encuentren dentro. Dicho de otra forma, su altura será automática.

A continuación, veamos la compatibilidad web al usar esta propiedad en los principales navegadores lo podemos observar en la figura 4.

Propiedad border:solid 4.3px #f7f7f9

En esta propiedad le damos un border sólido, esto quiere decir que el borde será visible, además de que será una línea definida sin interrupciones. Luego la otra característica es que tendrá una anchura de 4.3px, dicho de mejor forma será una línea fina casi invisible por ello tiene decimales no es un valor entero. Finalmente tenemos el color el cual es un color blanco humo.

A continuación, veamos la compatibilidad web al usar esta propiedad en los principales navegadores.

Figura 38. Compatibilidad web, propiedad border

Usage: % of all users: Global 97.05%

Current aligned Usage relative Date relative Filtered All

IE	Edge *	Firefox	Chrome	Safari	Opera	Safari on iOS *	Opera Mini *	Android Browser *	Opera Mobile *	Chrome for Android	Firefox for Android	UC Browser for Android	Samsung Internet	QQ Browser	Baidu Browser	KaiOS Browser
6-10	12-97	2-96	4-97	3.1-15.1	10-82	3.2-15.1		2.1-4.4.4	12-12.1				4-15.0			
11	98	97	98	15.3	83	15.3	all	98	64	98	96	12.12	16.0	10.4	7.12	2.5
		98-99	99-101	15.4-TP		15.4										

Propiedad margin: 0 auto

Esta propiedad como esta en el contenedor de input clásico lo que se busco era centrar todo el contenido en el centro para ello la mejor opción era usar los valores 0 arriba y debajo luego automático centrar a la derecha como a la izquierda. Ambos valores para hacerlo de forma abreviada solo se dispusieron a poner 0 y auto.

A continuación, veamos la compatibilidad web al usar esta propiedad en los principales navegadores.

Figura 39. Compatibilidad web, propiedad margin

IE	Edge	Firefox	Chrome	Safari	Opera	Safari on iOS	Opera Mini	Android Browser	Opera Mobile	Chrome for Android	Firefox for Android	UC Browser for Android	Samsung Internet	QQ Browser	Baidu Browser	KaiOS Browser
6-10	12-97	2-96	4-97	3.1-15.1	10-82	3.2-15.1		2.1-4.4.4	12-12.1				4-15.0			
11	98	97	98	15.3	83	15.3	all	98	64	98	96	12.12	16.0	10.4	7.12	2.5
		98-99	99-101	15.4-TP		15.4										

Propiedad border-radius:3px

Esta propiedad lo que hace es darle milésimas de bordes redondeados, esta propiedad nos brinda la ventaja de no depender de la resolución de pantalla del usuario siempre estará fija.

A continuación, veamos la compatibilidad web al usar esta propiedad en los principales navegadores.

Figura 40. Compatibilidad web, propiedad border-radius

IE	Edge	Firefox	Chrome	Safari	Opera	Safari on iOS	Opera Mini	Android Browser	Opera Mobile	Chrome for Android	Firefox for Android	UC Browser for Android	Samsung Internet	QQ Browser	Baidu Browser	KaiOS Browser
		2		3.1-4												
		3-3.6		5												
6-8		4-4.9	4	5.1-6.1	10.1	3.2		2.1								
9-10	12-97	50-96	5-97	7-15.1	11.5-82	4-15.1		2.2-4.4.4	12-12.1				4-15.0			
11	98	97	98	15.3	83	15.3	all	98	64	98	96	12.12	16.0	10.4	7.12	2.5
		98-99	99-101	15.4-TP		15.4										

Propiedad background-color:white

Esta propiedad se usa para darle un color de fondo para esta ocasión se decidió que sería blanco ya que la tonalidad blanco humo fue otorgado a los bordes.

A continuación, veamos la compatibilidad web al usar esta propiedad la podemos visualizar en la figura 5.

Propiedad color: #363636

La propiedad de color se definió en un label con una tonalidad que es semejante al negro, este color se eligió ya que es uno de los colores primarios. Además, la propiedad color solo es aplicable a textos e iconos.

A continuación, veamos la compatibilidad web al usar esta propiedad se puede apreciar en la figura 21.

Propiedad font-weight:700

La presente propiedad su función es agregar una tonalidad más oscura a la clase del texto donde se aplica, es similar a resaltar texto en negrita para su tonalidad es un poco menor.

A continuación, veamos la compatibilidad web al usar esta propiedad en los principales navegadores.

Figura 41. Compatibilidad web, propiedad font-weight

The screenshot shows a browser compatibility tool interface for the CSS property 'font-weight'. It displays a table of browser versions and their support for the property. The 'Usage' is listed as 97.41% globally. The table includes columns for IE, Edge, Firefox, Chrome, Safari, Opera, Safari on iOS, Opera Mini, Android Browser, Opera Mobile, Chrome for Android, Firefox for Android, UC Browser for Android, Samsung Internet, QQ Browser, Baidu Browser, and KaiOS Browser. Support is generally high, with most versions showing 96-100% compatibility.

IE	Edge	Firefox	Chrome	Safari	Opera	Safari on iOS	Opera Mini	Android Browser	Opera Mobile	Chrome for Android	Firefox for Android	UC Browser for Android	Samsung Internet	QQ Browser	Baidu Browser	KaiOS Browser
6-10	12-97	2-96	4-97	3.1-15.1	10-82	3.2-15.1		2.1-4.4.4	12-12.1				4-15.0			
11	98	97	98	15.3	83	15.3	all	98	64	98	96	12.12	16.0	10.4	7.12	2.5
		98-99	99-101	15.4-TP		15.4										

Propiedad font-size:0.91rem

La propiedad font-size es el tamaño de la letra en este caso se optó por usar decimales ya que es más preciso el tamaño que se necesitada además de que viene acompañado de la unidad de medida rem, lo que lo hace más receptivo a cambios de pantallas.

A continuación, vemos la compatibilidad web al usar esta propiedad en los principales navegadores.

Figura 42. Compatibilidad web, propiedad font-size

CSS property: font-size

Usage: Global 97.41%

Current aligned Usage relative Date relative Filtered All

IE	Edge *	Firefox	Chrome	Safari	Opera	Safari on iOS *	Opera Mini *	Android Browser *	Opera Mobile *	Chrome for Android	Firefox for Android	UC Browser for Android	Samsung Internet	QQ Browser	Baidu Browser	KaiOS Browser
6-10	12-97	2-96	4-97	3.1-15.1	10-82	3.2-15.1		2.1-4.4.4	12-12.1				4-15.0			
11	98	97	98	15.3	83	15.3	all	98	64	98	96	12.12	16.0	10.4	7.12	2.5
		98-99	99-101	15.4-TP		15.4										

Propiedad line-height:200px

Esta propiedad es para darle relleno solo de altura en donde se aplique similar al padding pero con uso más orientado a texto, aunque se puede aplicar también a etiquetas html.

A continuación, vemos la compatibilidad web al usar esta propiedad en los principales navegadores.

Figura 43. Compatibilidad web, propiedad line-height

CSS property: line-height

Usage: Global 97.41%

Current aligned Usage relative Date relative Filtered All

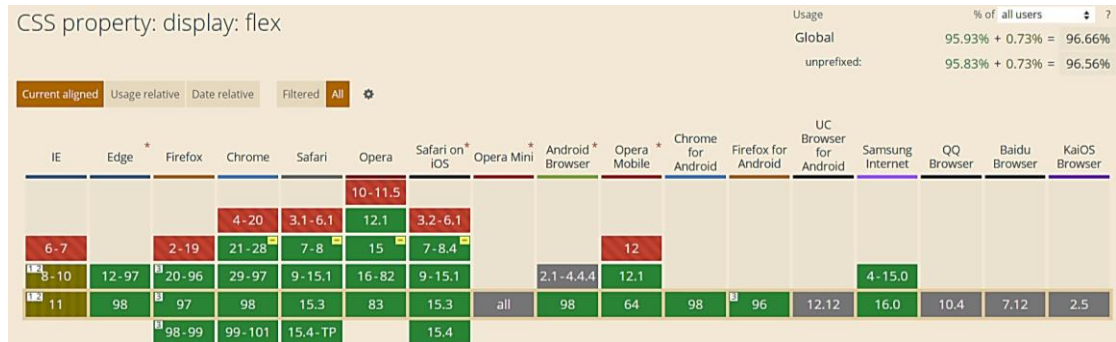
IE	Edge *	Firefox	Chrome	Safari	Opera	Safari on iOS *	Opera Mini *	Android Browser *	Opera Mobile *	Chrome for Android	Firefox for Android	UC Browser for Android	Samsung Internet	QQ Browser	Baidu Browser	KaiOS Browser
6-10	12-97	2-96	4-97	3.1-15.1	10-82	3.2-15.1		2.1-4.4.4	12-12.1				4-15.0			
11	98	97	98	15.3	83	15.3	all	98	64	98	96	12.12	16.0	10.4	7.12	2.5
		98-99	99-101	15.4-TP		15.4										

Propiedad display: flex

Con esta propiedad se puede aplicar la alineación de los elementos en una sola dirección sea horizontal o vertical, estos valores se puede conjugar con las distintos valores que tiene, se optó por esta propiedad ya que es una forma óptima de maquetar y de que los elementos no se desborden de sus contenedores como solía suceder al maquetar usando position o tables en la antigüedad además flexbox viene para usarlo con la versión 3 de css llamada css3 haciendo más robusta la maquetación de elementos.

A continuación, vemos la compatibilidad web al usar esta propiedad en los principales navegadores.

Figura 44. Compatibilidad web, propiedad `display:flex`



Propiedad `align-items: center`

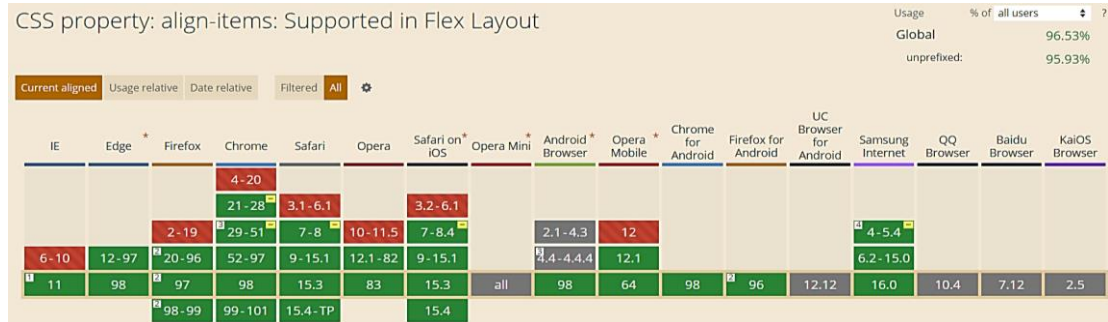
Para usar esta propiedad es un requisito indispensable tener habilitado `display:flex`, luego de eso se puede conjugar `align-items` con los siguientes valores:

- CENTER
- END
- FLEX-END
- FLEX-START
- INHERIT
- INITIAL
- NORMAL
- REVERT
- SELF-END
- START
- STRETCH
- UNSET

En este caso se usó con el valor `center` ya que lo que ara es centrar en el centro de gravedad a los elementos, pero en dirección vertical.

A continuación, vemos la compatibilidad web al usar esta propiedad en los principales navegadores.

Figura 45 Compatibilidad web, propiedad align-items



Propiedad justify-content:center

La propiedad justify-content tiene como prerrequisitos haber declarado display:flex en el mismo elemento o en un elemento padre. Tiene las siguientes conjugaciones:

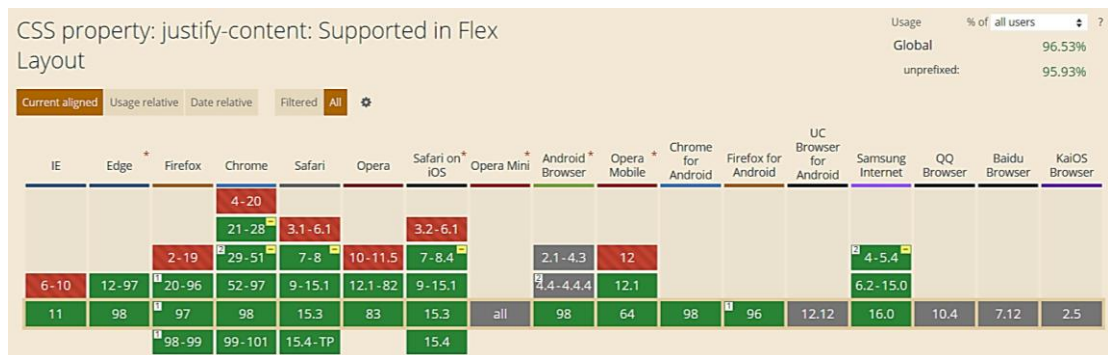
- CENTER
- END
- FLEX-END
- FLEX-START
- INHERIT
- INITIAL
- LEFT
- NORMAL
- REVERT
- RIGHT
- SPACE-AROUND
- SPACE-BETWEEN
- SPACE-EVENLY
- START

- STRETCH
- UNSET

Se optó por elegir el valor center ya que este centra en horizontal cualquier elemento.

A continuación, veamos la compatibilidad web al usar esta propiedad en los principales navegadores.

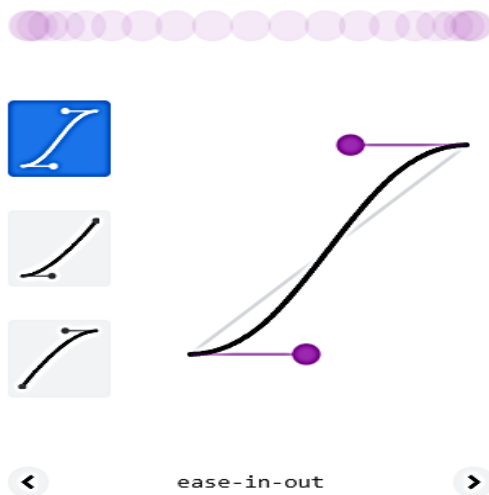
Figura 46 Compatibilidad web, propiedad justify-content



Propiedad transition: border-color 0.3s ease-in-out

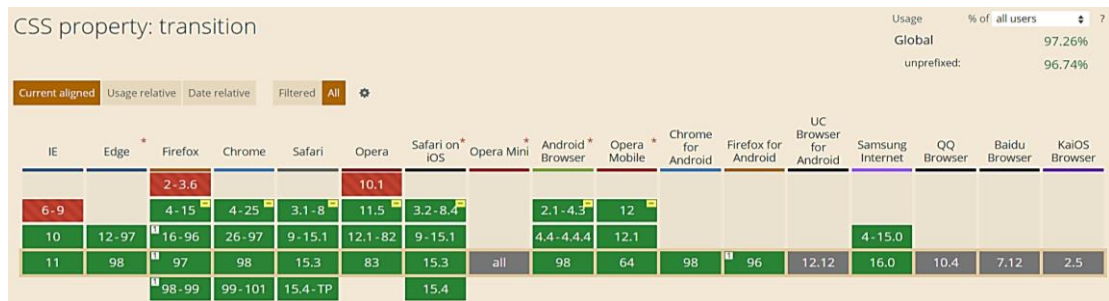
Esta propiedad en particular fue la más usada en los submódulos como en el módulo, su uso es mayormente en animaciones a las cuales se le retarda el tiempo para luego aplicar algún efecto según sea la combinación que se use ya que transition tiene varias combinaciones una de ellas es ease-in-out el cual es una función cubica.

Figura 47 Gráfico del valor ease-in-out en la propiedad transition



A continuación, vemos la compatibilidad web al usar esta propiedad en los principales navegadores.

Figura 48. Compatibilidad web, propiedad transition

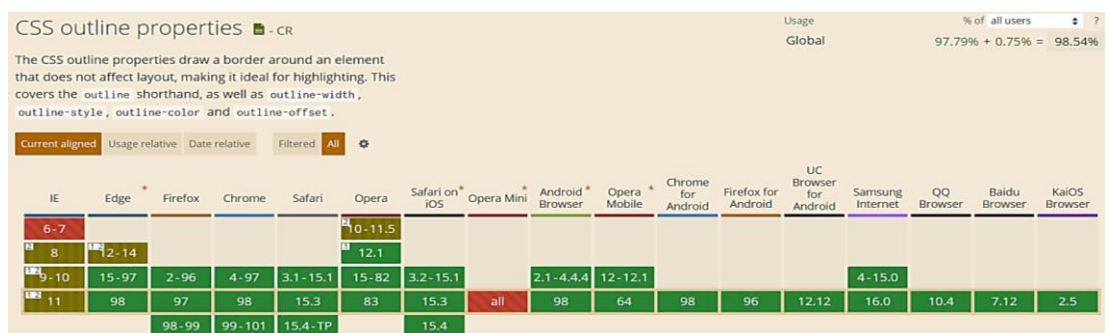


Propiedad outline:0

Esta propiedad fue usado para quitar el borde de los inputs que le asigna por default el navegador a las cajas de texto u otros tipos de inputs, si bien también se puede cancelar usando outline:none se optó por usar 0 ya que así solo se quita literalmente momentáneamente, ya que si se declara líneas de código más abajo otro diseño este las tomaría pero al poner outline:none serie mala práctica.

A continuación, vemos la compatibilidad web al usar esta propiedad en los principales navegadores.

Figura 49. Compatibilidad web, propiedad outline



Propiedad box-shadow: 0 0 0 3px rgba(13,110,253,0.04)

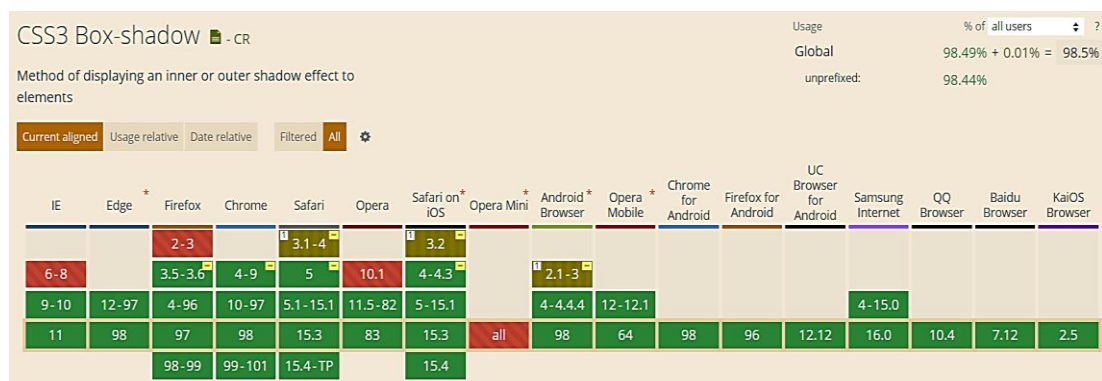
Esta propiedad en particular lo que hace es dar bordes con sombra a los componentes donde se apliquen en este caso se usó el posicionamiento de los componentes sin abreviatura, para hacer ello se usó los 4 puntos cardinales para se empieza en el siguiente orden:

- Arriba
- Derecha
- Abajo
- Izquierda

Luego se optó por usar la unidad de medida pixeles ya que serán fijos los valores y no dependerá del ancho de los dispositivos por el cual se conecten los usuarios. El color de las sombras el cual se eligió es un color humo, expresado en el formato rgba es: rgba(13,110,253,0.04).

A continuación, veamos la compatibilidad web al usar esta propiedad en los principales navegadores.

Figura 50. Compatibilidad web, propiedad box-shadow



Principales dificultades:

Para este efecto en concreto la principal dificultad fue organizar de distintas formas posibles, así como ver las posibles conjugaciones que pueden hacer con el efecto para que el usuario al final no tenga problemas al implementarlo. Esas dificultades surgieron a raíz de las siguientes preguntas:

- ¿Qué pasa si el usuario quiere usar uno de los inputs para una pantalla grande led de oficina, se vería bien?

En este caso la solución más óptima fue usar una clase en cada módulo y submódulo. Esta clase está construida con unidades de medida vh y rem.

- ¿Qué pasa si el usuario quiere usar uno de los inputs para una pantalla mediana que traen las laptops o tabletas, se vería bien?

Se previó que lo más común y más usado por los usuarios sería usar este tipo de pantalla por ello la clase por default cubre este tipo de pantallas.

- ¿Qué pasa si el usuario quiere usar uno de los inputs para una pantalla pequeña que traen los smartphones de gama baja?

Para este tipo de pantallas una solución óptima es tener una clase extra para estos tipos de pantallas, estos serán usados con las unidades de medida vh y rem. Otra solución viable era usar puntos de corte con @media query pero no hubiera sido óptima.

¿Qué usamos para hacer las pruebas?

Se usó el navegador Google Chrome y opera mini, así como la siguiente página web que es un simulador online:

<http://www.responsinator.com>

Este con motivos de asegurarnos que funcione en dispositivos móviles y tabletas. Ya que en base a lo trabajado no es del todo fiable usar el navegador para verificar la correcta maquetación de los elementos, así como tampoco es fiable depender de un software web, por ello se decidió diversificar en cuanto a las herramientas para probar las clases y efectos trabajados en el framework XRL8.

También usamos como editor de código visual studio code y su extensión live server para visualizar en vivo los efectos realizados en el css en tiempo real, también usamos el autoguardado que trae dicho editor de código para no estar guardando el proyecto a cada rato. Así conseguimos un mejor desempeño y evitamos el tedioso guardar cambios a cada rato, también evitamos el que la memoria cache se llene mostrándonos los mismos resultados.

FASE 3.- Hacerlo responsive

Con este efecto en particular fue sencillo ya que la propiedad width estaba en % además de que no se necesitó una maquetación extra como veremos más adelante en el presente informe. Otro factor fue decisivo el uso de clases externas para llegar a cualquier dispositivo del cual se pueden conectar los usuarios sin importar el dispositivo esas clases fueron las siguientes:

- Modulo:
 - ❖ Input clásico: max_clasi , min_clasi
- Submódulo:
 - ❖ input volador: max_desplazante, min_desplazante
 - ❖ input diseñador: max_progre, min_progre

En caso se hubiera dado la oportunidad de maquetar más el efecto para que tome forma y se moldee, lo que hubiéramos hecho en ese caso sería:

- Hubiéramos trabajado con la propiedad @media_query en las resoluciones de pantalla para celulares y tabletas. Siempre empezando a maquetar por la parte de celulares y luego tabletas.

FASE 4.- Testing asegurarse que no tenga conflicto con otros efectos.

En esta parte del desarrollo de la clase se dispuso a probar poniéndolo con las demás clases juntándolas. Para ello lo que se hizo fue colocar dentro de un archivo index.html todas las clases trabajadas y ver como el efecto input formulario respondió. En algunos casos había conflicto con los demás efectos para corregir esto se procedió a implementar en el desarrollo dos cosas básicas la primera fue:

- Para detectar que clase era la que causaba el conflicto y ver en que archivo scss estaba. Se procedía a inspeccionar la página para detectar donde estaba el problema. En muchas ocasiones no se daba con el archivo causante del error entonces en ese caso aplicábamos la frase “Divide y vencerás”. Lo que se procedió hacer fue comentar de uno en

uno cada efecto en el archivo principal que los llamaba esto reducía las líneas de código de css, además como teníamos en tiempo real los cambios fue mucho más rápido detectar el problema. Ya que no se tenía que recargar a cada rato además otro punto a favor era que se auto compilaba los cambios de sass automáticamente gracias a nodeJS el cual mediante la consola de comandos de Windows cuando detectaba un cambio en las líneas de código se auto compilaba y generando o reemplazando el archivo css en el cual hacíamos las pruebas, esto aumento la productividad para crear los efectos ya que al comentar de uno en uno cada archivo sass llegaba un momento donde desaparecía el error entonces revisábamos y nos dimos cuenta de que era uno o en algunos casos hasta tres efectos que hacían conflicto con la clase escrita.

- Se procedió a escribir mediante comentarios en un archivo llamado `_error_logs.scss` la clase que hace conflicto con la clase escrita, además de describir una nota breve en caso no se disponía de tiempo para solucionarlo luego, esto permitió tener un reporte más preciso de los incidentes que íbamos detectando.
- Uno de los problemas más cotidianos fue que al probar los inputs no era responsive así que la solución que implementamos fue crear clases extras para cada ocasión o resolución de pantalla, las cuales en la versión alfa aun no son mezclables.
- Se tuvo también conflictos al tratar de probar las dimensiones que trae por default los distintos tipos de inputs para rediseñarlos a nuestras propias dimensiones. Se tuvo que medir de manera manual con una regla en distintos tipos de pantallas, así como el uso de simuladores online.
- También tuvimos problemas en el submódulo input diseñador ya que la barra de carga no sabíamos cómo posicionarlo dentro del input al mismo nivel ya que solía mezclarse o hacer conflicto con la clase llamada maquetado. Al final se solucionó dándole posición absoluta y creando una etiqueta span.

FASE 5.- Aplicar la metodología OOCSS, llevándolo a SASS

- **Código css.** - Esta parte ya estaría completa por lo descrito anteriormente.

- **Código repetido.** - Identificación de código repetido

En este pequeño apartado luego de escribir y debugear el error se procedió a identificar el código css repetido para llevarlo a un archivo SCSS en sass, para su mayor optimización y aplicación de la metodología. Se procederá a identificar las clases las cuales comparten valores:

❖ **Input volador:**

Figura 51. Código repetido del componente input formulario parte I

```
1  .contenedor_form_volador {
2    margin: auto;
3    max-width: 80%;
4    padding: 2vw;
5    box-sizing: border-box;
6    border: 1px solid #dadce0;
7    -webkit-border-radius: 8px;
8    border-radius: 8px;
9    background-color: white;
10 }
```

Figura 52. Código repetido del componente input formulario parte II

```
1  .contenedor_form_volador h2 {
2    padding-bottom: 30px;
3    color: #fff;
4    text-align: center;
5    color: #202124;
6    font-family: "Google Sans", "Noto Sans Myanmar UI", arial, sans-serif;
7    font-size: 24px;
8    font-weight: 400;
9  }
```

Figura 53. Código repetido del componente input formulario parte III

```
1  .contenedor_form_volador .campo_bord_volador input {
2    width: 100%;
3    background: transparent;
4    margin-bottom: 1.875rem;
5    border-radius: 4px;
6    font-size: larger;
7    border: 1px solid #ccc;
8    padding: 0.5rem 0.7rem;
9  }
```

Figura 54. Código repetido del componente input formulario parte IV

```
1  .contenedor_form_volador .campo_bord_volador label {
2    position: absolute;
3    top: 0;
4    pointer-events: none;
5    transition: 0.5s;
6    padding: 0.5rem;
7    color: grey;
8    left: 30px;
9    font-size: 0.9rem;
10 }
```

Figura 55. Código repetido del componente input formulario parte V

```
1  .contenedor_form_volador .campo_bord_volador input:focus ~ label,
2  .contenedor_form_volador .campo_bord_volador input:valid ~ label,
3  .contenedor_form_volador .campo_bord_volador input:not([value=""]) ~ label {
4    font-size: 0.75rem;
5    top: -0.6rem;
6    height: 18px;
7    padding: 0px 5px !important;
8    text-align: center;
9    left: 30px;
10   color: #1a73e8;
11   background-color: white;
12 }
```

❖ Input diseñador:

Figura 56. Código repetido del componente input formulario parte VI

```
1  .contenedor_form_progre {
2    margin: auto;
3    max-width: 80%;
4    padding: 2vw;
5    box-sizing: border-box;
6    border: 1px solid #dadce0;
7    -webkit-border-radius: 8px;
8    border-radius: 8px;
9    background-color: white;
10 }
```

Figura 57. Código repetido del componente input formulario parte VII

```
1  .contenedor_form_progre h2 {
2    padding-bottom: 30px;
3    color: #fff;
4    text-align: center;
5    color: #202124;
6    font-family: "Google Sans", "Noto Sans Myanmar UI", arial, sans-serif;
7    font-size: 24px;
8    font-weight: 400;
9  }
```

Figura 58. Código repetido del componente input formulario parte VIII

```
1  .contenedor_form_progre .campo_bord_progre input {
2  width: 100%;
3  background: transparent;
4  width: 100%;
5  background: transparent;
6  padding: 0.5rem 0.29rem;
7  border: none;
8  padding-bottom: 0.3rem;
9  padding-top: 0.6rem;
10 border-bottom: 1px solid #c6c6c6;
11 font-size: 18px;
12 }
```

Figura 59. Código repetido del componente input formulario parte IX

```
1  .contenedor_form_progre .campo_bord_progre label {
2  position: absolute;
3  top: 0;
4  pointer-events: none;
5  transition: 0.5s;
6  padding: 0.5rem;
7  color: #c6c6c6;
8  font-size: 1rem;
9  }
10
```

Figura 60. Código repetido del componente input formulario parte X

```
1  .contenedor_form_progre .campo_bord_progre input:focus ~ label,
2  .contenedor_form_progre .campo_bord_progre input:valid ~ label,
3  .contenedor_form_progre .campo_bord_progre input:not([value=""]) ~ label {
4  font-size: 0.75rem;
5  top: -0.6rem;
6  height: 18px;
7  padding: 0px 5px !important;
8  text-align: center;
9  color: #2196F3;
10 }
```

- **Crear un objeto.** – Se describió la lógica de la función para luego crear un mixing.

Una vez identificado el código css repetido se procede hacer una función la cual se ejecute al llamado del código, mediante esta técnica nuestro código será más puro y limpio. Cabe recalcar que el submódulo input volador, así como el submódulo input diseñador comparten características similares.

Paralelamente en esta parte se determina la lógica de las funciones, el pase de parámetros, así como la condicional que llevarán dentro, por otra parte, y no menos importante se define el nombre con el cual se llamará a las funciones.

```
function general_Field_And_Desing(){
  margin:auto;
  max-width:80%;
  padding:2vw;
  box-sizing:border-box;
  border:1px solid #dadce0;
  -webkit-border-radius:8px;
  border-radius:8px
}
function h2_Field_And_Desing(){
  padding-bottom:30px;
  color:#fff;
  text-align:center;
  color:#202124;
  font-family:'Google Sans','Noto Sans Myanmar UI',arial,sans-serif;
  font-size:24px;
  font-weight:400;
}
function input_Fiel_And_Desing($condicion){
  width:100%;
  background:transparent;
  @if $condicion == 'borde_volador'{
  margin-bottom:1.875rem;
  border-radius:4px;
  font-size:larger;
  border:1px solid #ccc;
  padding:0.5rem 0.7rem;
  }
  @else{
  width:100%;
  background:transparent;
```

```

padding:0.5rem 0.29rem;
border:none;
padding-bottom:0.3rem;
padding-top:1px solid #c6c6c6;
font-size:18px;
}
}
function label_Fiel_And_Desing($condicion){
position:absolute;
top:0;
pointer-events:none;
transition:0.5s;
padding:0.5rem;
@if $condicion == 'borde_volador'{
color:grey
left: 30px;
font-size:0.9rem;
}@else{
color:#c6c6c6;
font-size:1rem;
}}
function condicional_label_Fiel_And_Desing($condicion){
font-size:0.75rem;
top:-0.6rem;
height:18px;
padding:0px 5px !important;
text-align:center;

@if $condicion == 'borde_volador'{
left:30px;
color:#1a73e8;
background-color:white;
}@else{
color:#2196F3;
}
}
}

```

Luego de tener la lógica se procede a llevarlo a SASS para ello tenemos dos opciones, la primera es usar @function ya que como su nombre ase referencia es para crear una función la otra opción es usar @mixin, pero ambas tienen diferencias. La primera retorna un valor, mientras @mixin solo llama a los valores en este caso no necesitamos que retorne un valor por ello usaremos @mixin. Entonces quedaría de la siguiente forma:

Figura 61. Creación del objeto general_fiel_and_desing en función mixin

```
1 @mixin general_Fiel_And_Desing{
2   margin: auto;
3   max-width:80%;
4   padding:$padding_contenedor_field_desing;
5   box-sizing: border-box;
6   border:$bordeGrosor_contenedor_field_desing solid $borde_contenedor_field_desing;
7   -webkit-border-radius: 8px;
8   border-radius:$bordeRadio_formulario_field_desing;
9   background-color:$fondo_contenedor_field_desing;
10 }
```

Figura 62. Creación del objeto h2_fiel_and_desing en función mixin

```
1 @mixin h2_Fiel_And_Desing{
2   padding-bottom: 30px;
3   color: #fff;
4   text-align: center;
5   color: #202124;
6   font-family: 'Google Sans','Noto Sans Myanmar UI',arial,sans-serif;
7   font-size: 24px;
8   font-weight: 400;
9 }
```

Figura 63. Creación del objeto input_fiel_and_desing en función mixin

```
1 @mixin input_Fiel_And_Desing($condicion){
2   width:100%;
3   background:$fondo_input_field;
4
5   @if $condicion == 'borde_volador'{
6     margin-bottom: 1.875rem;
7     border-radius:$redondeo_input_field;
8     font-size:larger;
9     border: 1px solid #ccc;
10    padding:0.5rem 0.7rem;
11  }
12  @else{
13    width: 100%;
14    background:$fondo_input_desing;
15    padding: 0.5rem 0.29rem;
16    border: none;
17    padding-bottom: 0.3rem;
18    padding-top: 0.6rem;
19    border-bottom: 1px solid $color_resistencia_input_desing;
20    font-size: 18px;
21  }
22 }
```

Figura 64. Creación del objeto `label_fiel_and_desing` en función `mixin`

```
1 @mixin label_Fiel_And_Desing($condicion){
2   position: absolute;
3   top: 0;
4   pointer-events: none; //Hace que no funcione input select
5   transition: 0.5s;
6   padding:0.5rem;
7
8   @if $condicion == 'borde_volador'{
9     color: $color_placeholder_field;
10    left: 30px;
11    font-size:0.9rem;
12  }@else{
13    color:$color_placeholder_desing;
14    font-size: 1rem;//Cuando es esto MATERIAL DESING
15  }
16 }
```

Figura 65. Creación del objeto `condicional_label_fiel_and_desing` en función `mixin`

```
1 @mixin condicional_label_Fiel_And_Desing($condicion){
2   font-size: 0.75rem;
3   top: -0.6rem;
4   height: 18px;
5   padding: 0px 5px!important;
6   text-align: center;
7   @if $condicion == 'borde_volador'{
8     left: 30px;
9     color:$color_placeholder_active_field;
10    background-color:$fondo_placeholder_active_field;
11  }@else{
12    color:$color_placeholder_active_desing;
13  }
14 }
```

- **OOCSS**. – Luego de tener la función o la lógica para reciclar el código se procede a integrar la metodología OOCSS para ello debemos llamar al `mixin` para ello solo necesitamos agregar `@include` “nombre del `mixin`”. Luego de integrarlo ya estará aplicada correctamente la metodología OOCSS

❖ Aplicando OOCSS en Input volador

Figura 66. Metodología OOCSS en el componente `input` formulario parte I

```
1 .contenedor_form_volador{
2   @include general_Fiel_And_Desing;
3
4   h2{
5     @include h2_Fiel_And_Desing;
6   }
7
8   input[type="submit"] {
9     @include btn_field;
10  }
11
12  .campo_bord_volador{
13    input {
14      @include input_Fiel_And_Desing('borde_volador');
15    }
16
17    label {
18      @include label_Fiel_And_Desing('borde_volador');
19    }
20  }
21 }
22
23 .contenedor_form_volador .campo_bord_volador input:focus ~ label,
24 .contenedor_form_volador .campo_bord_volador input:valid ~ label,
25 .contenedor_form_volador .campo_bord_volador input:not([value=""]) ~ label {
26   @include condicional_label_Fiel_And_Desing('borde_volador');
27 }
```


❖ Aplicando OOCSS en input diseñador

Figura 67. Metodología OOCSS en el componente input formulario parte II

```
1  .contenedor_form_progre{
2      @include general_Fiel_And_Desing;
3      h2{
4          @include h2_Fiel_And_Desing;
5      }
6      .campo_bord_progre{
7          input {
8              @include input_Fiel_And_Desing('borde_progre');
9              &:focus{
10                 outline: none;
11                 border: 2px solid #1a73e8;
12                 border:none!important;
13             }
14         }
15         label{
16             @include label_Fiel_And_Desing('borde_progre');
17         }
18     }
19 }
20 }
21
22 .contenedor_form_progre .campo_bord_progre input:focus ~ label,
23 .contenedor_form_progre .campo_bord_progre input:valid ~ label,
24 .contenedor_form_progre .campo_bord_progre input:not([value=""]) ~ label {
25     @include condicional_label_Fiel_And_Desing('borde_progre');
26 }
27 }
```

Volviendo al tema de las variables estas variables se encuentran en un archivo sass denominado variables valga la redundancia donde se encuentran las siguientes variables:

❖ Variables input clásico

Figura 68. Metodología OOCSS en el componente input formulario parte III

```
1  $input-normal-margenes:10px 20px;
2  $input-normal-altura-default:5vh;
3  $input-normal-ancho-contenedor:80%;
4  $input-normal-altura-contenedor:auto;
5  $input-normal-tipo-borde:solid;
6  $input-normal-grosor-borde:4.3px;
7  $input-normal-color-borde:#f7f7f9;
8  $input-normal-borde-radius:3px;
9  $input-normal-fondo:white;
10 $input-normal-label-color:#363636;
11 $input-normal-label-negrita:700;
12 $input-normal-label-tamano:0.91rem;
13 $campo-form-input-text-centrado:200px;
14 $input-normal-min-clasi:4vh;
15 $input-normal-max-clasi:5.9vh;
```

❖ Variables input volador y diseñador(compartidas)

Figura 69. Metodología OOCSS en el componente input formulario parte IV

```
1 $borde_contenedor_field_desing:#dadce0;  
2 $bordeGrosor_contenedor_field_desing:1px;  
3 $bordeRadio_formulario_field_desing:8px;  
4 $fondo_contenedor_field_desing:white;  
5 $padding_contenedor_field_desing:2vw;
```

❖ Variables input volador

Figura 70. Metodología OOCSS en el componente input formulario parte V

```
1 $fondo_input_field:transparent;  
2 $redondeo_input_field:4px;  
3 $color_placeholder_field:grey;  
4 $color_placeholder_active_field:#1a73e8;  
5 $fondo_placeholder_active_field:white;
```

❖ Variables input diseñador

Figura 71. Metodología OOCSS en el componente input formulario parte VI

```
1 $fondo_input_desing:transparent;  
2 $color_resistencia_input_desing:#c6c6c6;  
3 $color_placeholder_desing:#c6c6c6;  
4 $color_placeholder_active_desing:#2196F3;
```

Como se ha visto SASS es un preprocesador de css por ello se ejecuta antes del css tradicional esto permite añadir variables agregar condicionales, etc. Con la aplicación de la metodología como se observa se reutiliza el código css, además de que se hace escalable el proyecto. Ya que, si en un futuro queremos cambiar o añadir más propiedades css, solo cambiaríamos las variables o añadiríamos más propiedades en el mixing.

FASE 6.- Documentar en la web site www.phooldx.com

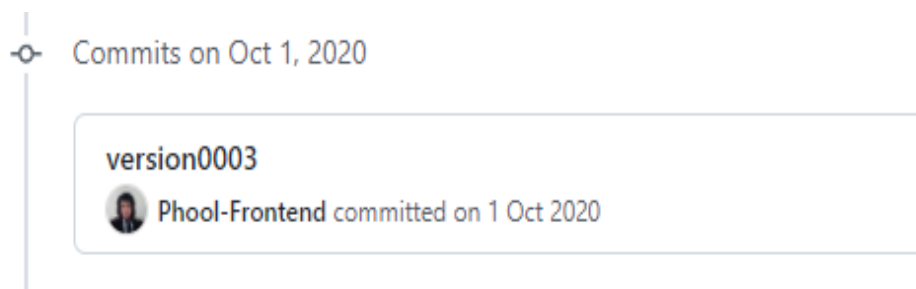
Finalmente, si el efecto paso por todos los de más filtros se procede a su documentación respectiva, para documentar todos los efectos se creó una página web de documentación usando clases y elementos del mismo framework. A continuación, se describirá como se procede a la documentación de la misma. Siguiendo estos pasos:

- Se copia las etiquetas de HTML5 junto con la clase a un txt temporal para no perder la sintaxis
- Se guarda los cambios en el compilador de SASS
- Como es la parte final se procedió hacer copias de seguridad para no perder el avance, en archivos RAR. Estos archivos estaban por versiones enumerados, además se agregaba un pequeño texto describiendo el avance. Este guardado se hacía de manera local en un USB.
- Para una mayor seguridad del avance del código se procedió luego a copiar la versión del archivo RAR junto a la descripción del mismo.

Guardado de código en GitHub. - Luego de tener la copia del proyecto a nivel físico en un USB, se procedió a guardarlo en un repositorio de GitHub para lo cual se usó Git, para ello se procedió de la siguiente forma:

- Se guardaba todo el proyecto con el siguiente comando que se aprecia mejor en la figura 12.
- Luego se verificaba los cambios guardados con el siguiente comando que se puede observar en la figura 13.
- Después se hacía el commit, es decir se le asignaba un nombre a los cambios realizados con el siguiente comando que se aprecia en la figura 14.
- Luego se procede a empaquetar los datos para que se suban al repositorio de GitHub con el siguiente código que se visualiza en la figura 15.
- Finalmente, luego de haber enviados el proyecto empaquetado hacia GitHub como se puede apreciar en la imagen, este queda guardado en dicho repositorio.

Figura 72. Visualización en github del componente input formulario



Se guarda en una escala de tiempo, con todos los cambios hechos desde la creación del proyecto, esto ayuda a tener un mejor control de versiones. Asimismo, también ayuda a que más personas se sumen al proyecto y puedan clonar el proyecto y trabajar por separado el proyecto, entre otras muchas ventajas.

- Así mismo se manejó un archivo llamado bitácora.txt donde aquí se registró todos los incidentes ocurridos, así como los cambios hechos, se registraba también las metas cumplidas, también las tareas nuevas para desarrollar al día siguiente. Al finalizar con un efecto por completo se quitaba de la lista de prioridades así dábamos pase al siguiente efecto o problema a resolver. Esto servía para tener un mejor orden y saber ejecutar prioridades.
- Luego de tener guardado el avance y de registrar el avanzase, se procedió a copiar del archivo Documentación Etiquetas.txt el fragmento de código que se copió anteriormente para luego pegarlo en www.nosetup.org esta página web se encargaba de convertir html en texto plano para poder visualizar html
- Antes de finalizar se procedió a crear un archivo llamado modulo_input_formulario.html donde se dividía en tres partes:
 - En la primera se describía un poco al efecto que era y que hacía.
 - En la segunda parte se procedía a copiar el código ya formateado obtenido de www.nosetup.com para luego pegarlo.
 - En la tercera parte lo que se hacía era mostrar un apartado llamado estructura donde se describía como se llamaban las variables para que sea modificado en SASS.

Figura 73. Código html5 final del componente input formulario parte I

Ejemplo en código - Input clásico

```
<div class="form_base">
<h1>Buenas tardes</h1>
<div class="maquetado">

<div class="campo_form col-largo-2 col-medi-2 col-peque-2 col-chiki-6">
<label>Nombres:</label>
<input class="min_clasi" type="text" name="" id="">
</div>

<div class="campo_form col-largo-2 col-medi-2 col-peque-2 col-chiki-6">
<label>Apellidos:</label>
<input type="text" name="" id="">
</div>

<div class="campo_form col-largo-2 col-medi-2 col-peque-2 col-chiki-6">
<label>Email or phone:</label>
<input class="max_clasi" type="text" name="" id="">
</div>

</div>
<div class="campo_form col-largo-2 col-medi-2 col-peque-2 col-chiki-6"><a href="#">Forgot Password?</a></div>
</div>
```

Figura 74. Código html5 final del componente input formulario parte II

Ejemplo en código - Input Volador

```
<div class="contenedor_form_volador">
<h2>Login</h2>
<div class="maquetado">
<div class="max_desplazante campo_bord_volador col-largo-2 col-medi-2 col-peque-3 col-chiki-6">
<input type="email" name="email" required onkeyup="this.setAttribute('value', this.value);" value="">
<label>Correo</label>
</div>

<div class="campo_bord_volador col-largo-2 col-medi-2 col-peque-3 col-chiki-6">
<input type="text" name="text" required value="">
<label>Verificacion</label>
</div>

<div class="campo_bord_volador col-largo-2 col-medi-2 col-peque-3 col-chiki-6">
<input type="password" name="text" required value="">
<label>contraseña</label>
</div>

<div class="min_desplazante campo_bord_volador col-largo-6 col-medi-6 col-peque-3 col-chiki-6">
<input type="text" name="text" required value="">
<label>Edad</label>
</div>

<input type="submit" name="siguiente" value="Siguiente">
</div><!--.maquetado-->
</div>
```

Figura 75. Código html5 final del componente input formulario parte III

Ejemplo en código - Input Diseñador

```
<div class="contenedor_form_progre">
<h2>Login</h2>
<div class="maquetado">
<div class="max_progre campo_bord_progre col-largo-6 col-medi-2 col-peque-3 col-chiki-6">
<input type="text" name="text" required value=""><span class="barra"></span>
<label>Nombre</label><br>
</div>
<div class="max_progre campo_bord_progre col-largo-6 col-medi-2 col-peque-3 col-chiki-6">
<input type="password" name="text" required value=""><span class="barra"></span>
<label>Contraseña</label><br>
</div>
<div class="max_progre campo_bord_progre col-largo-6 col-medi-2 col-peque-3 col-chiki-6">
<input type="email" name="number" required onkeyup="this.setAttribute('value', this.value);" value=""><span class="barra"></span>
<label>Correo</label><br>
</div>
<div class="max_progre campo_bord_progre col-largo-6 col-medi-2 col-peque-3 col-chiki-6">
<input type="number" name="text" required value=""><span class="barra"></span>
<label>edad</label><br>
</div>
<div class="max_progre campo_bord_progre col-largo-6 col-medi-2 col-peque-3 col-chiki-6">
<input type="text" name="text" required value=""><span class="barra"></span>
<label>Direccion</label><br>
</div>
<div class="max_progre campo_bord_progre col-largo-6 col-medi-2 col-peque-3 col-chiki-6">
<input type="text" name="number" required value=""><span class="barra"></span>
<label>Ciudad</label><br>
</div>
</div><!-- .maquetado-->
</div>
```

Responsive Desing

FASE 1.- Investigación del efecto y su importancia

Como primer paso lo primero que se iso fue investigar e indagar sobre el efecto, según la investigación se determinó que cada framework tiene un sistema responsive llamado también sistema de rejillas el cual consiste en clases preparadas para hacer responsive cualquier etiqueta HTML ganando así optimización en el proceso de maquetado.

Uno de las principales ventajas es que vienen con clases preparadas para cada tipo de pantalla las cuales son escritorio, tabletas y celulares. Estas clases preparadas permite que la maquetación de cualquier sitio web sea más rápida. Asimismo, beneficia a la construcción de un sitio web ya que garantiza que no se desborde la maquetación ya que estas son hechas con flexbox lo cual a diferencia de maquetarlo con posiciones son más resistentes a cambios. Fue elegido este efecto ya que no sería un framework completo sin este imprescindible efecto, además de que la existencia propiamente dicha de un framework es agilizar el desarrollo front-end con sus clases preparadas.

Otro del motivo el cual fue elegido el efecto para formar parte del framework XRL8 fue que el proyecto presente debe estar a la altura de los frameworks que están en la actualidad siendo usados en la industria del desarrollo web como son:

- **Bootstrap.** - El presente framework cuenta con la clase .row el cual es un contenedor de las clases responsive, dentro de dicha clase están las clases propiamente dichas las cuales tienen el prefijo col seguido de los tres tipos de pantalla pequeño, mediano y grande para finalmente tener números los cuales sumados llegan a doce, así se crea un equilibrio, si desean saber más sobre ello puede visitar la web oficial de documentación de dicho framework mediante el siguiente enlace:

<https://getbootstrap.com/docs/4.1/layout/grid/>

- **Bulma.**- Este framework también cuenta con una clase la cual para funcionar requiere de un contenedor el cual es la clase .columns luego de implementar esta clase recién se puede hacer uso de la propiedad is-mobile o is-desktop según sea las necesidades del usuario, para ver cómo se emplea y formas de uso revisar el siguiente enlace que los llevara a la documentación oficial de dicho framework:

<https://bulma.io/documentation/columns/responsiveness/>

- **TailwindCSS.**- Quisa sea este el framework de css más usado en los últimos 3 años sobre pasando a los demás por sus formas de integración con NodeJS y react. Este framework dentro de su contenido también tiene una clase llamada grid-cols-{n} la cual se usa conjugada con un contenedor llamado grid ambos deben estar en la misma clase pero separadas para su correcto funcionamiento, este sistema de columnas el modo de uso es sumar doce todas las clases en la etiqueta HTML donde se encuentre, mezclada con otras clases, se puede construir distintos proyectos la imaginación es el límite. Gracias a sus componentes, estos a diferencia de los demás se caracteriza por no tener clases predefinidas para efectos sino todo lo contrario tiene un conjunto de clases en el cual el usuario puede elegir los efectos más simples para colocar a su diseño.

Para revisar su documentación ingresar al siguiente enlace:

<https://tailwindcss.com/docs/grid-template-columns>

- **Foundation.-** Uno de los primeros frameworks en salir al mercado sin duda será foundation aunque con el pasar de los años ha perdido popularidad este framework también dentro de sus clases tiene una sección para el uso de rejillas el cual permite hacer responsive a cualquier etiqueta HTML dentro, tiene una clase llamada row el cual es un contenedor dentro de él, pueden ir unas clases hijas llamadas columns pero dentro de la misma clase donde va columns va las clases conjugadas las cuales son tres una llamada large que es para pantallas de escritorio, una clase médium así como una clase small las cuales se suman y la suma total debe ser doce para llegar a un equilibrio perfecto. Para ver el uso de este se puede recurrir al siguiente link:

<https://get.foundation/sites/docs/grid.html>

- **Materialize.-** Otro framework del medio es materialize el cual también cuenta con una sección llamada grid en el cual se puede hacer diseños responsive con las clases que cuenta dicho framework, estas clases se usan con un contenedor llamado row dentro del como si fueran clases hijas, estas se pueden usar con la clase col seguido de una clase llamada "s" la cual va acompañada de un numero el cual será el número de espacio dividido, en total debe sumar doce para que este en equilibrio y totalmente responsive. Lo podremos observar más detalladamente accediendo mediante el siguiente link:

<https://materializecss.com/grid.html>

- **UIKIT.-** También este framework cuenta con una clase para hacer responsive cualquier etiqueta HTML para ello usan la clase .uk-column-1{n} el cual no necesita tener una clase antes como contenedor para que pueda funcionar, para ver más detalles sobre este framework recomiendo revisar la documentación en el siguiente enlace:

<https://getuikit.com/docs/column>

- **Semantic UI.**- A continuación tenemos otro framework de CSS muy completo la verdad el cual tiene soporte y componentes para ser usado con React, la clase la cual ayuda a crear un sistema responsive se llama .ui grid esta clase es la clase padre ya que se debe usar con la clase column que es una clase hija, luego la suma de todas las clases debe dar dieciséis, podemos ver una gran cantidad de ejemplos mediante el siguiente enlace:

<https://semantic-ui.com/collections/grid.html#/definition>

Módulo: Un módulo según definición propia es una sección de trabajo principal a comparación de un submódulo este puede ser denominado la estructura padre y los submódulos estructuras hijas. En este efecto en particular el módulo vendría a ser la siguiente clase:

- **Responsive.** - Para este efecto nos basamos en flexbox ya que esta propiedad permite maquetar más robustamente, además de que se puede usar valores en porcentaje para una mayor efectividad.

Submódulos: Considere llamarle submódulo, a los efectos o clases que están dentro de un módulo en este caso el módulo es responsive. Un submódulo según denominación propia vendría hacer un efecto similar al original, pero con características diferentes, hagamos un ejemplo para comprender esto de una manera mejor.

- **Responsive grid.** - Este sería también otro submódulo en el cual se integraría la propiedad grid con la propiedad flex-box del módulo para alcanzar un maquetado más exquisito en cuanto a maquetación ya que ambas propiedades son complementarias.

Estos submódulos son muy útiles ya que desde mi punto de vista ayuda a diversificar los efectos y las clases. Beneficiando al usuario final en la construcción de sitios web. Actualmente el framework XRL8 no tiene pensado sacar sub módulos para el efecto responsive.

FASE 2.- Codificar el efecto

Antes de ver esta sección a fondo, quisiera explicar lo dificultoso que fue crear este efecto ya que me atrevo a decir que este efecto es el corazón de todo framework de css que existe y existirá en el mercado, sin él no tendría mucho sentido hacer la maquetación de otras clases o efectos. Debida a la alta complejidad que tiene el elaborar este efecto en particular me demoro tres meses hacerlo, ya que no es algo que se puede tomar a la ligera es trabajo duro y dedicación como horas de código, así como también la adquisición de curso avanzados de css ya que a diferencia de los demás para hacer este efecto se necesita conocimientos solidos de css.

Bien luego de tener claro que efecto queríamos construir y porque lo íbamos a construir como se vio en el capítulo anterior se procedió a escribir código como se muestra a continuación.

Código completo

[parte 01 - responsive]

Figura 76. Código de responsive desing parte I

```
1  .maquetado {
2    display: -webkit-box;
3    display: -ms-flexbox;
4    display: -webkit-flex;
5    display: flex;
6    flex-direction: row;
7    flex-wrap: wrap;
8  }
9
10 .maquetado > * {
11   padding: 1rem;
12 }
13
14 .maquetado > .col-chiki-1 {
15   flex: 0 0 16.6666666667%;
16   max-width: 16.6666666667%;
17 }
18
19 .maquetado > .col-chiki-2 {
20   flex: 0 0 33.3333333333%;
21   max-width: 33.3333333333%;
22 }
23
24 .maquetado > .col-chiki-3 {
25   flex: 0 0 50%;
26   max-width: 50%;
27 }
28
29 .maquetado > .col-chiki-4 {
30   flex: 0 0 66.6666666667%;
31   max-width: 66.6666666667%;
32 }
33
34 .maquetado > .col-chiki-5 {
35   flex: 0 0 83.3333333333%;
36   max-width: 83.3333333333%;
37 }
38
39 .maquetado > .col-chiki-6 {
40   flex: 0 0 100%;
41   max-width: 100%;
42 }
43
```

[parte 02 - responsive]

Figura 77. Código de responsive desing parte II

```
1
2 @media (min-width: 576px) {
3   .maquetado > .col-peque-1 {
4     flex: 0 0 16.6666666667%;
5     max-width: 16.6666666667%;
6   }
7
8   .maquetado > .col-peque-2 {
9     flex: 0 0 33.3333333333%;
10    max-width: 33.3333333333%;
11  }
12
13  .maquetado > .col-peque-3 {
14    flex: 0 0 50%;
15    max-width: 50%;
16  }
17
18  .maquetado > .col-peque-4 {
19    flex: 0 0 66.6666666667%;
20    max-width: 66.6666666667%;
21  }
22
23  .maquetado > .col-peque-5 {
24    flex: 0 0 83.3333333333%;
25    max-width: 83.3333333333%;
26  }
27
28  .maquetado > .col-peque-6 {
29    flex: 0 0 100%;
30    max-width: 100%;
31  }
32 }
```

[parte 03 - responsive]

Figura 78. Código de responsive desing parte III

```
1 @media (min-width: 768px) {
2   .maquetado > .col-medi-1 {
3     flex: 0 0 16.6666666667%;
4     max-width: 16.6666666667%;
5   }
6
7   .maquetado > .col-medi-2 {
8     flex: 0 0 33.3333333333%;
9     max-width: 33.3333333333%;
10  }
11
12  .maquetado > .col-medi-3 {
13    flex: 0 0 50%;
14    max-width: 50%;
15  }
16
17  .maquetado > .col-medi-4 {
18    flex: 0 0 66.6666666667%;
19    max-width: 66.6666666667%;
20  }
21
22  .maquetado > .col-medi-5 {
23    flex: 0 0 83.3333333333%;
24    max-width: 83.3333333333%;
25  }
26
27  .maquetado > .col-medi-6 {
28    flex: 0 0 100%;
29    max-width: 100%;
30  }
31 }
```

[parte 04 - responsive]

Figura 79. Código de responsive desing parte IV

```
1  @media (min-width: 992px) {
2    .maquetado > .col-largo-1 {
3      flex: 0 0 16.6666666667%;
4      max-width: 16.6666666667%;
5    }
6
7    .maquetado > .col-largo-2 {
8      flex: 0 0 33.3333333333%;
9      max-width: 33.3333333333%;
10   }
11
12   .maquetado > .col-largo-3 {
13     flex: 0 0 50%;
14     max-width: 50%;
15   }
16
17   .maquetado > .col-largo-4 {
18     flex: 0 0 66.6666666667%;
19     max-width: 66.6666666667%;
20   }
21
22   .maquetado > .col-largo-5 {
23     flex: 0 0 83.3333333333%;
24     max-width: 83.3333333333%;
25   }
26
27   .maquetado > .col-largo-6 {
28     flex: 0 0 100%;
29     max-width: 100%;
30   }
31 }
```

Descripción de las propiedades más significativas:

A continuación, se procederá a detallar las propiedades más usadas y su aplicación. También se mostrará la compatibilidad web de cada propiedad para su mejor comprensión dejamos una leyenda en la cual se indica si tiene compatibilidad web, entre otros factores todo ello se puede apreciar en la figura 2.

Propiedad max-width:100%:

Se uso max-width:100%, para el ancho del presente efecto se usó esta propiedad ya que tiene la particularidad de que como máximo llegara al 100% de su capacidad a diferencia de la propiedad width, que solo es un tipo de ancho específico mientras max-width impide que el ancho en width se más largo que max-width, dicho de una forma más sencilla max-width siempre será el valor más alto tanto que sobrescribirá al valor de width si este es menor al de max-width.

Se empleo max-width con el valor 100% para el ancho de este efecto ya que si bien esta propiedad tiene diversas unidades de medidas como:

- EM
- VH
- PIXELES
- REM
- PORCENTAJE

Se fue descartando una a una al momento de probarla, ya que la al usar la unidad EM este se distorsionaba por su propia naturaleza, básicamente esta propiedad usa el ancho de la pantalla del dispositivo donde estemos y las va ajustando así que fue descartado junto a las otras, para finalmente quedarnos con el porcentaje este valor ofrece una gran compensación a la hora de maquetar así que es excelente para crear una clase que haga responsive cualquier etiqueta HTML donde sea puesta.

A continuación, veamos la compatibilidad web al usar esta propiedad en los principales navegadores esto lo podemos observar en la figura 3.

Propiedad flex: 0 0 50%:

Esta propiedad también fue muy usada para la elaboración del presente efecto, para usar esta propiedad es un pre requisito tener una etiqueta padre con display:flex para que surjan efectos, flex es una forma abreviada ya que esta tiene tres instrucciones las cuales son:

- Flex-grow: Se usa un valor numérico, si se le deja en 0 usara el espacio que tenga el contenido o se repartirá el espacio en partes iguales debido al display:flex de la caja padre, por otro lado si el valor es mayor a 0 este incrementara su ancho con referencia a los demás componentes que estén con el, esto es equivalente a darle width a un elemento pero en flexbox.
- Flex-shrink: Se usa un valor numérico, esta propiedad lo que hace es encogerse o reducir el ancho que tienes a medida que se valla reduciendo los pixeles de la pantalla y viceversa va recuperando su tamaño a medida que se va expandiendo la pantalla hasta llegar a tener la misma medida que sus demás compañeros que estén en el contenedor, si lo dejamos en valor 0 lo que le estamos diciendo es que siempre mantenga su valor por default. Usualmente se usa para darle prioridad de espacio a uno u otro elemento.
- Flex-basis: Se usa una longitud o unidad de medida como es pixeles, rem, em, vh, porcentaje en esta propiedad a diferencia de las otras donde solo admiten números. Como su nombre lo indica se le asignara un valor base al elemento, esto quiere decir que ese valor base lo usara cuando puede usarlo, no necesariamente siempre va usar el valor asignado, además si requiere usar un valor menos a lo asignado este lo hara. Por otra parte, si hay más elementos en la etiqueta padre tomara lo máximo que pueda tomar respetando a los otros elementos que se encuentre de por medio en la misma etiqueta padre.

A continuación, veamos la compatibilidad web al usar esta propiedad en los principales navegadores podemos observarlo detenidamente en la figura 43.

Principales dificultades:

Para este efecto en concreto la principal dificultad fue el cómo maquetar ya que, si bien se podía hacer con posicionamientos, también se debía usar propiedades modernas así que el primer paso fue aprender flexbox a

profundidad, ya que los conceptos básicos no bastaban. Otra dificultad fue como hacer que se junten las clases para hacerlo responsive al 100%. Asi mismo apareció en el transcurso garantizar que no se desbordara el sistema de rejillas ya que si bien al conjugar otras propiedades se pueden distorsionar las medidas y por último otra dificultad fue si debíamos usar los `@media_query` en cada punto de corte ya sea para tabletas o pc de escritorio. Esas dificultades surgieron a raíz de las siguientes preguntas:

- ¿Si usamos en la maquetación con posicionamiento en vez de flexbox funcionara?

Si bien al empezar a realizarlo no se obtuvo un resultado optimo sirvió para darnos cuenta de cómo no se debe hacer las cosas.

- ¿Cómo logro saber en base a cuantas clases se logrará hacer alcanzar el 100% del total de pantalla?

Esta pregunta se logró resolver con base a como fue evolucionando el framework ya que en otros efectos que se trabajaba paralelamente se fue observando de que no se necesita demasiadas clases, es por ello que con 6 clases se llega a un equilibrio responsive.

- ¿Es necesario usar un `@media_query` en cada clase responsive para garantizar optimización en el proyecto?

Al principio creíamos que sí, pero el tiempo se encargó de mostrarnos que solo una clase en particular no necesitaría, esa clase fue la clase chiki ya que en los demás puntos de corte si se necesita para tener referencia en que puntos debe cambiar las propiedades responsive, pero como la clase chiki es el último punto de corte, el navegador debe tomar esto como referencia para dimensiones más pequeñas sin necesidad de seguir especificando. En este caso como se desconoce cuál será el dispositivo más pequeño por el cual el usuario navegará por la web se optó no ponerle un `@media_query` así agarraría cualquier dimensión más pequeña.

¿Qué usamos para hacer las pruebas?

Se usó el navegador Google Chrome y opera mini, así como la siguiente página web que es un simulador online:

<http://www.responsinator.com>

Este con motivos de asegurarnos que funcione en dispositivos móviles y tabletas. Ya que en base a lo trabajado no es del todo fiable usar el navegador para verificar la correcta maquetación de los elementos, así como tampoco es fiable depender de un software web, por ello se decidió diversificar en cuanto a las herramientas para probar las clases y efectos trabajados en el framework XRL8.

También usamos como editor de código visual studio code y su extensión live server para visualizar en vivo los efectos realizados en el css en tiempo real, también usamos el autoguardado que trae dicho editor de código para no estar guardando el proyecto a cada rato. Así conseguimos un mejor desempeño y evitamos el tedioso guardar cambios a cada rato, también evitamos el que la memoria cache se llene mostrándonos los mismos resultados.

FASE 3.- Hacerlo responsive

- Debe tener un contenedor que contenga al sistema responsive, para el correcto funcionamiento le di el nombre de maquetado.
- Dentro contiene a las clases responsive en este caso opte por conveniente que solo sean 6, gracias al uso de SASS estas pueden ser modificables al número que opten por conveniente. Opte por que sean seis por las siguientes razones:
 - Es más fácil maniobrar el total es seis toda la fila, además de que en dúos serian 3 + 3 que sumado seria 6, además de que considero innecesario usar más de 6.
 - Por cuestiones subjetivas, el cerebro asume que es más fácil aprender temas pequeños, así que considere que hacer combinaciones mayores al número 6 es un exceso.

- Col-largo-{n} se ejecuta en un mínimo de 992px dicho de paso es ahí también donde se encuentra el punto de corte para empezar a usar @media_query, se trabajó con la unidad de porcentaje ya que el responsive es excelente para ello. La lógica aplicada fue:
 - Col-largo-1 Esta parte vale 16.6666666667% que multiplicado por 6 daría un 100%.
 - Col-largo-2 Esta parte vale 33.3333333333% debido a que tendremos 3 clases más con el nombre col-largo-2 para que sume 100%
 - Col-largo-3 Esta parte vale 50% ya que solo puede haber una clase más con el nombre col-largo-3 que sume 100%
 - Col-largo-4 Esta parte vale 66.6666666667% ya que solo se le puede sumar una clase col-largo-2 para que sumado llegue al 100%
 - Col-largo-5 Esta parte vale 83.3333333333% que multiplicado por 6 daría un 100%.
 - Col-largo-6 Esta parte vale 100% debido a que abarcara todo el ancho de la pantalla o el contenedor dependiendo donde se encuentre.

- Col-medi-{n} se ejecuta en un mínimo de 768px dicho de paso es ahí también donde se encuentra el punto de corte para empezar a usar @media_query, se trabajó con la unidad de porcentaje ya que el responsive es excelente para ello. La lógica aplicada fue:
 - Col-medi-1 Esta parte vale 16.6666666667% que multiplicado por 6 daría un 100%.
 - Col-medi-2 Esta parte vale 33.3333333333% debido a que tendremos 3 clases más con el nombre col-largo-2 para que sume 100%
 - Col-medi-3 Esta parte vale 50% ya que solo puede haber una clase más con el nombre col-largo-3 que sume 100%
 - Col-medi-4 Esta parte vale 66.6666666667% ya que solo se le puede sumar una clase col-largo-2 para que sumado llegue al 100%

- Col-medi-5 Esta parte vale 83.3333333333% que multiplicado por 6 daría un 100%.
 - Col-medi-6 Esta parte vale 100% debido a que abarcara todo el ancho de la pantalla o el contenedor dependiendo donde se encuentre.
- Col-peque-{n} se ejecuta en un mínimo de 576px dicho de paso es ahí también donde se encuentra el punto de corte para empezar a usar @media_query, se trabajó con la unidad de porcentaje ya que el responsive es excelente para ello. La lógica aplicada fue:
 - Col-peque-1 Esta parte vale 16.6666666667% que multiplicado por 6 daría un 100%.
 - Col-peque-2 Esta parte vale 33.3333333333% debido a que tendremos 3 clases más con el nombre col-largo-2 para que sume 100%
 - Col-peque-3 Esta parte vale 50% ya que solo puede haber una clase más con el nombre col-largo-3 que sume 100%
 - Col-peque-4 Esta parte vale 66.6666666667% ya que solo se le puede sumar una clase col-largo-2 para que sumado llegue al 100%
 - Col-peque-5 Esta parte vale 83.3333333333% que multiplicado por 6 daría un 100%.
 - Col-peque-6 Esta parte vale 100% debido a que abarcara todo el ancho de la pantalla o el contenedor dependiendo donde se encuentre.
- Col-chiki-{n} se ejecuta sin un mínimo además tampoco cuenta con un punto de corte por ende no se usó el @media_query, ya que no es necesario porque las demás clases tenían un mínimo. Se trabajó con la unidad de porcentaje ya que el responsive es excelente para ello. La lógica aplicada fue:
 - Col-chiki-1 Esta parte vale 16.6666666667% que multiplicado por 6 daría un 100%.

- Col-chiki -2 Esta parte vale 33.3333333333% debido a que tendremos 3 clases más con el nombre col-largo-2 para que sume 100%
- Col-chiki-3 Esta parte vale 50% ya que solo puede haber una clase más con el nombre col-largo-3 que sume 100%
- Col-chiki-4 Esta parte vale 66.6666666667% ya que solo se le puede sumar una clase col-largo-2 para que sumado llegue al 100%
- Col-chiki-5 Esta parte vale 83.3333333333% que multiplicado por 6 daría un 100%.
- Col-chiki-6 Esta parte vale 100% debido a que abarcara todo el ancho de la pantalla o el contenedor dependiendo donde se encuentre.

FASE 4.- Testing asegurarse que no tenga conflicto con otros efectos.

En esta parte del desarrollo de la clase se dispuso a probar poniéndolo con las demás clases juntándolas. Para ello lo que se hizo fue colocar dentro de un archivo index.html todas las clases trabajadas y ver como el efecto de responsive desing respondía. En algunos casos había conflicto con los demás efectos para corregir esto se procedió a implementar en el desarrollo dos cosas básicas la primera fue:

- Para detectar que clase era la que causaba el conflicto y ver en que archivo scss estaba. Se procedía a inspeccionar la página para detectar donde estaba el problema. En muchas ocasiones no se daba con el archivo causante del error entonces en ese caso aplicábamos la frase “Divide y vencerás”. Lo que se procedió hacer fue comentar de uno en uno cada efecto en el archivo principal que los llamaba esto reducía las líneas de código de css, además como teníamos en tiempo real los cambios fue mucho más rápido detectar el problema. Ya que no se tenía que recargar a cada rato además otro punto a favor era que se auto compilaba los cambios de sass automáticamente gracias a nodeJS el cual mediante la consola de comandos de Windows cuando detectaba un cambio en las líneas de

código se auto compilaba y generando o reemplazando el archivo css en el cual hacíamos las pruebas, esto aumento la productividad para crear los efectos ya que al comentar de uno en uno cada archivo sass llegaba un momento donde desaparecía el error entonces revisábamos y nos dimos cuenta de que era uno o en algunos casos hasta tres efectos que hacían conflicto con la clase escrita.

- Se procedió a escribir mediante comentarios en un archivo llamado `_error_logs.scss` la clase que hace conflicto con la clase escrita, además de describir una nota breve en caso no se disponía de tiempo para solucionarlo luego, esto permitió tener un reporte más preciso de los incidentes que íbamos detectando.

FASE 5.- Aplicar la metodología OOCSS, llevándolo a SASS

- **Código css.** - Esta parte ya estaría completa por lo descrito anteriormente.
- **Código repetido.** - Identificación de código repetido

En este pequeño apartado luego de escribir y debugear el error se procedió a identificar el código css repetido para llevarlo a un archivo SCSS en sass, para su mayor optimización y aplicación de la metodología. El código repetido surgió en las siguientes clases:

- ✓ `col-largo-{n}`
- ✓ `col-medi-{n}`
- ✓ `col-peque-{n}`
- ✓ `col-chiki-{n}`

La cantidad de repeticiones fue seis veces por clase, esos códigos fueron:

Primer código.

Figura 80. Código repetido del componente responsive desing parte I

```
1 flex: 0 0 16.6666666667%;  
2 max-width: 16.6666666667%;
```

Segundo código.

Figura 81. Código repetido del componente responsive desing parte II

```
1 flex: 0 0 33.3333333333%;  
2 max-width: 33.3333333333%;
```

Tercer código:

Figura 82. Código repetido del componente responsive desing parte III

```
1 flex: 0 0 50%;  
2 max-width: 50%;
```

Cuarto código:

Figura 83. Código repetido del componente responsive desing parte IV

```
1 flex: 0 0 66.6666666667%;  
2 max-width: 66.6666666667%;
```

Quinto código:

Figura 84. Código repetido del componente responsive desing parte V

```
1 flex: 0 0 83.3333333333%;  
2 max-width: 83.3333333333%;
```

Sexto código:

Figura 85. Código repetido del componente responsive desing parte VI

```
1 flex: 0 0 100%;  
2 max-width: 100%;
```

- **Crear un objeto.** - Se describe la lógica de la función para luego crear un mixing.

Una vez identificado el código css repetido se procede hacer una función la cual se ejecuta al llamado del código, mediante esta técnica nuestro código será más puro y limpio.

Paralelamente en esta parte se determina la lógica de las funciones y ciclos como lo son for, while, each. También determinamos el pase de parámetros, así como la condicional que llevaran dentro, por otra parte, y no menos importante se define el nombre con el cual se llamara a las funciones.

- Esta función ayuda a determinar el número de elementos del cual se tendrá el sistema de responsive en el caso del framework XRL8 es de 6 pero puede ser modificable. Lleva la palabra return porque devolverá un valor.

```
function anchura_col($i){
  @return (100 / $num_elementos)*$i;
}
```

- En sass también se puede usar un ciclo for como en programación así que se diagramo como quedaría

```
for($i,$num_elementos<$i,$i++){
  .maquetado > .col-chiki-{$i}{
    flex: 0 0 {anchura_col($i)} "%";
    max-width: {anchura_col($i)} "%";
  }
}
```

- Otro dato asombroso también es que el sass al igual que un lenguaje de programación también se puede usar un foreach pero en sass se denomina each.

```
@each $k, $v in $puntos_de_corte{
  @media(min-width:#{ $v }){
    (for , $i=0; $num_elementos<$i, $i++){
      .maquetado > .col-#{ $k+"-"+ $i }{
        flex: 0 0 #{anchura_col($i)}+ "%";
        max-width: #{anchura_col($i)}+ "%";
      }
    }
  }
}
```

La función que se mostrara a continuación se implementó como parte del módulo, más no como parte de la metodología OCSS, estas funciones ayudan a tener un contenedor dentro del maquetado, así como ayuda a tener un contenedor antes de aplicarlo al sistema de rejillas también llamado módulo responsive en este caso.

```
function centrado(){
  margin:0 auto;
}
```

```
function flex_mixin(){
  display:-webkit-box;
  display:-ms-flexbox;
  display:flex;
  flex-direction:row;
  flex-wrap:wrap;
}
```

Luego de tener la lógica se procede a llevarlo a SASS para ello tenemos dos opciones, la primera es usar @function ya que como su nombre ase referencia es para crear una función la otra opción es usar @mixin, pero ambas tienen diferencias. La primera retorna un valor, mientras @mixin solo llama a los valores en este caso no necesitamos que retorne un valor por ello usaremos @mixin. Entonces quedaría de la siguiente forma:

Figura 86. Creación del objeto ancho_col en la función

```
1 @function anchura_col($i) {
2   @return (100 / $num_elementos)*$i ;
3 }
```

Figura 87. Creación del objeto en un ciclo for para la clase col-chiki-n

```
1 @for $i from 1 through $num_elementos {
2   .maquetado > .col-chiki-#{ $i } {
3     flex: 0 0 #{anchura_col($i)}+ "%";
4     max-width: #{anchura_col($i)}+ "%";
5   }
6 }
```

Figura 88. Creación del objeto usando puntos de corte dentro de un ciclo each

```
1 @each $k, $v in $puntos_de_corte{
2   @media (min-width: #{ $v }) {
3     //Bucle para generar las clases de las columnas responsivas
4     @for $i from 1 through $num_elementos {
5       .maquetado > .col-#{ $k }-#{ $i } {
6         flex: 0 0 #{anchura_col($i)}+ "%";
7         max-width: #{anchura_col($i)}+ "%";
8       }
9     }
10  }
11 }
```

- **OOCSS**. – Luego de tener la función o la lógica para reciclar el código se procede a integrar la metodología OOCSS para ello debemos llamar al mixin o la función, para ello solo necesitamos agregar @include “nombre del mixin”. Luego de integrarlo ya estará aplicada correctamente la metodología OOCSS

Figura 89. Metodología OOCSS en el componente responsive desing parte I

```
1  .maquetado > * {
2    padding: $espacion_interno_maquetado;
3  }
4
5  //Clase que estara centrada con ancho de 95%
6  .contenedor {
7    max-width: $contenedor_width;
8    @include centrado();
9    padding-top:0;
10 }
```

Figura 90. Metodología OOCSS en el componente responsive desing parte II

```
1  .maquetado {
2    @include flex_mixin;
3  }
```

Cabe hacer una aclaración en esta parte, si bien tuvimos funciones, for y un each todos esos métodos se aplicó de forma directa la metodología OOCSS y no se requirió llamarla como se hace con el mixin ya que dentro del for se ejecutó de forma automática. También muy importante se observa que al implementar la metodología queda fuera del mixin pero dentro de la clase contenedor algunos atributos, esto permite dos cosas:

- Usar condicionales dentro de SASS para acceder a clases de css, que veremos en el transcurso del proyecto.
- Hacer más personalizable el framework, ya para hacer cambio solo bastaría con cambiar la variable o añadir una en vez de revisar el código css ya compilado.

Bien volviendo al tema de las variables estas variables se encuentran en un archivo sass denominado variables valga la redundancia donde se encuentran las siguientes variables:

Figura 91. Variables Sass del componente responsive desing

```
1 // Padding para la separación entre las distintas columnas
2 $espacion_interno_maquetado: 1rem;
3
4 // Anchura del contenedor principal
5 $contenedor_width: 95%;
6
7 // Número de elementos máximos que voy a tener a lo ancho de la maquetacion
8 $num_elementos: 6;
9
10 // Mapa con dimensiones de maquetacion - puntos_rotos
11 $puntos_de_corte: (
12   "peque": 576px,
13   "medi": 768px,
14   "largo": 992px
15 );
```

Como se ha visto SASS es un preprocesador de css por ello se ejecuta antes del css tradicional esto permite añadir variables agregar condicionales, etc. Con la aplicación de la metodología como se observa se reutiliza el código css, además de que se hace escalable el proyecto. Ya que, si en un futuro queremos cambiar o añadir más propiedades css, solo cambiaríamos las variables o añadiríamos más propiedades en el mixin.

FASE 6.- Documentar en la web site www.phooldx.com

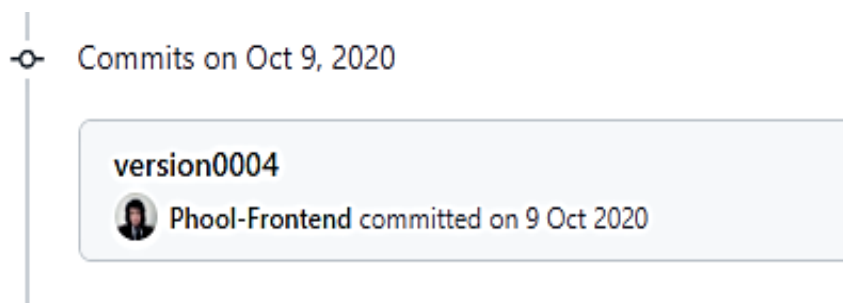
Finalmente, si el efecto paso por todos los demás filtros se procede a su documentación respectiva, para documentar todos los efectos se creó una página web de documentación usando clases y elementos del mismo framework. A continuación, se describirá como se procede a la documentación de la misma. Siguiendo estos pasos:

- Se copia las etiquetas de HTML5 junto con la clase a un txt temporal para no perder la sintaxis
- Se guarda los cambios en el compilador de SASS
- Como es la parte final se procedió hacer copias de seguridad para no perder el avance, en archivos RAR. Estos archivos estaban por versiones enumerados además se agregaba un pequeño texto describiendo el avance. Este guardado se hacía de manera local en un USB.
- Para una mayor seguridad del avance del código se procedió luego a copiar la versión del archivo RAR junto a la descripción del mismo.

Guardado de código en GitHub. - Luego de tener la copia del proyecto a nivel físico en un USB, se procedió a guardarlo en un repositorio de GitHub para lo cual se usó Git, para ello se procedió de la siguiente forma:

- Se guardaba todo el proyecto con el siguiente comando que se puede apreciar en la figura 12.
- Luego se verificaba los cambios guardados con el siguiente comando que se aprecia en la figura 13.
- Después se hacía el commit, es decir se le asignaba un nombre a los cambios realizados con el siguiente comando que se encuentra en la figura 14.
- Luego se procede a empaquetar los datos para que se suban al repositorio de GitHub con el siguiente código que lo podemos encontrar en la figura 15.
- Finalmente, luego de haber enviados el proyecto empaquetado hacia GitHub como se puede apreciar en la imagen, este queda guardado en dicho repositorio.

Figura 92. Visualización en github del componente responsive desing



Se guarda en una escala de tiempo, con todos los cambios hechos desde la creación del proyecto, esto ayuda a tener un mejor control de versiones. Asimismo, también ayuda a que más personas se sumen al proyecto y puedan clonar el proyecto y trabajar por separado el proyecto, entre otras muchas ventajas.

- Así mismo se manejó un archivo llamado bitácora.txt donde aquí se registró todos los incidentes ocurridos, así como los cambios hechos, se registraba también las metas cumplidas, también las tareas nuevas para desarrollar al día siguiente. Al finalizar con un efecto por completo se quitaba de la lista de prioridades así dábamos pase al siguiente efecto o problema a resolver. Esto servía para tener un mejor orden y saber ejecutar prioridades.
- Luego de tener guardado el avance y de registrar el avanzase, se procedió a copiar del archivo documentación Etiquetas.txt el fragmento de código que se copió anteriormente para luego pegarlo en www.nosetup.org esta página web se encargaba de convertir html en texto plano para poder visualizar html
- Antes de finalizar se procedió a crear un archivo llamado modulo_responsive_desing.html donde se dividía en tres partes:
 - En la primera se describía un poco al efecto que era y que hacia
 - En la segunda parte se procedía a copiar el código ya formateado obtenido de www.nosetup.com para luego pegarlo.
 - En la tercera parte lo que se hacía era mostrar un apartado llamado estructura donde se describía como se llamaban las variables para que sea modificado en SASS.
- Finalmente, el ejemplo listo para usarse queda de la siguiente forma:

Figura 93. Código html5 final del componente responsive desing

```

<div class="contenedor">
  <div class="maquetado">
    <div style="background:red"class="col-largo-2 col-medi-2 col-peque-3 col-chiki-6">
      <h1>Hola mundo</h1>
    </div>

    <section style="background:gold"class="col-largo-2 col-medi-2 col-peque-3 col-chiki-6">
    </section>

    <div style="background:blue" class="col-largo-2 col-medi-2 col-peque-0 col-chiki-6">
  </div>

```

Libro 3D

FASE 1.- Investigación del efecto y su importancia

Como primer paso lo primero que se hizo fue investigar e indagar sobre el efecto, así que se empezó a visitar páginas web sitios web y se observó que la mayoría de sitios web como la de restaurantes en su gran mayoría solo muestran imágenes en sus sitios web, fue aquí donde surgió la idea de empezar a crear un efecto que simule el tener un libro o una carta de algún restaurante o quisa un bar, donde con este efecto con forme vallas cambiando de hoja se valla cambiando la carta, propiamente dicho este efecto está enfocado a la industria del consumo como lo son restaurantes y bares aunque también puede ser aplicado a otros sectores como el sector de las prendas de vestir. Un gran beneficio es que se ahorraría espacio en mostrar contenido ya que el cliente no tendría que scrollear la página, fue elegido para ser un módulo de XRL8 fue elegido el efecto para formar parte del framework XRL8 fue que no se encontró un framework propiamente dicho enfocado a este tema en específico solo se encontró librerías dedicadas a la animación de elementos las cuales son:

- **BounceJS.** – La presente librería es una hecha con JavaScript enfocada a la animación de css3 con fotogramas, si desean saber más sobre ello puede visitar el repositorio de GitHub mediante el siguiente enlace:

<https://github.com/tictail/bounce.js>

- **AnimeJS.-** Esta librería también está enfocada a la creación de efectos y transiciones en 3D usando el lenguaje de programación JavaScript para la manipulación del DOM, para ver cómo se emplea y formas de uso revisar el siguiente enlace que los llevara a la documentación oficial de dicha librería:

<https://animejs.com/documentation/>

- **AnimateJS.-** Es una librería desarrollado en css, el cual permite hacer animaciones web, tiene integración al gestor de paquetes más grande de la comunidad web el cual es nodeJS mediante npm,también tiene

integración con yarn . Para revisar su documentación ingresar al siguiente enlace:

<https://animate.style/>

- **Magic animations.-** Es una librería de css hecha con sass y gulp, esta librería ayuda hacer animaciones en css, además cuenta con gestor de paquetes npm y yarn, así como también se puede usar con jquery. Para ver el uso de este se puede recurrir al siguiente link:

<https://github.com/miniMAC/magic>

- **Módulo:** Un módulo según definición propia es una sección de trabajo principal a comparación de un submódulo este puede ser denominado la estructura padre y los submódulos estructuras hijas. En este efecto en particular el módulo vendría a ser la siguiente clase:
- **Libro 3D.** – Este módulo cuenta con seis caras o páginas, muestras de frente el contenido, además de cuenta con botones de negación para que puedan dar la vuelta a las páginas, en estas páginas se pueden agregar contenido multimedia como imágenes, videos, audio todo completamente personalizable por el usuario.

Submódulos: Considere llamarle submódulo, a los efectos o clases que están dentro de un módulo en este caso el submódulo es carta 3D, hagamos un ejemplo para comprender esto de una manera mejor.

- **Carta 3D.** - Este sería un sub modulo ya que si bien no tiene atributos compartidos pertenecen a esta categoría ya que están en el mismo plano 3D además de que sus funciones son semejantes a las de libro 3D a diferencia del módulo esta gira sobre tu propio eje.

Este submódulo es útil ya que desde mi punto de vista ayuda a diversificar los efectos y las clases. Beneficiando al usuario final en la construcción de sitios web.

FASE 2.- Codificar el efecto

Antes de ver esta sección a fondo quisiera hacer énfasis, que fue complicado la codificación del módulo, la del submódulo no fue tan complicado ya que solo gira sobre su propio eje.

Bien luego de tener claro que efecto queríamos construir y porque lo íbamos a construir como se vio en el capítulo anterior se procedió a escribir código como se muestra a continuación.

Código completo

❖ Código del módulo libro 3D

[parte 01 – libro 3D]

Figura 94. Código de libro 3d parte I

```
1  /***** ZONA LIBRO 3D *****/
2  .btn_sigui_hoja {
3    background: white;
4    border: solid 3px;
5    cursor: pointer;
6    position: absolute;
7    bottom: 20px;
8    float: right;
9    clear: both;
10   padding: 5px 10px;
11 }
12
13 .btn_atras_hoja {
14   background: white;
15   border: solid 3px;
16   cursor: pointer;
17   position: absolute;
18   bottom: 20px;
19   right: 13px;
20   padding: 4px 10px;
21 }
22
23 #portada {
24   width: 250px;
25   height: 400px;
26 }
27
28 .tomo_libro_3d {
29   width: 250px;
30   height: 400px;
31   position: relative;
32   perspective: 1500px;
33 }
```

[parte 02 – libro 3D]

Figura 95. Código de libro 3d parte II

```
1
2 .pag_libro {
3   width: 100%;
4   height: 100%;
5   position: absolute;
6   top: 0;
7   left: 0;
8   transform-origin: left;
9   transform-style: preserve-3d;
10  transform: rotateY(0deg);
11  transition: 0.5s;
12  color: #000;
13 }
14
15 .hoja_pos {
16   width: 100%;
17   height: 100%;
18   position: absolute;
19   top: 0;
20   left: 0;
21   box-sizing: border-box;
22   padding: 13px;
23   box-shadow: inset 20px 0 50px rgba(0, 0, 0, 0.5) 0 2px 5px rgba(0, 0, 0, 0.5);
24   background-color: white;
25 }
```

[parte 03 – libro 3D]

Figura 96. Código de libro 3d parte III

```
1 .hoja_an {
2   width: 100%;
3   height: 100%;
4   position: absolute;
5   top: 0;
6   left: 0;
7   transform: rotateY(180deg);
8   backface-visibility: hidden;
9   z-index: 99;
10  background-color: white;
11 }
12
13 .cont_lib_3d input[type=checkbox] {
14   display: none;
15 }
16
17 .hoja_pos p {
18   font-size: 14p;
19   line-height: 24px;
20 }
21
22 .cont_lib_3d {
23   display: flex;
24   justify-content: center;
25   font-family: "Roboto", sans-serif;
26   align-items: center;
27 }
28
29 .cont_lib_3d img {
30   width: 100%;
31   height: 100%;
32 }
```

[parte 04 – libro 3D]

Figura 97. Código de libro 3d parte IV

```
1 @media screen and (max-width:496px){
2   body{
3     background-color: #f29720;
4   }
5   .cont_lib_3d{
6     display:block;
7   }
8   #portada{
9     width: 100%;
10    height: auto;
11  }
12  .pag_libro{
13    position:relative;
14    transform-origin: initial;
15  }
16  .cont_lib_3d img {
17    width: 100%;
18    // display:none;
19  }
20  .tomo_libro_3d {
21    width: 100%;
22  }
23 }
```

❖ Código del submódulo cartas 3D

[parte 01 – carta 3D]

Figura 98. Código de libro 3d parte V

```
1  /***** CARTA 3D *****/
2  .caja {
3    display: flex;
4    justify-content: center;
5    width: 1100px;
6    margin: 100px auto;
7  }
8
9  .adelante {
10   width: 100%;
11   height: 100%;
12   position: absolute;
13   top: 0;
14   left: 0;
15   backface-visibility: hidden;
16   -webkit-backface-visibility: hidden;
17 }
18
19 .atras {
20   width: 100%;
21   height: 100%;
22   position: absolute;
23   top: 0;
24   left: 0;
25   backface-visibility: hidden;
26   -webkit-backface-visibility: hidden;
27   display: flex;
28   justify-content: center;
29   transform: rotateY(180deg);
30   padding: 15px;
31   align-items: center;
32   text-align: center;
33   color: white;
34 }
```


[parte 02 – carta 3D]

Figura 99. Código de libro 3d parte VI

```
1  .tarjeta-caja:hover .tarjeta {
2    transform: rotateY(180deg);
3  }
4
5  .tarjeta-caja {
6    margin: 3%;
7    -webkit-perspective: 800;
8    perspective: 800;
9  }
10
11 .tarjeta {
12   width: 300px;
13   height: 350px;
14   background: #f29720;
15   position: relative;
16   transform-style: preserve-3d;
17   transition: 0.7s ease;
18   -webkit-box-shadow: 0px 10px 15px -5px rgba(0, 0, 0, 0.65);
19   box-shadow: 0px 10px 15px -5px rgba(0, 0, 0, 0.65);
20 }
21
22 .carta1 {
23   background-image: url("../multimedia/carta01.jpg");
24   background-size: cover;
25 }
26
27 .carta2 {
28   background-image: url("../multimedia/carta02.jpg");
29   background-size: cover;
30 }
31
32 .carta3 {
33   background-image: url("../multimedia/carta03.jpg");
34   background-size: cover;
35 }
```

Descripción de las propiedades más significativas:

A continuación, se procederá a detallar las propiedades más usadas y su aplicación. También se mostrará la compatibilidad web de cada propiedad para su mejor comprensión dejamos una leyenda en la cual se indica si tiene compatibilidad web esto podemos observarlo detenidamente en la figura 2.

Propiedad width:100% :

Se empleó width 100% para el ancho de este efecto ya que si bien esta propiedad tiene diversas unidades de medidas como:

- EM
- VH
- PIXELES
- REM
- PORCENTAJE

Se fue descartando una a una al momento de probarla, ya que la al usar la unidad EM este se distorsionaba por su propia naturaleza, básicamente esta propiedad usa el ancho de la pantalla del dispositivo donde estemos y las va ajustando así que fue descartado.

A continuación, veamos la compatibilidad web al usar esta propiedad en la figura 3.

Propiedad background-color:white

Esta propiedad se usa para darle un color de fondo para esta ocasión se decidió que sería blanco. Ya que es más comercial y combina con cualquier tonalidad.

A continuación, veamos la compatibilidad web al usar esta propiedad en la figura 5.

Propiedad display: flex

Con esta propiedad se puede aplicar la alineación de los elementos en una sola dirección sea horizontal o vertical, estos valores se puede conjugar con las distintos valores que tiene, se optó por esta propiedad ya que es una forma óptima de maquetar y de que los elementos no se desborden de sus contenedores como solía suceder al maquetar usando position o tables en la antigüedad además flexbox viene para usarlo con la versión 3 de css llamada css3 haciendo más robusta la maquetación de elementos.

A continuación, veamos la compatibilidad web al usar esta propiedad en la figura 43.

Propiedad align-items: center

Para usar esta propiedad es un requisito indispensable tener habilitado `display:flex`, luego de eso se puede conjugar `align-items` con los siguientes valores:

- CENTER
- END
- FLEX-END
- FLEX-START
- INHERIT
- INITIAL
- NORMAL
- REVERT
- SELF-END
- START
- STRETCH
- UNSET

En este caso se usó con el valor `center` ya que lo que ahora es centrar en el centro de gravedad a los elementos, pero en dirección vertical.

A continuación, veamos la compatibilidad web al usar esta propiedad en la figura 44.

Propiedad justify-content:center

La propiedad `justify-content` tiene como prerrequisitos haber declarado `display:flex` en el mismo elemento o en un elemento padre. Tiene las siguientes conjugaciones:

- CENTER
- END
- FLEX-END
- FLEX-START
- INHERIT
- INITIAL
- LEFT
- NORMAL
- REVERT
- RIGHT
- SPACE-AROUND
- SPACE-BETWEEN
- SPACE-EVENLY
- START
- STRETCH
- UNSET

Se optó por elegir el valor center ya que este centra en horizontal cualquier elemento.

A continuación, veamos la compatibilidad web al usar esta propiedad en la figura 45.

Propiedad line-height:24px

Esta propiedad es para darle relleno solo de altura en donde se aplique similar al padding pero con uso más orientado a texto, aunque se puede aplicar también a etiquetas html.

A continuación, veamos la compatibilidad web al usar esta propiedad en la figura 42.

Propiedad border:solid 3px

En esta propiedad le damos un border sólido, esto quiere decir que el borde será visible, además de que será una línea definida sin interrupciones. Luego la otra característica es que tendrá una anchura de 3px, dicho de mejor forma será una línea fina.

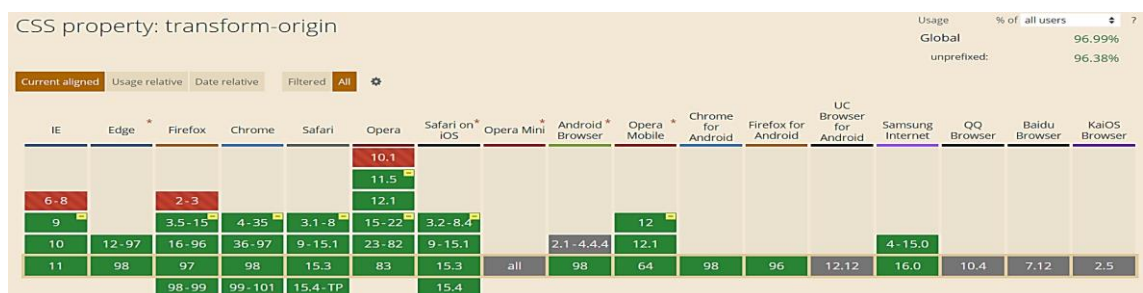
A continuación, veamos la compatibilidad web al usar esta propiedad en la figura 37.

Propiedad transform-origin:left

En esta propiedad lo que se hace es definir el origen de la transformación en este caso en particular para hacer las tarjetas 3D el giro es hacia la derecha así que se puso left.

A continuación, veamos la compatibilidad web al usar esta propiedad en los principales navegadores.

Figura 100. Compatibilidad web, propiedad transform-origin



Propiedad perspective:1500px

Sin mucha ciencia y explicándolo con palabras sencillas esta propiedad su función es asignar un centro de visión para poder apreciar los efectos 3D en un plano de z, ya que los planos x e y se pueden apreciar perfectamente en la pantalla.

A continuación, veamos la compatibilidad web al usar esta propiedad en los principales navegadores.

Figura 101. Compatibilidad web, propiedad perspective



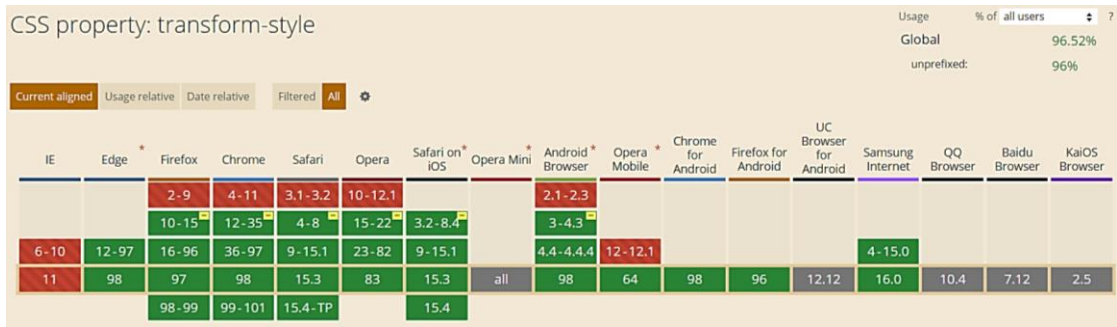
Propiedad transform-style:preserve-3d

Se establece en la propiedad transform-style que sea un elemento padre para que surjan efectos en el elemento hijo, este especifica como se representaran los elementos anidados en el espacio 3D, así mismo la propiedad transform-style se conjuga con las siguientes propiedades:

- ❖ **Flat.** - Indica a los elementos secundarios en el mismo nivel, los conserva en un valor predeterminado el cual no es el 3D.
- ❖ **Preserve-3d.**- indica que el elemento debe estar posicionado en el espacio 3D dicho de otra forma permite que conserve su posicionamiento 3D
- ❖ **Initial.**- Se utiliza para establecer la propiedad en su valor predeterminado.
- ❖ **Inherit.** - Se utiliza para heredar la propiedad de su elemento padre.

A continuación, veamos la compatibilidad web al usar esta propiedad en los principales navegadores.

Figura 102. Compatibilidad web, propiedad transform-style



Propiedad transform:rotateY(0deg)

La función que cumple es rotar la etiqueta sobre la cual se aplicó la propiedad en el eje Y por otra parte el prefijo deg hace referencia a los grados de inclinación en esta propiedad en particular es 0 lo que quiere decir que se mantendrá en una posición sin inclinación o dicho de otra forma se mantendrá estático.

A continuación, veamos la compatibilidad web al usar esta propiedad en los principales navegadores.

Figura 103. Compatibilidad web, propiedad transforms



Propiedad transition:0.5s

La propiedad transition tiene muchas conjugaciones en este caso en particular lo que indica es el tiempo de velocidad al cambiar de una propiedad de css a otra. Dichos valores están expresados en segundos y se le agrega el prefijo s para indicarlo.

A continuación, veamos la compatibilidad web al usar esta propiedad en la figura 47.

Principales dificultades:

Para este efecto en concreto la principal dificultad fue organizar de distintas formas posibles, así como ver las posibles conjugaciones que pueden hacer con el efecto para que el usuario al final no tenga problemas al implementarlo. También fue difícil organizar las animaciones para que resulten en 3D y estas no se rompan al agregarle más capas. Esas dificultades surgieron a raíz de las siguientes preguntas:

- ¿Si el usuario quiere puede agregar más cartas en 3D?

Al principio creíamos que se emplearía más clases para poder cubrir la demanda, pero se decidió solo tener 3 clases consecutivas que no tienen conflictos entre si al estar en una misma fila, las misma que pueden replicarse y aumentar.

- ¿Cómo hacemos para tener varias capas para el libro en 3D?

Esta pregunta se solucionó con los z-index los cuales ocultan según la prioridad del número que se le asigne.

¿Qué usamos para hacer las pruebas?

Se usó el navegador Google Chrome y opera mini, así como la siguiente página web que es un simulador online:

<http://www.responsinator.com>

Este con motivos de asegurarnos que funcione en dispositivos móviles y tabletas. Ya que en base a lo trabajado no es del todo fiable usar el navegador para verificar la correcta maquetación de los elementos, así como tampoco es fiable depender de un software web, por ello se decidió diversificar en cuanto a las herramientas para probar las clases y efectos trabajados en el framework XRL8.

También usamos como editar de código visual studio code y su extensión live server para visualizar en vivo los efectos realizados en el css en tiempo real, también usamos el autoguardado que trae dicho editor de código para no estar guardando el proyecto a cada rato. Así conseguimos un mejor desempeño y evitamos el tedioso guardar cambios a cada rato, también evitamos el que la memoria cache se llene mostrándonos los mismos resultados.

FASE 3.- Hacerlo responsive

Con este efecto en particular se usó dimensiones, para hacerlo responsive se tuvo que usar el siguiente método:

- Usar la propiedad flex-box y sus posibles conjugaciones con cada uno de sus valores según sea el caso.
- Se trabajó con la propiedad @media_query en las resoluciones de pantalla para celulares y tabletas. Siempre empezando a maquetar por la parte de celulares y luego tabletas. En la siguiente imagen se observa que se implementó la parte responsive en el punto de corte 496px ya que ahí es donde se distorsiona el proyecto.

Figura 104. Uso de media query en el componente libro 3d

```
1  @media screen and (max-width:496px){
2    body{
3      background-color: #f29720;
4    }
5    .cont_lib_3d{
6      display:block;
7    }
8    #portada{
9      width: 100%;
10     height: auto;
11   }
12   .pag_libro{
13     position:relative;
14     transform-origin: initial;
15   }
16   .cont_lib_3d img {
17     width: 100%;
18     // display:none;
19   }
20   .tomo_libro_3d {
21     width: 100%;
22   }
23 }
```


FASE 4.- Testing asegurarse que no tenga conflicto con otros efectos.

En esta parte del desarrollo de la clase se dispuso a probar poniéndolo con las demás clases juntándolas. Para ello lo que se hizo fue colocar dentro de un archivo index.html todas las clases trabajadas y ver como el efecto de libro_3d respondió. En algunos casos había conflicto con los demás efectos para corregir esto se procedió a implementar en el desarrollo dos cosas básicas la primera fue:

- Para detectar que clase era la que causaba el conflicto y ver en que archivo scss estaba. Se procedía a inspeccionar la página para detectar donde estaba el problema. En muchas ocasiones no se daba con el archivo causante del error entonces en ese caso aplicábamos la frase “Divide y vencerás”. Lo que se procedió hacer fue comentar de uno en uno cada efecto en el archivo principal que los llamaba esto reducía las líneas de código de css, además como teníamos en tiempo real los cambios fue mucho más rápido detectar el problema. Ya que no se tenía que recargar a cada rato además otro punto a favor era que se auto compilaba los cambios de sass automáticamente gracias a nodeJS el cual mediante la consola de comandos de Windows cuando detectaba un cambio en las líneas de código se auto compilaba y generando o reemplazando el archivo css en el cual hacíamos las pruebas, esto aumento la productividad para crear los efectos ya que al comentar de uno en uno cada archivo sass llegaba un momento donde desaparecía el error entonces revisábamos y nos dimos cuenta de que era uno o en algunos casos hasta tres efectos que hacían conflicto con la clase escrita.
- Se procedió a escribir mediante comentarios en un archivo llamado _error_logs.scss la clase que hace conflicto con la clase escrita, además de describir una nota breve en caso no se disponía de tiempo para solucionarlo luego, esto permitió tener un reporte más preciso de los incidentes que íbamos detectando.

FASE 5.- Aplicar la metodología OOCSS, llevándolo a SASS

- **Código css.** - Esta parte ya estaría completa por lo descrito anteriormente.

- **Código repetido.** - Identificación de código repetido

En este pequeño apartado luego de escribir y debugear el error se procedió a identificar el código css repetido para llevarlo a un archivo SCSS en sass, para su mayor optimización y aplicación de la metodología. En la clase descrita a continuación se observó que se repetía el siguiente código:

❖ Libro 3D

Figura 105. Código repetido del componente libro 3d parte I

```
1  .btn_sigui_hoja {
2    background: white;
3    border: solid 3px;
4    cursor: pointer;
5    position: absolute;
6    bottom: 20px;
7    float: right;
8    clear: both;
9    padding: 5px 10px;
10 }
```

Figura 106. Código repetido del componente libro 3d parte II

```
1  .btn_atras_hoja {
2    background: white;
3    border: solid 3px;
4    cursor: pointer;
5    position: absolute;
6    bottom: 20px;
7    right: 13px;
8    padding: 4px 10px;
9  }
```

Figura 107. Código repetido del componente libro 3d parte III

```
1  #portada {
2    width: 250px;
3    height: 400px;
4  }
```

Figura 108. Código repetido del componente libro 3d parte IV

```
1 .tomo_libro_3d {
2   width: 250px;
3   height: 400px;
4   position: relative;
5   perspective: 1500px;
6 }
```

Figura 109. Código repetido del componente libro 3d parte V

```
1 .pag_libro {
2   width: 100%;
3   height: 100%;
4   position: absolute;
5   top: 0;
6   left: 0;
7   transform-origin: left;
8   transform-style: preserve-3d;
9   transform: rotateY(0deg);
10  transition: 0.5s;
11  color: #000;
12 }
```

Figura 110. Código repetido del componente libro 3d parte VI

```
1 .hoja_pos {
2   width: 100%;
3   height: 100%;
4   position: absolute;
5   top: 0;
6   left: 0;
7   box-sizing: border-box;
8   padding: 13px;
9   box-shadow: inset 20px 0 50px rgba(0, 0, 0, 0.5) 0 2px 5px rgba(0, 0, 0, 0.5);
10  background-color: white;
11 }
```

Figura 111. Código repetido del componente libro 3d parte VII

```
1 .hoja_an {
2   width: 100%;
3   height: 100%;
4   position: absolute;
5   top: 0;
6   left: 0;
7   transform: rotateY(180deg);
8   backface-visibility: hidden;
9   z-index: 99;
10  background-color: white;
11 }
```

❖ Carta 3D

Figura 112. Código repetido del componente libro 3d parte VIII

```
1  .caja {
2    display: flex;
3    justify-content: center;
4    width: 1100px;
5    margin: 100px auto;
6  }
```

Figura 113. Código repetido del componente libro 3d parte IX

```
1  .adelante {
2    width: 100%;
3    height: 100%;
4    position: absolute;
5    top: 0;
6    left: 0;
7    backface-visibility: hidden;
8    -webkit-backface-visibility: hidden;
9  }
```

Figura 114. Código repetido del componente libro 3d parte X

```
1  .atras {
2    width: 100%;
3    height: 100%;
4    position: absolute;
5    top: 0;
6    left: 0;
7    backface-visibility: hidden;
8    -webkit-backface-visibility: hidden;
9    display: flex;
10   justify-content: center;
11   transform: rotateY(180deg);
12   padding: 15px;
13   align-items: center;
14   text-align: center;
15   color: white;
16 }
```

- **Crear un objeto.** - Se describe la lógica de la función para luego crear un mixing.

Una vez identificado el código css repetido se procede hacer una función la cual se ejecuta al llamado del código, mediante esta técnica nuestro código será más puro y limpio.

Paralelamente en esta parte se determina la lógica de las funciones y ciclos como lo son for, while, each. También determinamos el pase de parámetros, así como la condicional que llevaran dentro, por otra parte, y no menos importante se define el nombre con el cual se llamara a las funciones.

❖ Libro 3D

```
function btn_hoja_navega(){
background:white;
border:solid 3px;
cursor:pointer;
position:absolute;
bottom:20px;
}
function hojas_pag_libro(){
width:100%;
height:100%;
position:absolute;
top:0;
left:0;
}

function porta_and_tomo(){
width:250px;
height:400px;
}

for($i,$num_paginas < $i,$i++){
#desplegar_hoja_#{i}:checked~ .tomo_libreo_3d #pagina#{$i}{
transform:rotateY(-180deg);
z-index:$i;
}
}

for($i,$num_paginas < $i,$i++){
#pagina#{$i}{
z-index:$num_paginas;
}
$num_paginas:$num_paginas – 1;
}
```

❖ Carta 3D

```
function dimension_caja_and_atras(){
  display:flex;
  justify-content:center;
}

function adelante_and_atras(){
  width:100%;
  height:100%;
  position:absolute;
  top:0;
  left:0;
  backface-visibility:hidden;
}

for($i,$num_cartas_3D < $i,$i++){
  .carta#{$i}{
    background-image:url("../multimedia/carta0#{$i}.jpg");
    background-size:cover;
  }
}
```

Luego de tener la lógica se procede a llevarlo a SASS para ello tenemos dos opciones, la primera es usar `@function` ya que como su nombre ase referencia es para crear una función la otra opción es usar `@mixin`, pero ambas tienen diferencias. La primera retorna un valor, mientras `@mixin` solo llama a los valores en este caso no necesitamos que retorne un valor por ello usaremos `@mixin`. Entonces quedaría de la siguiente forma:

❖ Libro 3D

Figura 115. Creación del objeto `btn_hoja_navega` en función `mixin`

```
1 @mixin btn_hoja_navega{
2     background: white;
3     border: solid 3px;
4     cursor: pointer;
5     position:absolute;
6     bottom: 20px;
7 }
```

Figura 116. Creación del objeto `hojas_pag_libro` en función `mixin`

```
1 @mixin hojas_pag_libro{
2     width:100%;
3     height:100%;
4     position:absolute;
5     top:0;
6     left:0;
7 }
```

Figura 117. Creación del objeto `porta_and_tomo` en función `mixin`

```
1 @mixin porta_and_tomo{
2     width:250px;
3     height:400px;
4 }
```

Figura 118 Creación del objeto en un ciclo para la clase `tomo_libro_3d`

```
1 @for $i from 1 through $num_paginas {
2     #desplegar_hoja_#{ $i } :checked~ .tomo_libro_3d #pagina#{ $i }{
3         transform:rotateY(-180deg);
4         z-index:$i;
5     }
6 }
```

Figura 119. Creación del objeto en un ciclo para el identificador `pagina`

```
1 @for $i from 1 through $num_paginas {
2     #pagina#{ $i }{
3         z-index: $num_paginas;
4     }
5     $num_paginas:$num_paginas - 1;
6 }
```

❖ Carta 3D

Figura 120. Creación del objeto `dimencion_caja_and_tareas` en función `mixin`

```
1 @mixin dimencion_caja_and_atras{
2     display: flex;
3     justify-content: center;
4 }
```

Figura 121. Creación del objeto `adelante_and_tareas` en función `mixin`

```
1 @mixin adelante_and_atras{
2     width: 100%;
3     height:100%;
4     position:absolute;
5     top:0;
6     left:0;
7     backface-visibility: hidden;
8     -webkit-backface-visibility: hidden;
9 }
```

Figura 122. Creación del objeto en un ciclo para la clase `carta`

```
1 @for $i from 1 through $num_cartas_3D {
2     .carta#{$i}{
3         background-image: url("../multimedia/carta0#{$i}.jpg");
4         background-size: cover;
5     }
6 }
```

- **OOCSS**. – Luego de tener la función o la lógica para reciclar el código se procede a integrar la metodología OOCSS para ello debemos llamar al `mixin` para ello solo necesitamos agregar `@include` “nombre del `mixin`”. Luego de integrarlo ya estará aplicada correctamente la metodología OOCSS

❖ Libro 3D

Figura 123. Metodología OOCSS en el componente `libro 3d` parte I

```
1 .btn_sigui_hoja {
2     @include btn_hoja_navega;
3     float: right;
4     clear: both;
5     padding:$espacion_interno_botones;
6 }
```

Figura 124. Metodología OOCSS en el componente `libro 3d` parte II

```
1 .btn_atras_hoja {
2     @include btn_hoja_navega;
3     right:13px;
4     padding: $espacion_interno_botones;
5 }
```

Figura 125. Metodología OOCSS en el componente `libro 3d` parte III

```
1 #portada{
2     @include porta_and_tomo;
3 }
```


Figura 126. Metodología OOCSS en el componente libro 3d parte IV

```
1 .tomo_libro_3d{
2   @include porta_and_tomo;
3   position:relative;
4   perspective:$dimension_perspectiva_3d;
5 }
```

Figura 127. Metodología OOCSS en el componente libro 3d parte V

```
1 .pag_libro{
2   @include hojas_pag_libro;
3   transform-origin:left;
4   transform-style:preserve-3d;
5   transform:rotateY(0deg);
6   transition:.5s;
7   color:#000;
8 }
```

Figura 128. Metodología OOCSS en el componente libro 3d parte VI

```
1 .hoja_pos{
2   @include hojas_pag_libro;
3   box-sizing:border-box;
4   padding:13px;
5   box-shadow:inset 20px 0 50px rgba(0,0,0,0.5) 0 2px 5px rgba(0,0,0,.5);
6   background-color:white; $color_fondo_hoja_posterior:white;
7 }
```

Figura 129. Metodología OOCSS en el componente libro 3d parte VII

```
1 .hoja_an{
2   @include hojas_pag_libro;
3   transform:rotateY(180deg);
4   backface-visibility:hidden;
5   z-index:99;
6   background-color:white; $color_fondo_hoja_posterior:white;
7 }
```

❖ Carta 3D

Figura 130. Metodología OOCSS en el componente libro 3d parte VIII

```
1 .caja
2 {
3     @include dimencion_caja_and_atras;
4     width: 1100px;
5     margin: 100px auto;
6 }
```

Figura 131. Metodología OOCSS en el componente libro 3d parte IX

```
1  .adelante{
2      @include adelante_and_atras;
3  }
```

Figura 132. Metodología OOCSS en el componente libro 3d parte X

```
1  .atras{
2      @include adelante_and_atras;
3      @include dimencion_caja_and_atras;
4      transform: rotateY(180deg);
5      padding: 15px;
6      align-items: center;
7      text-align: center;
8      color:$color-blanco-1;
9  }
```

Bien volviendo al tema de las variables estas variables se encuentran en un archivo sass denominado variables valga la redundancia donde se encuentran las siguientes variables:

Figura 133. Variables Sass del componente libro 3d

```
1  /***** CARTA 3D && LIBRO 3D *****/
2  $num_paginas: 3;//Libro 3D
3  $espacion_interno_botones: 5px 10px;//Libro 3D
4  $tamano_letra_hoja_post:14px;//Libro 3D
5  $dimension_perspectiva_3d:1500px;//Libro 3D
6
7  $num_cartas_3D:3;//Carta 3D
8  $ancho_carta:300px;//Carta 3D
9  $alto_carta:350px;//Carta 3D
10 $color_fondo:#f29720;//Carta 3D
```

Como se ha visto SASS es un preprocesador de css por ello se ejecuta antes del css tradicional esto permite añadir variables agregar condicionales, etc. Con la aplicación de la metodología como se observa se reutiliza el código css, además de que se hace escalable el proyecto. Ya que, si en un futuro queremos cambiar o añadir más propiedades css, solo cambiaríamos las variables o añadiríamos más propiedades en el mixin.

FASE 6.- Documentar en la web site www.phooldx.com

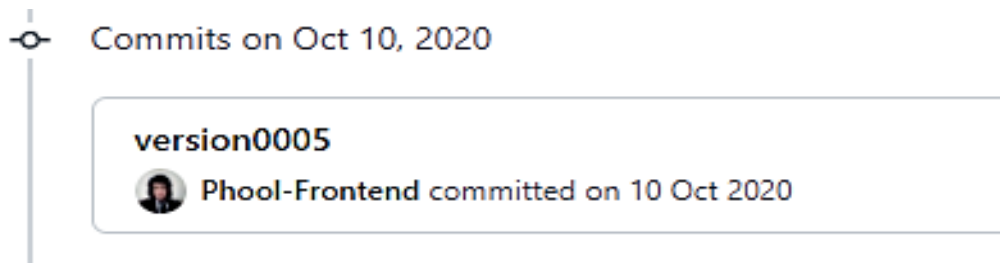
Finalmente, si el efecto paso por todos los de más filtros se procede a su documentación respectiva, para documentar todos los efectos se creó una página web de documentación usando clases y elementos del mismo framework. A continuación, se describirá como se procede a la documentación de la misma. Siguiendo estos pasos:

- Se copia las etiquetas de HTML5 junto con la clase a un txt temporal para no perder la sintaxis
- Se guarda los cambios en el compilador de SASS
- Como es la parte final se procedió hacer copias de seguridad para no perder el avance, en archivos RAR. Estos archivos estaban por versiones enumerados además se agregaba un pequeño texto describiendo el avance. Este guardado se hacía de manera local en un USB.
- Para una mayor seguridad del avance del código se procedió luego a copiar la versión del archivo RAR junto a la descripción del mismo.

Guardado de código en GitHub. - Luego de tener la copia del proyecto a nivel físico en un USB, se procedió a guardarlo en un repositorio de GitHub para lo cual se usó Git, para ello se procedió de la siguiente forma:

- Se guardaba todo el proyecto con el siguiente comando que se observa en la figura 12.
- Luego se verificaba los cambios guardados con el siguiente comando que lo podemos apreciar en la figura 13.
- Después se hacía el commit, es decir se les asignaba un nombre a los cambios realizados con el siguiente comando que se observa en la figura 14.
- Luego se procede a empaquetar los datos para que se suban al repositorio de GitHub con el siguiente código que lo podemos apreciar en la figura 15.
- Finalmente, luego de haber enviados el proyecto empaquetado hacia GitHub como se puede apreciar en la imagen, este queda guardado en dicho repositorio.

Figura 134. Visualización en github del componente libro 3d



Se guarda en una escala de tiempo, con todos los cambios hechos desde la creación del proyecto, esto ayuda a tener un mejor control de versiones. Asimismo, también ayuda a que más personas se sumen al proyecto y puedan clonar el proyecto como trabajar por separado el proyecto, entre otras muchas ventajas.

- Así mismo se manejó un archivo llamado bitácora.txt donde aquí se registró todos los incidentes ocurridos, así como los cambios hechos, se registraba también las metas cumplidas, también las tareas nuevas para desarrollar al día siguiente. Al finalizar con un efecto por completo se quitaba de la lista de prioridades así dábamos pase al siguiente efecto o problema a resolver. Esto servía para tener un mejor orden y saber ejecutar prioridades.
- Luego de tener guardado el avance y de registrar el avanzase, se procedió a copiar del archivo documentación Etiquetas.txt el fragmento de código que se copió anteriormente para luego pegarlo en www.nosetup.org esta página web se encargaba de convertir html en texto plano para poder visualizar html
- Antes de finalizar se procedió a crear un archivo llamado modulo_libro_3D.html donde se dividía en tres partes:
 - En la primera se describía un poco al efecto que era y que hacia
 - En la segunda parte se procedía a copiar el código ya formateado obtenido de www.nosetup.com para luego pegarlo.
 - En la tercera parte lo que se hacía era mostrar un apartado llamado estructura donde se describía como se llamaban las variables para que sea modificado en SASS.
- Finalmente, el ejemplo listo para usarse queda de la siguiente forma:

❖ Libro 3D

Figura 135. Código html5 final del componente libro 3d

```
<div class="book">
  <input type="checkbox" id="c1">
  <input type="checkbox" id="c2">
  <input type="checkbox" id="c3">
  <div id="cover"></div>
  <div class="flip-book">
    <div class="flip" id="p1">
      <div class="back">
        
        <label class="back-btn" for="c1" style="color:black;">Back</label>
      </div>

      <div class="front">
        <h2>Manzana</h2>
        <p><La manzana es el fruto comestible de la especie Malus domestica, el manzano común. Es una fruta pomácea de forma redonda y sabor muy dulce, dependiendo de la variedad. Los manzanos se cultivan en todo el mundo y son las especies más utilizadas del género Malus.</p>
        <label class="next-btn" for="c1">Next</label>
      </div>
    </div>

    <div class="flip" id="p2">
      <div class="back">
        
        <label class="back-btn" for="c2" style="color:black;">Back</label>
      </div>

      <div class="front">
        <h2>Fresa</h2>
        <p><Fragaria, llamado comúnmente fresa o frutilla, es un género de plantas rastreras estoloníferas de la familia Rosaceae. Agrupa unos 400 taxones descritos, de los cuales solo Son cultivadas por su fruto comestible llamado de la misma manera, fresa o frutilla.</p>
        <label class="next-btn" for="c2">Next</label>
      </div>
    </div>

    <div class="flip" id="p3">
      <div class="back">
        <label class="back-btn" for="c3" style="color:white;">Back</label>
      </div>

      <div class="front">
        <h2>Piña</h2>
        <p><Ananas comosus es una especie de la familia de las bromeliáceas, nativa de América del Sur. Planta de escaso porte y con hojas duras y lanceoladas de hasta 1 m de largo, fructifica una vez al año produciendo un único fruto fragante y dulce, muy apreciado en gastronomía.</p>
        <label class="next-btn" for="c3">Next</label>
      </div>
    </div>
  </div>
</div>
```

❖ Carta 3D

Figura 136. Código html5 final del submodulo carta 3d

```
<div class="caja">
  <div class="tarjeta-caja">
    <div class="tarjeta">
      <div class="adelante carta2"><h1>Carta N°2</h1></div>
      <div class="atras">
        <p>Hola p mano estoy sin estilos</p>
      </div>
    </div>
  </div>
</div>
```

PRE - CARGA

FASE 1.- Investigación del efecto y su importancia

Como primer paso lo primero que se hizo fue investigar e indagar sobre el efecto, mencionando un poco de lo que trata el efecto, este es usado para mostrar al usuario mientras la petición de datos al servidor se está demorando y en vez de mostrar un error o en el peor de los casos no mostrar nada una alternativa es mostrar una precarga o en inglés también llamado un preloader. El beneficio de usar este precarga es transmitirle de forma subjetiva al usuario de que su petición o la respuesta de este tardara unos segundos. El motivo por el cual fue elegido para entrar al framework XRL8 es que normalmente los desarrolladores front-end tienen que buscar librerías externas para poder aplicarlos, así mismo entre los desarrolladores novatos se les complica el implementar librerías externas ya que cada librería trae sus propios prerequisites, lo que se busca en XRL8 es que puedan encontrar todo en el mismo framework sin tener que buscar en librerías extras.

Otro del motivo el cual fue elegido el efecto para formar parte del framework XRL8 fue que el proyecto presente debe estar a la altura de los frameworks que están en la actualidad siendo usados en la industria del desarrollo web como son:

- **Bootstrap.** - El presente framework cuenta dentro de la sección de componentes con la sección de spinners en ella se puede encontrar animaciones de precarga las cuales se puede usar en botones, texto, si desean saber más sobre ello pueden visitar la web oficial de documentación de dicho framework mediante el siguiente enlace:

<https://getbootstrap.com/docs/4.2/components/spinners/>

- **Bulma.**- Este framework también cuenta con una sección de preloader, estas están limitadas solo a los botones, se accede mediante la clase is-loading, para ver cómo se emplea y formas de uso revisar el siguiente enlace que los llevara a la documentación oficial de dicho framework:

<https://bulma.io/documentation/elements/button/#states>

- **UIKIT.-** También este framework cuenta con una clase para hacer animación de preloader, pero carece de variedad solo tienen un diseño, la clase empleada en este caso tiene por nombre .uk-spinner para ver más sobre este framework recomiendo revisar la documentación en el siguiente enlace:

<https://getuikit.com/docs/spinner>

- **Semantic UI.-** A continuación tenemos otro framework de CSS muy completo la verdad el cual tiene soporte y componentes para ser usado con React, se encontró que tiene una clase denominada ui loading button la cual crea efecto de precarga a nivel de botones, podemos ver varios ejemplos mediante el siguiente enlace:

<https://semantic-ui.com/elements/button.html>

Modulo: Un módulo según definición propia es una sección de trabajo principal a comparación de un submódulo este puede ser denominado la estructura padre y los submódulos estructuras hijas. Cabe aclarar que en esta sección en específico no hubo submódulos ya que estos son módulos independientes por la misma razón que no comparten similitudes o propiedades. Así que son cuatro módulos que pertenecen a la misma sección llamada precarga o en inglés preloaders. Y esos módulos están expresados en las siguientes clases:

- **Pre carga 1.** - Para este efecto nos basamos en una bolita pintoresca que rebota sobre su propio eje acompañando al usuario en su espera.
- **Pre carga 2.-** Este efecto tiene la particularidad que está representada en forma de ondas las cuales están impacientes a la espera de respuesta al igual que el usuario.
- **Pre carga 3.-** Un efecto un poco más clásico se quiso presentar como propuesta y el resultado fue bolitas que se mueven en forma de onda, un poco más ligera e ideal para todo público.
- **Pre carga 4.-** Este módulo visualmente se observa una bola que observe a la otra indeterminadamente. Ya que se quiso variar un poco la dinámica de solo moverse como en los módulos anteriores.

FASE 2.- Codificar el efecto

La parte de codificación fue poca, lo realmente difícil fue aprender a usar @keyframes los cuales fueron usados en la parte de la animación, así como la secuencias que deberían de seguir y el posicionamiento de los elementos para que se mostrasen, reemplazando a otros elementos que estuvieran antes que el efecto de precarga.

Bien luego de tener claro que efecto queríamos construir y porque lo íbamos a construir como se vio en el capítulo anterior se procedió a escribir código como se muestra a continuación.

Código completo

❖ Código del módulo precarga 1

[parte 01 – precarga 1]

Figura 137. Código de pre-carga parte I

```
1  /***** Pre-carga1 Bolita *****/
2  .pre-carga1 {
3    width: 40px;
4    height: 40px;
5    position: absolute;
6    border-radius: 50%;
7    background: transparent;
8    top: 50%;
9    left: 50%;
10   margin-top: 80px;
11   transform: translate(-50%, -50%);
12   z-index: 1;
13 }
14
15 .pre-carga1:before {
16   content: "";
17   width: 100%;
18   height: 100%;
19   position: absolute;
20   border-radius: 50%;
21   background: #188FA7;
22   box-shadow: inset 0 -10px rgba(0, 0, 0, 0.1);
23   animation: pre-carga1 0.75s infinite linear;
24   z-index: 10;
25 }
```


[parte 02 – precarga 1]

Figura 138. Código de pre-carga parte II

```
1 .pre-carga1:after {
2   width: 100%;
3   height: 100%;
4   position: absolute;
5   border-radius: 50%;
6   content: "";
7   background: rgba(0, 0, 0, 0.1);
8   transform: scale(1, 0.25);
9   animation: pre-carga1shadow 0.75s infinite linear;
10 }
11
12 @keyframes pre-carga1 {
13   0% {
14     transform: translate(0, -50%) scale(1, 0.8);
15   }
16   30% {
17     transform: translate(0, -250%) scale(0.8, 1);
18   }
19   60% {
20     transform: translate(0, -300%) scale(1, 0.8);
21   }
22   85% {
23     transform: translate(0, -50%) scale(1, 0.8);
24   }
25   88% {
26     transform: translate(0, -50%) scale(1, 0.8);
27   }
28   95%, 100% {
29     transform: translate(0, -50%) scale(1, 0.8);
30   }
31 }
```

[parte 03 – precarga 1]

Figura 139. Código de pre-carga parte III

```
1 @keyframes pre-carga1shadow {
2   0% {
3     transform: scale(1, 0.25);
4   }
5   30% {
6     transform: scale(1.5, 0.25);
7   }
8   60% {
9     transform: scale(2, 0.25);
10  }
11  85% {
12    transform: scale(1, 0.25);
13  }
14  90% {
15    transform: scale(1, 0.25);
16  }
17  100% {
18    transform: scale(1, 0.25);
19  }
20 }
```

❖ Código del módulo precarga 2

[parte 01 – preloader 2]

Figura 140. Código de pre-carga parte IV

```
1  /***** Pre-carga2 BarraMusic *****/
2  .pre-carga2 {
3    display: flex;
4    align-items: center;
5    height: 40px;
6    position: absolute;
7    top: 50%;
8    left: 50%;
9    transform: translate(-50%, -50%);
10   z-index: 1;
11 }
12
13 .obj {
14   display: flex;
15   align-items: center;
16   height: 40px;
17   width: 6px;
18   background: #2980b9;
19   margin: 0px 3px;
20   border-radius: 10px;
21   animation: pre-carga2 0.8s infinite;
22 }
23
24 @keyframes pre-carga2 {
25   0% {
26     height: 0;
27   }
28   50% {
29     height: 40px;
30   }
31   100% {
32     height: 0;
33   }
34 }
```

❖ Código del módulo precarga 3

[parte 01 – preloader 3]

Figura 141. Código de pre-carga parte V

```
1  /***** Pre-carga3 HolaBolita *****/
2  .pre-carga3 {
3    position: absolute;
4    z-index: 1;
5    display: flex;
6    justify-content: center;
7    align-items: center;
8    top: 50%;
9    left: 50%;
10 }
11
12 .pre-carga3 span {
13   position: absolute;
14   z-index: 1;
15   display: flex;
16   justify-content: center;
17   align-items: center;
18   width: 1em;
19   height: 1em;
20   background: white;
21   left: 0;
22   top: 0;
23   border-radius: 50%;
24   animation: pre-carga3 2s ease-in-out infinite;
25 }
26
27 .pre-carga3 span:nth-child(1) {
28   left: -2em;
29   animation-delay: 0s;
30 }
```

[parte 02 – preloader 3]

Figura 142. Código de pre-carga parte VI

```
1  .pre-carga3 span:nth-child(2) {
2    left: -0.5em;
3    animation-delay: 0.1s;
4  }
5
6  .pre-carga3 span:nth-child(3) {
7    left: 1em;
8    animation-delay: 0.2s;
9  }
10
11 .pre-carga3 span:nth-child(4) {
12   left: 2.5em;
13   animation-delay: 0.3s;
14 }
15
16 @keyframes pre-carga3 {
17   0%, 75%, 100% {
18     transform: translateY(0) scale(1);
19   }
20   25% {
21     transform: translateY(2.5em);
22   }
23   50% {
24     transform: translateY(-2.5em) scale(1.1);
25   }
26 }
```

❖ **Código del módulo precarga 4**

[parte 01 – preloader 4]

Figura 143. Código de pre-carga parte VII

```
1  /***** PRE-CARGA4 BolitaAbsorvida *****/
2  .pre-carga4 {
3    width: 80vw;
4    height: -webkit-fill-available;
5    position: absolute;
6    top: 50%;
7    left: 50%;
8    transform: translate(-50%, -50%);
9    background: white;
10   filter: blur(10px) contrast(20);
11   z-index: 1;
12 }
13
14 .bola-1 {
15   width: 70px;
16   height: 70px;
17   background: black;
18   border-radius: 50%;
19   position: absolute;
20   top: 50%;
21   left: 50%;
22   transform: translate(-50%, -50%);
23   left: 20%;
24   animation: obsorver 2.5s ease infinite;
25 }
```

[parte 02 – preloader 4]

Figura 144. Código de pre-carga parte VIII

```
1  .bola-2 {
2    width: 70px;
3    height: 70px;
4    background: black;
5    border-radius: 50%;
6    position: absolute;
7    top: 50%;
8    left: 50%;
9    transform: translate(-50%, -50%);
10   left: 80%;
11   animation: envolver 2.5s ease infinite;
12   background: #0ff;
13 }
14
15 @keyframes obsorver {
16   0% {
17     left: 20%;
18   }
19   50% {
20     left: 50%;
21   }
22   100% {
23     left: 20%;
24   }
25 }
```

[parte 03 – preloader 4]

Figura 145. Código de pre-carga parte IX

```
1  @keyframes envolver {
2    0% {
3      left: 80%;
4    }
5    50% {
6      left: 50%;
7    }
8    100% {
9      left: 80%;
10   }
11 }
12
13 @media only screen and (min-width: 421px) {
14   .pre-carga4 {
15     width: 24rem;
16   }
17 }
```

Descripción de las propiedades más significativas:

A continuación, se procederá a detallar las propiedades más usadas y su aplicación. También se mostrará la compatibilidad web de cada propiedad para su mejor comprensión dejamos una leyenda en la cual se indica si tiene compatibilidad web esto lo podemos visualizar en la figura 2.

Propiedad width:40px

Se empleó width 40px para el ancho de este efecto ya que si bien esta propiedad tiene diversas unidades de medidas como:

- EM
- VH
- PÍXELES
- REM
- PORCENTAJE

Se fue descartando una a una al momento de probarla, ya que al usar la unidad EM este se distorsionaba por su propia naturaleza, básicamente esta propiedad usa el ancho de la pantalla del dispositivo donde estemos y las va ajustando así que fue descartado también fueron descartados la propiedad vh ya que este usa el alto de la pantalla y estos valores pueden variar así que la decisión era usar porcentaje o pixeles. Al final se decidió usar pixeles ya que se necesitaba un valor fijo como estático lo cual no se conseguía con el valor en porcentaje.

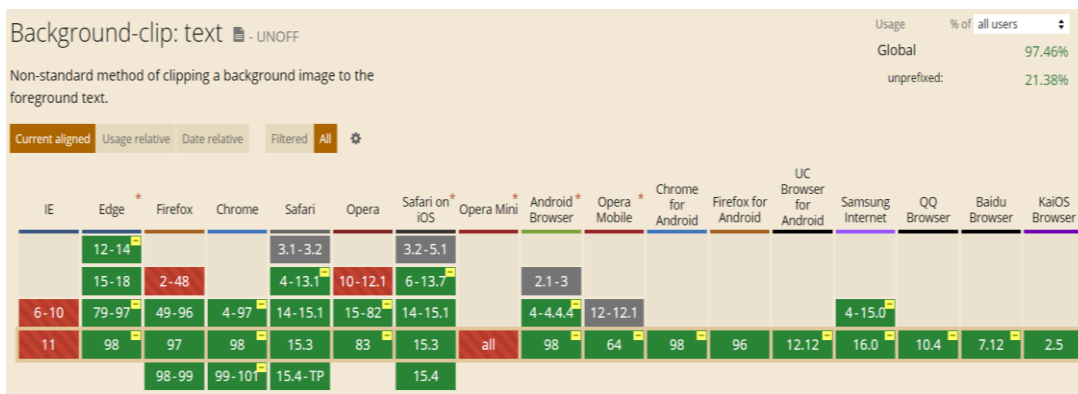
A continuación, veamos la compatibilidad web al usar esta propiedad lo podemos observar en la figura 3.

Propiedad background:transparent

Esta propiedad se usa para darle transparencia al fondo o clase sobre la cual se aplica. De forma predeterminada, el color de fondo será transparente, lo que significa que no existirá un color de fondo.

A continuación, veamos la compatibilidad web al usar esta propiedad en los principales navegadores.

Figura 146. Compatibilidad web, propiedad background

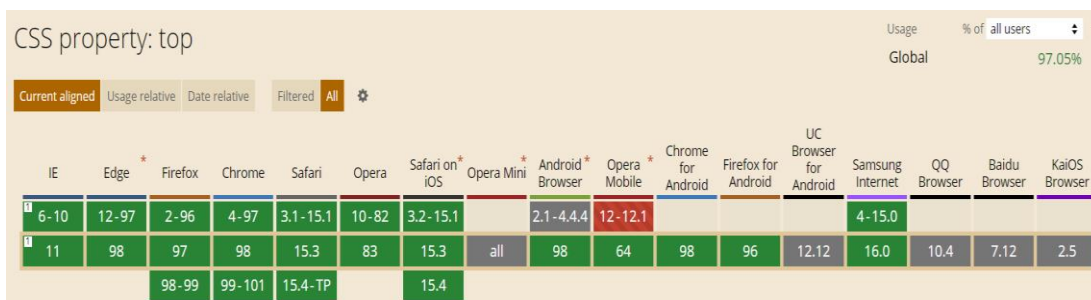


Propiedad top:50%

Con esta propiedad lo que se busca es centrar hacia arriba en un 50% junto a este elemento se puede combinar con la propiedad left:50% para un centrado más preciso, si bien existe muchas formas de centrar, se optó de esta forma para no distorsionar el efecto.

A continuación, veamos la compatibilidad web al usar esta propiedad en los principales navegadores.

Figura 147. Compatibilidad web, propiedad top



Propiedad left:50%

Esta propiedad se usó para no crear conflicto interno con la animación hecha con @keyframes además cumple con su propósito el cual es centrar un elemento, con la ayuda de la propiedad top:50% se logró, además se optó por usar porcentaje para que sea responsive y no se quede estático en un único estilo de pantalla.

A continuación, veamos la compatibilidad web al usar esta propiedad en los principales navegadores.

Figura 148. Compatibilidad web, propiedad left

CSS property: left														Usage	% of all users	
														Global	97.41%	
IE	Edge	Firefox	Chrome	Safari	Opera	Safari on iOS	Opera Mini	Android	Opera Mobile	Chrome for Android	Firefox for Android	UC Browser for Android	Samsung Internet	QQ Browser	Baidu Browser	KaiOS Browser
6-10	12-97	2-96	4-97	3.1-15.1	10-82	3.2-15.1		2.1-4.4.4	12-12.1				4-15.0			
11	98	97	98	15.3	83	15.3	all	98	64	98	96	12.12	16.0	10.4	7.12	2.5
		98-99	99-101	15.4-TP		15.4										

Propiedad content:””

Para usar esta propiedad es requisito primordial usar un pseudo-elemento como:

- Before: Lo que hace esta propiedad es insertar elementos antes de la clase.
- After: Lo que hace esta propiedad es insertar elementos después de la clase.

Estos pseudo-elementos son usados para añadir contenido a un elemento con la propiedad content, aunque también en content puede ir palabras y estas se insertaran.

A continuación, veamos la compatibilidad web al usar esta propiedad en los principales navegadores.

Figura 149. Compatibilidad web, propiedad content

CSS property: content														Usage	% of all users	
														Global	97.39%	
IE	Edge	Firefox	Chrome	Safari	Opera	Safari on iOS	Opera Mini	Android	Opera Mobile	Chrome for Android	Firefox for Android	UC Browser for Android	Samsung Internet	QQ Browser	Baidu Browser	KaiOS Browser
6-7																
8-10	12-97	2-96	4-97	3.1-15.1	10-82	3.2-15.1		2.1-4.4.4	12-12.1				4-15.0			
11	98	97	98	15.3	83	15.3	all	98	64	98	96	12.12	16.0	10.4	7.12	2.5
		98-99	99-101	15.4-TP		15.4										

Propiedad box-shadow:inset 0 -10 rgba(0,0,0,0.1)

Esta propiedad tiene la función de agregar sombras alrededor del elemento, se puede definir varios valores, los primeros valores son 0 y -10 son la posición en la cual estará la sombra, el primer valor es para arriba y abajo en este caso es 0 y el -10 es para la izquierda y derecha. Luego tenemos que

colocar el valor del color en este caso en particular está en `rgba(0,0,0,0.1)` el cual es una sombra ligeramente débil una sombra en pocas palabras. La palabra reservada `inset` hace referencia a que el valor externo lo tomara como un valor interno a la sombra.

A continuación, veamos la compatibilidad web al usar esta propiedad en la figura 49 podemos verlo más detalladamente.

Propiedad `transform:scale(1,0.25)`

Para poder usar esta propiedad es necesario usar un `@keyframes`. El valor `scale` indica la expansión de la transformación en 2D recibe asimismo dos parámetros los cuales será X y Y.

A continuación, veamos la compatibilidad web al usar esta propiedad en los principales navegadores, se puede apreciar en la figura 101.

Propiedad `transform:translate(-50%,-50%)`

Aquí se define la traslación del elemento en 2D, en el paréntesis admite dos valores el cual será el movimiento que se generará en el plano cartesiano el primer elemento se moverá en el eje de las abscisas, así mismo se moverá el segundo elemento en el eje de las ordenadas todo en el plano (X,Y) .En este caso particular se optó por usar valores en porcentajes ya que estos son más efectivos para cualquier tipo de pantalla.

A continuación, veamos la compatibilidad web al usar esta propiedad en los principales navegadores en la figura 101 se observa detalladamente lo mencionado.

Propiedad `transform:translate(0,-50%) scale (1 0.8)`

Como se observó en la propiedad anterior esta también comparte los mismos atributos lo único nuevo es el valor `scale` dicho valor admite dos valores los cuales representan la expansión del elemento en 2D.

A continuación, veamos la compatibilidad web al usar esta propiedad en los principales navegadores en la figura 101 se observa detalladamente lo mencionado.

Propiedad transform:translateY(-2.5em) scale (1.1)

En esta propiedad es similar a la anterior, la diferencia radica en que solo recibirá un solo valor en el eje Y, posteriormente como ya se mencionó scale revive el valor de la expansión de la transformación.

A continuación, veamos la compatibilidad web al usar esta propiedad en los principales navegadores en la figura 101 se observa detalladamente lo mencionado.

Propiedad left:-2em

Esta propiedad lo que hace es mover el elemento al lado izquierdo ya que como se observa tiene un valor negativo, ahora el valor que tiene es em se eligió ese valor debido a que em es un múltiplo del tamaño de fuente del elemento también conocido como font-size.

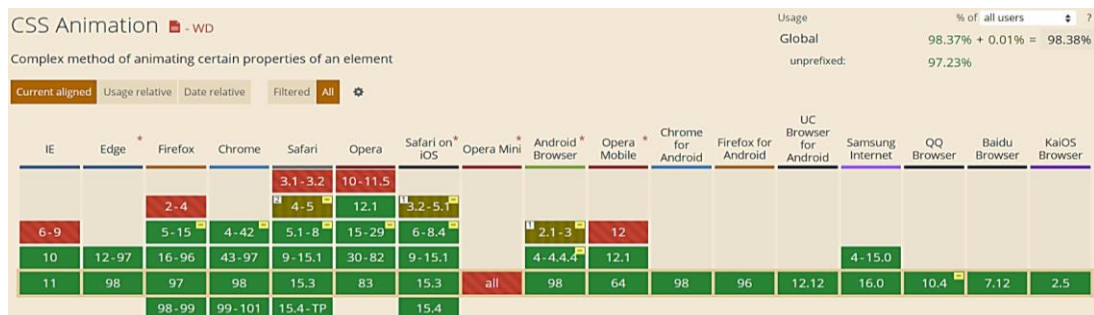
A continuación, veamos la compatibilidad web al usar esta propiedad en los principales navegadores en la figura 143 se observa detalladamente lo mencionado.

Propiedad animation:pre-carga2 0.8s infinite

En esta propiedad es un requisito tener en un @keyframes ya que es el primer valor que nos pedirá, luego como segundo valor tenemos el tiempo de duración expresado en segundos para finalmente colocar el tiempo de duración del efecto el cual será expresado en segundos si no se le coloca por default será uno, también podemos colocar la palabra reservada infinite lo cual indica que la duración del efecto es infinita.

A continuación, veamos la compatibilidad web al usar esta propiedad en los principales navegadores.

Figura 150. Compatibilidad web, propiedad animation

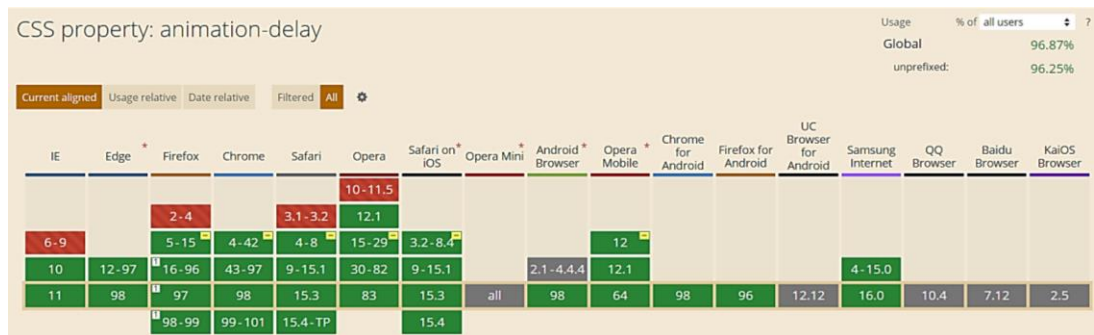


Propiedad animation-delay:0s

La función de esta propiedad es sencillamente retrasar las animaciones que tengan la clase, recibirá como parámetro un numero entero el cual indicara el retraso en segundos.

A continuación, vemos la compatibilidad web al usar esta propiedad en los principales navegadores.

Figura 151. Compatibilidad web, propiedad animation-delay



Propiedad filter:blur(10px) contrast(20)

La propiedad filter hace un desenfocado y satura los efectos visuales para un elemento, el valor blur es un desenfocado gaussiano hace que se mezclen los pixeles recibe como parámetro el valor o la cantidad a mezclar además no se acepta porcentaje. El segundo valor que se observa es el contrast el cual añade una tonalidad gris, el parámetro mínimo que se acepta es 1 así mismo este es el valor por default mientras más alto sea el valor de dicho parámetro será mayor el contraste como la iluminación en esta.

A continuación, vemos la compatibilidad web al usar esta propiedad en los principales navegadores.

Figura 152. Compatibilidad web, propiedad filter



Propiedad animation:absorber 2.5s ease infinite

Para poder usar esta propiedad es necesario usar un @keyframes ya que el primer parámetro es el nombre asignado al keyframe, seguido tenemos al valor en segundos, como tercer parámetro tenemos el tipo de animación según las cubicas de Bezier el cual no es más que la trayectoria en la curva que correrá la animación, esta puede ser configurable en esta propiedad en particular se eligió ease la cual especifica por default que la transición CSS comenzara lentamente para luego ir más rápido y terminara reduciendo su velocidad la equivalencia de esto es cubic Bezier (0.25,0.1,0.25,1).

A continuación, veamos la compatibilidad web al usar esta propiedad en los principales navegadores en la figura 145 se observa detalladamente lo mencionado.

Principales dificultades:

Para este efecto en concreto la principal dificultad fue organizar de distintas formas posibles, así como ver las posibles conjugaciones que pueden hacer con el efecto para que el usuario al final no tenga problemas al implementarlo. Esas dificultades surgieron a raíz de las siguientes preguntas:

- ¿Qué pasa si el sitio web o aplicación web tiene propiedades css para superponerse al efecto de precarga?

Efectivamente al momento de probarlo en sitios web, este no se superponía existía otros elementos que se anteponían al efecto de precarga, para ello en primera instancia se usó posicionamiento absoluto, luego tras las pruebas tampoco dieron resultados para ello investigando entre las muchas propiedades css se encontró la propiedad z-index la cual da énfasis a la clase que tenga el mayor valor dentro de sus atributos.

- ¿Cómo hacemos para detener la animación de un efecto de precarga?

Otra pregunta que surgió también fue como detener al efecto de precarga ya que estos son infinitos, en este punto se llegó a la conclusión que se puede implementarlo según las necesidades de los usuarios, por lo general los desarrolladores web se encargan de determinar el tiempo de duración de dicho efecto de precarga.

¿Qué usamos para hacer las pruebas?

Se usó el navegador Google Chrome y opera mini, así como la siguiente página web que es un simulador online:

<http://www.responsinator.com>

Esto con motivos de asegurarnos que funcione en dispositivos móviles y tabletas. Ya que en base a lo trabajado no es del todo fiable usar el navegador para verificar la correcta maquetación de los elementos, así como tampoco es fiable depender de un software web, por ello se decidió diversificar en cuanto a las herramientas para probar las clases y efectos trabajados en el framework XRL8.

También usamos como editor de código visual studio code y su extensión live server para visualizar en vivo los efectos realizados en el css en tiempo real, también usamos el autoguardado que trae dicho editor de código para no estar guardando el proyecto a cada rato. Así conseguimos un mejor desempeño y evitamos el tedioso guardar cambios a cada rato, también evitamos el que la memoria cache se llene mostrándonos los mismos resultados.

Adicionalmente se usó plantillas web para implementar el efecto de precarga en el lado del front-end, lo que se procedió hacer fue lo siguiente:

- Se crearon los archivos respectivos, así como el enlace respectivo para usar la plantilla HTML adicionalmente se usó jquery para usar Ajax.
- Se creó una etiqueta div en en la plantilla HTML, luego se llamó al efecto preloader y se inyecto código html mediante jquery usando el método html() de jquery.
- En el archivo JavaScript se creó una función setTimeout en el cual se simula el tiempo de espera como el tiempo de respuesta por parte del servidor al traer la data respectiva para mostrar al usuario. En dicha función se le indicaba que quitara la clase de carga pasado los 6 segundos, con esto se demostró que no se necesita una clase extra para paralizar al efecto de precarga. Asimismo, sirvió para corregir los problemas que surgía al usar el efecto de precarga y se repitió el proceso para los cuatro efectos de precarga.

FASE 3.- Hacerlo responsive

Con este efecto en particular hubo circunstancias desfavorables ya que necesita un valor estático y fijo para su correcto funcionamiento, así que una forma de hacerlo responsive y funcional al mismo tiempo fue reducir el umbral de trabajo esto quiere decir se les asignó al ancho del efecto en si valores mínimos que soportara desde el dispositivo de pantalla más pequeño y agrandar el margen del umbral mínimo permitido.

Para los módulos de precarga en el cual se pudo hacer responsive se aplicó las siguientes buenas prácticas:

- Se usó la propiedad flex-box y sus posibles conjugaciones con cada uno de sus valores según sea el caso.
- Se usó las unidades de medida vh o rem para texto y contenedores ya que estas son imprescindibles a la hora de hacer responsive un sitio web.
- Se trabajó con la propiedad @media_query en las resoluciones de pantalla para celulares y tabletas. Siempre empezando a maquetar por la parte de celulares y luego tabletas.

Figura 153. Uso de media query en el componente pre carga

```
13 @media only screen and (min-width: 421px) {
14   .pre-carga4 {
15     width: 24rem;
16   }
17 }
```

FASE 4.- Testing asegurarse que no tenga conflicto con otros efectos.

En esta parte del desarrollo de la clase se dispuso a probar poniéndolo con las demás clases juntándolas. Para ello lo que se hizo fue colocar dentro de un archivo index.html todas las clases trabajadas y ver como el efecto de precarga respondió. En algunos casos había conflicto con los demás efectos para corregir esto se procedió a implementar en el desarrollo dos cosas básicas la primera fue:

- Para detectar que clase era la que causaba el conflicto y ver en que archivo scss estaba. Se procedía a inspeccionar la página para detectar donde estaba el problema. En muchas ocasiones no se daba con el archivo causante del error entonces en ese caso aplicábamos la frase “Divide y vencerás”. Lo que se procedió hacer fue comentar de uno en uno cada efecto en el archivo principal que los llamaba esto reducía las líneas de código de css, además como teníamos en tiempo real los cambios fue mucho más rápido detectar el problema. Ya que no se tenía que recargar a cada rato además otro punto a favor era que se auto compilaba los cambios de sass automáticamente gracias a nodeJS el cual mediante la consola de comandos de Windows cuando detectaba un cambio en las líneas de código se auto compilaba y generando o reemplazando el archivo css en el cual hacíamos las pruebas, esto aumento la productividad para crear los efectos ya que al comentar de uno en uno cada archivo sass llegaba un momento donde desaparecía el error entonces revisábamos y nos dimos cuenta de que era uno o en algunos casos hasta tres efectos que hacían conflicto con la clase escrita.
- Se procedió a escribir mediante comentarios en un archivo llamado `_error_logs.scss` la clase que hace conflicto con la clase escrita, además de describir una nota breve en caso no se disponía de tiempo para solucionarlo luego, esto permitió tener un reporte más preciso de los incidentes que íbamos detectando.
- Como se pudo observar en la sección de pruebas uno de los factores era que en la mayoría de casos los demás efectos le tapaban, dicho de otra forma, lo ocultaban para ello se puso un z-index con esto se solucionó el problema.

FASE 5. Aplicar la metodología OOCSS, llevándolo a SASS

- **Código css.** - Esta parte ya estaría completa por lo descrito anteriormente.

- **Código repetido.** - Identificación de código repetido

En este pequeño apartado luego de escribir y debugear el error se procedió a identificar el código css repetido para llevarlo a un archivo SCSS en sass, para su mayor optimización y aplicación de la metodología. En la clase descrita a continuación se observó que se repetía el siguiente código.

❖ Precarga 1

Figura 154. Código repetido del componente pre-carga parte I

```
1 .pre-carga1 {
2   width: 40px;
3   height: 40px;
4   position: absolute;
5   border-radius: 50%;
6   background: transparent;
7   top: 50%;
8   left: 50%;
9   margin-top: 80px;
10  transform: translate(-50%, -50%);
11  z-index: 1;
12 }
```

Figura 155. Código repetido del componente pre-carga parte II

```
1 .pre-carga1:before {
2   content: "";
3   width: 100%;
4   height: 100%;
5   position: absolute;
6   border-radius: 50%;
7   background: #188FA7;
8   box-shadow: inset 0 -10px rgba(0, 0, 0, 0.1);
9   animation: pre-carga1 0.75s infinite linear;
10  z-index: 10;
11 }
```

Figura 156. Código repetido del componente pre-carga parte III

```
1 .pre-carga1:after {
2   width: 100%;
3   height: 100%;
4   position: absolute;
5   border-radius: 50%;
6   content: "";
7   background: rgba(0, 0, 0, 0.1);
8   transform: scale(1, 0.25);
9   animation: pre-carga1shadow 0.75s infinite linear;
10 }
```

❖ Precarga 2

Figura 157. Código repetido del componente pre-carga parte IV

```
1 .pre-carga2 {
2   display: flex;
3   align-items: center;
4   height: 40px;
5   position: absolute;
6   top: 50%;
7   left: 50%;
8   transform: translate(-50%, -50%);
9   z-index: 1;
10 }
```

Figura 158. Código repetido del componente pre-carga parte V

```
1 .obj {
2   display: flex;
3   align-items: center;
4   height: 40px;
5   width: 6px;
6   background: #2980b9;
7   margin: 0px 3px;
8   border-radius: 10px;
9   animation: pre-carga2 0.8s infinite;
10 }
```

❖ Precarga 3

Figura 159. Código repetido del componente pre-carga parte VI

```
1 .pre-carga3 {
2   position: absolute;
3   z-index: 1;
4   display: flex;
5   justify-content: center;
6   align-items: center;
7   top: 50%;
8   left: 50%;
9 }
```

Figura 160. Código repetido del componente pre-carga parte VII

```
1 .pre-carga3 span {
2   position: absolute;
3   z-index: 1;
4   display: flex;
5   justify-content: center;
6   align-items: center;
7   width: 1em;
8   height: 1em;
9   background: white;
10  left: 0;
11  top: 0;
12  border-radius: 50%;
13  animation: pre-carga3 2s ease-in-out infinite;
14 }
```


❖ Precarga 4

Figura 161. Código repetido del componente pre-carga parte VIII

```
1  .pre-carga4 {
2    width: 80vw;
3    height: -webkit-fill-available;
4    position: absolute;
5    top: 50%;
6    left: 50%;
7    transform: translate(-50%, -50%);
8    background: white;
9    filter: blur(10px) contrast(20);
10   z-index: 1;
11 }
```

Figura 162. Código repetido del componente pre-carga parte IX

```
1  .bola-1 {
2    width: 70px;
3    height: 70px;
4    background: black;
5    border-radius: 50%;
6    position: absolute;
7    top: 50%;
8    left: 50%;
9    transform: translate(-50%, -50%);
10   left: 20%;
11   animation: obsorver 2.5s ease infinite;
12 }
```

Figura 163. Código repetido del componente pre-carga parte X

```
1  .bola-2 {
2    width: 70px;
3    height: 70px;
4    background: black;
5    border-radius: 50%;
6    position: absolute;
7    top: 50%;
8    left: 50%;
9    transform: translate(-50%, -50%);
10   left: 80%;
11   animation: envolver 2.5s ease infinite;
12   background: #0ff;
13 }
```

- **Crear un objeto.** - Se describe la lógica de la función para luego crear un mixing.

Una vez identificado el código css repetido se procede hacer una función la cual se ejecuta al llamado del código, mediante esta técnica nuestro código será más puro y limpio.

Paralelamente en esta parte se determina la lógica de las funciones y ciclos como lo son for, while, each. También determinamos el pase de parámetros, así como la condicional que llevaran dentro, por otra parte, y no menos importante se define el nombre con el cual se llamara a las funciones.

❖ Precarga 1

```
function precarga1_be_af(){
  if($condicion == 'dimension_bola'){
    width:$pre_carga_ancho_bolita;
    height:$pre_carga_alto_bolita;
  }else{
    width:100%;
    height:100%;
  }
  position:absolute;
  border-radius:50%;
}
```

❖ Precarga 2

```
function pre_carga_base(){
  display:flex;
  align-items:center;
  height:40px;
}

@for($i=0;$i<$pre_carga_2_num_barras;$i++){
  .obj:nth-child($i+1){
    Animation-delay:($i/10)+s;
  }
}
```

❖ Precarga 3

```
function pre_carga_3_base(){
  position:absolute;
  z-index:$pre_carga_3_visibilidad;
  display:flex;
  justify-content:center;
  align-items:center;
}
```

❖ Precarga 4

```
function pre_carga_centrado(){
  position:absolute;
  top:50%;
  left:50%;
  transform:translate(-50%,-50%);
}
```

```
function bola_1_and_bola_2(){
    width:$pre_carga_4_dimencion_bolas;
    height:$pre_carga_4_dimencion_bolas;
    background:$color-negro-1;
    border-radius:$pre_carga_4_borde_redondeado;
}
```

Luego de tener la lógica se procede a llevarlo a SASS para ello tenemos dos opciones, la primera es usar `@function` ya que como su nombre ase referencia es para crear una función la otra opción es usar `@mixin`, pero ambas tienen diferencias. La primera retorna un valor, mientras `@mixin` solo llama a los valores en este caso no necesitamos que retorne un valor por ello usaremos `@mixin`. Entonces quedaría de la siguiente forma:

❖ Precarga 1

Figura 164. Creación del objeto `pre_carga1_be_af` en función `mixin`

```
1 @mixin pre_carga1_be_af($condicion){
2     @if $condicion == 'dimension_bola'{
3         width:$pre_carga_ancho_bolita;
4         height:$pre_carga_alto_bolita;
5     }@else{
6         width: 100%;
7         height:100%;
8     }
9     position: absolute;
10    border-radius: 50%;
11 }
```

❖ Precarga 2

Figura 165. Creación del objeto `pre_carga_base` en función `mixin`

```
1 @mixin pre_carga_base{
2     display: flex;
3     align-items: center;
4     height:40px;
5 }
```

Figura 166. Creación del objeto en un ciclo para la clase `obj`

```
1 @for $i from 1 through $pre_carga_2_num_barras {
2     .obj:nth-child(#{ $i+1 }){
3         animation-delay: ($i/10)+s;
4     }
5 }
```

❖ Precarga 3

Figura 167. Creación del objeto `pre_carga_3_base` en función `mixin`

```
1 @mixin pre_carga_3_base{
2   position: absolute;
3   z-index:$pre_carga_3_visibilidad;
4   display:flex;
5   justify-content: center;
6   align-items: center;
7 }
```

❖ Precarga 4

Figura 168. Creación del objeto `pre_carga_centrado` en función `mixin`

```
1 @mixin pre_carga_centrado{
2   position: absolute;
3   top: 50%;
4   left: 50%;
5   transform: translate(-50%,-50%);
6 }
```

- **OOCSS**. Luego de tener la función o la lógica para reciclar el código se procede a integrar la metodología OOCSS para ello debemos llamar al `mixin` para ello solo necesitamos agregar `@include` “nombre del `mixin`”. Luego de integrarlo ya estará aplicada correctamente la metodología OOCSS

❖ Precarga 1

Figura 169. Metodología OOCSS en el componente `pre-carga parte I`

```
1 .pre-carga1{
2   @include precarga1_be_af('dimension_bola');
3   background: transparent;
4   top: 50%;
5   left:50%;
6   margin-top:80px;
7   transform:translate(-50%,-50%);
8   z-index:1;
9 }
```

Figura 170. Metodología OOCSS en el componente `pre-carga parte II`

```
1 .pre-carga1:before{
2   content: '';
3   @include precarga1_be_af('before');
4   background: $pre_carga_fondo_uno;
5   box-shadow: inset 0 -10px rgba(0,0,0,0.1);
6   animation: pre-carga1 .75s infinite linear;
7   z-index: 10;
8 }
```

Figura 171. Metodología OOCSS en el componente pre-carga parte III

```
1 .pre-carga1:after{
2   @include precarga1_be_af('after');
3   content: '';
4   background: rgba(0,0,0,0.1);
5   transform: scale(1,.25);
6   animation: pre-carga1shadow .75s infinite linear;
7 }
```

❖ Precarga 2

Figura 172. Metodología OOCSS en el componente pre-carga parte IV

```
1 .pre-carga2{
2   @include pre_carga_base;
3   position: absolute;
4   top:50%;
5   left: 50%;
6   transform: translate(-50%,-50%);
7   z-index:$pre_carga_2_prioridad_visibilidad;
8 }
```

Figura 173. Metodología OOCSS en el componente pre-carga parte V

```
1 .obj{
2   @include pre_carga_base;
3   width: 6px;
4   background:$pre_carga_2_color_barras;
5   margin: 0px 3px;
6   border-radius:$pre_carga_2_borde_barra;
7   animation: pre-carga2 0.8s infinite;
8 }
```

❖ Precarga 3

Figura 174. Metodología OOCSS en el componente pre-carga parte VI

```
1 .pre-carga3{
2   @include pre_carga_3_base;
3   top: 50%;
4   left: 50%;
5 }
```

Figura 175. Metodología OOCSS en el componente pre-carga parte VII

```
1 .pre-carga3 span{
2   @include pre_carga_3_base;
3   width:$pre_carga_3_tamano_bola;
4   height:$pre_carga_3_tamano_bola;
5   background: $color-blanco-1;
6   left: 0;
7   top: 0;
8   border-radius:$pre_carga_3_borde_redondeado;
9   animation: pre-carga3 2s ease-in-out infinite;
10 }
```

Cabe hacer una aclaración en esta parte, si bien tuvimos funciones, for y un each todos esos métodos se aplicó de forma directa la metodología OOCSS y no se requirió llamarla como se hace con el mixin ya que dentro del for se ejecutó de forma automática.

También muy importante se observa que al implementar la metodología queda fuera del mixin pero dentro de la clase la gran mayoría de las clases presentadas.

Esto permite dos cosas:

- Usar condicionales dentro de SASS para acceder a clases de css
- Hacer más personalizable el framework, ya para hacer cambio solo bastaría con cambiar la variable o añadir una en vez de revisar el código css ya compilado.

Bien volviendo al tema de las variables estas variables se encuentran en un archivo sass denominado variables valga la redundancia donde se encuentran las siguientes variables:

Figura 176. Variables Sass del componente pre-carga parte I

```
1 //Pre carga 1
2 $pre_carga_fondo_uno:#188FA7;//pre_carga_1
3 $pre_carga_ancho_bolita:40px;//pre_carga_1
4 $pre_carga_alto_bolita:40px;//pre_carga_1
```

Figura 177. Variables Sass del componente pre-carga parte II

```
1 //Pre carga 2
2 $pre_carga_2_num_barras:7;//pre_carga_2
3 $pre_carga_2_color_barras:#2980b9;//pre_carga_2
4 $pre_carga_2_prioridad_visibilidad:1;//pre_carga_2
5 $pre_carga_2_borde_barra:10px;//pre_carga_2
```

Figura 178. Variables Sass del componente pre-carga parte III

```
1 //Pre carga 3
2 $pre_carga_3_visibilidad:1;//pre_carga_3
3 $pre_carga_3_tamano_bola:1em;//pre_carga_3
4 $pre_carga_3_borde_redondeado:50%;//pre_carga_3
```

Figura 179. Variables SASS del componente pre-carga parte IV

```
1 //Pre carga 4
2 $pre_carga_4_dimencion_bolas:70px;//pre_carga_4
3 $pre_carga_4_borde_redondeado:50%;//pre_carga_4
4 $pre_carga_4_color_bola_2:#0ff;//pre_carga_4
```

Como se ha visto SASS es un preprocesador de css por ello se ejecuta antes del css tradicional esto permite añadir variables agregar condicionales, etc. Con la aplicación de la metodología como se observa se reutiliza el código css, además de que se hace escalable el proyecto. Ya que, si en un futuro queremos cambiar o añadir más propiedades css, solo cambiaríamos las variables o añadiríamos más propiedades en el mixin.

FASE 6.- Documentar en la web site www.phooldx.com

Finalmente, si el efecto paso por todos los de más filtros se procede a su documentación respectiva, para documentar todos los efectos se creó una página web de documentación usando clases y elementos del mismo framework. A continuación, se describirá como se procede a la documentación de la misma. Siguiendo estos pasos:

- Se copia las etiquetas de HTML5 junto con la clase a un txt temporal para no perder la sintaxis
- Se guarda los cambios en el compilador de SASS
- Como es la parte final se procedió hacer copias de seguridad para no perder el avance, en archivos RAR. Estos archivos estaban por versiones enumerados además se agregaba un pequeño texto describiendo el avance. Este guardado se hacía de manera local en un USB.
- Para una mayor seguridad del avance del código se procedió luego a copiar la versión del archivo RAR junto a la descripción del mismo.

Guardado de código en GitHub. - Luego de tener la copia del proyecto a nivel físico en un USB, se procedió a guardarlo en un repositorio de GitHub para lo cual se usó Git, para ello se procedió de la siguiente forma:

- Se guardaba todo el proyecto con el siguiente comando que se observa en la figura 12.
- Luego se verificaba los cambios guardados con el siguiente comando que se observa en la figura 13.
- Después se hacía el commit, es decir se le asignaba un nombre a los cambios realizados con el siguiente comando que se observa en la figura 14.
- Luego se procede a empaquetar los datos para que se suban al repositorio de GitHub con el siguiente código que se puede apreciar en la figura 15.
- Finalmente, luego de haber enviados el proyecto empaquetado hacia GitHub como se puede apreciar en la imagen, este queda guardado en dicho repositorio.

Figura 180. Visualización en github del componente pre-carga



Se guarda en una escala de tiempo, con todos los cambios hechos desde la creación del proyecto, esto ayuda a tener un mejor control de versiones. Asimismo, también ayuda a que más personas se sumen al proyecto y puedan clonar el proyecto como trabajar por separado el proyecto, entre otras muchas ventajas.

- Así mismo se manejó un archivo llamado bitácora.txt donde aquí se registró todos los incidentes ocurridos, así como los cambios hechos, se registraba también las metas cumplidas, también las tareas nuevas para desarrollar al día siguiente. Al finalizar con un efecto por completo se quitaba de la lista de prioridades así dábamos pase al siguiente efecto o problema ha resolver. Esto servía para tener un mejor orden y saber ejecutar prioridades.
- Luego de tener guardado el avance y de registrar el avanzase, se procedió a copiar del archivo documentación Etiquetas.txt el fragmento de código

que se copió anteriormente para luego pegarlo en www.nosetup.org esta página web se encargaba de convertir html en texto plano para poder visualizar html

- Antes de finalizar se procedió a crear un archivo llamado `modulo_precarga.html` donde se dividía en tres partes:
 - En la primera se describía un poco al efecto que era y que hacía
 - En la segunda parte se procedía a copiar el código ya formateado obtenido de www.nosetup.com para luego pegarlo.
 - En la tercera parte lo que se hacía era mostrar un apartado llamado estructura donde se describía como se llamaban las variables para que sea modificado en SASS.
- Finalmente, el ejemplo listo para usarse queda de la siguiente forma:

❖ Precarga 1

Figura 181. Código html5 final del componente pre-carga parte I

```
<div id="loader"></div>
```

❖ Precarga 2

Figura 182. Código html5 final del componente pre-carga parte II

```
<div class="pre-carga2">  
  <div class="obj"></div>  
  <div class="obj"></div>  
  <div class="obj"></div>  
  <div class="obj"></div>  
  <div class="obj"></div>  
  <div class="obj"></div>  
  <div class="obj"></div>  
  <div class="obj"></div>  
  <div class="obj"></div>  
</div>
```

❖ Precarga 3

Figura 183. Código html5 final del componente pre-carga parte III

```
<div class="pre-carga3">  
  <span class="Negro"></span>  
  <span class="Negro"></span>  
  <span class="Negro"></span>  
  <span class="Negro"></span>  
</div>
```

❖ Precarga 4

Figura 184. Código html5 final del componente pre-carga parte IV

```
<div class="pre-carga4">  
  <div class="bola-1"></div>  
  <div class="bola-2"></div>  
</div>
```

SLIDER AUTOMATICO

FASE 1.- Investigación del efecto y su importancia

Como primer paso lo primero que se iso fue investigar e indagar sobre el efecto, dicho efecto es uno de los más populares en la comunidad de desarrollo web, se quiso integrar este efecto a las filas del framework XRL8 por su alto impacto, un efecto slider funciona mostrando las imágenes más resaltantes en un pequeño recuadro las cuales van pasando automáticamente cada cierto tiempo. Un beneficio de usar el efecto slider es que apenas abran la página web los usuarios visualizaran las fotografías más representativas del sitio para así tener más contenido en un pequeño espacio. La razón más fuerte por la cual se decidió incluir en la lista de efectos es para destacar contenido.

Otro del motivo el cual fue elegido el efecto fue para formar parte del framework XRL8 es que debe estar a la altura de los frameworks que están en la actualidad siendo usados en la industria del desarrollo web como son:

- **Bootstrap.** El presente framework cuenta con la clase .carousel slide para poder crear el efecto slider, pueden encontrar varios casos de uso en la

documentación de dicho framework mediante el siguiente enlace:

<https://getbootstrap.com/docs/5.1/components/carousel/#slides-only>

- **Materialize.-** Otro framework del medio es materialize el cual también cuenta con una sección llamada carousel en el cual se puede elaborar un slider y este puede ser personalizado. Lo podremos observar más detalladamente accediendo mediante el siguiente link:

<https://materializecss.com/carousel.html#one!>

- **UIKIT.-** También este framework cuenta con una clase para hacer el efecto slider la clase es llamada uk-slider la cual sirve como contenedor del efecto, cabe recalcar que este efecto el framework usa JavaScript para darle comportamiento al efecto para saber más sobre este framework recomiendo revisar la documentación en el siguiente enlace:

<https://getuikit.com/docs/slider>

Modulo: Un módulo según definición propia es una sección de trabajo principal a comparación de un submódulo este puede ser denominado la estructura padre y los submódulos estructuras hijas. En este efecto en particular el módulo vendría a ser la siguiente clase:

- **Slider automático.** - Para este efecto nos basamos en mostrar imágenes desplazándose infinitamente de derecha hacia izquierda de forma automática con un retraso de seis segundos. Con el cual se garantiza que se muestren las imágenes más representativas del sitio web.

Submódulos: Considere llamarle submódulo, a los efectos o clases que están dentro de un módulo en este caso es slider automático. Los submódulos pueden compartir atributos del módulo, pero en otros casos pueden ser independientes, pero seguir clasificados como el módulo original.

Estos submódulos son muy útiles ya que desde mi punto de vista ayuda a diversificar los efectos y las clases. Beneficiando al usuario final en la construcción de sitios web. Actualmente el framework XRL8 no tiene pensado sacar sub módulos para el efecto slider automático.

FASE 2.- Codificar el efecto

Antes de ver esta sección a fondo quisiera hacer énfasis que la parte de codificación fue entender los conceptos básicos de css y con ello ir avanzando poco a poco.

Bien luego de tener claro que efecto queríamos construir y porque lo íbamos a construir como se vio en el capítulo anterior se procedió a escribir código como se muestra a continuación.

Código completo

❖ Código del módulo slider automático

Figura 185. Código de slider automatico parte I

```
1  .slider {
2    width: 100%;
3    margin: auto;
4    padding: 0;
5    overflow: hidden;
6  }
7  .slider ul {
8    display: flex;
9    padding: 0px;
10   width: 400%;
11   animation: cambio 20s infinite alternate linear;
12  }
13  .slider li {
14    width: 100%;
15    margin: auto;
16    padding: 0;
17    list-style: none;
18  }
19  .slider img {
20    width: 100%;
21    margin: auto;
22    padding: 0;
23    height: 40vh;
24  }
```

Figura 186. Código de slider automatico parte II

```
1  @keyframes cambio {
2    0% {
3      margin-left: 0;
4    }
5    20% {
6      margin-left: 0;
7    }
8    25% {
9      margin-left: -100%;
10   }
11   45% {
12     margin-left: -100%;
13   }
14   50% {
15     margin-left: -200%;
16   }
17   70% {
18     margin-left: -200%;
19   }
20   75% {
21     margin-left: -300%;
22   }
23   100% {
24     margin-left: -300%;
25   }
26 }
```

Descripción de las propiedades más significativas:

A continuación, se procederá a detallar las propiedades más usadas y su aplicación. También se mostrará la compatibilidad web de cada propiedad para su mejor comprensión dejamos una leyenda en la cual se indica si tiene compatibilidad web la cual se puede observar en la figura 2.

Propiedad width:100% :

Se empleo width 100% para el ancho de este efecto ya que si bien esta propiedad tiene diversas unidades de medidas como:

- EM
- VH
- PIXELES
- REM
- PORCENTAJE

Se fue descartando una a una al momento de probarla, ya que la al usar la unidad EM este se distorsionaba por su propia naturaleza, básicamente esta propiedad usa el ancho de la pantalla del dispositivo donde estemos y las va ajustando así que fue descartado.

A continuación, veamos la compatibilidad web al usar esta propiedad en la figura 3.

Propiedad display: flex

Con esta propiedad se puede aplicar la alineación de los elementos en una sola dirección sea horizontal o vertical, estos valores se puede conjugar con las distintos valores que tiene, se optó por esta propiedad ya que es una forma óptima de maquetar y de que los elementos no se desborden de sus contenedores como solía suceder al maquetar usando position o tables en la antigüedad además flexbox viene para usarlo con la versión 3 de css llamada css3 haciendo más robusta la maquetación de elementos.

A continuación, veamos la compatibilidad web al usar esta propiedad en la figura 43.

Propiedad height:40vh

Se uso esta propiedad para definir una altura ideal para que el efecto se pueda apreciar con facilidad asimismo este no pueda estropear la funcionalidad de la web donde se implementen. Así mismo como se aprecia tiene la unidad de medida vh dicha unidad de medida lo que hace es dar un valor con referencia al alto del dispositivo de donde nos conectemos, siendo 100vh la el alto total con referencia a la pantalla del dispositivo de donde nos conectemos.

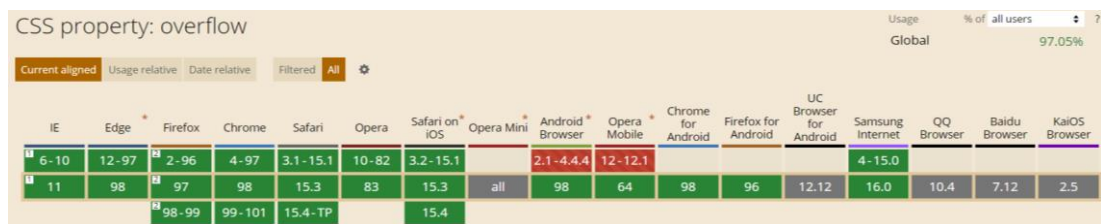
A continuación, veamos la compatibilidad web al usar esta propiedad en la figura 4.

Propiedad overflow:hidden

Esta propiedad lo que hace es ocultar los elementos que salgan de su contenedor mostrando solo una parte del contenido que encaje exacto en dicho contenedor al mismo tiempo lo que hace es eliminar el scroll horizontalmente.

A continuación, veamos la compatibilidad web al usar esta propiedad en los principales navegadores.

Figura 187. Compatibilidad web, propiedad overflow



IE	Edge	Firefox	Chrome	Safari	Opera	Safari on iOS	Opera Mini	Android Browser	Opera Mobile	Chrome for Android	Firefox for Android	UC Browser for Android	Samsung Internet	QQ Browser	Baidu Browser	KaiOS Browser
6-10	12-97	2-96	4-97	3.1-15.1	10-82	3.2-15.1		2.1-4.4.4	12-12.1				4-15.0			
11	98	97	98	15.3	83	15.3	all	98	64	98	96	12.12	16.0	10.4	7.12	2.5
		98-99	99-101	15.4-TP		15.4										

Propiedad list-style:none

Con esto se le asigna un estilo a las listas que están en las etiquetas como en las cuales se les asigna un estilo a dichas listas, con la propiedad none, lo que hace es desaparecer los puntos por default que trae dicha lista pueden ser símbolos o enumeración dependiendo del estilo son más de 60 estilos.

A continuación, veamos la compatibilidad web al usar esta propiedad en los principales navegadores.

Figura 188. Compatibilidad web, propiedad list-style

Browser	Version
IE	6-10
Edge	12-97
Firefox	2-96
Chrome	4-97
Safari	3.1-15.1
Opera	10-82
Safari on iOS	3.2-15.1
Opera Mini	all
Android Browser	2.1-4.4.4
Opera Mobile	12-12.1
Chrome for Android	98
Firefox for Android	96
UC Browser for Android	12.12
Samsung Internet	4-15.0
QQ Browser	16.0
Baidu Browser	10.4
KaiOS Browser	7.12
	2.5

Propiedad @keyframe cambio

Con esta propiedad se pueden crear animaciones más sofisticadas ya que se pueden mezclar propiedades de css en su interior además de combinarlas con la propiedad animation y transform. Esta se declara usando la palabra reservada @keyframe seguido de un nombre en este caso el nombre que le asignamos es cambio, luego dentro se puede usar de dos formas:

- ❖ Usando la estructura from y to de la siguiente manera:

Figura 189. Estructura de un keyframes parte I

```
1 @keyframes nombreAnimacion {
2   from { /*tus propiedades aqui*/ }
3   to { /*tus propiedades aqui*/ }
4 }
```

- ❖ También podemos usar porcentaje para especificar a fondo lo que queremos que haga la animación:

Figura 190. Estructura de un keyframes parte II

```
1 @keyframes nombreAnimacion {
2   0% { /*Tus propiedades aqui*/ }
3   50% { /*Tus propiedades aqui*/ }
4   50% { /*Tus propiedades aqui*/ }
5   100% { /*Tus propiedades aqui*/ }
6 }
```

A continuación, veamos la compatibilidad web al usar esta propiedad en los principales navegadores.

Figura 191. Compatibilidad web propiedad keframes



Principales dificultades:

Para este efecto en concreto la principal dificultad fue organizar de distintas formas posibles, así como ver las posibles conjugaciones que pueden hacer con el efecto para que el usuario al final no tenga problemas al implementarlo. Esas dificultades surgieron a raíz de las siguientes preguntas:

- ¿Se podría colocar botones para mejorar la experiencia de usuario?

Claro que sí, haciendo uso del lenguaje de programación JavaScript.

- ¿Cómo hago que el efecto slider recorra indeterminadamente en ambos sentidos?

Al principio no se tuvo noción de cómo hacerlo, pero luego investigando las propiedades de animación se encontró con los valores infinite el cual hace referencia a que será infinito el efecto, seguidamente nos topamos con la propiedad alternate la cual como su propio nombre hace referencia alterna de inicio a fin es decir al llegar al final de la animación empieza otra vez, pero en sentido contrario.

Figura 192. Ejemplo de uso de la propiedad animation

```
animation: cambio 20s infinite alternate
```

¿Qué usamos para hacer las pruebas?

Se uso el navegador Google Chrome y opera mini, así como la siguiente página web que es un simulador online:

<http://www.responsinator.com>

Esto con motivos de asegurarnos que funcione en dispositivos móviles y tabletas. Ya que en base a lo trabajado no es del todo fiable usar el navegador para verificar la correcta maquetación de los elementos, así como tampoco es fiable depender de un software web, por ello se decidió diversificar en cuanto a las herramientas para probar las clases y efectos trabajados en el framework XRL8.

También usamos como editor de código visual studio code y su extensión live server para visualizar en vivo los efectos realizados en el css en tiempo real, también usamos el autoguardado que trae dicho editor de código para no estar guardando el proyecto a cada rato. Así conseguimos un mejor desempeño y evitamos el tedioso guardar cambios a cada rato, también evitamos el que la memoria cache se llene mostrándonos los mismos resultados.

FASE 3.- Hacerlo responsive

Con este efecto en particular fue sencillo ya que la propiedad width estaba en % además de que no se necesitó una maquetación extra como veremos más adelante en el presente informe. En caso se hubiera dado la oportunidad de maquetar más el efecto para que tome forma y se moldee, lo que hubiéramos hecho en ese caso sería:

- Usar la propiedad flex-box y sus posibles conjugaciones con cada uno de sus valores según sea el caso.
- Hubiéramos trabajado con la propiedad @media_query en las resoluciones de pantalla para celulares y tabletas. Siempre empezando a maquetar por la parte de celulares y luego tabletas.
- También hubiéramos usado las unidades de medida vh o rem para texto y contenedores ya que estas son imprescindibles a la hora de hacer responsive un sitio web.
-

FASE 4.- Testing asegurarse que no tenga conflicto con otros efectos.

En esta parte del desarrollo de la clase se dispuso a probar poniéndolo con las demás clases juntándolas. Para ello lo que se hizo fue colocar dentro de un archivo index.html todas las clases trabajadas y ver como el efecto de slider respondía. En algunos casos había conflicto con los demás efectos para corregir esto se procedió a implementar en el desarrollo dos cosas básicas la primera fue:

- Para detectar que clase era la que causaba el conflicto y ver en que archivo scss estaba. Se procedía a inspeccionar la página para detectar donde estaba el problema. En muchas ocasiones no se daba con el archivo causante del error entonces en ese caso aplicábamos la frase “Divide y vencerás”. Lo que se procedió hacer fue comentar de uno en uno cada efecto en el archivo principal que los llamaba esto reducía las líneas de código de css, además como teníamos en tiempo real los cambios fue mucho más rápido detectar el problema. Ya que no se tenía que recargar a cada rato además otro punto a favor era que se auto compilaba los cambios de sass automáticamente gracias a nodeJS el cual mediante la consola de comandos de Windows cuando detectaba un cambio en las líneas de código se auto compilaba y generando o reemplazando el archivo css en el cual hacíamos las pruebas, esto aumento la productividad para crear los efectos ya que al comentar de uno en uno cada archivo sass llegaba un momento donde desaparecía el error entonces revisábamos y nos dimos cuenta de que era uno o en algunos casos hasta tres efectos que hacían conflicto con la clase escrita.
- Se procedió a escribir mediante comentarios en un archivo llamado `_error_logs.scss` la clase que hace conflicto con la clase escrita, además de describir una nota breve en caso no se disponía de tiempo para solucionarlo luego, esto permitió tener un reporte más preciso de los incidentes que íbamos detectando.

FASE 5.- Aplicar la metodología OOCSS, llevándolo a SASS

- **Código css.** - Esta parte ya estaría completa por lo descrito anteriormente.

- **Código repetido.** - Identificación de código repetido

En este pequeño apartado luego de escribir y debugear el error se procedió a identificar el código css repetido para llevarlo a un archivo SCSS en sass, para su mayor optimización y aplicación de la metodología. Así que se procedió a reciclar el siguiente código.

❖ Slider automático

Figura 193. Código repetido del componente slider automático

```
1  .slider {
2    width: 100%;
3    margin: auto;
4    padding: 0;
5    overflow: hidden;
6  }
7
8  .slider li {
9    width: 100%;
10   margin: auto;
11   padding: 0;
12   list-style: none;
13  }
14
15  .slider img {
16    width: 100%;
17    margin: auto;
18    padding: 0;
19    height: 40vh;
20  }
```

- **Crear un objeto.** - Se describe la lógica de la función para luego crear un mixing.

Una vez identificado el código css repetido se procede hacer una función la cual se ejecuta al llamado del código, mediante esta técnica nuestro código será más puro y limpio. Cabe recalcar que slider comparte características similares.

Paralelamente en esta parte se determina la lógica de las funciones y ciclos como lo son for, while, each. También determinamos el pase de

parámetros, así como la condicional que llevaran dentro, por otra parte, y no menos importante se define el nombre con el cual se llamara a las funciones.

```
function slider_base(){
width:100%;
margin:auto;
padding:0;
}
```

Luego de tener la lógica se procede a llevarlo a SASS para ello tenemos dos opciones, la primera es usar @function ya que como su nombre ase referencia es para crear una función la otra opción es usar @mixin, pero ambas tienen diferencias. La primera retorna un valor, mientras @mixin solo llama a los valores en este caso no necesitamos que retorne un valor por ello usaremos @mixin. Entonces quedaría de la siguiente forma:

❖ Slider

Figura 194. Creación del objeto slider_base en función mixin

```
1 @mixin slider_base{
2     width:$slider-automatico-ancho;
3     margin: auto;
4     padding: 0;
5 }
6
```

- **OOCSS**. – Luego de tener la función o la lógica para reciclar el código se procede a integrar la metodología OOCSS para ello debemos llamar al mixin para hacerlo solo necesitamos agregar @include “nombre del mixin”. Luego de integrarlo ya estará aplicada correctamente la metodología OOCSS

❖ Slider automático

Figura 195. Metodología OOCSS en el componente slider automático

```
1 .slider{
2     @include slider_base;
3     overflow: hidden;
4     ul{
5         display: flex;
6         padding: 0px;
7         width: 400%;
8         animation:cambio 20s infinite alternate linear;
9     }
10    li{
11        @include slider_base;
12        list-style:none;
13    }
14    img{
15        @include slider_base;
16        height:40vh;
17    }
18 }
19
```

También muy importante se observa que al implementar la metodología queda fuera del mixin pero dentro de la clase slider automático, además se aprecia que está construido en cascada las clases, esto es gracias a SASS que permite anidar más clases a la clase principal.

Esto permite dos cosas:

- Usar condicionales dentro de SASS para acceder a clases de css, que veremos en el transcurso del proyecto.
- Hacer más personalizable el framework, para hacer cambio solo bastaría con cambiar la variable o añadir una en vez de revisar el código css ya compilado.

Bien volviendo al tema de las variables estas variables se encuentran en un archivo sass denominado variables valga la redundancia donde se encuentran las siguientes variables:

Figura 196. Variables sass del componente slider automático

```
1 $slider-automatico-ancho:100%;  
2 $slider_margenes:auto;  
3 $slider_padding:0;
```

Como se ha visto SASS es un preprocesador de css por ello se ejecuta antes del css tradicional esto permite añadir variables agregar condicionales, etc. Con la aplicación de la metodología como se observa se reutiliza el código css, además de que se hace escalable el proyecto. Ya que, si en un futuro queremos cambiar o añadir más propiedades css, solo cambiaríamos las variables o añadiríamos más propiedades en el mixin.

FASE 6.- Documentar en la web site www.phooldx.com

Finalmente, si el efecto paso por todos los de más filtros se procede a su documentación respectiva, para documentar todos los efectos se creó una página web de documentación usando clases y elementos del mismo framework. A continuación, se describirá como se procede a la documentación de la misma. Siguiendo estos pasos:

- Se copia las etiquetas de HTML5 junto con la clase a un txt temporal para no perder la sintaxis
- Se guarda los cambios en el compilador de SASS
- Como es la parte final se procedió hacer copias de seguridad para no perder el avance, en archivos RAR. Estos archivos estaban por versiones enumerados además se agregaba un pequeño texto describiendo el avance. Este guardado se hacía de manera local en un USB.
- Para una mayor seguridad del avance del código se procedió luego a copiar la versión del archivo RAR junto a la descripción del mismo.

Guardado de código en GitHub. - Luego de tener la copia del proyecto a nivel físico en un USB, se procedió a guardarlo en un repositorio de GitHub para lo cual se usó Git, para ello se procedió de la siguiente forma:

- Se guardaba todo el proyecto con el siguiente comando que se observa en la figura 12.
- Luego se verificaba los cambios guardados con el siguiente comando que se puede observar en la figura 13.
- Después se hacía el commit, es decir se le asignaba un nombre a los cambios realizados con el siguiente comando que se puede observar en la figura 14.
- Luego se procede a empaquetar los datos para que se suban al repositorio de GitHub con el siguiente código que se observa en la figura 15.
- Finalmente, luego de haber enviados el proyecto empaquetado hacia GitHub como se puede apreciar en la imagen, este queda guardado en dicho repositorio.

Figura 197. Visualización en github del componente slider automático



Se guarda en una escala de tiempo, con todos los cambios hechos desde la creación del proyecto, esto ayuda a tener un mejor control de versiones. Asimismo, también ayuda a que más personas se sumen al proyecto y puedan clonar el proyecto como trabajar por separado el proyecto, entre otras muchas ventajas.

- Así mismo se manejó un archivo llamado bitácora.txt donde aquí se registró todos los incidentes ocurridos, así como los cambios hechos, se registraba también las metas cumplidas, también las tareas nuevas para desarrollar al día siguiente. Al finalizar con un efecto por completo se quitaba de la lista de prioridades así dábamos pase al siguiente efecto o problema ha resolver. Esto servía para tener un mejor orden y saber ejecutar prioridades.
- Luego de tener guardado el avance y de registrar el avanzase, se procedió a copiar del archivo documentación Etiquetas.txt el fragmento de código que se copió anteriormente para luego pegarlo en www.nosetup.org esta página web se encargaba de convertir html en texto plano para poder visualizar html
- Antes de finalizar se procedió a crear un archivo llamado modulo_slider.html donde se dividía en tres partes:
 - En la primera se describía un poco al efecto que era y que hacia
 - En la segunda parte se procedía a copiar el código ya formateado obtenido de www.nosetup.com para luego pegarlo.
 - En la tercera parte lo que se hacía era mostrar un apartado llamado estructura donde se describía como se llamaban las variables para que sea modificado en SASS.
- Finalmente, el ejemplo listo para usarse queda de la siguiente forma:

❖ Slider automático

Figura 198. Código html5 final del componente slider automático

```
<div class="slider">
  <ul>
    <li></li>
    <li></li>
    <li></li>
    <li></li>
  </ul>
</div>
```

PAGINACION

FASE 1.- Investigación del efecto y su importancia

Como primer paso lo primero que se hizo fue investigar e indagar sobre el efecto, una paginación sirve para poder navegar entre el contenido de un sitio web, haciendo analogía es como estar entre las páginas de un libro y la paginación vendría hacer las páginas enumeradas de un libro. Uno de los beneficios de usarlo es que se ahorra mucho en los espacios de los contenidos ya que no tendrá que cargar todo el contenido de golpe en una sola vez, es decir cargara la primera página, luego ira cargando las demás según el usuario valla dando clic en las páginas así nos aseguramos tener un rendimiento más estable en el servidor como en el lado del cliente. Por todo lo expuesto anteriormente y los beneficios que trae el usarlo se decidió implementar este efecto en XRL8 por sus altos beneficios y por la demanda que existe en el mercado.

Otro del motivo el cual fue elegido el efecto para formar parte del framework XRL8 fue que el proyecto presente debe estar a la altura de los frameworks que están en la actualidad siendo usados en la industria del desarrollo web como son:

- **Bootstrap.** - El presente framework cuenta con la clase .pagination la cual sirve de contenedor para la paginación dicho de paso es ahí donde irán las demás etiquetas que complementen el efecto, si desean saber más sobre ello puede visitar la web oficial de documentación de dicho framework mediante el siguiente enlace:

<https://getbootstrap.com/docs/4.0/components/pagination/>

- **Bulma.-** Este framework también cuenta con un modelo para hacer la paginación, también tiene la opción de personalizarlo por tamaño, para ver cómo se emplea y formas de uso revisar el siguiente enlace que los llevara a la documentación oficial de dicho framework:

<https://bulma.io/documentation/components/pagination/>

- **TailwindCSS.-** Quisa sea este el framework de css más usado en los últimos 3 años sobre pasando a los demás por sus formas de integración con NodeJS y react. Este framework dentro de su contenido tiene la particularidad de que se mezcla con clases preparadas para su construcción, este a diferencia de los demás se caracteriza por no tener clases predefinidas para efectos sino todo lo contrario tiene un conjunto de clases en el cual el usuario puede elegir los efectos más simples para colocar a su diseño, la diferencia está en que al ser clases preparadas y redefinidas son muchas más que el usar una clase de cualquier otro framework incluido XRL8. Para revisar su documentación y ver como se construye un efecto de paginación ingresar al siguiente enlace:

<https://tailwindui.com/components/application-ui/navigation/pagination>

- **Foundation.-** Uno de los primeros frameworks en salir al mercado sin duda será foundation aunque con el pasar de los años a perdido popularidad este framework también dentro de sus clases tiene una sección para hacer paginación, la clase que se emplea para hacer dicho efecto es denominado .pagination esta clase sirve como contenedor para que dentro de ella se pueda implementar las demás clases y etiquetas que se requiere para su funcionamiento. Para ver el uso de este se puede recurrir al siguiente link:

<https://get.foundation/sites/docs/pagination.html>

- **Materialize.-** Otro framework del medio es materialize el cual también cuenta con una sección llamada pagination el cual ayuda a construir páginas de navegación. Lo podremos observar más detalladamente accediendo mediante el siguiente link:

<https://materializecss.com/pagination.html>

- **UIKIT.-** También este framework cuenta con una clase para hacer una paginación sencilla, la clase empleada en este caso tiene por nombre .uk-pagination para ver más sobre este framework recomiendo revisar la documentación en el siguiente enlace:

<https://getuikit.com/docs/pagination>

- **Semantic UI.-** A continuación tenemos otro framework de css muy completo la verdad el cual tiene soporte y componentes para ser usado con react, la clase la cual ayuda a crear una paginación se denomina pagination, podemos ver una gran cantidad de ejemplos mediante el siguiente enlace:

<https://react.semantic-ui.com/addons/pagination/>

Módulo: Un módulo según definición propia es una sección de trabajo principal a comparación de un submódulo este puede ser denominado la estructura padre y los submódulos estructuras hijas. En este efecto en particular el módulo vendría a ser la siguiente clase:

- **Paginación.** - Para este efecto nos basamos en un diseño clásico el cual contiene dos botones para la navegación individual de atrás y siguiente. Así mismo se muestra las posibles páginas que tendrá para poder visualizar el contenido, finalmente tiene una tonalidad verde claro.

Submódulos: Considere llamarle submódulo, a los efectos o clases que están dentro de un módulo en este caso el módulo es paginación un submódulo vendría a ser como una clase hija que heredara atributos como también por el contrario podría ser una versión mejorada del módulo, aunque no comparta rasgos o atributos.

Estos submódulos son muy útiles ya que desde mi punto de vista ayuda a diversificar los efectos y las clases. Beneficiando al usuario final en la construcción de sitios web. Actualmente el framework XRL8 no tiene pensado sacar submódulos para el efecto paginación.

FASE 2.- Codificar el efecto

Antes de ver esta sección a fondo quería hacer énfasis en cómo fue la codificación del módulo, no fue engorrosa ya que se partió de elementos muy básicos para su construcción.

Bien luego de tener claro que efecto queríamos construir y porque lo íbamos a construir como se vio en el capítulo anterior se procedió a escribir código como se muestra a continuación.

Código completo

❖ Código del módulo paginación

[parte 01 – paginación]

Figura 199. Código de paginacion parte I

```
1 .paginacion {
2   display: flex;
3   flex-wrap: wrap;
4 }
5
6 .paginacion a {
7   float: left;
8   padding: 8px 16px;
9   text-decoration: none;
10  border: 1px solid #ddd;
11  color: black;
12 }
13
14 .paginacion a.activado {
15   background-color: #4CAF50;
16   border: 1px solid #4CAF50;
17   color: white;
18 }
```

[parte 02 – paginación]

Figura 200. Código de paginacion parte II

```
1 .paginacion a:hover:not(.activado) {
2   background-color: #ddd;
3 }
4
5 .paginacion a:first-child {
6   border-top-left-radius: 5px;
7   border-bottom-right-radius: 5px;
8 }
9
10 .paginacion a:last-child {
11   border-top-left-radius: 5px;
12   border-bottom-right-radius: 5px;
13 }
```

Descripción de las propiedades más significativas:

A continuación, se procederá a detallar las propiedades más usadas y su aplicación. También se mostrará la compatibilidad web de cada propiedad para su mejor comprensión dejamos una leyenda la cual se puede apreciar en la figura 2.

Propiedad background-color:#4CAF50

Esta propiedad se usa para darle un color de fondo para esta ocasión se decidió que sería un verde claro. Ya que es más comercial y combina con cualquier tonalidad.

A continuación, veamos la compatibilidad web al usar esta propiedad en los principales navegadores lo podemos visualizar en la figura 5.

Propiedad border:solid 1px #4CAF50

En esta propiedad le damos un borde sólido, esto quiere decir que el borde será visible, además de que será una línea definida sin interrupciones. Luego la otra característica es que tendrá una anchura de 1px, dicho de mejor forma será una línea fina. Finalmente, el tercer campo indica el color del borde en este caso es #4CAF50 que dicho de una forma sencilla es un color verde claro.

A continuación, veamos la compatibilidad web al usar esta propiedad en los principales navegadores lo podemos visualizar en la figura 37.

Propiedad padding:8px 16px

En el uso de esta propiedad se define como primera instancia a los valores que tendrán arriba y abajo que en este caso será de 8px luego como segundo parámetro tenemos 16px que será a la derecha y a la izquierda. Dicho de otra forma, cuando se pone un solo valor este pertenecerá a las cuatro posiciones de css arriba, derecha, abajo e izquierda pero cuando existe dos valores estos se distribuyen como lo mencione.

A continuación, veamos la compatibilidad web al usar esta propiedad en los principales navegadores lo podemos visualizar en la figura 20.

Propiedad color:black

Como color por default se eligió al negro ya que este hará resaltar a la numeración que tendrá el efecto de paginación además de que es un color básico como el blanco y no se verá brusco.

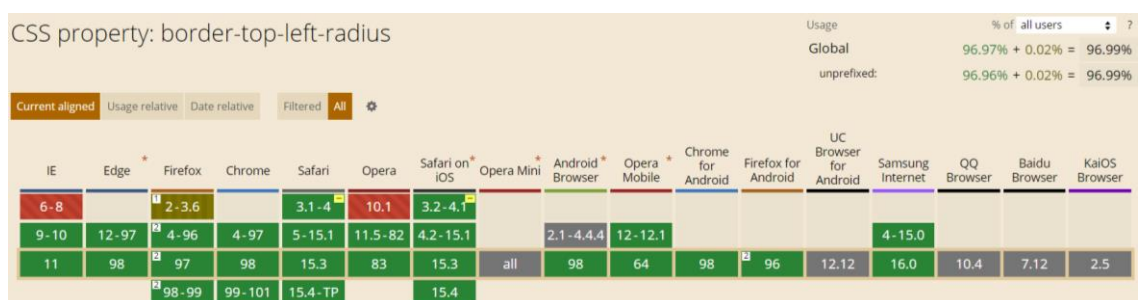
A continuación, veamos la compatibilidad web al usar esta propiedad en los principales navegadores lo podemos visualizar en la figura 21.

Propiedad border-top-left-radius:5px

Esta propiedad es bastante compleja ya que como se observa lleva la propiedad borde como prefijo esto nos quiere decir que la propiedad se encarga de dar cierto tipo de borde de acuerdo a la configuración, así como los parámetros que le pongamos, en términos más sencillos, lo que ara esta propiedad es dar un borde en la parte superior e inferior al mismo tiempo para tener como resultado final un arco en el lado izquierdo, pero será muy sutil el borde ya que solo tiene como parámetro 5px.

A continuación, veamos la compatibilidad web al usar esta propiedad en los principales navegadores.

Figura 201. Compatibilidad web, propiedad border-top-left-radius



Principales dificultades:

Para este efecto en concreto la principal dificultad fue organizar de distintas formas posibles, así como ver las posibles conjugaciones que pueden hacer con el efecto para que el usuario al final no tenga problemas al implementarlo. Esas dificultades surgieron a raíz de las siguientes preguntas:

- ¿Qué pasa con la numeración de la paginación si estos valores son enormes y vienen del servidor, se podrá adecuar?

Ante esa pregunta, surgió una respuesta conforme se iba construyendo dicho proyecto nos dimos cuenta que eso ya dependía del front-end ya que puede manipular el DOM usando Javascript.

¿Qué usamos para hacer las pruebas?

Se uso el navegador Google Chrome y opera mini, así como la siguiente página web que es un simulador online:

<http://www.responsinator.com>

Esto con motivos de asegurarnos que funcione en dispositivos móviles y tabletas. Ya que en base a lo trabajado no es del todo fiable usar el navegador para verificar la correcta maquetación de los elementos, así como tampoco es fiable depender de un software web, por ello se decidió diversificar en cuanto a las herramientas para probar las clases y efectos trabajados en el framework XRL8.

También usamos como editor de código visual studio code y su extensión live server para visualizar en vivo los efectos realizados en el css en tiempo real, también usamos el autoguardado que trae dicho editor de código para no estar guardando el proyecto a cada rato. Así conseguimos un mejor desempeño y evitamos el tedioso guardar cambios a cada rato, también evitamos el que la memoria cache se llene mostrándonos los mismos resultados.

FASE 3.- Hacerlo responsive

Para este módulo solo se aplicó la propiedad flex-box para hacerlo responsive ya que no se consideró necesario añadir la propiedad width o usar porcentaje ya que fue suficiente con el padding en cada elemento para cubrir el ancho de los elementos. En caso se hubiera dado la oportunidad de maquetar más el efecto para que tome forma y se moldee, lo que hubiéramos hecho en ese caso sería:

- Hubiéramos trabajado con la propiedad @media_query en las resoluciones de pantalla para celulares y tabletas. Siempre empezando a maquetar por la parte de celulares y luego tabletas.
- También hubiéramos usado las unidades de medida vh o rem para texto y contenedores ya que estas son imprescindibles a la hora de hacer responsive un sitio web.

FASE 4.- Testing asegurarse que no tenga conflicto con otros efectos.

En esta parte del desarrollo de la clase se dispuso a probar poniéndolo con las demás clases juntándolas. Para ello lo que se hizo fue colocar dentro de un archivo index.html todas las clases trabajadas y ver como el efecto de paginación respondía. En algunos casos había conflicto con los demás efectos para corregir esto se procedió a implementar en el desarrollo dos cosas básicas la primera fue:

- Para detectar que clase era la que causaba el conflicto y ver en que archivo scss estaba. Se procedía a inspeccionar la página para detectar donde estaba el problema. En muchas ocasiones no se daba con el archivo causante del error entonces en ese caso aplicábamos la frase “Divide y vencerás”. Lo que se procedió hacer fue comentar de uno en uno cada efecto en el archivo principal que los llamaba esto reducía las líneas de código de css, además como teníamos en tiempo real los cambios fue mucho más rápido detectar el problema. Ya que no se tenía que recargar a cada rato además otro punto a favor era que se auto compilaba los cambios de sass automáticamente gracias a nodeJS el cual mediante la consola de comandos de Windows cuando detectaba un cambio en las líneas de código se auto compilaba y generando o reemplazando el archivo css en el cual hacíamos las pruebas, esto aumento la productividad para crear los efectos ya que al comentar de uno en uno cada archivo sass llegaba un momento donde desaparecía el error entonces revisábamos y nos dimos cuenta de que era uno o en algunos casos hasta tres efectos que hacían conflicto con la clase escrita.
- Se procedió a escribir mediante comentarios en un archivo llamado _error_logs.scss la clase que hace conflicto con la clase escrita, además de describir una nota breve en caso no se disponía de tiempo para solucionarlo luego, esto permitió tener un reporte más preciso de los incidentes que íbamos detectando.

FASE 5.- Aplicar la metodología OOCSS, llevándolo a SASS

- **Código css.** - Esta parte ya estaría completa por lo descrito anteriormente.

- **Código repetido.** - Identificación de código repetido

En este pequeño apartado luego de escribir y debugear el error se procedió a identificar el código css repetido para llevarlo a un archivo SCSS en sass, para su mayor optimización y aplicación de la metodología. Así que se procedió a reciclar el siguiente código.

Figura 202. Código repetido del componente paginación parte I

```
1 .paginacion a:first-child {  
2   border-top-left-radius: 5px;  
3   border-bottom-right-radius: 5px;  
4 }
```

Figura 203. Código repetido del componente paginación parte II

```
1 .paginacion a:last-child {  
2   border-top-left-radius: 5px;  
3   border-bottom-right-radius: 5px;  
4 }
```

- **Crear un objeto.** - Se describe la lógica de la función para luego crear un mixing.

Una vez identificado el código css repetido se procede hacer una función la cual se ejecuta al llamado del código, mediante esta técnica nuestro código será más puro y limpio. Paralelamente en esta parte se determina la lógica de las funciones y ciclos como lo son for, while, each. También determinamos el pase de parámetros, así como la condicional que llevaran dentro, por otra parte, y no menos importante se define el nombre con el cual se llamara a las funciones.

```
function paginacion_retrocesos(){  
  border-top-left-radius:5px;  
  border-bottom-right-radius:5px;  
}
```

Luego de tener la lógica se procede a llevarlo a SASS para ello tenemos dos opciones, la primera es usar @function ya que como su nombre ase referencia es para crear una función la otra opción es usar @mixin, pero ambas tienen diferencias. La primera retorna un valor, mientras @mixin solo

llama a los valores en este caso no necesitamos que retorne un valor por ello usaremos @mixin. Entonces quedaría de la siguiente forma:

Figura 204. Creación del objeto paginacion_retrocesos en función mixin

```
1 @mixin paginacion_retrocesos{
2     border-top-left-radius:$border_arco_superior_redondeado;
3     border-bottom-right-radius:$border_arco_inferior_redondeado;
4 }
```

- **OOCSS**. – Luego de tener la función o la lógica para reciclar el código se procede a integrar la metodología OOCSS para ello debemos llamar al mixin para hacerlo solo necesitamos agregar @include “nombre del mixin”. Luego de integrarlo ya estará aplicada correctamente la metodología OOCSS

Figura 205. Metodología OOCSS en el componente paginación parte I

```
1 .paginacion a:first-child{
2     @include paginacion_retrocesos();
3 }
```

Figura 206. Metodología OOCSS en el componente paginación parte II

```
1 .paginacion a:last-child{
2     @include paginacion_retrocesos();
3 }
```

Bien volviendo al tema de las variables estas variables se encuentran en un archivo sass denominado variables valga la redundancia donde se encuentran las siguientes variables:

Figura 207. Variables sass del componente paginación

```
1 $border_arco_superior_redondeado:5px;
2 $border_arco_inferior_redondeado:5px;
```

Como se ha visto SASS es un preprocesador de css por ello se ejecuta antes del css tradicional esto permite añadir variables, agregar condicionales, etc. Con la aplicación de la metodología como se observa se reutiliza el código css, además de que se hace escalable el proyecto. Ya que, si en un futuro queremos cambiar o añadir más propiedades css, solo cambiaríamos las variables o añadiríamos más propiedades en el mixin.

FASE 6.- Documentar en la web site www.phooldx.com

Finalmente, si el efecto paso por todos los de más filtros se procede a su documentación respectiva, para documentar todos los efectos se creó una página web de documentación usando clases y elementos del mismo framework. A continuación, se describirá como se procede a la documentación de la misma. Siguiendo estos pasos:

- Se copia las etiquetas de HTML5 junto con la clase a un txt temporal para no perder la sintaxis
- Se guarda los cambios en el compilador de SASS
- Como es la parte final se procedió hacer copias de seguridad para no perder el avance, en archivos RAR. Estos archivos estaban por versiones enumerados además se agregaba un pequeño texto describiendo el avance. Este guardado se hacía de manera local en un USB.
- Para una mayor seguridad del avance del código se procedió luego a copiar la versión del archivo RAR junto a la descripción del mismo.

Guardado de código en GitHub. - Luego de tener la copia del proyecto a nivel físico en un USB, se procedió a guardarlo en un repositorio de GitHub para lo cual se usó Git, para ello se procedió de la siguiente forma:

- Se guardaba todo el proyecto con el siguiente comando que se muestra en la figura 12.
- Luego se verificaba los cambios guardados con el siguiente comando que se muestra en la figura 13.
- Después se hacía el commit, es decir se le asignaba un nombre a los cambios realizados con el siguiente comando que se muestra en la figura 14.
- Luego se procede a empaquetar los datos para que se suban al repositorio de GitHub con el siguiente código que se muestra en la figura 15.
- Finalmente, luego de haber enviados el proyecto empaquetado hacia GitHub como se puede apreciar en la imagen, este queda guardado en dicho repositorio.

Figura 208. Visualización en github del componente paginación



Se guarda en una escala de tiempo, con todos los cambios hechos desde la creación del proyecto, esto ayuda a tener un mejor control de versiones. Asimismo, también ayuda a que más personas se sumen al proyecto y puedan clonar el proyecto como trabajar por separado el proyecto, entre otras muchas ventajas.

- Así mismo se manejó un archivo llamado bitácora.txt donde aquí se registró todos los incidentes ocurridos, así como los cambios hechos, se registraba también las metas cumplidas, también las tareas nuevas para desarrollar al día siguiente. Al finalizar con un efecto por completo se quitaba de la lista de prioridades así dábamos pase al siguiente efecto o problema ha resolver. Esto servía para tener un mejor orden y saber ejecutar prioridades.
- Luego de tener guardado el avance y de registrar el avanzase, se procedió a copiar del archivo documentación Etiquetas.txt el fragmento de código que se copió anteriormente para luego pegarlo en www.nosetup.org esta página web se encargaba de convertir html en texto plano para poder visualizar html
- Antes de finalizar se procedió a crear un archivo llamado modulo_paginacion.html donde se dividía en tres partes:
 - En la primera se describía un poco al efecto que era y que hacia
 - En la segunda parte se procedía a copiar el código ya formateado obtenido de www.nosetup.com para luego pegarlo.
 - En la tercera parte lo que se hacía era mostrar un apartado llamado estructura donde se describía como se llamaban las variables para que sea modificado en SASS.
- Finalmente, el ejemplo listo para usarse queda de la siguiente forma:

Figura 209. Código html5 final del componente paginación

```
<div class="paginacion">
  <a href="#">&laquo;</a>
  <a href="#" class="activado">1</a>
  <a href="#">2</a>
  <a href="#">3</a>
  <a href="#">4</a>
  <a href="#">5</a>
  <a href="#">6</a>
  <a href="#">7</a>
  <a href="#">8</a>
  <a href="#">&raquo;</a>
</div>
```

TEXTURAS

FASE 1.- Investigación del efecto y su importancia

Como primer paso lo primero que se ise fue investigar e indagar sobre el efecto o modulo. Esta clase sirve para cambiar el puntero del mouse en la página web donde estamos. Uno de los beneficios es que tu página web se diferenciara del resto a tal punto de convertirlo toda la experiencia de usuario premium. Además de lo ya mencionado dicho efecto fue elegido por su eficiencia para lograr convertir el típico puntero de mouse en algo personalizable adicional a esto o dicho de otra manera como valor agregado estos conjuntos de punteros funcionan sin ninguna dificultad o retraso que haga que corra lento la página en el navegador.

Otro motivo el cual fue elegido el efecto para formar parte del framework XRL8 fue que el proyecto presente debe estar a la altura de los frameworks que están en la actualidad siendo usados en la industria del desarrollo web como son:

- **Magicmouse.**- La presente librería es una escrita en JavaScript el cual cambia el típico cursor del mouse por uno con diseño, si desean saber más sobre ello puede visitar la web oficial de documentación de dicho framework mediante el siguiente enlace:

<https://magicmousejs.web.app/>

- **TailwindCSS.**- Quisa sea este el framework de css más usado en los últimos 3 años sobre pasando a los demás por sus formas de integración

con NodeJS y react. Este framework dentro de su contenido también tiene una clase llamada .navbar que mezclada con otras clases se puede construir un menú de navegación. Gracias a sus componentes, estos a diferencia de los demás se caracteriza por no tener clases predefinidas para efectos sino todo lo contrario tiene un conjunto de clases en el cual el usuario puede elegir los efectos más simples para colocar a su diseño. Para revisar su documentación ingresar al siguiente enlace:

<https://tailwindcss.com/docs/installation>

- **KursorJS.** Este es una biblioteca de JavaScript para implementar un cursor (mouse) en el sitio web, es muy simple de implementar y con muchos tipos de temas para elegir, visite la documentación en el siguiente enlace:

<https://lusaxweb.github.io/Kursor/>

Modulo: Un módulo según definición propia es una sección de trabajo principal a comparación de un submódulo este puede ser denominado la estructura padre y los submódulos estructuras hijas. En este efecto en particular el módulo vendría a ser la siguiente clase:

- **Texturas.** - Para realizar este efecto en concreto nos basamos en insertar una clase en la etiqueta <html> una sola vez para que mediante este acción cambien todo el sitio web el cursor del mouse, también se puede aplicar esta clase a un elemento en específico en ese caso solo se vería afectado el área donde se aplicó la clase, las texturas del framework están enfocadas solo al cambio del cursor, estos pueden ser personalizados por el usuario siempre que este tenga la imagen del cursor en formato o extensión .cur

Submódulos: Considere llamarle submódulo, a los efectos o clases que están dentro de un módulo en este caso el módulo es texturas, si tendríamos un efecto similar con atributos diferentes en ese caso sería un submódulo, como también podría ser un submódulo clases o elementos que hereden del módulo atributos.

Estos submódulos son muy útiles ya que desde mi punto de vista ayuda a diversificar los efectos y las clases. Beneficiando al usuario final en la construcción de sitios web. Actualmente el framework XRL8 no tiene pensado sacar sub módulos para el efecto texturas.

FASE 2.- Codificar el efecto

Antes de ver esta sección a fondo quiero hacer hincapié en que se requirió de mucho ingenio, así como de investigación para hacer este efecto ya que mi persona no creía que fuera posible. A nivel de codificación también requirió capacitaciones y constante disciplina.

Bien luego de tener claro que efecto queríamos construir y porque lo íbamos a construir como se vio en el capítulo anterior se procedió a escribir código como se muestra a continuación.

Código completo

[parte 01 – texturas]

Figura 210. Código de texturas parte I

```
1 .puntero-1, .puntero-1 a {
2   margin: 0;
3   padding: 0;
4   box-sizing: border-box;
5   cursor: url("../multimedia/cursos/1_cursor.cur"), url("../multimedia/cursos/1_cursor.cur"), pointer;
6 }
7
8 .puntero-2, .puntero-2 a {
9   margin: 0;
10  padding: 0;
11  box-sizing: border-box;
12  cursor: url("../multimedia/cursos/2_cursor.cur"), url("../multimedia/cursos/2_cursor.cur"), pointer;
13 }
14
15 .puntero-3, .puntero-3 a {
16   margin: 0;
17   padding: 0;
18   box-sizing: border-box;
19   cursor: url("../multimedia/cursos/3_cursor.cur"), url("../multimedia/cursos/3_cursor.cur"), pointer;
20 }
21
22 .puntero-4, .puntero-4 a {
23   margin: 0;
24   padding: 0;
25   box-sizing: border-box;
26   cursor: url("../multimedia/cursos/4_cursor.cur"), url("../multimedia/cursos/4_cursor.cur"), pointer;
27 }
```

[parte 02 – texturas]

Figura 211. Código de texturas parte II

```
1 .puntero-5, .puntero-5 a {
2   margin: 0;
3   padding: 0;
4   box-sizing: border-box;
5   cursor: url("../multimedia/cursosores/5_cursor.cur"), url("../multimedia/cursosores/5_cursor.cur"), pointer;
6 }
7
8 .puntero-6, .puntero-6 a {
9   margin: 0;
10  padding: 0;
11  box-sizing: border-box;
12  cursor: url("../multimedia/cursosores/6_cursor.cur"), url("../multimedia/cursosores/6_cursor.cur"), pointer;
13 }
14
15 .puntero-7, .puntero-7 a {
16  margin: 0;
17  padding: 0;
18  box-sizing: border-box;
19  cursor: url("../multimedia/cursosores/7_cursor.cur"), url("../multimedia/cursosores/7_cursor.cur"), pointer;
20 }
21
22 .puntero-8, .puntero-8 a {
23  margin: 0;
24  padding: 0;
25  box-sizing: border-box;
26  cursor: url("../multimedia/cursosores/8_cursor.cur"), url("../multimedia/cursosores/8_cursor.cur"), pointer;
27 }
```

[parte 03 – texturas]

Figura 212. Código de texturas parte III

```
1 .puntero-9, .puntero-9 a {
2   margin: 0;
3   padding: 0;
4   box-sizing: border-box;
5   cursor: url("../multimedia/cursosores/9_cursor.cur"), url("../multimedia/cursosores/9_cursor.cur"), pointer;
6 }
7
8 .puntero-10, .puntero-10 a {
9   margin: 0;
10  padding: 0;
11  box-sizing: border-box;
12  cursor: url("../multimedia/cursosores/10_cursor.cur"), url("../multimedia/cursosores/10_cursor.cur"), pointer;
13 }
14
15 .puntero-11, .puntero-11 a {
16  margin: 0;
17  padding: 0;
18  box-sizing: border-box;
19  cursor: url("../multimedia/cursosores/11_cursor.cur"), url("../multimedia/cursosores/11_cursor.cur"), pointer;
20 }
21
22 .puntero-12, .puntero-12 a {
23  margin: 0;
24  padding: 0;
25  box-sizing: border-box;
26  cursor: url("../multimedia/cursosores/12_cursor.cur"), url("../multimedia/cursosores/12_cursor.cur"), pointer;
27 }
28
29 .puntero-13, .puntero-13 a {
30  margin: 0;
31  padding: 0;
32  box-sizing: border-box;
33  cursor: url("../multimedia/cursosores/13_cursor.cur"), url("../multimedia/cursosores/13_cursor.cur"), pointer;
34 }
```

[parte 04 – texturas]

Figura 213. Código de texturas parte IV

```
1 .puntero-14, .puntero-14 a {
2   margin: 0;
3   padding: 0;
4   box-sizing: border-box;
5   cursor: url("../multimedia/cursosores/14_cursor.cur"), url("../multimedia/cursosores/14_cursor.cur"), pointer;
6 }
7
8 .puntero-15, .puntero-15 a {
9   margin: 0;
10  padding: 0;
11  box-sizing: border-box;
12  cursor: url("../multimedia/cursosores/15_cursor.cur"), url("../multimedia/cursosores/15_cursor.cur"), pointer;
13 }
```

Descripción de las propiedades más significativas:

A continuación, se procederá a detallar las propiedades más usadas y su aplicación. También se mostrará la compatibilidad web de cada propiedad para su mejor comprensión dejamos una leyenda en la cual se observa en la figura 2.

Propiedad margin:0

Esta propiedad con el valor 0 lo que hace se desaparecer los márgenes exteriores que estos traigan por defecto ya que el propio navegador trae estilos por default para cada elemento HTML. Se puede observar que navegadores soportan esta propiedad en la figura 38.

Propiedad padding:0

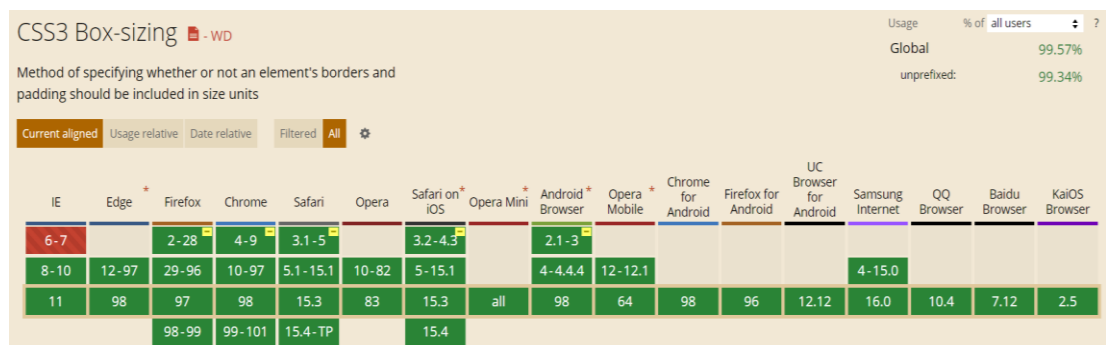
Con esta propiedad lo que se busca es desaparecer por completo los márgenes internos de cada etiqueta html que vienen por defecto en el navegador, por otra parte padding son los espacio internos que trae cada etiqueta web maquetado en HTML5 y margin son los espacios externos que le asigna el navegador a las etiquetas. Se puede observar que navegadores soportan esta propiedad en la figura 20.

Propiedad box-sizing:border-box

El navegador como ya lo mencionaba trae por default muchas propiedades una de ellas es box-sizing:content-box el cual hace que los

márgenes internos y el padding interno incrementen sus valores conforme le vallamos agregando mas contenido y atributos, esto es un problema ya que las medidas no son exactas y para hacerlas exactas se tendría que calcular también esos bordes de mas así como el padding de más para evitar eso sirve la propiedad `box-sizing: border-box` la cual hace que las dimensiones asignadas de alto y ancho se respete aun si le agregamos un padding o márgenes extras a cada clase.

Figura 214. Compatibilidad web, propiedad box-sizing



Principales dificultades:

Para este efecto en concreto la principal dificultad fue organizar de distintas formas posibles, así como ver las posibles conjugaciones que pueden hacer con el efecto para que el usuario al final no tenga problemas al implementarlo. Esas dificultades surgieron a raíz de las siguientes preguntas:

- ¿Si el usuario desea personalizar el cursor de su mouse como aria?

Al principio no se tenía idea luego de hacer pruebas y de tener el efecto creado, puedo afirmar que la mejor opción es solo cambiar la ruta de las imágenes cur por el de nuestra imagen, como también lo que se puede hacer es borrar la imagen por default del curso y reescribir el nombre del archivo borrado para que de esa forma el navegador agarre el curso que emos elegidos así evitamos escribir cualquier clase de css que a futuro puede hacer conflicto con las demás. Para que convertir su curso de jpg u otro formato de imagen a formato cur se usó la siguiente página web:

<https://convertio.co/nl/png-cur/>

¿Qué usamos para hacer las pruebas?

Se uso el navegador Google Chrome y opera mini, así como la siguiente página web que es un simulador online:

<http://www.responsinator.com>

Esto con motivos de asegurarnos que funcione en dispositivos móviles y tabletas. Ya que en base a lo trabajado no es del todo fiable usar el navegador para verificar la correcta maquetación de los elementos, así como tampoco es fiable depender de un software web, por ello se decidió diversificar en cuanto a las herramientas para probar las clases y efectos trabajados en el framework XRL8.

También usamos como editar de código visual studio code y su extensión live server para visualizar en vivo los efectos realizados en el css en tiempo real, también usamos el autoguardado que trae dicho editor de código para no estar guardando el proyecto a cada rato. Así conseguimos un mejor desempeño y evitamos el tedioso guardar cambios a cada rato, también evitamos el que la memoria cache se llene mostrándonos los mismos resultados.

FASE 3.- Hacerlo responsive

En caso se hubiera dado la oportunidad de maquetar más el efecto para que tome forma y se moldee, lo que hubiéramos hecho en ese caso sería:

- Usar la propiedad flex-box y sus posibles conjugaciones con cada uno de sus valores según sea el caso.
- Hubiéramos trabajado con la propiedad @media_query en las resoluciones de pantalla para celulares y tabletas. Siempre empezando a maquetar por la parte de celulares y luego tabletas.
- También hubiéramos usado las unidades de medida vh o rem para texto y contenedores ya que estas son imprescindibles a la hora de hacer responsive un sitio web.

Este modulo en especifico no se necesito maquetarlo, pero si se isieron las pruebas responsive correspondientes en caso hubiera habido algún suceso hubiéramos aplicados las propiedades correspondientes.

FASE 4.- Testing asegurarse que no tenga conflicto con otros efectos.

En esta parte del desarrollo de la clase se dispuso a probar poniéndolo con las demás clases juntándolas. Para ello lo que se hizo fue colocar dentro de un archivo index.html todas las clases trabajadas y ver como el efecto de texturas respondió. En algunos casos había conflicto con los demás efectos para corregir esto se procedió a implementar en el desarrollo dos cosas básicas la primera fue:

- Para detectar que clase era la que causaba el conflicto y ver en que archivo scss estaba. Se procedía a inspeccionar la página para detectar donde estaba el problema. En muchas ocasiones no se daba con el archivo causante del error entonces en ese caso aplicábamos la frase “Divide y vencerás”. Lo que se procedió hacer fue comentar de uno en uno cada efecto en el archivo principal que los llamaba esto reducía las líneas de código de css, además como teníamos en tiempo real los cambios fue mucho más rápido detectar el problema. Ya que no se tenía que recargar a cada rato además otro punto a favor era que se auto compilaba los cambios de sass automáticamente gracias a nodeJS el cual mediante la consola de comandos de Windows cuando detectaba un cambio en las líneas de código se auto compilaba y generando o reemplazando el archivo css en el cual hacíamos las pruebas, esto aumento la productividad para crear los efectos ya que al comentar de uno en uno cada archivo sass llegaba un momento donde desaparecía el error entonces revisábamos y nos dimos cuenta de que era uno o en algunos casos hasta tres efectos que hacían conflicto con la clase escrita.
- Se procedió a escribir mediante comentarios en un archivo llamado `_error_logs.scss` la clase que hace conflicto con la clase escrita, además de describir una nota breve en caso no se disponía de tiempo para solucionarlo luego, esto permitió tener un reporte más preciso de los incidentes que íbamos detectando.

FASE 5.- Aplicar la metodología OOCSS, llevándolo a SASS

- **Código css.** - Esta parte ya estaría completa por lo descrito anteriormente.

- **Código repetido.** - Identificación de código repetido

En este pequeño apartado luego de escribir y debugear el error se procedió a identificar el código css repetido para llevarlo a un archivo SCSS en sass, para su mayor optimización y aplicación de la metodología. En la clase descrita a continuación se observó que se repetía el siguiente código.

Figura 215 Código repetido del componente texturas.

```
1 margin: 0;  
2 padding: 0;  
3 box-sizing: border-box;
```

- **Crear un objeto.** - Se describe la lógica de la función para luego crear un mixing.

Una vez identificado el código css repetido se procede hacer una función la cual se ejecuta al llamado del código, mediante esta técnica nuestro código será más puro y limpio.

Paralelamente en esta parte se determina la lógica de las funciones y ciclos como lo son for, while, each. También determinamos el pase de parámetros, así como la condicional que llevaran dentro, por otra parte, y no menos importante se define el nombre con el cual se llamara a las funciones.

```
function textura_base(){  
margin:0;  
padding:0;  
box-sizing:border-box;  
}
```

Luego de tener la lógica se procede a llevarlo a SASS para ello tenemos dos opciones, la primera es usar @function ya que como su nombre ase referencia es para crear una función la otra opción es usar @mixin, pero ambas tienen diferencias. La primera retorna un valor, mientras @mixin solo llama a los valores en este caso no necesitamos que retorne un valor por ello usaremos @mixin. Entonces quedaría de la siguiente forma:

Figura 216 Creación del objeto `texturas_base` en función `mixin`

```
1 @mixin texturas_base(){
2     margin:0;
3     padding: 0;
4     box-sizing: border-box;
5 }
```

Figura 217 Creación del objeto en un ciclo `for` para la clase `puntero-n`

```
1 @for $i from 1 through $num_texturas {
2     .puntero-#{ $i },.puntero-#{ $i } a{
3         @include texturas_base();
4         cursor: url("../multimedia/cursosores/#{ $i }_cursor.cur"), url("../multimedia/cursosores/#{ $i }_cursor.cur"),pointer;
5     }
6 }
```

- **OOCSS**. – Luego de tener la función o la lógica para reciclar el código se procede a integrar la metodología OOCSS para ello debemos llamar al `mixin` para hacerlo solo necesitamos agregar `@include` “nombre del `mixin`”. Luego de integrarlo ya estará aplicada correctamente la metodología OOCSS

Figura 218 Metodología OOCSS en el componente `texturas`

```
1 .puntero-#{ $i },.puntero-#{ $i } a{
2     @include texturas_base();
3     cursor: url("../multimedia/cursosores/#{ $i }_cursor.cur"), url("../multimedia/cursosores/#{ $i }_cursor.cur"),pointer;
4 }
```

Cabe hacer una aclaración en esta parte, si bien tuvimos funciones, `for` y un `each` todos esos métodos se aplicó de forma directa la metodología OOCSS y no se requirió llamarla como se hace con el `mixin` ya que dentro del `for` se ejecutó de forma automática. Paralelamente en este caso además de ejecutarse el `for` dentro del mismo se ejecutó el `mixin`.

Bien volviendo al tema de las variables estas variables se encuentran en un archivo sass denominado variables valga la redundancia donde se encuentran las siguientes variables:

Figura 219 .Variables sass del componente `texturas`

```
1 $num_texturas:15;
```

Como se ha visto SASS es un preprocesador de css por ello se ejecuta antes del css tradicional esto permite añadir variables agregar condicionales, etc. Con la aplicación de la metodología como se observa se reutiliza el código css, además de que se hace escalable el proyecto. Ya que, si en un futuro queremos cambiar o añadir más propiedades css, solo cambiaríamos las variables o añadiríamos más propiedades en el `mixin`.

FASE 6.- Documentar en la web site www.phooldx.com

Finalmente, si el efecto paso por todos los de más filtros se procede a su documentación respectiva, para documentar todos los efectos se creó una página web de documentación usando clases y elementos del mismo framework. A continuación, se describirá como se procede a la documentación de la misma. Siguiendo estos pasos:

- Se copia las etiquetas de HTML5 junto con la clase a un txt temporal para no perder la sintaxis
- Se guarda los cambios en el compilador de SASS
- Como es la parte final se procedió hacer copias de seguridad para no perder el avance, en archivos RAR. Estos archivos estaban por versiones enumerados además se agregaba un pequeño texto describiendo el avance. Este guardado se hacía de manera local en un USB.
- Para una mayor seguridad del avance del código se procedió luego a copiar la versión del archivo RAR junto a la descripción del mismo.

Guardado de código en GitHub. - Luego de tener la copia del proyecto a nivel físico en un USB, se procedió a guardarlo en un repositorio de GitHub para lo cual se usó Git, para ello se procedió de la siguiente forma:

- Se guardaba todo el proyecto con el siguiente comando que se observa en la figura 12.
- Luego se verificaba los cambios guardados con el siguiente comando que se observa en la figura 13.
- Después se hacía el commit, es decir se le asignaba un nombre a los cambios realizados con el siguiente comando que se observa en la figura 14.
- Luego se procede a empaquetar los datos para que se suban al repositorio de GitHub con el siguiente código que se observa en la figura 15.
- Finalmente, luego de haber enviados el proyecto empaquetado hacia GitHub como se puede apreciar en la imagen, este queda guardado en dicho repositorio.

Figura 220. Visualización en github del componente texturas.



Se guarda en una escala de tiempo, con todos los cambios hechos desde la creación del proyecto, esto ayuda a tener un mejor control de versiones. Asimismo, también ayuda a que más personas se sumen al proyecto y puedan clonar el proyecto como trabajar por separado el proyecto, entre otras muchas ventajas.

- Así mismo se manejó un archivo llamado bitácora.txt donde aquí se registró todos los incidentes ocurridos, así como los cambios hechos, se registraba también las metas cumplidas, también las tareas nuevas para desarrollar al día siguiente. Al finalizar con un efecto por completo se quitaba de la lista de prioridades así dábamos pase al siguiente efecto o problema ha resolver. Esto servía para tener un mejor orden y saber ejecutar prioridades.
- Luego de tener guardado el avance y de registrar el avanzase, se procedió a copiar del archivo documentación Etiquetas.txt el fragmento de código que se copió anteriormente para luego pegarlo en www.nosetup.org esta página web se encargaba de convertir html en texto plano para poder visualizar html
- Antes de finalizar se procedió a crear un archivo llamado modulo_texturas.html donde se dividía en tres partes:
 - En la primera se describía un poco al efecto que era y que hacia
 - En la segunda parte se procedía a copiar el código ya formateado obtenido de www.nosetup.com para luego pegarlo.
 - En la tercera parte lo que se hacía era mostrar un apartado llamado estructura donde se describía como se llamaban las variables para que sea modificado en SASS.
 - Finalmente, el ejemplo listo para usarse queda de la siguiente forma:

Figura 221. Código html5 final del componente texturas parte I

```
<html lang="en" class="puntero-1">
  <body>
</body>
</html>
```

Figura 222. Código html5 final del componente texturas parte II

```
<html lang="en" class="puntero-2">
  <body>
  </body>
</html>
```

Figura 223. Código html5 final del componente texturas parte III

```
<html lang="en" class="puntero-3">
  <body>
  </body>
</html>
```

Figura 224. Código html5 final del componente texturas parte IV

```
<html lang="en" class="puntero-4">
  <body>
  </body>
</html>
```

Figura 225. Código html5 final del componente texturas parte V

```
<html lang="en" class="puntero-5">
  <body>
  </body>
</html>
```

Figura 226. Código html5 final del componente texturas parte VI

```
<html lang="en" class="puntero-6">
  <body>
  </body>
</html>
```

Figura 227. Código html5 final del componente texturas parte VII

```
<html lang="en" class="puntero-7">
  <body>
  </body>
</html>
```

Figura 228. Código html5 final del componente texturas parte VIII

```
<html lang="en" class="puntero-8">
  <body>
  </body>
</html>
```


Figura 229 .Código html5 final del componente texturas parte IX

```
<html lang="en" class="puntero-9">
  <body>
  </body>
</html>
```

Figura 230. Código html5 final del componente texturas parte X

```
<html lang="en" class="puntero-10">
  <body>
  </body>
</html>
```

Figura 231. Código html5 final del componente texturas parte XI

```
<html lang="en" class="puntero-11">
  <body>
  </body>
</html>
```

Figura 232.Código html5 final del componente texturas parte XII

```
<html lang="en" class="puntero-12">
  <body>
  </body>
</html>
```

Figura 233. Código html5 final del componente texturas parte XIII

```
<html lang="en" class="puntero-13">
  <body>
  </body>
</html>
```

Figura 234. Código html5 final del componente texturas parte XIV

```
<html lang="en" class="puntero-14">
  <body>
  </body>
</html>
```

Figura 235. Código html5 final del componente texturas parte XV

```
<html lang="en" class="puntero-15">
  <body>
  </body>
</html>
```

PARALLAX

FASE 1.- Investigación del efecto y su importancia

Como primer paso lo primero que se hizo fue investigar e indagar sobre el efecto parallax. El presente efecto sirve para simular el desplazamiento de nuestro contenido en una imagen mientras hacemos scroll, se creó este efecto porque le da un poco de más profesionalismo implementarlo en nuestra web. Este efecto entre tanto de sus beneficios el principal es la sensación que tendrá el usuario con la web y su interacción ya que debo admitir que la primera vez es bastante impactante.

Otro del motivo el cual fue elegido el efecto para formar parte del framework XRL8 fue que el proyecto presente debe estar a la altura de los frameworks que están en la actualidad siendo usados en la industria del desarrollo web como son:

- **TailwindCSS.**- Quiza sea este el framework de css más usado en los últimos 3 años sobre pasando a los demás por sus formas de integración con NodeJS y react. Este framework dentro de su contenido también tiene una clase llamada .bg-fixed que mezclada con otras clases se puede construir un efecto parallax. Gracias a sus componentes, estos a diferencia de los demás se caracteriza por no tener clases predefinidas para efectos sino todo lo contrario tiene un conjunto de clases en el cual el usuario puede elegir los efectos más simples para colocar a su diseño. Para revisar su documentación ingresar al siguiente enlace:

<https://tailwindcss.com/docs/background-attachment>

- **Materialize.**- Otro framework del medio es materialize el cual también cuenta con una sección llamada parallax el cual ayuda a construir este asombroso efecto. Lo podremos observar más detalladamente accediendo mediante el siguiente link:

<https://materializecss.com/parallax.html>

- **UIKIT.-** También este framework cuenta con una clase para hacer el efecto parallax la clase empleada en este caso tiene por nombre .uk-parallax para ver más sobre este framework recomiendo revisar la documentación en el siguiente enlace:

<https://getuikit.com/docs/parallax>

Modulo: Un módulo según definición propia es una sección de trabajo principal a comparación de un submódulo este puede ser denominado la estructura padre y los submódulos estructuras hijas. En este efecto en particular el módulo vendría a ser la siguiente clase:

- **Parallax sin before.** – Este efecto es el modulo principal ya que es compatible con las diferentes conjugaciones de la propiedad display excepto con la propiedad absolute.

Submódulos: Considere llamarle submódulo, a los efectos o clases que están dentro de un módulo en este caso el módulo es parallax sin before si bien comparten pocos atributos. Un submódulo según denominación propia vendría hacer un efecto similar al original, pero con características diferentes, hagamos un ejemplo para comprender esto de una manera mejor.

- **Parallax con before.** – Este es el submódulo ya que comparte pocos atributos con el primero, además de que son el mismo efecto la diferencia radica en que este efecto es compatible con la propiedad absolute, pero con las demás propiedades como el relative no se llevan bien.

Estos submódulos son muy útiles ya que desde mi punto de vista ayuda a diversificar los efectos y las clases. Beneficiando al usuario final en la construcción de sitios web.

FASE 2.- Codificar el efecto

Antes de ver esta sección a fondo mencionare como fue la parte de codificación y de lo que se requirió para poder crearlo, para la creación del módulo solo se requirió investigar en que consistía el efecto y decidir que propiedades eran las mas adecuadas para su construcción, luego de codificar

el modulo y pasarlo por las respectivas pruebas se nota que hacía conflicto con algunas propiedades así que a partir de ese punto surgió el submodulo. El submódulo si fue un dolor de cabeza hacer que haga lo mismo que el primero pero sin entrar en conflicto con las demás propiedades.

Bien luego de tener claro que efecto queríamos construir y porque lo íbamos a construir como se vio en el capítulo anterior se procedió a escribir código como se muestra a continuación.

Código completo

❖ Código del módulo parallax sin before

[parte 01 – parallax sin before]

Figura 236. Código parallax parte I

```
1  .parallax-contenido {
2    width: 100%;
3    box-sizing: border-box;
4    display: flex;
5    justify-content: center;
6    align-items: center;
7    margin: 0;
8    padding: 0;
9    height: 60vh;
10   background-image: url("../multimedia/i3.jpg");
11   background-size: cover;
12   background-attachment: fixed;
13 }
14
15 .parallax-contenido h2 {
16   width: 100%;
17   box-sizing: border-box;
18   display: flex;
19   justify-content: center;
20   align-items: center;
21   margin: 0;
22   padding: 0;
23   flex-direction: column;
24   font-family: arial black;
25   font-size: 2em;
26   color: tomato;
27   text-align: center;
28   text-transform: uppercase;
29 }
30
31 p.caracter {
32   color: #fe4918;
33   font-size: 2.2em;
34   margin: 0 0 -10px 0;
35   font-family: "Oswald", sans-serif;
36 }
```

❖ Código del submódulo parallax con before

[parte 01 – parallax con before]

Figura 237. Código parallax parte I

```
1 .contenido {
2   width: 100%;
3   margin: 0 auto;
4 }
5
6 .contenido ul {
7   list-style: none;
8   margin: 0;
9   padding: 0;
10 }
11
12 .parallax::before {
13   width: 100%;
14   height: 100%;
15   top: 0;
16   left: 0;
17   position: absolute;
18   background-repeat: no-repeat;
19   background-size: cover;
20   background-position: center center;
21   background-attachment: fixed;
22   -webkit-filter: brightness(0.8);
23   filter: brightness(0.8);
24 }
25
26 div.contador {
27   position: relative;
28   height: auto;
29 }
```

[parte 02 – parallax con before]

Figura 238. Código parallax parte II

```
1 div.contador::before {
2   background-image: url("../multimedia/bg-resumen.jpg");
3   content: "";
4 }
5
6 ul.resumen-evento {
7   position: relative;
8   padding: 120px 0px;
9 }
10
11 ul.resumen-evento li {
12   width: 24%;
13   float: left;
14   text-align: center;
15   color: white;
16   text-transform: uppercase;
17   font-family: "Oswald", sans-serif;
18   font-size: 24px;
19 }
20
21 p.numero {
22   color: #fe4918;
23   font-size: 4em;
24   display: block;
25   margin: 0 0 10px 0;
26   font-family: "Oswald", sans-serif;
27 }
```

[parte 03 – parallax con before]

Figura 239. Código parallax parte III

```
1  .biforito:before,  
2  .biforito:after {  
3    content: " ";  
4    display: table;  
5  }  
6  
7  .biforito:after {  
8    clear: both;  
9  }  
10  
11 @media only screen and (min-width: 768px) {  
12   ul.resumen-evento li {  
13     width: 25%;  
14   }  
15 }  
16  
17 @media only screen and (min-width: 480px) {  
18   .contenido {  
19     width: 95%;  
20   }  
21 }  
22  
23 @media only screen and (min-width: 768px) {  
24   .contenido {  
25     width: 90%;  
26   }  
27 }  
28  
29 @media only screen and (min-width: 992px) {  
30   .contenido {  
31     width: 90%;  
32     max-width: 1100px;  
33   }  
34 }
```

Descripción de las propiedades más significativas:

A continuación, se procederá a detallar las propiedades más usadas y su aplicación. También se mostrará la compatibilidad web de cada propiedad para su mejor comprensión dejamos una leyenda la cual se puede observar en la figura 2.

Propiedad background-repeat:no-repeat

Esta propiedad se usa para indicar si una imagen se va a repetir ya sea de manera automática, verticalmente, horizontalmente como también si simplemente no deseamos que se repita con el atributo no-repeat. La compatibilidad web de esta propiedad se observa a continuación.

Figura 240. Compatibilidad web propiedad background-repeat

CSS property: background-repeat

Usage: Global 97.41%

Current aligned Usage relative Date relative Filtered All

IE	Edge	Firefox	Chrome	Safari	Opera	Safari on iOS	Opera Mini	Android Browser	Opera Mobile	Chrome for Android	Firefox for Android	UC Browser for Android	Samsung Internet	QQ Browser	Baidu Browser	KaiOS Browser
6-10	12-97	2-96	4-97	3.1-15.1	10-82	3.2-15.1		2.1-4.4.4	12-12.1				4-15.0			
11	98	97	98	15.3	83	15.3	all	98	64	98	96	12.12	16.0	10.4	7.12	2.5
		98-99	99-101	15.4-TP		15.4										

Propiedad background-attachment:fixed

Esta propiedad lo que ara es establecer una imagen de fondo la cual se podrá desplazar con el resto de la página o si estará fija el valor fixed lo que hace es que se desplace. La compatibilidad web de esta propiedad se observa a continuación.

Figura 241. Compatibilidad web propiedad background-attachment

CSS property: background-attachment: fixed

Usage: Global 78.13% + 18.7% = 96.83%

Current aligned Usage relative Date relative Filtered All

IE	Edge	Firefox	Chrome	Safari	Opera	Safari on iOS	Opera Mini	Android Browser	Opera Mobile	Chrome for Android	Firefox for Android	UC Browser for Android	Samsung Internet	QQ Browser	Baidu Browser	KaiOS Browser
6-8		2-24		3.1-13.1	10.1	3.2-4.3		2.1-4.4.4	12-12.1				4-15.0			
9-10	12-97	25-96	4-97	14-15.1	11.5-82	5-15.1		2.1-4.4.4	12-12.1				4-15.0			
11	98	97	98	15.3	83	15.3	all	98	64	98	96	12.12	16.0	10.4	7.12	2.5
		98-99	99-101	15.4-TP		15.4										

Propiedad filter:brightness(0.8)

La propiedad filter hace un desenfoco y satura los efectos visuales para un elemento, el valor brightness se encarga de agregar brillo siendo 0 sin nada de brillo es decir color negro todo, el valor 1 es el brillo normal que trajo así mientras más sigo subiendo el valor se va agregando más brillo hasta llegar a quedar en blanco. También brightness acepta porcentajes y decimales.

La compatibilidad web de esta propiedad se observa en la figura 147.

Propiedad content:""

Para usar esta propiedad es requisito primordial usar un pseudo-elemento como:

- Before: Lo que hace esta propiedad es insertar elementos antes de la clase.
- After: Lo que hace esta propiedad es insertar elementos después de la clase.

Estos pseudo-elementos son usados para añadir contenido a un elemento con la propiedad content, aunque también en content puede ir palabras y estas se insertaran.

A continuación, veamos la compatibilidad web al usar esta propiedad, se observa lo mencionado en la figura 144.

Principales dificultades:

Para este efecto en concreto la principal dificultad fue organizar de distintas formas posibles, así como ver las posibles conjugaciones que pueden hacer con el efecto para que el usuario al final no tenga problemas al implementarlo. Esas dificultades surgieron a raíz de las siguientes preguntas:

- ¿Si el usuario quiere como podría hacer estático el efecto?

Cambiando la propiedad background-attachment:fixed por una con background-attachment:inherit, además una de las ventajas de escribir código css limpio es que se puede chancar el código de la clase reescribiendo el código y en ultimas instancias usar de la propiedad !important. Asi mismo la manera más fácil seria retirando la clase de la etiqueta HTML.

- ¿Si tengo problemas de integración o de distorsionamiento con el módulo parallax sin before que hago?

Así nació el submódulo parallax con before, es su complemento del modulo ya que si tienes problemas con el modulo en el submódulo esta maquetado con diferentes propiedades para solucionarlo. Así mismo funciona de manera inversa si tienes problemas de implementar el submódulo se recomienda usar el módulo.

¿Qué usamos para hacer las pruebas?

Se uso el navegador Google Chrome y opera mini, así como la siguiente página web que es un simulador online:

<http://www.responsinator.com>

Esto con motivos de asegurarnos que funcione en dispositivos móviles y tabletas. Ya que en base a lo trabajado no es del todo fiable usar el navegador para verificar la correcta maquetación de los elementos, así como tampoco es fiable depender de un software web, por ello se decidió diversificar en cuanto a las herramientas para probar las clases y efectos trabajados en el framework XRL8.

También usamos como editor de código visual studio code y su extensión live server para visualizar en vivo los efectos realizados en el css en tiempo real, también usamos el autoguardado que trae dicho editor de código para no estar guardando el proyecto a cada rato. Así conseguimos un mejor desempeño y evitamos el tedioso guardar cambios a cada rato, también evitamos el que la memoria cache se llene mostrándonos los mismos resultados.

FASE 3.- Hacerlo responsive

❖ Módulo parallax sin before

En este submódulo se aplicó solo flex-box para hacerlo responsive no se requirió de ninguna otra técnica.

❖ Submódulo parallax con before

Para hacerlo responsive se aplicaron las siguientes técnicas:

- Se uso valores en porcentaje en el ancho del efecto.
- Se uso flex-box junto a la conjugación de sus valores.
- Se uso @media en los siguientes puntos de corte:

Figura 242. Uso de media query en el componente parrallax parte I

```
1 @media only screen and (min-width:480px)
2 {
3   .contenido
4   {
5     width: 95%;
6   }
7 }
```

Figura 243. Uso de media query en el componente parrallax parte II

```
1 @media only screen and(min-width:768px){
2   ul.resumen-evento li{
3     width: 25%;
4   }
5   .contenido
6   {
7     width: 90%;
8   }
9 }
```

Figura 244. Uso de media query en el componente parrallax parte III

```
1 @media only screen and (min-width:992px)
2 {
3   .contenido
4   {
5     width:90%;
6     max-width: 1100px;
7   }
8 }
```

FASE 4.- Testing asegurarse que no tenga conflicto con otros efectos.

En esta parte del desarrollo de la clase se dispuso a probar poniéndolo con las demás clases juntándolas. Para ello lo que se hizo fue colocar dentro de un archivo index.html todas las clases trabajadas y ver como el módulo y el submódulo respondían. En algunos casos había conflicto con los demás efectos para corregir esto se procedió a implementar en el desarrollo dos cosas básicas la primera fue:

- Para detectar que clase era la que causaba el conflicto y ver en que archivo scss estaba. Se procedía a inspeccionar la página para detectar donde estaba el problema. En muchas ocasiones no se daba con el archivo causante del error entonces en ese caso aplicábamos la frase “Divide y vencerás”. Lo que se procedió hacer fue comentar de uno en uno cada efecto en el archivo principal que los llamaba esto reducía las líneas de código de css, además como teníamos en tiempo real los cambios fue mucho más rápido detectar el problema. Ya que no se tenía que recargar a cada rato además otro punto a favor era que se auto compilaba los cambios

de sass automáticamente gracias a nodeJS el cual mediante la consola de comandos de Windows cuando detectaba un cambio en las líneas de código se auto compilaba y generando o reemplazando el archivo css en el cual hacíamos las pruebas, esto aumento la productividad para crear los efectos ya que al comentar de uno en uno cada archivo sass llegaba un momento donde desaparecía el error entonces revisábamos y nos dimos cuenta de que era uno o en algunos casos hasta tres efectos que hacían conflicto con la clase escrita.

- Se procedió a escribir mediante comentarios en un archivo llamado `_error_logs.scss` la clase que hace conflicto con la clase escrita, además de describir una nota breve en caso no se disponía de tiempo para solucionarlo luego, esto permitió tener un reporte más preciso de los incidentes que íbamos detectando.
- Cuando teníamos el módulo se creyó que estaba listo y que no daría conflicto a un futuro, pero luego al probarlo con las otras clases nos dimos cuenta que hacia conflicto con los que tenía posicionamiento absoluto, una de las formas mas optimas de solucionar esto fue crear otra clase que haga lo mismo en este caso es el submodulo,el cual cumple la misma función que el modulo la diferencia esta en que el submódulo no entrara en conflicto con una etiqueta o clase con posicionamiento absoluto, pero se observó que este submódulo si hace conflicto con los posicionamientos relativos que pudieran tener alguna etiqueta o clase, bien entonces se decidió dejarlo ahí porque sería difícil encontrar una etiqueta HTML o clase que tenga posicionamiento absoluto y relativo a la vez aunque existe una propiedad que mezcla ambas la cual es `position:fixed` el cual mezcla ambas propiedades mencionadas para ese caso se probó tanto el modulo como el submódulo y para nuestra sorpresa nos dimos cuenta que prima en su mayor parte del posicionamiento absoluto así que no iso conflicto en lo mas absoluto con el submódulo.

FASE 5.- Aplicar la metodología OOCSS, llevándolo a SASS

- **Código css.** - Esta parte ya estaría completa por lo descrito anteriormente.

- **Código repetido.** - Identificación de código repetido

En este pequeño apartado luego de escribir y debugear el error se procedió a identificar el código css repetido para llevarlo a un archivo SCSS en sass, para su mayor optimización y aplicación de la metodología. Así que se procedió a reciclar el siguiente código:

❖ Modulo - Parallax sin before

Figura 245. Código repetido del componente parallax parte I

```
1  .parallax-contenido {
2    width: 100%;
3    box-sizing: border-box;
4    display: flex;
5    justify-content: center;
6    align-items: center;
7    margin: 0;
8    padding: 0;
9    height: 60vh;
10   background-image: url("../multimedia/i3.jpg");
11   background-size: cover;
12   background-attachment: fixed;
13 }
14
15 .parallax-contenido h2 {
16   width: 100%;
17   box-sizing: border-box;
18   display: flex;
19   justify-content: center;
20   align-items: center;
21   margin: 0;
22   padding: 0;
23   flex-direction: column;
24   font-family: arial black;
25   font-size: 2em;
26   color: tomato;
27   text-align: center;
28   text-transform: uppercase;
29 }
```

❖ Submódulo - Parallax con before

Figura 246. Código repetido del componente parallax parte II

```
1  .contenido ul {
2    list-style: none;
3    margin: 0;
4    padding: 0;
5  }
```

- **Crear un objeto.** - Se describe la lógica de la función para luego crear un `mixing`.

Una vez identificado el código `css` repetido se procede hacer una función la cual se ejecuta al llamado del código, mediante esta técnica nuestro código será más puro y limpio. Cabe recalcar que la `parallax sin before` comparte características similares al submódulo.

Paralelamente en esta parte se determina la lógica de las funciones y ciclos como lo son `for`, `while`, `each`. También determinamos el pase de parámetros, así como la condicional que llevaran dentro, por otra parte, y no menos importante se define el nombre con el cual se llamara a las funciones.

```
function sin_margen_espacios(){
  margin:0;
  padding:0;
}
```

```
function parallax_sin_before_base(){
  width:100%;
  box-sizing:border-box;
  display:flex;
  justify-content:center;
  align-items:center;
}
```

Luego de tener la lógica se procede a llevarlo a `SASS` para ello tenemos dos opciones, la primera es usar `@function` ya que como su nombre ase referencia es para crear una función la otra opción es usar `@mixin`, pero ambas tienen diferencias. La primera retorna un valor, mientras `@mixin` solo llama a los valores en este caso no necesitamos que retorne un valor por ello usaremos `@mixin`. Entonces quedaría de la siguiente forma:

Figura 247. Creación del objeto `parallax_sin_before_base` en función `mixin`

```
1 @mixin parallax_sin_before_base{
2   width:$parallax_sin_before_ancho;
3   box-sizing: border-box;
4   display: flex;
5   justify-content: center;
6   align-items: center;
7 }
```

Figura 248. Creación del objeto `sin_margen_espacios` en función `mixin`

```
1 @mixin sin_margen_espacios{
2     margin:0;
3     padding:0;
4 }
```

- **OOCSS**. – Luego de tener la función o la lógica para reciclar el código se procede a integrar la metodología OOCSS para ello debemos llamar al `mixin` para hacerlo solo necesitamos agregar `@include` “nombre del `mixin`”. Luego de integrarlo ya estará aplicada correctamente la metodología OOCSS

❖ Parallax sin before

Figura 249. Metodología OOCSS en el componente `parallax` parte I

```
1 .parallax-contenido{
2     @include parallax_sin_before_base;
3     @include sin_margen_espacios;
4     height: 60vh;
5     background-image: url($parallax_sin_before_ruta_img_fondo);
6     background-size: cover;
7     background-attachment: fixed;
8 }
```

Figura 250. Metodología OOCSS en el componente `parallax` parte II

```
1 .parallax-contenido h2{
2     @include parallax_sin_before_base;
3     @include sin_margen_espacios;
4     flex-direction: column;
5     font-family: arial black;
6     font-size:$parallax_sin_before_tamano_letra;
7     color: tomato;
8     text-align: center;
9     text-transform: uppercase;
10 }
```

❖ Parallax con before

Figura 251 .Metodología OOCSS en el componente `parallax` parte III

```
1 .contenido ul{
2     list-style: none;
3     @include sin_margen_espacios;
4 }
```

Bien volviendo al tema de las variables estas variables se encuentran en un archivo `sass` denominado `variables` valga la redundancia donde se encuentran las siguientes variables:

Figura 252. Variables sass del componente parallax

```
1 $parallax_sin_before_ancho: 100%;  
2 $parallax_sin_before_tamano_letra: 2em;  
3 $parallax_sin_before_ruta_img_fondo: "../multimedia/i3.jpg";
```

Como se ha visto SASS es un preprocesador de css por ello se ejecuta antes del css tradicional esto permite añadir variables agregar condicionales, etc. Con la aplicación de la metodología como se observa se reutiliza el código css, además de que se hace escalable el proyecto. Ya que, si en un futuro queremos cambiar o añadir más propiedades css, solo cambiaríamos las variables o añadiríamos más propiedades en el mixin.

FASE 6.- Documentar en la web site www.phooldx.com

Finalmente, si el efecto paso por todos los de más filtros se procede a su documentación respectiva, para documentar todos los efectos se creó una página web de documentación usando clases y elementos del mismo framework. A continuación, se describirá como se procede a la documentación de la misma. Siguiendo estos pasos:

- Se copia las etiquetas de HTML5 junto con la clase a un txt temporal para no perder la sintaxis
- Se guarda los cambios en el compilador de SASS
- Como es la parte final se procedió hacer copias de seguridad para no perder el avance, en archivos RAR. Estos archivos estaban por versiones enumerados además se agregaba un pequeño texto describiendo el avance. Este guardado se hacía de manera local en un USB.
- Para una mayor seguridad del avance del código se procedió luego a copiar la versión del archivo RAR junto a la descripción del mismo.

Guardado de código en GitHub. - Luego de tener la copia del proyecto a nivel físico en un USB, se procedió a guardarlo en un repositorio de GitHub para lo cual se usó Git, para ello se procedió de la siguiente forma:

- Se guardaba todo el proyecto con el siguiente comando que se observa en la figura 12.

- Luego se verificaba los cambios guardados con el siguiente comando que se observa en la figura 13.
- Después se hacía el commit, es decir se le asignaba un nombre a los cambios realizados con el siguiente comando que se observa en la figura 14.
- Luego se procede a empaquetar los datos para que se suban al repositorio de GitHub con el siguiente código que se observa en la figura 15.
- Finalmente, luego de haber enviados el proyecto empaquetado hacia GitHub como se puede apreciar en la imagen, este queda guardado en dicho repositorio

Figura 253. Visualización en github del componente parallax



Se guarda en una escala de tiempo, con todos los cambios hechos desde la creación del proyecto, esto ayuda a tener un mejor control de versiones. Asimismo, también ayuda a que más personas se sumen al proyecto y puedan clonar el proyecto como trabajar por separado el proyecto, entre otras muchas ventajas.

- Así mismo se manejó un archivo llamado bitácora.txt donde aquí se registró todos los incidentes ocurridos, así como los cambios hechos, se registraba también las metas cumplidas, también las tareas nuevas para desarrollar al día siguiente. Al finalizar con un efecto por completo se quitaba de la lista de prioridades así dábamos pase al siguiente efecto o problema ha resolver. Esto servía para tener un mejor orden y saber ejecutar prioridades.
- Luego de tener guardado el avance y de registrar el avanzase, se procedió a copiar del archivo documentación Etiquetas.txt el fragmento de código que se copió anteriormente para luego pegarlo en www.nosetup.org esta página web se encargaba de convertir html en texto plano para poder visualizar html

- Antes de finalizar se procedió a crear un archivo llamado modulo_parallax.html donde se dividía en tres partes:
 - En la primera se describía un poco al efecto que era y que hacia
 - En la segunda parte se procedía a copiar el código ya formateado obtenido de www.nosetup.com para luego pegarlo.
 - En la tercera parte lo que se hacía era mostrar un apartado llamado estructura donde se describía como se llamaban las variables para que sea modificado en SASS.
- Finalmente, el ejemplo listo para usarse queda de la siguiente forma:

❖ Modulo parallax sin before

Figura 254. Código html5 final del componente parallax parte I

```
<div class="parallax-contenido">
  <h2><p class="caracter">2</p>Papillas</h2>
  <h2><p class="caracter">4</p>Caiman</h2>
  <h2><p class="caracter">5</p>Honduras</h2>
</div>
```

❖ Submódulo parallax con before

Figura 255. Código html5 final del componente parallax parte II

```
<div class="contador parallax">
  <div class="contenido">
    <ul class="resumen-evento biforito">
      <li><p class="numero">6</p>Invitados</li>
      <li><p class="numero">15</p>Talleres</li>
      <li><p class="numero">3</p>Dias</li>
      <li><p class="numero">9</p>Conferencias</li>
    </ul>
  </div>
</div>
```

TABLAS

FASE 1.- Investigación del efecto y su importancia

Como primer paso lo primero que se iso fue investigar e indagar sobre el efecto, lo que se encontró que es imprescindible para hacer reportes de valores para el usuario. Uno de sus mayores beneficios es sin duda la legibilidad de los datos, se eligió también ya que es muy usado en sistemas web por los administradores para ver todo tipo de reportes por otra parte en las páginas web es muy limitado su uso ya que existen otros medios para mostrar a los usuarios datos el cual iremos viendo poco a poco.

Otro del motivo el cual fue elegido el efecto para formar parte del framework XRL8 fue que el proyecto presente debe estar a la altura de los frameworks que están en la actualidad siendo usados en la industria del desarrollo web como son:

- **Bootstrap.** - El presente framework cuenta con la clase `.table` para crear tablas para hacer reportes además cuenta con diversos diseños, si desean saber más sobre ello puede visitar la web oficial de documentación de dicho framework mediante el siguiente enlace:
<https://getbootstrap.com/docs/4.0/content/tables/>
- **Bulma.**- Este framework también cuenta con un modelo para hacer tablas, la clase con la cual se trabaja esta denominada como `.table`, para ver cómo se emplea y formas de uso revisar el siguiente enlace que los llevara a la documentación oficial de dicho framework:
<https://bulma.io/documentation/elements/table/>
- **TailwindCSS.**- Quisa sea este el framework de css más usado en los últimos 3 años sobre pasando a los demás por sus formas de integración con NodeJS y react. Este framework dentro de su contenido también una sección llamada componentes donde tiene de muestra diversas combinaciones para poder armar una tabla de contenido para listar información. Gracias a sus componentes, estos a diferencia de los demás se caracteriza por no tener clases predefinidas para efectos sino todo lo contrario tiene un conjunto de clases en el cual el usuario puede elegir los

efectos más simples para colocar a su diseño. Para revisar su documentación ingresar al siguiente enlace:

<https://tailwindcomponents.com/components/tables>

- **Foundation.-** Uno de los primeros frameworks en salir al mercado sin duda será foundation aunque con el pasar de los años ha perdido popularidad este framework también dentro de sus clases tiene una sección para hacer header o menú de navegación usando la clase `.menú`, que por lo general este menú es un menú de navegación en el cual se suele usar y poner en la parte superior de página web que luego servirá de como una barra de navegación. Para ver el uso de este se puede recurrir al siguiente link:

<https://get.foundation/sites/docs-v5/components/tables.html>

- **Materialize.-** Otro framework del medio es materialize el cual también cuenta con una sección llamada table el cual ayuda a construir tablas para visualizar contenido en dicha sección cuenta con cuatro clases distintas. Lo podremos observar más detalladamente accediendo mediante el siguiente link:

<https://materializecss.com/table>

- **UIKIT.-** También este framework cuenta con una clase para hacer tablas con listado de datos la clase empleada en este caso tiene por nombre `.uk-table` de la misma forma tenemos más clases empleadas ya que no es la única, para ver más sobre este framework recomiendo revisar la documentación en el siguiente enlace:

<https://getuikit.com/docs/table>

- **Semantic UI.-** A continuación tenemos otro framework de css muy completo la verdad el cual tiene soporte y componentes para ser usado con react, la clase la cual ayuda a crear el listado de datos en las tablas es denominada `.table` pero esta es mezclado junto a otras clases más para ver el ejemplo completo les recomiendo ir al siguiente enlace:

<https://semantic-ui.com/collections/table.html>

Módulo: Un módulo según definición propia es una sección de trabajo principal a comparación de un submódulo este puede ser denominado la estructura padre y los submódulos estructuras hijas. En este efecto en particular el módulo vendría a ser la siguiente clase:

- **Tabla simple.** - Para este efecto nos basamos en las tablas simple como su propio nombre lo menciona, ya que en muchos casos se usa para mostrar pocos valores. Este efecto tiene lo básico como lo son un efecto hover, colores y bordes finos para que no se vea muy tosco el diseño así mismo pueden ser personalizables por el usuario debido a que no tienen en sus propiedades la palabra reservada !important.

Submódulos: Considere llamarle submódulo, a los efectos o clases que están dentro de un módulo en este caso el módulo es tabla simple porque comparten atributos similares, los cuales los iremos viendo más adelante. Un submódulo según denominación propia vendría hacer un efecto similar al original, pero con características diferentes, hagamos un ejemplo para comprender esto de una manera mejor.

- **Tabla compleja.** - Este sería el submódulo ya que comparte algunas propiedades con el módulo y nos basamos en ello para su construcción, este efecto en particular también se usa en una tabla lo que lo hace diferente es usar el efecto hover tanto en horizontal como en vertical. Esto ayuda a encontrar el punto de intersección entre los datos es muy recomendable cuando se maneja muchos datos en una tabla con muchas filas y columnas.

FASE 2.- Codificar el efecto

Antes de ver esta sección a fondo quisiera hacer hincapié en lo muy duro que fue la elaboración del submódulo a nivel de codificación ya que se tuvo que indagar a profundidad a tal punto de revisar documentación en inglés y diversos foros.

Bien luego de tener claro que efecto queríamos construir y porque lo íbamos a construir como se vio en el capítulo anterior se procedió a escribir código como se muestra a continuación.

Código completo

❖ Código del módulo tabla simple

[parte 01 – tabla simple]

Figura 256. Código tablas parte I

```
1  .tabla-simple {
2    width: 100%;
3    padding: 0;
4    margin: 0 auto;
5    top: 0;
6    left: 0;
7    border-collapse: collapse;
8    background-color: black;
9    color: white;
10   border: solid 5px #13967b;
11 }
12 .tabla-simple tr {
13   border: solid 2px #13967b;
14 }
15 .tabla-simple tr:hover {
16   background-color: #96dbda;
17 }
18 .tabla-simple td, .tabla-simple th {
19   padding: 0.5em;
20 }
```

❖ Código del submódulo tabla compleja

[parte 01 – tabla compleja]

Figura 257. Código tablas parte II

```
1  *,
2  *::after
3  *::before {
4    box-sizing: border-box;
5  }
6
7  main {
8    @include tabla_simple_and_compleja_base(false);
9  }
10
11 .tabla-compleja{
12   font-family: sans-serif;
13   @include tabla_simple_and_compleja_base(position);
14   table-layout: fixed;
15   border-collapse: collapse;
16   overflow: hidden;
17   tr{
18     position: relative;
19     display: -ms-flexbox;
20     display: flex;
21     &:hover{
22       background:$tabla_compleja_hover_horizontal;
23     }
24   }
25   tr::after{
26     content: "";
27     @include tabla_simple_and_compleja_base(position);
28     height: 100%;
29     top: 0;
30     left: 0;
31     background: $color-plomo-3;
32     z-index: -2;
33   }
34   tr:nth-child(even)::after {
35     background: #ddd;
36   }
```

[parte 02 – tabla compleja]

Figura 258. Código tablas parte III

```
37     td{
38         @include tabla_simple_and_compleja_base(false);
39         line-height: 2.5;
40         text-align: center;
41         -ms-flex: auto;
42         flex: auto;
43         &:hover{
44             position: relative;
45         }
46     }
47     td:hover::before{
48         content: "";
49         @include tabla_simple_and_compleja_base(position);
50         height: 1000px;
51         background:$tabla_compleja_hover_vertical;
52         top: -500px;
53         z-index: -1;
54     }
55     th{
56         @include tabla_simple_and_compleja_base(false);
57         display: flex;
58         justify-content: center;
59         align-items: center;
60     }
61 }
62 }
```

Descripción de las propiedades más significativas:

A continuación, se procederá a detallar las propiedades más usadas y su aplicación. También se mostrará la compatibilidad web de cada propiedad para su mejor comprensión dejamos una leyenda en la cual se indica si tiene compatibilidad web, esto se puede apreciar en la figura 2.

Propiedad background-color: #ddd

Esta propiedad se usa para darle un color de fondo para esta ocasión se decidió que sería un blanco humo. Ya que es más comercial y combina con cualquier tonalidad.

A continuación, veamos la compatibilidad web al usar esta propiedad en los principales navegadores esto se puede apreciar en la figura 5.

Propiedad border: solid 2px rgba(19,150,123,1.00)

En esta propiedad le damos un border sólido, esto quiere decir que el borde será visible, además de que será una línea definida sin interrupciones.

Luego la otra característica es que tendrá una anchura de 2px, dicho de mejor forma será una línea fina. A continuación, veamos la compatibilidad web al usar esta propiedad en los principales navegadores esto se puede apreciar en la figura 37.

Propiedad content:””

Para usar esta propiedad es requisito primordial usar un pseudo-elemento como:

- Before: Lo que hace esta propiedad es insertar elementos antes de la clase.
- After: Lo que hace esta propiedad es insertar elementos después de la clase.

Estos pseudo-elementos son usados para añadir contenido a un elemento con la propiedad content, aunque también en content puede ir palabras y estas se insertaran.

A continuación, veamos la compatibilidad web al usar esta propiedad en los principales navegadores esto se puede observar en la figura 144.

Propiedad overflow:hidden

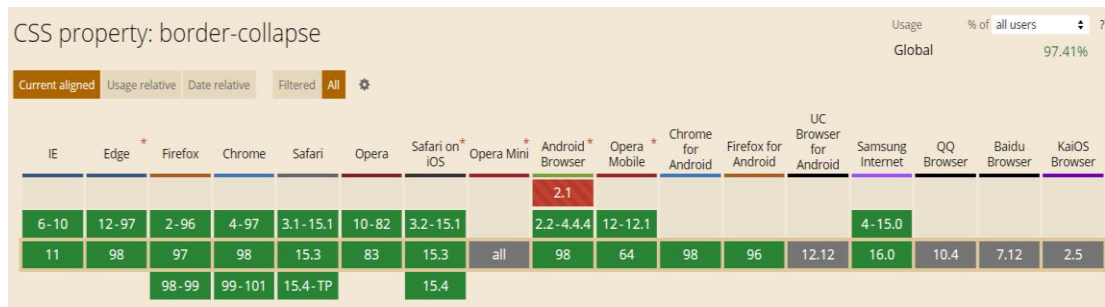
Esta propiedad lo que hace es ocultar los elementos que salgan de su contenedor mostrando solo una parte del contenido que encaje exacto en dicho contenedor al mismo tiempo lo que hace es eliminar el scroll horizontalmente.

A continuación, veamos la compatibilidad web al usar esta propiedad en los principales navegadores esto lo podemos apreciar en la figura 181.

Propiedad border-collapse:collapse

Con esta propiedad lo que se hace es fusionar automáticamente todos los bordes adyacentes lo cual viene en la propiedad por default al aplicar la propiedad border-collapse con el atributo collapse lo que sucede es que se juntaran esos bordes para que se pueda visualizar como un solo borde.

Figura 259. Compatibilidad web propiedad border-collapse



Principales dificultades:

Para este efecto en concreto la principal dificultad fue organizar de distintas formas posibles, así como ver las posibles conjugaciones que pueden hacer con el efecto para que el usuario al final no tenga problemas al implementarlo. Esas dificultades surgieron a raíz de las siguientes preguntas:

- ¿Si el usuario quiere como podría hacerlo responsive?

Esta fue la pregunta más difícil de contestar ya que ambas clases tanto el módulo como el submódulo fueron construidos bajo estándares de responsive pero llegan a un punto de quiebre cuando tiene demasiadas columnas. La solución fue crear una clase enfocada al responsive pero aún sigue en desarrollo, esto también sucede en los frameworks más usados pero en XRL8 lo podemos solucionar reduciendo las columnas en las tablas hasta que sea lanzado oficialmente la versión responsive para las tablas.

- ¿Cómo hago que el hover se intercepte verticalmente con el hover horizontal?

Esto surgió cuando codificábamos el submodulo, la solución puede sonar simple, pero implicó leer mucha documentación de terceros y probar diferentes propiedades hasta que al final llegamos a la conclusión que debíamos usar un hover junto a un before en una misma línea de código.

¿Qué usamos para hacer las pruebas?

Se uso el navegador Google Chrome y opera mini, así como la siguiente página web que es un simulador online:

<http://www.responsinator.com>

Esto con motivos de asegurarnos que funcione en dispositivos móviles y tabletas. Ya que en base a lo trabajado no es del todo fiable usar el navegador para verificar la correcta maquetación de los elementos, así como tampoco es fiable depender de un software web, por ello se decidió diversificar en cuanto a las herramientas para probar las clases y efectos trabajados en el framework XRL8.

También usamos como editor de código visual studio code y su extensión live server para visualizar en vivo los efectos realizados en el css en tiempo real, también usamos el autoguardado que trae dicho editor de código para no estar guardando el proyecto a cada rato. Así conseguimos un mejor desempeño y evitamos el tedioso guardar cambios a cada rato, también evitamos el que la memoria cache se llene mostrándonos los mismos resultados.

FASE 3.- Hacerlo responsive

En caso se hubiera dado la oportunidad de maquetar más el efecto para que tome forma y se moldee, lo que hubiéramos hecho en ese caso sería:

- Usar la propiedad flex-box y sus posibles conjugaciones con cada uno de sus valores según sea el caso.
- Hubiéramos trabajado con la propiedad @media_query en las resoluciones de pantalla para celulares y tabletas. Siempre empezando a maquetar por la parte de celulares y luego tabletas.
- También hubiéramos usado las unidades de medida vh o rem para texto y contenedores ya que estas son imprescindibles a la hora de hacer responsive un sitio web.

FASE 4.- Testing asegurarse que no tenga conflicto con otros efectos.

En esta parte del desarrollo de la clase se dispuso a probar poniéndolo con las demás clases juntándolas. Para ello lo que se hizo fue colocar dentro de un archivo index.html todas las clases trabajadas y ver como el efecto de tablas respondía. En algunos casos había conflicto con los demás efectos

para corregir esto se procedió a implementar en el desarrollo dos cosas básicas la primera fue:

- Para detectar que clase era la que causaba el conflicto y ver en que archivo scss estaba. Se procedía a inspeccionar la página para detectar donde estaba el problema. En muchas ocasiones no se daba con el archivo causante del error entonces en ese caso aplicábamos la frase “Divide y vencerás”. Lo que se procedió hacer fue comentar de uno en uno cada efecto en el archivo principal que los llamaba esto reducía las líneas de código de css, además como teníamos en tiempo real los cambios fue mucho más rápido detectar el problema. Ya que no se tenía que recargar a cada rato además otro punto a favor era que se auto compilaba los cambios de sass automáticamente gracias a nodeJS el cual mediante la consola de comandos de Windows cuando detectaba un cambio en las líneas de código se auto compilaba y generando o reemplazando el archivo css en el cual hacíamos las pruebas, esto aumento la productividad para crear los efectos ya que al comentar de uno en uno cada archivo sass llegaba un momento donde desaparecía el error entonces revisábamos y nos dimos cuenta de que era uno o en algunos casos hasta tres efectos que hacían conflicto con la clase escrita.
- Se procedió a escribir mediante comentarios en un archivo llamado `_error_logs.scss` la clase que hace conflicto con la clase escrita, además de describir una nota breve en caso no se disponía de tiempo para solucionarlo luego, esto permitió tener un reporte más preciso de los incidentes que íbamos detectando.

FASE 5.- Aplicar la metodología OOCSS, llevándolo a SASS

- **Código css.** - Esta parte ya estaría completa por lo descrito anteriormente.
- **Código repetido.** - Identificación de código repetido

En este pequeño apartado luego de escribir y debugear el error se procedió a identificar el código css repetido para llevarlo a un archivo SCSS en sass, para su mayor optimización y aplicación de la metodología. Así que se procedió a reciclar el siguiente código:

❖ Modulo tabla simple

Figura 260. Código repetido del componente tablas parte I

```
1  .tabla-simple {
2    width: 100%;
3    padding: 0;
4    margin: 0 auto;
5    top: 0;
6    left: 0;
7    border-collapse: collapse;
8    background-color: black;
9    color: white;
10   border: solid 5px #13967b;
11 }
```

❖ Submódulo tabla compleja

Figura 261. Código repetido del componente tablas parte II

```
1  main {
2    width: 100%;
3    padding: 0;
4    margin: 0 auto;
5    top: 0;
6    left: 0;
7 }
```

Figura 262. Código repetido del componente tablas parte III

```
1  .tabla-compleja {
2    font-family: sans-serif;
3    width: 100%;
4    padding: 0;
5    margin: 0 auto;
6    top: 0;
7    left: 0;
8    position: absolute;
9    table-layout: fixed;
10   border-collapse: collapse;
11   overflow: hidden;
12 }
```

Figura 263. Código repetido del componente tablas parte IV

```
1  .tabla-compleja tr::after {
2    content: "";
3    width: 100%;
4    padding: 0;
5    margin: 0 auto;
6    top: 0;
7    left: 0;
8    position: absolute;
9    height: 100%;
10   top: 0;
11   left: 0;
12   background: #eee;
13   z-index: -2;
14 }
```

Figura 264. Código repetido del componente tablas parte V

```
1  .tabla-compleja td {
2    width: 100%;
3    padding: 0;
4    margin: 0 auto;
5    top: 0;
6    left: 0;
7    line-height: 2.5;
8    text-align: center;
9    -ms-flex: auto;
10   flex: auto;
11 }
```

Figura 265. Código repetido del componente tablas parte VI

```
1  .tabla-compleja td:hover::before {
2    content: "";
3    width: 100%;
4    padding: 0;
5    margin: 0 auto;
6    top: 0;
7    left: 0;
8    position: absolute;
9    height: 10000px;
10   background: white;
11   top: -5000px;
12   z-index: -1;
13 }
```

Figura 266. Código repetido del componente tablas parte VII

```
1  .tabla-compleja th {
2    width: 100%;
3    padding: 0;
4    margin: 0 auto;
5    top: 0;
6    left: 0;
7    display: flex;
8    justify-content: center;
9    align-items: center;
10 }
```

- **Crear un objeto.** - Se describe la lógica de la función para luego crear un mixing.

Una vez identificado el código css repetido se procede hacer una función la cual se ejecuta al llamado del código, mediante esta técnica nuestro código será más puro y limpio. Cabe recalcar que el módulo tabla simple comparte características con el submódulo.

Paralelamente en esta parte se determina la lógica de las funciones y ciclos como lo son for, while, each. También determinamos el pase de parámetros, así como la condicional que llevaran dentro, por otra parte, y no menos importante se define el nombre con el cual se llamara a las funciones.

```
function table_simple_and_compleja_base (){  
  width:100%;  
  padding:0;  
  margin:0 auto;  
  top:0;  
  left:0;  
}
```

Luego de tener la lógica se procede a llevarlo a SASS para ello tenemos dos opciones, la primera es usar @function ya que como su nombre ase referencia es para crear una función la otra opción es usar @mixin, pero ambas tienen diferencias. La primera retorna un valor, mientras @mixin solo llama a los valores en este caso no necesitamos que retorne un valor por ello usaremos @mixin. Entonces quedaría de la siguiente forma:

Figura 267. Creación del objeto `tabla_simple_and_compleja_base` en función `mixin`

```
1 @mixin tabla_simple_and_compleja_base($condicion){  
2   width:$tabla_compleja_y_simple_ancho;  
3   padding:0;  
4   margin:0 auto;  
5   top: 0;  
6   left: 0;  
7   @if($condicion == 'position'){  
8     position:$tabla_compleja_position;  
9   }  
10 }
```

- **OOCSS**. – Luego de tener la función o la lógica para reciclar el código se procede a integrar la metodología OOCSS para ello debemos llamar al mixin para hacerlo solo necesitamos agregar @include “nombre del mixin”. Luego de integrarlo ya estará aplicada correctamente la metodología OOCSS

❖ Modulo tabla simple

Figura 268. Metodología OOCSS en el componente tablas parte I

```
1  .tabla-simple
2  {
3      @include tabla_simple_and_compleja_base(false);
4      border-collapse:$tabla-simple-tipo-linea;
5      background-color: $color-negro-1;
6      color: $color-blanco-1;
7      border: solid 5px rgba(19,150,123,1.00);
8
9      tr{
10         border: solid 2px rgba(19,150,123,1.00);
11         &:hover{
12             background-color:rgba(150,219,218,1.00);
13         }
14     }
15     td,th{
16         padding:$tabla-simple-padding-casilleros;
17     }
18 }
```

❖ Submódulo tabla compleja

Figura 269. Metodología OOCSS en el componente tablas parte II

```
2  main {
3      @include tabla_simple_and_compleja_base(false);
4  }
```

Figura 270. Metodología OOCSS en el componente tablas parte III

```
1  .tabla-compleja{
2      font-family: sans-serif;
3      @include tabla_simple_and_compleja_base(position);
4      table-layout: fixed;
5      border-collapse: collapse;
6      overflow: hidden;
```

Figura 271. Metodología OOCSS en el componente tablas parte IV

```
1  tr::after{
2      content: "";
3      @include tabla_simple_and_compleja_base(position);
4      height: 100%;
5      top: 0;
6      left: 0;
7      background: $color-plomo-3;
8      z-index: -2;
9  }
```

Cabe recalcar que la escritura del modulo y submódulo fue anidada para evitar así mayor redundancia en el código. Bien volviendo al tema de las variables estas variables se encuentran en un archivo sass denominado variables valga la redundancia donde se encuentran las siguientes variables:

❖ Modulo tabla simple

Figura 272. Variables sass del componente tablas parte I

```
1 $tabla-simple-tipo-linea: collapse;
2 $tabla-simple-padding-casilleros: 0.5em;
```

❖ Submódulo tabla alterna

Figura 273. Variables sass del componente tablas parte II

```
1 $tabla_compleja_position:absolute;
2 $tabla_compleja_y_simple_ancho:100%;
3 $tabla_compleja_hover_horizontal:rgb(195, 224, 226);
4 $tabla_compleja_hover_vertical:white;
5 $tabla-color-borde: #333;
6 $tabla-cabecera-color: #333 !default;
7 $tabla-cabecera-texto-color: #fff;
8 $tabla-fila-impar: aliceblue;
9 $tabla-fila-par: #a9a9a93b;
10 $tabla-hover: teal;
11 $tabla-thead-borde: 0.2rem;
12 $tabla-letra: 0.8rem;
```

Como se ha visto SASS es un preprocesador de css por ello se ejecuta antes del css tradicional esto permite añadir variables agregar condicionales, etc. Con la aplicación de la metodología como se observa se reutiliza el código css, además de que se hace escalable el proyecto. Ya que, si en un futuro queremos cambiar o añadir más propiedades css, solo cambiaríamos las variables o añadiríamos más propiedades en el mixin.

FASE 6.- Documentar en la web site www.phooldx.com

Finalmente, si el efecto paso por todos los de más filtros se procede a su documentación respectiva, para documentar todos los efectos se creó una página web de documentación usando clases y elementos del mismo framework. A continuación, se describirá como se procede a la documentación de la misma. Siguiendo estos pasos:

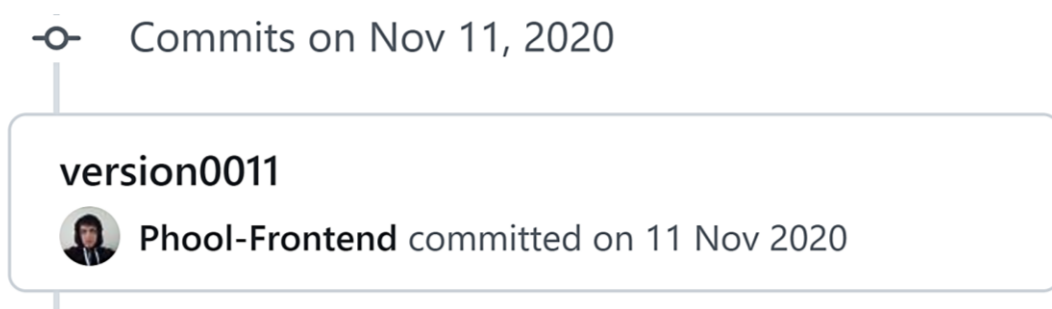
- Se copia las etiquetas de HTML5 junto con la clase a un txt temporal para no perder la sintaxis
- Se guarda los cambios en el compilador de SASS

- Como es la parte final se procedió hacer copias de seguridad para no perder el avance, en archivos RAR. Estos archivos estaban por versiones enumerados además se agregaba un pequeño texto describiendo el avance. Este guardado se hacía de manera local en un USB.
- Para una mayor seguridad del avance del código se procedió luego a copiar la versión del archivo RAR junto a la descripción del mismo.

Guardado de código en GitHub. - Luego de tener la copia del proyecto a nivel físico en un USB, se procedió a guardarlo en un repositorio de GitHub para lo cual se usó Git, para ello se procedió de la siguiente forma:

- Se guardaba todo el proyecto con el siguiente comando que se observa en la figura 12.
- Luego se verificaba los cambios guardados con el siguiente comando que se observa 13.
- Después se hacía el commit, es decir se le asignaba un nombre a los cambios realizados con el siguiente comando que se observa en la figura14.
- Luego se procede a empaquetar los datos para que se suban al repositorio de GitHub con el siguiente código que se observa en la figura 15.
- Finalmente, luego de haber enviados el proyecto empaquetado hacia GitHub como se puede apreciar en la imagen, este queda guardado en dicho repositorio.

Figura 274. Visualización en github del componente tablas



Se guarda en una escala de tiempo, con todos los cambios hechos desde la creación del proyecto, esto ayuda a tener un mejor control de versiones. Asimismo, también ayuda a que más personas se sumen al proyecto y puedan clonar el proyecto como trabajar por separado el proyecto, entre otras muchas ventajas.

- Así mismo se manejó un archivo llamado bitácora.txt donde aquí se registró todos los incidentes ocurridos, así como los cambios hechos, se registraba también las metas cumplidas, también las tareas nuevas para desarrollar al día siguiente. Al finalizar con un efecto por completo se quitaba de la lista de prioridades así dábamos pase al siguiente efecto o problema ha resolver. Esto servía para tener un mejor orden y saber ejecutar prioridades.
- Luego de tener guardado el avance y de registrar el avanzase, se procedió a copiar del archivo documentación Etiquetas.txt el fragmento de código que se copió anteriormente para luego pegarlo en www.nosetup.org esta página web se encargaba de convertir html en texto plano para poder visualizar html
- Antes de finalizar se procedió a crear un archivo llamado modulo_tablas.html donde se dividía en tres partes:
 - En la primera se describía un poco al efecto que era y que hacia
 - En la segunda parte se procedía a copiar el código ya formateado obtenido de www.nosetup.com para luego pegarlo.
 - En la tercera parte lo que se hacía era mostrar un apartado llamado estructura donde se describía como se llamaban las variables para que sea modificado en SASS.
- Finalmente, el ejemplo listo para usarse queda de la siguiente forma:

❖ Modulo tabla simple

Figura 275. Código html5 final del componente tablas parte I

```
<table border="1" width="100%" class="tabla-simple">
  <tr>
    <th></th>
    <th>NOMBRE</th>
    <th>INGRESOS</th>
    <th>EGRESOS</th>
    <th>PATRIMONIO</th>
  </tr>
  <tr>
    <th>1</th>
    <td>Martin</td>
    <td>$/890</td>
    <td>$/90</td>
    <td>$/10 000</td>
  </tr>
  <tr>
    <th>2</th>
    <td>Alejandro</td>
    <td>$/1 113</td>
    <td>$/104</td>
    <td>$/11 005</td>
  </tr>
  <tr>
    <th>3</th>
    <td>Don Juan</td>
    <td>$/1 418</td>
    <td>$/19</td>
    <td>$/20 000</td>
  </tr>
  <tr>
    <th>4</th>
    <td>Patrick</td>
    <td>$/1 223</td>
    <td>$/224</td>
    <td>$/12 115</td>
  </tr>
  <tr>
    <th>5</th>
    <td>Cheiz</td>
    <td>$/208</td>
    <td>$/129</td>
    <td>$/7 930</td>
  </tr>
</table>
```

❖ Submódulo tabla compleja

Figura 276. Código html5 final del componente tablas parte II

```
<table border="1" class="tabla-compleja">
  <tr>
    <th></th>
    <th>NOMBRE</th>
    <th>INGRESOS</th>
    <th>EGRESOS</th>
    <th>PATRIMONIO</th>
  </tr>
  <tr>
    <th>1</th>
    <td>Martin</td>
    <td>$/890</td>
    <td>$/90</td>
    <td>$/10 000</td>
  </tr>
  <tr>
    <th>2</th>
    <td>Alejandro</td>
    <td>$/1 113</td>
    <td>$/104</td>
    <td>$/11 005</td>
  </tr>
  <tr>
    <th>3</th>
    <td>Don Juan</td>
    <td>$/1 418</td>
    <td>$/19</td>
    <td>$/20 000</td>
  </tr>
  <tr>
    <th>4</th>
    <td>Patrick</td>
    <td>$/1 223</td>
    <td>$/224</td>
    <td>$/12 115</td>
  </tr>
  <tr>
    <th>5</th>
    <td>Cheiz</td>
    <td>$/208</td>
    <td>$/129</td>
    <td>$/7 930</td>
  </tr>
</table>
```

IMAGEN REDONDA

FASE 1.- Investigación del efecto y su importancia

Como primer paso lo primero que se iso fue investigar e indagar sobre el efecto en este caso se descubrió que es muy usado en foros como también en fotos de perfil de los sitios web como puede ser un panel de administrador, etc. Con este efecto quedara en cuestión de segundos cualquier imagen con bordes redondeados. Por otra parte, como valor agregado también se puede dar estos bordes redondeados a cualquier etiqueta HTML que tenga la clase.

Otro del motivo el cual fue elegido el efecto para formar parte del framework XRL8 fue que el proyecto presente debe estar a la altura de los frameworks que están en la actualidad siendo usados en la industria del desarrollo web como son:

- **Bootstrap.** - El presente framework cuenta con la clase `.rounded-circle` para poder crear bordes en las imágenes si desean saber más sobre ello puede visitar la web oficial de documentación de dicho framework mediante el siguiente enlace:
<https://getbootstrap.com/docs/4.3/utilities/borders/>

- **Bulma.**- Este framework también cuenta con un modelo para dar borde redondeado a las imágenes con la clase `image`, para ver cómo se emplea y formas de uso revisar el siguiente enlace que los llevara a la documentación oficial de dicho framework:
<https://bulma.io/documentation/elements/image/>

- **TailwindCSS.**- Quisa sea este el framework de css más usado en los últimos 3 años sobre pasando a los demás por sus formas de integración con NodeJS y react. Este framework dentro de su contenido también tiene una clase llamada `.rounded-full` que mezclada con otras clases se puede aplicar a etiquetas e imágenes con un borde redondeado. Gracias a sus componentes, estos a diferencia de los demás se caracteriza por no tener clases predefinidas para efectos sino todo lo contrario tiene un conjunto de clases en el cual el usuario puede elegir los efectos más simples para

colocar a su diseño. Para revisar su documentación ingresar al siguiente enlace:

<https://tailwindcss.com/docs/border-radius>

- **Materialize.**- Otro framework del medio es materialize el cual también cuenta con una sección llamada media en la cual se encuentra una clase llamada responsive-img el cual ayuda a construir imágenes con bordes redondeados. Lo podremos observar más detalladamente accediendo mediante el siguiente link:

<https://materializecss.com/media-css.html>

- **UIKIT.**- También este framework cuenta con una clase para hacer imágenes redondeadas, con la clase .uk-border-circle se puede emplear para obtener el resultado que queremos, para ver más sobre este framework recomiendo revisar la documentación en el siguiente enlace:

<https://getuikit.com/docs/utility>

- **Semantic UI.**- A continuación tenemos otro framework de css muy completo la verdad el cual tiene soporte y componentes para ser usado con react, la clase la cual ayuda a crear imágenes con bordes redondeados es denominada .ui.circular , podemos ver una gran cantidad de ejemplos mediante el siguiente enlace:

<https://semantic-ui.com/elements/image.html>

Modulo: Un módulo según definición propia es una sección de trabajo principal a comparación de un submódulo este puede ser denominado la estructura padre y los submódulos estructuras hijas. En este efecto en particular el módulo vendría a ser la siguiente clase:

- **Imagen redonda.** - Para este efecto nos basamos en la idea de hacer que el usuario pueda darles un formato circular a sus imágenes, así como etiquetas HTML para ello la imagen o etiqueta HTML toma el máximo valor que puedan tener en ancho y alto para luego aplicar cualquiera de nuestras tres clases según se desee estas están expresadas en valores pequeño, mediano, grande y uno por default.

FASE 2.- Codificar el efecto

Antes de ver esta sección a fondo quisiera mencionar que a nivel de código no fue tan complicado como se creía pero, lo que si dio dificultad fueron las imágenes que perdían nitidez al usar la clase, en capítulos posteriores veremos cómo se solucionó.

Bien luego de tener claro que efecto queríamos construir y porque lo íbamos a construir como se vio en el capítulo anterior se procedió a escribir código como se muestra a continuación.

Código completo

[parte 01 – img redonda]

Figura 277 Código de imagen redonda parte I

```
1  .img-redondo {
2    min-width: 80px !important;
3    min-height: 80px !important;
4    max-width: 20%;
5    max-height: 20%;
6    border: solid 3px gold;
7    border-radius: 100%;
8  }
9
10 .img_redondo_grande {
11   min-width: 80px !important;
12   min-height: 80px !important;
13   max-width: 20%;
14   max-height: 20%;
15   border: solid 3px gold;
16   border-radius: 100%;
17   width: 15vh;
18   height: 15vh;
19 }
20
21 .img_redondo_mediano {
22   min-width: 80px !important;
23   min-height: 80px !important;
24   max-width: 20%;
25   max-height: 20%;
26   border: solid 3px gold;
27   border-radius: 100%;
28   width: 12vh;
29   height: 12vh;
30 }
```

[parte 02 – img redonda]

Figura 278 Código de imagen redonda parte II

```
1  .img_redondo_peque {  
2    min-width: 80px !important;  
3    min-height: 80px !important;  
4    max-width: 20%;  
5    max-height: 20%;  
6    border: solid 3px gold;  
7    border-radius: 100%;  
8    width: 3vh;  
9    height: 3vh;  
10 }
```

Descripción de las propiedades más significativas:

A continuación, se procederá a detallar las propiedades más usadas y su aplicación. También se mostrará la compatibilidad web de cada propiedad para su mejor comprensión dejamos una leyenda la cual se puede observar en la figura 2.

Propiedad border-radius:100%

Esta propiedad lo que hace es darle el 100% de borde a las etiquetas HTML así como a la imagen donde se aplique, con esto lo que se consigue es que se convierta en una esfera totalmente redonda.

A continuación, veamos la compatibilidad web al usar esta propiedad en la figura 39.

Propiedad max-width:20%:

Se uso max-width:20%, para el ancho del presente efecto se usó esta propiedad ya que tiene la particularidad de que como máximo llegara al 20% de su capacidad a diferencia de la propiedad width, que solo es un tipo de ancho especifico mientras max-width impide que el ancho en width se más largo que max-width, dicho de una forma más sencilla max-width siempre será el valor más alto tanto que sobrescribirá al valor de width si este es menor al de max-width.

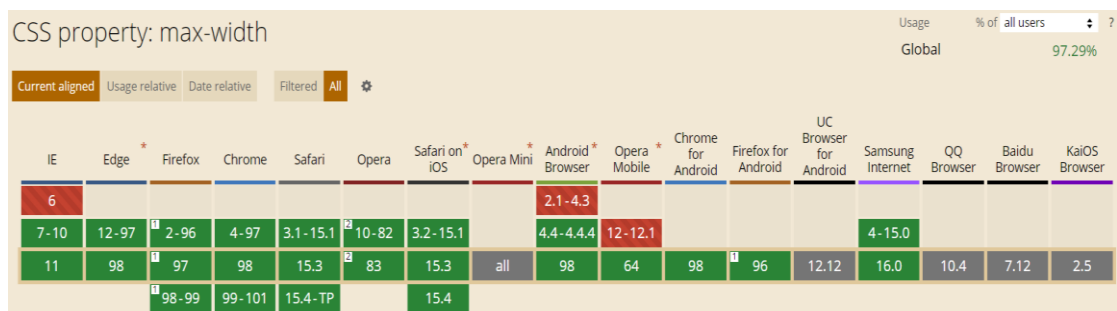
Se empleo max-width con el valor 20% para el ancho de este efecto ya que si bien esta propiedad tiene diversas unidades de medidas como:

- EM
- VH
- PIXELES
- REM
- PORCENTAJE

Se fue descartando una a una al momento de probarla, ya que la al usar la unidad EM este se distorsionaba por su propia naturaleza, básicamente esta propiedad usa el ancho de la pantalla del dispositivo donde estemos y las va ajustando así que fue descartado junto a las otras, para finalmente quedarnos con el porcentaje este valor ofrece una gran compensación a la hora de maquetar así que es excelente para crear una clase que haga responsive cualquier etiqueta HTML donde sea puesta.

A continuación, veamos la compatibilidad web al usar esta propiedad en los principales navegadores.

Figura 279. Compatibilidad web propiedad max-width

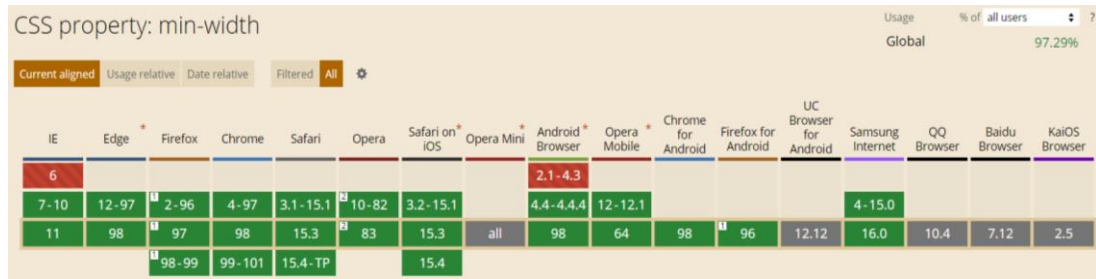


Propiedad min-width:20%:

Esta propiedad tiene la particularidad de que como mínimo llegara al 20% de su capacidad a diferencia de la propiedad width, ya que solo es un tipo de ancho especifico mientras min- width impide que el ancho en width se más bajo que min-width, dicho de una forma más sencilla min- width siempre será el valor más bajo tanto que sobrescribirá al valor de width si este es menor al de min-width.

A continuación, vemos la compatibilidad web al usar esta propiedad en los principales navegadores.

Figura 280 Compatibilidad web propiedad min-width

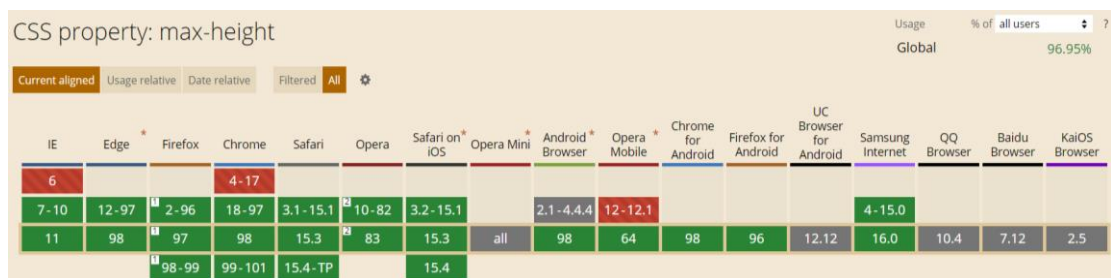


Propiedad max-height:20%:

Esta propiedad tiene la particularidad de que como máximo llegara al 20% de su capacidad a diferencia de la propiedad height, que solo es un tipo de alto especifico mientras max-height impide que el alto en height se más largo que max- height, dicho de una forma más sencilla max- height siempre será el valor más alto tanto que sobrescribirá al valor de height si este es mayor al de max-height.

A continuación, vemos la compatibilidad web al usar esta propiedad en los principales navegadores.

Figura 281 Compatibilidad web propiedad max-height



Propiedad min-height:20%:

Esta propiedad tiene la particularidad de que como mínimo llegara al 20% de su capacidad a diferencia de la propiedad height, ya que solo es un tipo de alto especifico mientras min- height impide que el alto en height se más bajo que min-height, dicho de una forma más sencilla min-height siempre será el valor más bajo tanto que sobrescribirá al valor de height si este es menor al de min-height.

A continuación, vemos la compatibilidad web al usar esta propiedad en los principales navegadores.

Figura 282 Compatibilidad web propiedad min-height



Principales dificultades:

Para este efecto en concreto la principal dificultad fue organizar de distintas formas posibles, así como ver las posibles conjugaciones que pueden hacer con el efecto para que el usuario al final no tenga problemas al implementarlo. Esas dificultades surgieron a raíz de las siguientes preguntas:

- ¿Si el usuario quiere una altura y ancho en específico como lo aria?

Para solucionarlo de forma eficiente y sin mucho rodeos se decidió crear tres tipos de bordes redondeados uno default que te redondea toda la imagen sin tomar en cuenta el ancho y largo que pueda tener y añadiendo estas propiedades por default. Para eso creamos uno en formato grande con extensión en vh para que se adapte al alto del dispositivo desde donde entre el usuario para que sea así proporcional, luego el segundo valor es un valor mediano también expresado en vh, finalmente creamos uno con valor mínimo lo suficiente para una foto de perfil o la foto de algún foro en internet.

¿Qué usamos para hacer las pruebas?

Se uso el navegador Google Chrome y opera mini, así como la siguiente página web que es un simulador online:

Esto con motivos de asegurarnos que funcione en dispositivos móviles y tabletas. Ya que en base a lo trabajado no es del todo fiable usar el navegador para verificar la correcta maquetación de los elementos, así como

tampoco es fiable depender de un software web, por ello se decidió diversificar en cuanto a las herramientas para probar las clases y efectos trabajados en el framework XRL8.

También usamos como editor de código visual studio code y su extensión live server para visualizar en vivo los efectos realizados en el css en tiempo real, también usamos el autoguardado que trae dicho editor de código para no estar guardando el proyecto a cada rato. Así conseguimos un mejor desempeño y evitamos el tedioso guardar cambios a cada rato, también evitamos el que la memoria cache se llene mostrándonos los mismos resultados.

FASE 3.- Hacerlo responsive

Con este efecto en particular fue sencillo ya que las propiedades max-width, max-height, min-width, min-height estaban en % además de que no se necesitó una maquetación extra como veremos más adelante en el presente informe. En caso se hubiera dado la oportunidad de maquetar más el efecto para que tome forma y se moldee, lo que hubiéramos hecho en ese caso sería:

- Usar la propiedad flex-box y sus posibles conjugaciones con cada uno de sus valores según sea el caso.
- Hubiéramos trabajado con la propiedad @media_query en las resoluciones de pantalla para celulares y tabletas. Siempre empezando a maquetar por la parte de celulares y luego tabletas.
- También hubiéramos usado las unidades de medida vh o rem para texto y contenedores ya que estas son imprescindibles a la hora de hacer responsive un sitio web.

FASE 4.- Testing asegurarse que no tenga conflicto con otros efectos.

En esta parte del desarrollo de la clase se dispuso a probar poniéndolo con las demás clases juntándolas. Para ello lo que se hizo fue colocar dentro de un archivo index.html todas las clases trabajadas y ver como el efecto imagen redonda respondía. En algunos casos había conflicto con los demás

efectos para corregir esto se procedió a implementar en el desarrollo dos cosas básicas la primera fue:

- Para detectar que clase era la que causaba el conflicto y ver en que archivo scss estaba. Se procedía a inspeccionar la página para detectar donde estaba el problema. En muchas ocasiones no se daba con el archivo causante del error entonces en ese caso aplicábamos la frase “Divide y vencerás”. Lo que se procedió hacer fue comentar de uno en uno cada efecto en el archivo principal que los llamaba esto reducía las líneas de código de css, además como teníamos en tiempo real los cambios fue mucho más rápido detectar el problema. Ya que no se tenía que recargar a cada rato además otro punto a favor era que se auto compilaba los cambios de sass automáticamente gracias a nodeJS el cual mediante la consola de comandos de Windows cuando detectaba un cambio en las líneas de código se auto compilaba y generando o reemplazando el archivo css en el cual hacíamos las pruebas, esto aumento la productividad para crear los efectos ya que al comentar de uno en uno cada archivo sass llegaba un momento donde desaparecía el error entonces revisábamos y nos dimos cuenta de que era uno o en algunos casos hasta tres efectos que hacían conflicto con la clase escrita.
- Se procedió a escribir mediante comentarios en un archivo llamado `_error_logs.scss` la clase que hace conflicto con la clase escrita, además de describir una nota breve en caso no se disponía de tiempo para solucionarlo luego, esto permitió tener un reporte más preciso de los incidentes que íbamos detectando.
- Nos dimos una gran sorpresa cuando vimos que no bastaba con aplicar `border-radius` a las imágenes ya que quedaban desproporcionadas por el tamaño de la imagen, necesariamente debía haber un ancho y alto como propiedades base. Pero esto crearía una mala practica ya que si el desarrollador web quiere modificar la clase tendría que agregar `important` a la clase o sino no podría visualizar los cambios o en el peor de los casos tendría que aplicar estilos en línea es decir en la propia etiqueta HTML. Para ello luego de leer documentación se encontró la propiedad `max-width` y `min-width` así como las propiedades `max-height`, `min-height` las cuales resolvieron los dos problemas iniciales que tuvimos.

FASE 5.- Aplicar la metodología OOCSS, llevándolo a SASS

- **Código css.** - Esta parte ya estaría completa por lo descrito anteriormente.

- **Código repetido.** - Identificación de código repetido

En este pequeño apartado luego de escribir y debugear el error se procedió a identificar el código css repetido para llevarlo a un archivo SCSS en sass, para su mayor optimización y aplicación de la metodología. Así que se procedió a reciclar el siguiente código:

Figura 283 Código repetido del componente imagen redonda parte I

```
1 .img-redondo {
2   min-width: 80px !important;
3   min-height: 80px !important;
4   max-width: 20%;
5   max-height: 20%;
6   border: solid 3px gold;
7   border-radius: 100%;
8 }
9
10 .img_redondo_grande {
11   min-width: 80px !important;
12   min-height: 80px !important;
13   max-width: 20%;
14   max-height: 20%;
15   border: solid 3px gold;
16   border-radius: 100%;
17   width: 15vh;
18   height: 15vh;
19 }
```

Figura 284 Código repetido del componente imagen redonda parte II

```
1 .img_redondo_mediano {
2   min-width: 80px !important;
3   min-height: 80px !important;
4   max-width: 20%;
5   max-height: 20%;
6   border: solid 3px gold;
7   border-radius: 100%;
8   width: 12vh;
9   height: 12vh;
10 }
11
12 .img_redondo_peque {
13   min-width: 80px !important;
14   min-height: 80px !important;
15   max-width: 20%;
16   max-height: 20%;
17   border: solid 3px gold;
18   border-radius: 100%;
19   width: 3vh;
20   height: 3vh;
21 }
```

- **Crear un objeto.** - Se describe la lógica de la función para luego crear un mixing.

Una vez identificado el código css repetido se procede hacer una función la cual se ejecuta al llamado del código, mediante esta técnica nuestro código será más puro y limpio.

Paralelamente en esta parte se determina la lógica de las funciones y ciclos como lo son for, while, each. También determinamos el pase de parámetros, así como la condicional que llevaran dentro, por otra parte, y no menos importante se define el nombre con el cual se llamara a las funciones.

```
function img_redonda_base(){
  min-width:80px!important;
  min-height:80px!important;
  max-width:20%;
  max-height:20%;
  border:solid 3px gold;
  border-radius:100%;
}
```

Luego de tener la lógica se procede a llevarlo a SASS para ello tenemos dos opciones, la primera es usar @function ya que como su nombre ase referencia es para crear una función la otra opción es usar @mixin, pero ambas tienen diferencias. La primera retorna un valor, mientras @mixin solo llama a los valores en este caso no necesitamos que retorne un valor por ello usaremos @mixin. Entonces quedaría de la siguiente forma:

Figura 285 Creación del objeto *img_redonda_base* en función *mixin*

```
1 @mixin img_redonda_base{
2   min-width:80px!important;
3   min-height:80px!important;
4   max-width:20%;
5   max-height:20%;
6   border: solid 3px $color-amarillos-1;
7   border-radius:100%;
8 }
```

- **OOCSS**. – Luego de tener la función o la lógica para reciclar el código se procede a integrar la metodología OOCSS para ello debemos llamar al mixin para hacerlo solo necesitamos agregar @include “nombre del mixin”. Luego de integrarlo ya estará aplicada correctamente la metodología OOCSS

Figura 286 Metodología OOCSS en el componente imagen redonda

```
1  .img-redondo{
2      @include img_redonda_base;
3  }
4
5  .img_redondo_grande{
6      @include img_redonda_base;
7      width:15vh;
8      height:15vh;
9  }
10
11 .img_redondo_mediano{
12     @include img_redonda_base;
13     width:12vh;
14     height:12vh;
15 }
16
17 .img_redondo_peque{
18     @include img_redonda_base;
19     width:3vh;
20     height:3vh;
21 }
```

Bien volviendo al tema de las variables estas variables se encuentran en un archivo sass denominado variables valga la redundancia donde se encuentran las siguientes variables:

Figura 287 Variables sass del componente imagen redonda parte I

```
$color-amarillos-1: gold;
```

Figura 288 Variables sass del componente imagen redonda parte II

```
$imagen_redonda_redondeo:100%;
$imagen_redonda_grande_ancho:15vh;
$imagen_redonda_mediano_ancho:15vh;
$imagen_redonda_peque_ancho:15vh;
```

Como se ha visto SASS es un preprocesador de css por ello se ejecuta antes del css tradicional esto permite añadir variables, así como agregar condicionales, etc. Con la aplicación de la metodología como se observa se reutiliza el código css, además de que se hace escalable el proyecto. Ya que si en un futuro queremos cambiar o añadir más propiedades css, solo cambiaríamos las variables o añadiríamos más propiedades en el mixin.

FASE 6.- Documentar en la web site www.phooldx.com

Finalmente, si el efecto paso por todos los de más filtros se procede a su documentación respectiva, para documentar todos los efectos se creó una página web de documentación usando clases y elementos del mismo framework. A continuación, se describirá como se procede a la documentación de la misma. Siguiendo estos pasos:

- Se copia las etiquetas de HTML5 junto con la clase a un txt temporal para no perder la sintaxis
- Se guarda los cambios en el compilador de SASS
- Como es la parte final se procedió hacer copias de seguridad para no perder el avance, en archivos RAR. Estos archivos estaban por versiones enumerados además se agregaba un pequeño texto describiendo el avance. Este guardado se hacía de manera local en un USB.
- - Para una mayor seguridad del avance del código se procedió luego a copiar la versión del archivo RAR junto a la descripción del mismo.

Guardado de código en GitHub. - Luego de tener la copia del proyecto a nivel físico en un USB, se procedió a guardarlo en un repositorio de GitHub para lo cual se usó Git, para ello se procedió de la siguiente forma:

- Se guardaba todo el proyecto con el siguiente comando que se observa en la figura 12.
- Luego se verificaba los cambios guardados con el siguiente comando que se observa en la figura 13.
- Después se hacía el commit, es decir se le asignaba un nombre a los cambios realizados con el siguiente comando que se observa en la figura 14.
- Luego se procede a empaquetar los datos para que se suban al repositorio de GitHub con el siguiente código que se observa en la figura 15.
- Finalmente, luego de haber enviados el proyecto empaquetado hacia GitHub como se puede apreciar en la imagen, este queda guardado en dicho repositorio.

Figura 289. Visualización en github del componente imagen redonda



Se guarda en una escala de tiempo, con todos los cambios hechos desde la creación del proyecto, esto ayuda a tener un mejor control de versiones. Asimismo, también ayuda a que más personas se sumen al proyecto y puedan clonar el proyecto como trabajar por separado el proyecto, entre otras muchas ventajas.

- Así mismo se manejó un archivo llamado bitácora.txt donde aquí se registró todos los incidentes ocurridos, así como los cambios hechos, se registraba también las metas cumplidas, también las tareas nuevas para desarrollar al día siguiente. Al finalizar con un efecto por completo se quitaba de la lista de prioridades así dábamos pase al siguiente efecto o problema ha resolver. Esto servía para tener un mejor orden y saber ejecutar prioridades.
- Luego de tener guardado el avance y de registrar el avanzase, se procedió a copiar del archivo documentación Etiquetas.txt el fragmento de código que se copió anteriormente para luego pegarlo en www.nosetup.org esta página web se encargaba de convertir html en texto plano para poder visualizar html
- Antes de finalizar se procedió a crear un archivo llamado modulo_img_redonda.html donde se dividía en tres partes:
 - En la primera se describía un poco al efecto que era y que hacia
 - En la segunda parte se procedía a copiar el código ya formateado obtenido de www.nosetup.com para luego pegarlo.
 - En la tercera parte lo que se hacía era mostrar un apartado llamado estructura donde se describía como se llamaban las variables para que sea modificado en SASS.
- Finalmente, el ejemplo listo para usarse queda de la siguiente forma:

Figura 290. Código html5 final del componente imagen redonda

```
  
  

```

BOTONES

FASE 1.- Investigación del efecto y su importancia

Como primer paso lo primero que se iso fue investigar e indagar sobre el efecto aplicado a los botones, lo que se obtuvo fue que los botones sirven para enviar formularios. También se puede aplicar a paginas ecomerse en el cual se hace clic para comprar algún servicio, sea cual sea el caso un botón hace que el usuario interactúe con el sistema mediante un clic esta interacción se puede dar al dar clic en un botón para ver más contenido en la sección, como también en una paginación de un sitio web de noticias, indistintamente ese botón suele tener estilos como también suele tener formas en XRL8 lo que se iso fue tener desde el botón clásico e ir implementando más diseño conforme valla pasando el tiempo.

Otro del motivo el cual fue elegido el efecto para formar parte del framework XRL8 fue que el proyecto presente debe estar a la altura de los frameworks que están en la actualidad siendo usados en la industria del desarrollo web como son:

- **Bootstrap.** - El presente framework cuenta con la clase btn seguido del nombre del botón el cual se desea implementar así mismo cuenta con diferentes tamaños para poder combinarlos de acuerdo a la ocasión, si desean saber más sobre ello puede visitar la web oficial de documentación de dicho framework mediante el siguiente enlace:
<https://getbootstrap.com/docs/4.1/components/buttons/>
- **Bulma.-** Este framework también cuenta con una gran variedad de botones desde botones responsive como también botones de diversos colores como también estilos y diversos bordes no podría faltar los botones que estén en disabled, para ver cómo se emplea y formas de uso revisar el siguiente enlace que los llevara a la documentación oficial de dicho framework:
<https://bulma.io/documentation/elements/button/>
- **TailwindCSS.-** Quisa sea este el framework de css más usado en los últimos 3 años sobre pasando a los demás por sus formas de integración con NodeJS y react. Este framework dentro de su contenido tiene en su

mayor parte clases semi preparadas, con esto me refiero a que las clases son propiedades que modifican a las propiedades por default sobrescribiéndolas y modificando esos valores, asimismo también cuenta con una sección de componentes en ellos si existen ejemplos de cómo construir botones personalizados, pero tiene un costo adquirirlos. Para revisar su documentación ingresar al siguiente enlace:

<https://tailwindui.com/components/application-ui/elements/buttons>

- **Foundation.-** Uno de los primeros frameworks en salir al mercado sin duda será foundation aunque con el pasar de los años a perdido popularidad este framework nos ofrece botones en diferentes tamaños y colores siendo el color celeste el color default de la gran mayoría de sus colores, otras de sus virtudes es que tienen clases para hacer responsive los botones, estos botones pueden ser personalizados ya que están escritos en el preprocesador de css SASS . Para ver el uso de este se puede recurrir al siguiente link:

<https://get.foundation/sites/docs/button.html>

- **Materialize.-** Otro framework del medio es materialize el cual también cuenta con una sección llamada buttons en la cual muestra los diferentes estilos y características de sus botones y algo que me llamo la atención es que los botones tienen un efecto al darle clic que se asemeja a la caída de una gota de agua en formas de ondas además de ofrecer las mismas clases que los frameworks mostrados anteriormente. Lo podremos observar más detalladamente accediendo mediante el siguiente link:

<https://materializecss.com/buttons.html>

- **UIKIT.-** También este framework cuenta con una clase para hacer botones esa clase principal esta denominada como uk-button. recomiendo revisar la documentación en el siguiente enlace:

<https://getuikit.com/docs/button>

- **Semantic UI.**- A continuación tenemos otro framework de CSS muy completo la verdad el cual tiene soporte y componentes para ser usado con React, algo que me llamo la atención es que cuentan algunos botones con animación, la clase la cual ayuda a crear estos botones es `ui button`, podemos ver una gran cantidad de ejemplos mediante el siguiente enlace:

<https://semantic-ui.com/elements/button.html>

Modulo: Un módulo según definición propia es una sección de trabajo principal a comparación de un submódulo este puede ser denominado la estructura padre y los submódulos estructuras hijas. Cabe aclarar que en esta sección en específico no hubo submódulos ya que estos son módulos independientes por la misma razón que no comparten similitudes o propiedades. Así que son cuatro módulos que pertenecen a la misma sección llamada botones. Estos módulos están expresados en las siguientes clases:

- Botón clásico. - Para este efecto nos basamos en las propiedades básicas que tiene un botón, así como el color que suele ser azulado, con una capa de borde sencilla por último con letras blancas para resaltar el texto.
- Botón luminoso. - Este efecto al aplicarlo a un botón se logra tener colores muy llamativos, para que estos hagan contraste el color de fondo debe ser negro como principal requisito, así mismo están disponibles dos diseños el primero con la clase `btn_led` es algo sutil ya que está enfocado hacia los bordes dichos bordes se mueven hacia la derecha como un letrero luminoso. El segundo diseño es algo más extravagante ya que es similar al primer diseño, pero se diferencia en el comportamiento ya que al darle clic este muestra todo el botón colores de tipo plasmáticos por ello le dimos una clase con el nombre `btn plasma`.
- Botón espejo. – Este efecto surgió para dar mayor libertad al usuario a la hora de elegir ya que el efecto anterior es usado normalmente cuando existe fondos oscuros en este caso este el botón espejo está creado para usar en tonalidades claras. Cuenta con la peculiaridad de que al pasar el mouse se muestra una ráfaga con si fuera el reflejo de un espejo mientras

el botón adquiere un relleno, así mismo se puede revertir el proceso añadiendo la clase `btn_inverso` el cual hace que con un `hover` se quite el relleno.

- Botón animado. Finalmente luego de tener botones clásicos, botones oscuros, botones claros se procedió a crear algo que nos diferencie del resto para ello creamos un botón con animación pero que esta animación sea algo sutil como para todo público nada en específico la animación consiste en manchas que se esparcen o se podría decir que parece que se oxida el botón con solo hacer `hover` en él, se le asigno la clase `btn_termita` por el aspecto visual que llegase a tener.

FASE 2.- Codificar el efecto

Antes de ver esta sección a fondo quisiera mencionar que a nivel de codificación el módulo que más trabajo costo tanto a nivel de codificación como de investigación fue el botón animado.

Bien luego de tener claro que efecto queríamos construir y porque lo íbamos a construir como se vio en el capítulo anterior se procedió a escribir código como se muestra a continuación.

Código completo

❖ Código del módulo botón clásico

Figura 291. Código de botones parte I

```
1 .btn_clasico {
2   border: none;
3   outline: none;
4   color: white;
5   background-color: #1a73e8;
6   padding: 0.625rem 1.25rem;
7   cursor: pointer;
8   border-radius: 0.312rem;
9   font-size: 1rem;
10  margin: 0 auto;
11  box-sizing: border-box;
12 }
13
14 .btn_clasico:hover {
15   background-color: #287ae6;
16   box-shadow: 0 1px 1px 0 rgba(66, 133, 244, 0.45), 0 1px 3px 1px rgba(66, 133, 244, 0.3);
17   margin: 0 auto;
18   box-sizing: border-box;
19 }
20
21 .btn_clasico:active {
22   outline: none;
23   border: 2px solid #1a73e8;
24 }
```

❖ Código del módulo botón luminoso

[parte nº1 - botón luminoso - led]

Figura 292. Código de botones parte II

```
1 .btn_led:active {
2   width: initial;
3   height: initial;
4   border: 2px solid skyblue;
5 }
6
7 .btn_led {
8   cursor: pointer;
9   text-decoration: none;
10  width: calc(20vw + 6px);
11  height: calc(8vw + 6px);
12  background-image: linear-gradient(90deg, #00C0FF 0,
13  border-radius: 5px;
14  border: none;
15  text-transform: uppercase;
16  font-size: 3vw;
17  color: white;
18  font-weight: 700;
19  display: flex;
20  align-items: center;
21  justify-content: center;
22 }
23
24 .btn_led:after {
25  content: attr(alt);
26  width: 20vw;
27  height: 8vw;
28  background-color: #191919;
29  display: flex;
30  align-items: center;
31  justify-content: center;
32 }
33
34 .btn_led:hover {
35  animation: animacion_btn_led 2s linear infinite;
36 }
37
38 @keyframes animacion_btn_led {
39  to {
40    background-position: 20vw;
41  }
42 }
```

[parte nº2 - botón luminoso - plasma]

Figura 293. Código de botones parte III

```
1 .btn_plasma {
2   width: 220px;
3   height: 50px;
4   border: none;
5   outline: none;
6   color: white;
7   background: #111;
8   cursor: pointer;
9   position: relative;
10  z-index: 0;
11  border-radius: 10px;
12  display: flex;
13  align-items: center;
14  justify-content: center;
15 }
16
17 .btn_plasma:before {
18  content: "";
19  width: calc(100% + 4px);
20  height: calc(100% + 4px);
21  background: linear-gradient(45deg, #ff0000, #ff7300,
22  position: absolute;
23  top: -2px;
24  left: -2px;
25  background-size: 400%;
26  z-index: -1;
27  filter: blur(5px);
28  animation: animacion_plasma 20s linear infinite;
29  opacity: 0;
30  transition: opacity 0.3s ease-in-out;
31  border-radius: 10px;
32  display: flex;
33  align-items: center;
34  justify-content: center;
35 }
```

[parte nº3 - botón luminoso - plasma]

Figura 294. Código de botones parte IV

```
1  .btn_plasma:active {
2    color: #000;
3  }
4
5  .btn_plasma:active:after {
6    background: transparent;
7    width: 100%;
8    height: 100%;
9  }
10
11 .btn_plasma:hover:before {
12   opacity: 1;
13 }
14
15 .btn_plasma:after {
16   width: 100%;
17   height: 100%;
18   z-index: -1;
19   content: "";
20   position: absolute;
21   background: #111;
22   left: 0;
23   top: 0;
24   border-radius: 10px;
25 }
26
27 @keyframes animacion_plasma {
28   0% {
29     background-position: 0 0;
30   }
31   50% {
32     background-position: 400% 0;
33   }
34   100% {
35     background-position: 0 0;
36   }
37 }
```

❖ Código del módulo botón espejo

[Parte nº1 – botón espejo]

Figura 295. Código de botones parte V

```
1  .btn_espejo {
2    padding: 8px 20px;
3    position: relative;
4    margin-right: 25px;
5    cursor: pointer;
6    font-size: 18px;
7    border-radius: 4px;
8    border: 4px solid transparent;
9    background-clip: padding-box;
10   transition: 0.5s all;
11 }
12 .btn_espejo:after {
13   transition: 0.5s all;
14   position: absolute;
15   top: -4px;
16   left: -4px;
17   right: -4px;
18   bottom: -4px;
19   content: "";
20   z-index: -1;
21   border-radius: 4px;
22 }
23
24 .btn_bordeado {
25   border-radius: 50px;
26 }
27 .btn_bordeado.btn_espejo:after {
28   border-radius: 50px;
29 }
```

[Parte nº2 – botón espejo - funda]

Figura 296. Código de botones parte VI

```
1 .btn_funda {
2   transition: 0.6s all;
3 }
4 .btn_funda:hover {
5   background-color: transparent;
6   color: white;
7 }
8 .btn_funda.btn_inverso {
9   background-color: transparent;
10  color: white;
11 }
12 .btn_funda.btn_inverso:hover {
13   background-color: white;
14   color: #333;
15 }
16 .btn_funda:after {
17   background-size: 200% 100%;
18   background-position: 0% 0;
19 }
20 .btn_funda:hover:after {
21   background-position: 100% 0;
22 }
23 .btn_funda:after {
24   background-image: linear-gradient(135deg, #f68918 0%, #f68918 31%, #ae3e9f 31%,
25 }
```

[Parte nº3 – botón espejo - barrido]

Figura 297. Código de botones parte VII

```
1 .btn_barrido {
2   transition: 0.6s all;
3 }
4 .btn_barrido:hover {
5   background-color: transparent;
6   color: white;
7 }
8 .btn_barrido.btn_inverso {
9   background-color: transparent;
10  color: white;
11 }
12 .btn_barrido.btn_inverso:hover {
13   background-color: white;
14   color: #333;
15 }
16 .btn_barrido:after {
17   background-size: 200% 100%;
18   background-position: 0% 0;
19 }
20 .btn_barrido:hover:after {
21   background-position: 100% 0;
22 }
23 .btn_barrido:after {
24   background-image: linear-gradient(135deg, #8ae553 0%, #8ae553
25 }
```


[Parte nº3 – botón espejo - cortina]

Figura 298. Código de botones parte VIII

```
1 .btn_cortina {
2   transition: 0.6s all;
3 }
4 .btn_cortina:hover {
5   background-color: transparent;
6   color: white;
7 }
8 .btn_cortina.btn_inverso {
9   background-color: transparent;
10  color: white;
11 }
12 .btn_cortina.btn_inverso:hover {
13   background-color: white;
14   color: #333;
15 }
16 .btn_cortina:after {
17   background-size: 200% 100%;
18   background-position: 0% 0;
19 }
20 .btn_cortina:hover:after {
21   background-position: 100% 0;
22 }
23 .btn_cortina:after {
24   background-image: linear-gradient(45deg, rgba(49,
25 }
```

❖ Código del módulo botón animado

[Parte nº1 – botón animado]

Figura 299. Código de botones parte IX

```
1 .btn_termita {
2   position: relative;
3   width: calc(0.8 * 200px);
4   height: calc(0.7 * 100px);
5   cursor: pointer;
6   overflow: hidden;
7   margin: 0 0.8rem;
8   box-shadow: 0 2px 5px rgba(0, 0, 0, 0.2), 0 3px 8px rgba(0, 0, 0, 0.1);
9   transition: all 0.3s cubic-bezier(0, 0.22, 0.3, 1);
10  display: flex;
11  justify-content: center;
12  align-items: center;
13 }
14 .btn_termita:before {
15   content: "";
16   position: absolute;
17   top: 0;
18   left: 0;
19   right: 0;
20   bottom: 0;
21   background: rgba(0, 0, 0, 0.1);
22   display: flex;
23   justify-content: center;
24   align-items: center;
25 }
```

[Parte nº2 – botón animado]

Figura 300. Código de botones parte X

```
1  .btn_termita span {
2    color: #fff;
3    font-size: 1rem;
4    z-index: 10;
5    text-transform: uppercase;
6    letter-spacing: 2px;
7  }
8  .btn_termita.btn_termita_direccion {
9    background: #16a085;
10 }
11 .btn_termita .btn_termita_activacion {
12   position: absolute;
13   width: 0;
14   height: 0;
15   filter: url(#filter);
16   border-radius: 50%;
17   z-index: 5;
18   transition: all 2.5s cubic-bezier(0.1, 0.22, 0.3, 1);
19 }
```

[Parte nº3 – botón animado]

Figura 301. Código de botones parte XI

```
1  .btn_termita.btn_termita_direccion .btn_termita_activacion {
2    left: -50%;
3    bottom: -50%;
4    background: #34495e;
5  }
6  .btn_termita:hover .btn_termita_activacion {
7    width: calc(2 * 200px);
8    height: calc(2 * 100px);
9  }
10
11 @media only screen and (max-width: 750px) {
12   .btn_termita {
13     margin: 0.8rem 0;
14   }
15 }
```

Descripción de las propiedades más significativas:

A continuación, se procederá a detallar las propiedades más usadas y su aplicación. También se mostrará la compatibilidad web de cada propiedad para su mejor comprensión dejamos una leyenda la cual se puede observar en la figura 2.

Propiedad width:100% :

Se empleo width 100% para el ancho de este efecto ya que si bien esta propiedad tiene diversas unidades de medidas como:

- EM
- VH
- PIXELES
- REM
- PORCENTAJE

Se fue descartando una a una al momento de probarla, ya que la al usar la unidad EM este se distorsionaba por su propia naturaleza, básicamente esta propiedad usa el ancho de la pantalla del dispositivo donde estemos y las va ajustando así que fue descartado.

A continuación, veamos la compatibilidad web al usar esta propiedad en la figura 3.

Propiedad height: calc(0.7*100px)

Luego de ver las unidades de medida en la propiedad anterior se decidió trabar en pixeles, ya que se necesitaba un valor estático poniendo como un límite que no tenga en cuenta las dimensiones del dispositivo del usuario, en ese sentido se descartó las propiedades que toman el ancho del dispositivos como son REM y EM por otra parte también se descartó a los que toman el alto del dispositivo como son VH . Luego de reducir nuestro marco de trabajo se tenía PORCENTAJE o PIXELES así que descartamos a PORCENTAJE porque tampoco necesitábamos que sea ajustable el alto ya que normalmente para eso tendríamos que dejarlo en automático ya que por default el navegador le asigna ese valor.

El prefijo calc que lleva se debe a que se empleara una operación matemática en este caso es la multiplicación que está representado por el símbolo asterisco el resultado de dicha operación es 70px si bien se pudo haber directamente el valor se optó por usar la función calc para calcular el valor con la intención de hacerlo algo más académico y demostrar que se puede calcular el valor con usar esta función.

A continuación, veamos la compatibilidad web al usar esta propiedad en la figura 4.

Propiedad background:transparent

Esta propiedad se usa para darle transparencia al fondo o clase sobre la cual se aplica. De forma predeterminada, el color de fondo será transparente, lo que significa que no existirá un color de fondo.

A continuación, veamos la compatibilidad web al usar esta propiedad en la figura 141.

Propiedad width:initial

Es un valor global el cual indica que volverá a tomar las dimensiones de su valor inicial el cual trae el navegador por defecto. Para ver que navegadores soportan esta propiedad, observamos la figura 3.

Propiedad display: flex

Con esta propiedad se puede aplicar la alineación de los elementos en una sola dirección sea horizontal o vertical, estos valores se puede conjugar con los distintos valores que tiene, se optó por esta propiedad ya que es una forma óptima de maquetar y de que los elementos no se desborden de sus contenedores como solía suceder al maquetar usando position o tables en la antigüedad además flexbox viene para usarlo con la versión 3 de css llamada css3 haciendo más robusta la maquetación de elementos. A continuación, veamos la compatibilidad web al usar esta propiedad en la figura 43.

Propiedad align-items: center

Para usar esta propiedad es un requisito indispensable tener habilitado display:flex ,luego de eso se puede conjugar align-items con los siguientes valores:

- CENTER
- END
- FLEX-END
- FLEX-START
- INHERIT
- INITIAL
- NORMAL
- REVERT
- SELF-END

- START
- STRETCH
- UNSET

En este caso se usó con el valor center ya que lo que ara es centrar en el centro de gravedad a los elementos, pero en dirección vertical. A continuación, veamos la compatibilidad web al usar esta propiedad en la figura 44.

Propiedad justify-content:center

La propiedad justify-content tiene como prerequisites haber declarado display:flex en el mismo elemento o en un elemento padre. Tiene las siguientes conjugaciones:

- CENTER
- END
- FLEX-END
- FLEX-START
- INHERIT
- INITIAL
- LEFT
- NORMAL
- REVERT
- RIGHT
- SPACE-AROUND
- SPACE-BETWEEN
- SPACE-EVENLY
- START
- STRETCH
- UNSET

Se opto por elegir el valor center ya que este centra en horizontal cualquier elemento. A continuación, veamos la compatibilidad web al usar esta propiedad en la figura 45.

Propiedad filter:blur(5px)

La propiedad filter hace un desenfocado y satura los efectos visuales para un elemento, el valor blur es un desenfocado gaussiano hace que se mezclen los pixeles recibe como parámetro el valor o la cantidad a mezclar además no se acepta porcentaje. A continuación, veamos la compatibilidad web al usar esta propiedad en la figura 147.

Propiedad margin:0 auto

Esta propiedad como esta en el contenedor de input clásico lo que se busco era centrar todo el contenido en el centro para ello la mejor opción era usar los valores 0 arriba y debajo luego automático centrar a la derecha como a la izquierda. Ambos valores para hacerlo de forma abreviada solo se dispusieron a poner 0 y auto. A continuación, veamos la compatibilidad web al usar esta propiedad en la figura 38.

Propiedad content:””

Para usar esta propiedad es requisito primordial usar un pseudo-elemento como:

- Before: Lo que hace esta propiedad es insertar elementos antes de la clase.
- After: Lo que hace esta propiedad es insertar elementos después de la clase.

Estos pseudo-elementos son usados para añadir contenido a un elemento con la propiedad content, aunque también en content puede ir palabras y estas se insertaran.

A continuación, veamos la compatibilidad web al usar esta propiedad en la figura 144.

Propiedad animation: animación_plasma 20s linear infinite

En esta propiedad es un requisito tener en un @keyframes ya que es el primer valor que nos pedirá, luego como segundo valor tenemos el tiempo de duración expresado en segundos para finalmente colocar el tiempo de duración del efecto el cual será expresado en segundos si no se le coloca por default será uno, luego vemos la palabra linear esto hace referencia que el efecto tendrá la misma velocidad de principio a fin expresado de otra forma será lineal sin variables en su trayectoria, también podemos colocar la palabra reservada infinite lo cual indica que la duración del efecto es infinita.

A continuación, veamos la compatibilidad web al usar esta propiedad en la figura145.

Propiedad overflow: hidden

Esta propiedad lo que hace es ocultar los elementos que salgan de su contenedor mostrando solo una parte del contenido que encaje exacto en dicho contenedor al mismo tiempo lo que hace es eliminar el scroll horizontalmente.

A continuación, veamos la compatibilidad web al usar esta propiedad en la figura 181.

Propiedad @keyframe animación_plasma

Con esta propiedad se pueden crear animaciones más sofisticadas ya que se pueden mezclar propiedades de css en su interior además de combinarlas con la propiedad animation y transform. Esta se declara usando la palabra reservada @keyframe seguido de un nombre en este caso el nombre que le asignamos es animación_plasma, luego dentro se puede usar de dos formas:

- ❖ Usando la estructura from y to como se aprecia en la figura 183:
- ❖ También podemos usar porcentaje para especificar a fondo lo que queremos que haga la animación como se observa en la figura 184.

A continuación, veamos la compatibilidad web al usar esta propiedad en la figura 185.

Principales dificultades:

Para este efecto en concreto la principal dificultad fue organizar de distintas formas posibles, así como ver las posibles conjugaciones que pueden hacer con el efecto para que el usuario al final no tenga problemas al implementarlo. Esas dificultades surgieron a raíz de las siguientes preguntas:

- ¿Qué sucede si el usuario no desea relleno o si desea un efecto inverso en su botón?

Dentro de todos los módulos el único donde podíamos implementarlo haciendo contraste con los colores fue el módulo botón espejo para ello lo que se tuvo que implementar para solucionar eficazmente el problema fue una clase adicional a la cual le denominamos btn_inverso la función de esta clase consistió en dar relleno sin hacer hover y al quitar la clase da relleno al hacer hover.

¿Qué usamos para hacer las pruebas?

Se uso el navegador Google Chrome y opera mini, así como la siguiente página web que es un simulador online:

<http://www.responsinator.com>

Esto con motivos de asegurarnos que funcione en dispositivos móviles y tabletas. Ya que en base a lo trabajado no es del todo fiable usar el navegador para verificar la correcta maquetación de los elementos, así como tampoco es fiable depender de un software web, por ello se decidió diversificar en cuanto a las herramientas para probar las clases y efectos trabajados en el framework XRL8.

También usamos como editor de código visual studio code y su extensión live server para visualizar en vivo los efectos realizados en el css en tiempo real, también usamos el autoguardado que trae dicho editor de código para no estar guardando el proyecto a cada rato. Así conseguimos un mejor desempeño y evitamos el tedioso guardar cambios a cada rato, también evitamos el que la memoria cache se llene mostrándonos los mismos resultados.

FASE 3.- Hacerlo responsive

- Se uso propiedad flex-box y sus posibles conjugaciones con cada uno de sus valores según sea el caso.
- Se uso @media_query para la clase btn_termita

Figura 302. Uso de media query en el componente botones

```
1 @media only screen and (max-width: 750px) {  
2   .btn_termita {  
3     margin: 0.8rem 0;  
4   }  
5 }
```

- También se usó las unidades de medida vh o rem para texto y contenedores ya que estas son imprescindibles a la hora de hacer responsive un sitio web.

FASE 4.- Testing asegurarse que no tenga conflicto con otros efectos.

En esta parte del desarrollo de la clase se dispuso a probar poniéndolo con las demás clases juntándolas. Para ello lo que se hizo fue colocar dentro de un archivo index.html todas las clases trabajadas y ver como todos los módulos con sus respectivas clases respondían. En algunos casos había conflicto con los demás efectos para corregir esto se procedió a implementar en el desarrollo dos cosas básicas la primera fue:

- Para detectar que clase era la que causaba el conflicto y ver en que archivo scss estaba. Se procedía a inspeccionar la página para detectar donde estaba el problema. En muchas ocasiones no se daba con el archivo causante del error entonces en ese caso aplicábamos la frase “Divide y vencerás”. Lo que se procedió hacer fue comentar de uno en uno cada efecto en el archivo principal que los llamaba esto reducía las líneas de código de css, además como teníamos en tiempo real los cambios fue mucho más rápido detectar el problema. Ya que no se tenía que recargar a cada rato además otro punto a favor era que se auto compilaba los cambios de sass automáticamente gracias a nodeJS el cual mediante la consola de comandos de Windows cuando detectaba un cambio en las líneas de código se auto compilaba y generando o reemplazando el archivo css en el cual hacíamos las pruebas, esto aumento la productividad para crear los efectos ya que al comentar de uno en uno cada archivo sass llegaba un momento donde desaparecía el error entonces revisábamos y nos dimos cuenta de que era uno o en algunos casos hasta tres efectos que hacían conflicto con la clase escrita.
- Se procedió a escribir mediante comentarios en un archivo llamado `_error_logs.scss` la clase que hace conflicto con la clase escrita, además de describir una nota breve en caso no se disponía de tiempo para solucionarlo luego, esto permitió tener un reporte más preciso de los incidentes que íbamos detectando.

FASE 5.- Aplicar la metodología OOCSS, llevándolo a SASS

- **Código css.** - Esta parte ya estaría completa por lo descrito anteriormente.

- **Código repetido.** - Identificación de código repetido

En este pequeño apartado luego de escribir y debugear el error se procedió a identificar el código css repetido para llevarlo a un archivo SCSS en sass, para su mayor optimización y aplicación de la metodología. Así que se procedió a reciclar el siguiente código:

❖ Botón clásico

Figura 303. Código repetido del componente botones parte I

```
1  .btn_clasico {
2    border: none;
3    outline: none;
4    color: white;
5    background-color: #1a73e8;
6    padding: 0.625rem 1.25rem;
7    cursor: pointer;
8    border-radius: 0.312rem;
9    font-size: 1rem;
10   margin: 0 auto;
11   box-sizing: border-box;
12 }
13
14 .btn_clasico:hover {
15   background-color: #287ae6;
16   box-shadow: 0 1px 1px 0 rgba(66, 133, 244, 0.45), 0 1px 3px 1px rgba(66, 133, 244, 0.3);
17   margin: 0 auto;
18   box-sizing: border-box;
19 }
```

❖ Botón luminoso

Figura 304. Código repetido del componente botones parte II

```
1  .btn_led {
2    cursor: pointer;
3    text-decoration: none;
4    width: calc(20vw + 6px);
5    height: calc(8vw + 6px);
6    background-image: linear-gradient(90deg,
7    border-radius: 5px;
8    border: none;
9    text-transform: uppercase;
10   font-size: 3vw;
11   color: white;
12   font-weight: 700;
13   display: flex;
14   align-items: center;
15   justify-content: center;
16 }
```

Figura 305. Código repetido del componente botones parte III

```
1 .btn_led:after {
2   content: attr(alt);
3   width: 20vw;
4   height: 8vw;
5   background-color: #191919;
6   display: flex;
7   align-items: center;
8   justify-content: center;
9 }
```

Figura 306. Código repetido del componente botones parte IV

```
1 .btn_plasma {
2   width: 220px;
3   height: 50px;
4   border: none;
5   outline: none;
6   color: white;
7   background: #111;
8   cursor: pointer;
9   position: relative;
10  z-index: 0;
11  border-radius: 10px;
12  display: flex;
13  align-items: center;
14  justify-content: center;
15 }
16
17 .btn_plasma:before {
18  content: "";
19  width: calc(100% + 4px);
20  height: calc(100% + 4px);
21  background: linear-gradient(45deg, #ff0000, #ff7300,
22  position: absolute;
23  top: -2px;
24  left: -2px;
25  background-size: 400%;
26  z-index: -1;
27  filter: blur(5px);
28  animation: animacion_plasma 20s linear infinite;
29  opacity: 0;
30  transition: opacity 0.3s ease-in-out;
31  border-radius: 10px;
32  display: flex;
33  align-items: center;
34  justify-content: center;
35 }
```

Figura 307. Código repetido del componente botones parte V

```
1 .btn_plasma:active:after {
2   background: transparent;
3   width: 100%;
4   height: 100%;
5 }
```

Figura 308. Código repetido del componente botones parte VI

```
1 .btn_plasma:after {
2   width: 100%;
3   height: 100%;
4   z-index: -1;
5   content: "";
6   position: absolute;
7   background: #111;
8   left: 0;
9   top: 0;
10  border-radius: 10px;
11 }
```

❖ Botón espejo

Figura 309. Código repetido del componente botones parte VII

```
1 .btn_funda {
2   transition: 0.6s all;
3 }
4 .btn_barrido {
5   transition: 0.6s all;
6 }
7 .btn_cortina {
8   transition: 0.6s all;
9 }
```

❖ Botón animado

Figura 310. Código repetido del componente botones parte VIII

```
1 .btn_termita {
2   position: relative;
3   width: calc(0.8 * 200px);
4   height: calc(0.7 * 100px);
5   cursor: pointer;
6   overflow: hidden;
7   margin: 0 0.8rem;
8   box-shadow: 0 2px 5px rgba(0, 0, 0, 0.2);
9   transition: all 0.3s cubic-bezier(0, 0.22
10  display: flex;
11  justify-content: center;
12  align-items: center;
13 }
```

Figura 311. Código repetido del componente botones parte IX

```
1 .btn_termita:before {
2   content: "";
3   position: absolute;
4   top: 0;
5   left: 0;
6   right: 0;
7   bottom: 0;
8   background: rgba(0, 0, 0, 0.1);
9   display: flex;
10  justify-content: center;
11  align-items: center;
12 }
```

- **Crear un objeto.** - Se describe la lógica de la función para luego crear un mixing.

Una vez identificado el código css repetido se procede hacer una función la cual se ejecuta al llamado del código, mediante esta técnica nuestro código será más puro y limpio.

Paralelamente en esta parte se determina la lógica de las funciones y ciclos como lo son for, while, each. También determinamos el pase de parámetros, así como la condicional que llevaran dentro, por otra parte, y no menos importante se define el nombre con el cual se llamara a las funciones.

```
function boton_clasico_base(){  
  margin:0 auto;  
  box-sizing:border-box;  
}
```

```
function boton_luminoso_base(){  
  display:flex;  
  align-items:center;  
  justify-content:center;  
}
```

```
function botón_luminoso_plasma(){  
  width:100%;  
  height:100%;  
}
```

```
function botón_espejo_base(){  
  transition:.06 all;  
  &:hover{  
    background-color:transparent;  
  }  
  &:btn_inverso{  
    background-color:transparent;  
    color:white;
```

```
  &:hover{  
    background-color:white;  
    color:#333;  
  }  
  }  
  &:after{  
    background-size:200% 100%;  
    background-position:0% 0;  
  }  
  &:hover{  
    &:after{  
      Background-position:100% 0;  
    }  
  }  
}
```

Luego de tener la lógica se procede a llevarlo a SASS para ello tenemos dos opciones, la primera es usar `@function` ya que como su nombre ase referencia es para crear una función la otra opción es usar `@mixin`, pero ambas tienen diferencias. La primera retorna un valor, mientras `@mixin` solo llama a los valores en este caso no necesitamos que retorne un valor por ello usaremos `@mixin`. Entonces quedaría de la siguiente forma:

❖ Botón clásico

Figura 312. Creación del objeto botón_clasico_base en función mixin

```
1 @mixin boton_clasico_base{
2     margin: 0 auto;
3     box-sizing: border-box;
4 }
```

❖ Botón luminoso

Figura 313. Creación del objeto botón_luminoso_base en función mixin

```
1 @mixin boton_luminoso_base{
2     display: flex;
3     align-items: center;
4     justify-content: center;
5 }
```

Figura 314. Creación del objeto botón_luminoso_plasma en función mixin

```
1 @mixin boton_luminoso_plasma{
2     width: 100%;
3     height:100%;
4 }
```

❖ Botón espejo

Figura 315. Creación del objeto botón_espejo_base en función mixin

```
1 @mixin boton_espejo_base{
2     transition: .6s all;
3     &:hover {
4         background-color: transparent;
5         color: white;
6     }
7     &.btn_inverso {
8         background-color: transparent;
9         color: white;
10    &:hover {
11        background-color: white;
12        color: #333;
13    }
14 }
15 &:after {
16     background-size: 200% 100%;
17     background-position: 0% 0;
18 }
19
20 &:hover {
21     &:after {
22         background-position: 100% 0;
23     }
24 }
25 }
```

❖ Botón animado

Figura 316. Creación del objeto boton_animado_termina en función mixin

```
1 @mixin boton_animado_termina{
2     display: flex;
3     justify-content: center;
4     align-items: center;
5 }
```

- **OOCSS**. – Luego de tener la función o la lógica para reciclar el código se procede a integrar la metodología OOCSS para ello debemos llamar al mixin para hacerlo solo necesitamos agregar @include “nombre del mixin”. Luego de integrarlo ya estará aplicada correctamente la metodología OOCSS

❖ Botón clásico

Figura 317. Metodología OOCSS en el componente botones parte I

```
1 .btn_clasico {
2   border: none;
3   outline: none;
4   color:$btn_uno_color_letra;
5   background-color:$btn_uno_color_fondo;
6   padding: 0.625rem 1.25rem;
7   cursor: pointer;
8   border-radius: 0.312rem;
9   font-size:$btn_uno_tamano_letra;
10  @include boton_clasico_base;
11 }
```

Figura 318. Metodología OOCSS en el componente botones parte II

```
1 .btn_clasico:hover{
2   background-color:$btn_uno_color_fondo_hover;
3   box-shadow: 0 1px 1px 0 rgb(66 133 244 / 45%), 0 1px 3px 1px rgb(66 133 244 / 30%);
4   @include boton_clasico_base;
5 }
```

❖ Botón luminoso

Figura 319. Metodología OOCSS en el componente botones parte III

```
1 .btn_led{
2   cursor: pointer;
3   text-decoration:none;
4   width: calc(20vw + 6px);
5   height: calc(8vw + 6px);
6   background-image: linear-gradient(90deg,#00C0FF 0,#FFCF00 49%,#FC4F4F 80%,#00C0FF 100%);
7   border-radius: 5px;
8   border: none;
9   text-transform: uppercase;
10  font-size: 3vw;
11  color: white;
12  font-weight: 700;
13  @include boton_luminoso_base;
14 }
```

Figura 320. Metodología OOCSS en el componente botones parte IV

```
1 .btn_led:after{
2   content:attr(alt);
3   width:20vw;
4   height:8vw;
5   background-color:#191919;
6   @include boton_luminoso_base;
7 }
```

Figura 321. Metodología OOCSS en el componente botones parte V

```
1 .btn_plasma {
2   width:$btn_tres_ancho;
3   height:$btn_tres_alto;
4   border: none;
5   outline: none;
6   color: white;
7   background: #111;
8   cursor: pointer;
9   position: relative;
10  z-index: 0;
11  border-radius:$btn_tres_grosor_borde;
12  @include boton_luminoso_base;
13 }
```


Figura 322. Metodología OOCSS en el componente botones parte VI

```
1 .btn_plasma:before {
2   content: '';
3   width: calc(100% + 4px);
4   height: calc(100% + 4px);
5   background: linear-gradient(45deg, #ff0000, #ff7300,
6   position: absolute;
7   top: -2px;
8   left: -2px;
9   background-size: 400%;
10  z-index: -1;
11  filter: blur(5px);
12  animation: animacion_plasma 20s linear infinite;
13  opacity: 0;
14  transition: opacity .3s ease-in-out;
15  border-radius:$btn_tres_grosor_borde;
16  @include boton_luminoso_base;
17 }
```

Figura 323. Metodología OOCSS en el componente botones parte VII

```
1 .btn_plasma:active:after {
2   background: transparent;
3   @include boton_luminoso_plasma;
4 }
```

Figura 324. Metodología OOCSS en el componente botones parte VIII

```
1 .btn_plasma:after {
2   @include boton_luminoso_plasma;
3   z-index: -1;
4   content: '';
5   position: absolute;
6   background: #111;
7   left: 0;
8   top: 0;
9   border-radius:$btn_tres_grosor_borde;
10 }
```

❖ Botón espejo

Figura 325. Metodología OOCSS en el componente botones parte IX

```
1 .btn_funda{
2   @include boton_espejo_base;
3   &:after {
4     background-image: linear-gradient(135deg, rgb(246, 137, 24) 0%,
5   }
6 }
```

Figura 326. Metodología OOCSS en el componente botones parte X

```
1 .btn_barrido{
2   @include boton_espejo_base;
3   &:after {
4     background-image: linear-gradient(135deg, rgb(138, 229, 83) 0%,
5   }
6 }
```

Figura 327. Metodología OOCSS en el componente botones parte XI

```
1 .btn_cortina{
2   @include boton_espejo_base;
3   &:after {
4     background-image: linear-gradient(45deg, rgba(49, 74, 89, 0.45) 0%,
5   }
6 }
```

❖ Botón animado

Figura 328. Metodología OOCSS en el componente botones parte XII

```
1 .btn_termita{
2   position: relative;
3   width: calc(0.8 * #{$btn_cinco_ancho});
4   height: calc(0.7 * #{$btn_cinco_alto});
5   cursor: pointer;
6   overflow: hidden;
7   margin: 0 0.8rem;
8   box-shadow: 0 2px 5px rgba(0,0,0,0.2), 0 3px 8px rgba(0,0,0,0.1);
9   transition: all 0.3s cubic-bezier(0, 0.22, .3, 1);
10  @include boton_animado_termita;
11
12  &:before{
13    content: '';
14    position: absolute;
15    top: 0;
16    left: 0;
17    right: 0;
18    bottom: 0;
19    background: rgba(0,0,0,0.1);
20    @include boton_animado_termita;
21  }
22 }
```

Bien volviendo al tema de las variables estas variables se encuentran en un archivo sass denominado variables valga la redundancia donde se encuentran las siguientes variables:

Figura 329. Variables sass del componente botones

```
1 //Boton clasico
2 $btn_clasico_color_fondo: #1a73e8;
3 $btn_clasico_tamano_letra: 1rem;
4 $btn_clasico_color_letra: white;
5 $btn_clasico_color_fondo_hover: #287ae6;
6
7 //Boton luminoso --> led
8 $btn_luminoso_led_ancho_largo: initial;
9 $btn_luminoso_led_color_borde: skyblue;
10 $btn_luminoso_led_grosor_borde: 2px;
11
12 //Boton luminoso --> plasma
13 $btn_luminoso_plasma_ancho: 220px;
14 $btn_luminoso_plasma_alto: 50px;
15 $btn_tres_grosor_borde: 10px;
16
17 //Boton espejo
18 $btn_espejo_ancho: 4px;
19 $btn_espejo_tamano_letra: 18px;
20 $btn_espejo_tamano_grosor_borde: 4px;
21
22 //Boton animado
23 $btn_animado_ancho: 200px;
24 $btn_animado_alto: 100px;
```

Como se ha visto SASS es un preprocesador de css por ello se ejecuta antes del css tradicional esto permite añadir variables así como agregar

condicionales, etc. Con la aplicación de la metodología como se observa se reutiliza el código css, además de que se hace escalable el proyecto. Ya que, si en un futuro queremos cambiar o añadir más propiedades css, solo cambiaríamos las variables o añadiríamos más propiedades en el mixin.

FASE 6.- Documentar en la web site www.phooldx.com

Finalmente, si el efecto paso por todos los de más filtros se procede a su documentación respectiva, para documentar todos los efectos se creó una página web de documentación usando clases y elementos del mismo framework. A continuación, se describirá como se procede a la documentación de la misma. Siguiendo estos pasos:

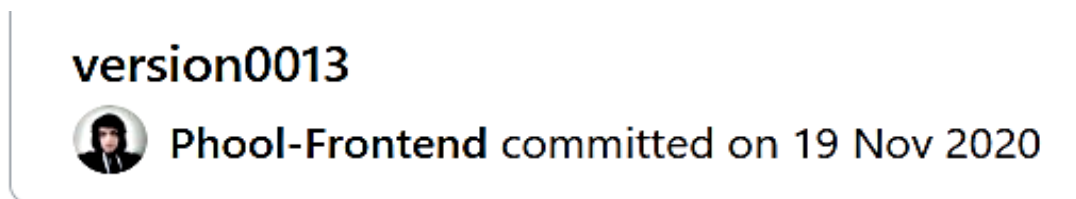
- Se copia las etiquetas de HTML5 junto con la clase a un txt temporal para no perder la sintaxis
- Se guarda los cambios en el compilador de SASS
- Como es la parte final se procedió hacer copias de seguridad para no perder el avance, en archivos RAR. Estos archivos estaban por versiones enumerados además se agregaba un pequeño texto describiendo el avance. Este guardado se hacía de manera local en un USB.
- Para una mayor seguridad del avance del código se procedió luego a copiar la versión del archivo RAR junto a la descripción del mismo.

Guardado de código en GitHub. - Luego de tener la copia del proyecto a nivel físico en un USB, se procedió a guardarlo en un repositorio de GitHub para lo cual se usó Git, para ello se procedió de la siguiente forma:

- Se guardaba todo el proyecto con el siguiente comando que observamos en la figura 12
- Luego se verificaba los cambios guardados con el siguiente comando que observamos en la figura 13.
- Después se hacía el commit, es decir se le asignaba un nombre a los cambios realizados con el siguiente comando que observamos en la figura 14.

- Luego se procede a empaquetar los datos para que se suban al repositorio de GitHub con el siguiente código que se observa en la figura 15.
- Finalmente, luego de haber enviados el proyecto empaquetado hacia GitHub como se puede apreciar en la imagen, este queda guardado en dicho repositorio.

Figura 330. Visualización en github del componente botones



Se guarda en una escala de tiempo, con todos los cambios hechos desde la creación del proyecto, esto ayuda a tener un mejor control de versiones. Asimismo, también ayuda a que más personas se sumen al proyecto y puedan clonar el proyecto como trabajar por separado el proyecto, entre otras muchas ventajas.

- Así mismo se manejó un archivo llamado bitácora.txt donde aquí se registró todos los incidentes ocurridos, así como los cambios hechos, se registraba también las metas cumplidas, como las tareas nuevas para desarrollar al día siguiente. Al finalizar con un efecto por completo se quitaba de la lista de prioridades así dábamos pase al siguiente efecto o problema ha resolver. Esto servía para tener un mejor orden y saber ejecutar prioridades.
- Luego de tener guardado el avance y de registrar el avanzase, se procedió a copiar del archivo documentación Etiquetas.txt el fragmento de código que se copió anteriormente para luego pegarlo en www.nosetup.org esta página web se encargaba de convertir html en texto plano para poder visualizar html
- Antes de finalizar se procedió a crear un archivo llamado modulo_botones.html donde se dividía en tres partes:
 - En la primera se describía un poco al efecto que era y que hacia

- En la segunda parte se procedía a copiar el código ya formateado obtenido de www.nosetup.com para luego pegarlo.
 - En la tercera parte lo que se hacía era mostrar un apartado llamado estructura donde se describía como se llamaban las variables para que sea modificado en SASS.
- Finalmente, el ejemplo listo para usarse queda de la siguiente forma:

❖ Botón clásico

Figura 331. Código html5 final del componente botones parte I

```
<input class="btn_clasico" type="submit" value="Siguiete">
```

❖ Botón luminoso

Figura 332. Código html5 final del componente botones parte II

```
<button class="btn_led" alt="Ingresar"></button>
```

Figura 333. Código html5 final del componente botones parte III

```
<button class="btn_plasma" type="button">Entrar</button>
```

❖ Botón espejo

Figura 334. Código html5 final del componente botones parte IV

```
<button class="btn_espejo btn_funda">Adelante</button>
<button class="btn_espejo btn_bordeado btn_barrido">Siguiete</button>
<button class="btn_espejo btn_inverso btn_cortina">Suscribirse</button>
```

❖ Botón animado

Figura 335. Código html5 final del componente botones parte V

```
<svg width="0" height="0">
  <filter id="filter">
    <feTurbulence type="fractalNoise" baseFrequency=".01" numOctaves="6"></feTurbulence>
    <feDisplacementMap in="SourceGraphic" scale="100"></feDisplacementMap>
  </filter>
</svg>
<div class="btn_termita btn_termita_direccion">
  <span>Termita</span><div class="btn_termita_activacion"></div>
</div>
```

FORMULARIOS

FASE 1.- Investigación del efecto y su importancia

Como primer paso lo primero que se hizo fue investigar e indagar sobre el efecto, a diferencia de los efectos antecesores este se caracteriza por solo usarse para formularios donde el usuario tenga que ingresar sus datos personales. En esta ocasión se decidió trabajar con formularios transparentes ya que los formularios estáticos y con fondo de colores es fácil de hacer, nos vimos en la necesidad de implementar algo con un poco más de nivel así que se creó formularios transparentes son dos en este caso una es algo sutil que no pasara desapercibido y el otro es un formulario con un poco más de efectos para que el usuario interactúe con el formulario. Uno de los beneficios de usar un formulario es la organización de estos y la garantía de que será responsive al momento de implementarlo.

Otro del motivo el cual fue elegido el efecto para formar parte del framework XRL8 fue que el proyecto presente debe estar a la altura de los frameworks que están en la actualidad siendo usados en la industria del desarrollo web como son:

- **Bootstrap.** - El presente framework cuenta con la clase `.form-control` para poder crear barras de navegación, si desean saber más sobre ello puede visitar la web oficial de documentación de dicho framework mediante el siguiente enlace:
<https://getbootstrap.com/docs/4.0/components/forms/>
- **Bulma.**- Este framework también cuenta con un modelo para realizar formularios o al menos el contenedor denominado `.control`, para ver cómo se emplea y formas de uso revisar el siguiente enlace que los llevara a la documentación oficial de dicho framework:
<https://bulma.io/documentation/form/general/#form-control>
- **TailwindCSS.**- Quisa sea este el framework de css más usado en los últimos 3 años sobre pasando a los demás por sus formas de integración con NodeJS y react. Este framework dentro de su repertorio tiene un apartado dedicado a realizar formularios mezclada con otras clases esta se puede construir. Gracias a sus componentes, estos a diferencia de los demás se caracteriza por no tener clases predefinidas para efectos sino

todo lo contrario tiene un conjunto de clases en el cual el usuario puede elegir los efectos más simples para colocar a su diseño. Para revisar su documentación ingresar al siguiente enlace:

<https://tailwindui.com/components/application-ui/forms/form-layouts>

- **Foundation.-** Uno de los primeros frameworks en salir al mercado sin duda será foundation aunque con el pasar de los años a perdido popularidad este framework también dentro de sus clases tiene una sección para formulario básicos usando sus contenedores globales denominado .grid-container. Para ver el uso de este se puede recurrir al siguiente link:

<https://get.foundation/sites/docs/forms.html>

- **UIKIT.-** Por otra parte este framework cuenta con la peculiaridad de que funciona con la etiqueta <form> sin agregar ninguna clase, ya que reescribieron los estilos que trae por defecto el navegador y decidieron darle ellos los nuevos estilos así que lo agregaron en la raíz que viene a ser el mismo elemento, para ver más sobre este framework recomiendo revisar la documentación en el siguiente enlace:

<https://getuikit.com/docs/form>

- **Semantic UI.-** A continuación tenemos otro framework de css muy completo la verdad el cual tiene soporte y componentes para ser usado con react, la clase la cual ayuda a crear formularios es denominada ui form, podemos ver una gran cantidad de ejemplos mediante el siguiente enlace:

<https://semantic-ui.com/collections/form.html>

Modulo: Un módulo según definición propia es una sección de trabajo principal a comparación de un submódulo este puede ser denominado la estructura padre y los submódulos estructuras hijas. Cabe aclarar que en esta sección en específico no hubo submódulos ya que estos son módulos independientes además en esta ocasión comparten similitudes. Estos módulos están expresados en las siguientes clases:

- **Formulario Transparente I.** - Para este efecto quisimos hacer algo diferente que un simple formulario para ello le agregamos transparencia además de animación como efectos visuales en los inputs los cuales hacen

que el usuario tenga mejor experiencia en el web así mismo contiene técnicas de maquetado la cual garantiza que estos sean responsive.

- **Formulario Transparente II.** – Este efecto es similar al primero, pero se caracteriza por ser más simple y sencillo ya que no cuenta con efectos visuales ni animación ya que su apariencia es un poco más conservador, manteniendo el funcionamiento responsive.

FASE 2.- Codificar el efecto

Antes de ver esta sección a fondo quisiera hacer énfasis en lo tedioso que fue codificar ya que se tuvo que indagar además de probar las diferentes técnicas y posibles combinaciones para que este funcione a la perfección.

Bien luego de tener claro que efecto queríamos construir y porque lo íbamos a construir como se vio en el capítulo anterior se procedió a escribir código como se muestra a continuación.

Código completo

❖ Código del módulo formulario transparente I

[parte 01 – formulario transparente I]

Figura 336. Código de formularios parte I

```
1 .form_trans_I {
2   width: 55%;
3   height: auto;
4   margin: 80px auto;
5   padding: 20px;
6   border-radius: 9px;
7   border: solid 4.4px black;
8   color: white;
9   background-color: rgba(0, 0, 0, 0.5);
10  transition: all 2s ease;
11  text-align: justify;
12  font-family: Courier, monospace;
13  display: flex;
14  flex-direction: column;
15  box-sizing: border-box;
16 }
17 .form_trans_I:hover {
18  transition: all 0.3s ease;
19  transform: scale(1.1);
20  transform: rotate(9.9);
21  color: white;
22 }
23 .form_trans_I input {
24  width: 100%;
25  padding: 5px;
26  margin-bottom: 20px;
27  transition: all 0.3s ease;
28  display: flex;
29  flex-direction: column;
30  box-sizing: border-box;
31 }
```


[parte 02 – formulario transparente I]

Figura 337. Código de formularios parte II

```
1 .form_trans_I input:hover {
2   transition: all 0.3s ease;
3   color: white;
4   background-color: rgba(0, 0, 0, 0.2);
5 }
6 .form_trans_I h1, .form_trans_I h2, .form_trans_I h3 {
7   padding: 0 0 15px 0;
8   text-align: center;
9   color: white;
10 }
11 .form_trans_I .btn-response {
12   padding: 9px 9%;
13   border-radius: 10px;
14   margin-bottom: 10px;
15   width: 49%;
16   margin: 0 29%;
17   margin-bottom: 9px;
18   background-color: rgba(0, 0, 0, 0.6);
19   color: white;
20   display: flex;
21   flex-direction: column;
22   box-sizing: border-box;
23   justify-content: center;
24   align-items: center;
25 }
26 .form_trans_I .btn-response:active {
27   transform: scale(0.9);
28 }
29 .form_trans_I h1:hover, .form_trans_I .btn-response:hover {
30   color: turquoise;
31 }
```

[parte 03 – formulario transparente I]

Figura 338. Código de formularios parte III

```
1 @media only screen and (max-width: 1000px) {
2   .form_trans_I {
3     width: 59vh;
4   }
5
6   .form_trans_I h1, .form_trans_I h2, .form_trans_I h3 {
7     font-size: 3.4vh;
8   }
9
10  .form_trans_I:hover {
11    transform: scale(1.03);
12  }
13
14  .form_trans_I .btn-response:active {
15    transform: scale(1.09);
16  }
17 }
18 @media only screen and (max-width: 580px) {
19   .form_trans_I {
20     width: 95%;
21   }
22
23   .form_trans_I h1, .form_trans_I h2, .form_trans_I h3 {
24     font-size: 3vh;
25   }
26
27   .form_trans_I .btn-response:active {
28     transform: scale(1.03);
29   }
30
31   .form_trans_I:hover {
32     transform: scale(1);
33   }
34 }
35 @media only screen and (max-width: 240px) {
36   .form_trans_I {
37     width: 100%;
38   }
39
40   .form_trans_I h1, .form_trans_I h2, .form_trans_I h3 {
41     font-size: 2vh;
42   }
43 }
```

❖ Código del módulo Formulario transparente II

[parte 01 – formulario transparente II]

Figura 339. Código de formularios parte IV

```
1  .form_trans_II {
2    border: 1px solid rgba(255, 255, 255, 0.3);
3    border-radius: 5px;
4    padding: 3em;
5    position: absolute;
6    top: 50%;
7    left: 50%;
8    transform: translateX(-50%) translateY(-50%);
9    display: flex;
10   flex-direction: column;
11   box-sizing: border-box;
12 }
13
14 .form_trans_II h2 {
15   margin-top: 0px;
16   font-family: Source Sans Pro;
17   font-weight: lighter;
18   color: white;
19   font-size: 50px;
20   text-align: center;
21   margin-bottom: 35px;
22 }
```

[parte 02 – formulario transparente II]

Figura 340. Código de formularios parte V

```
1  .form_trans_II input {
2    display: block;
3    width: 320px;
4    height: 50px;
5    background: rgba(0, 0, 0, 0.3);
6    outline: none;
7    border: 1px solid rgba(0, 0, 0, 0.5);
8    font-family: Source Sans Pro;
9    font-weight: lighter;
10   font-size: 14px;
11   margin-bottom: 10px;
12   padding-left: 10px;
13   border-radius: 5px;
14   color: white;
15   display: flex;
16   flex-direction: column;
17   box-sizing: border-box;
18 }
19
20 .form_trans_II button {
21   width: 321px;
22   height: 50px;
23   font-size: 16px;
24   background: #000;
25   font-weight: lighter;
26   color: white;
27   border: 0px;
28   border-radius: 5px;
29   display: flex;
30   flex-direction: column;
31   box-sizing: border-box;
32   justify-content: center;
33   align-items: center;
34 }
```

[parte 03 – formulario transparente II]

Figura 341. Código de formularios parte VI

```
1 @media only screen and (max-width: 460px) {
2   .form_trans_II {
3     width: 91%;
4   }
5
6   .form_trans_II input {
7     width: 100%;
8   }
9
10  .form_trans_II button {
11    width: 100%;
12  }
13
14  .form_trans_II h2 {
15    font-size: 5vh;
16  }
17 }
18 @media only screen and (max-width: 400px) {
19   .form_trans_II {
20     width: 80%;
21     padding: 20px;
22   }
23
24   .form_trans_II h2 {
25     font-size: 3.5vh;
26   }
27 }
28 @media only screen and (max-width: 200px) {
29   .form_trans_II {
30     width: 83%;
31     padding: 15px;
32   }
33
34   .form_trans_II h2 {
35     font-size: 2.8vh;
36   }
37 }
```

Descripción de las propiedades más significativas:

A continuación, se procederá a detallar las propiedades más usadas y su aplicación. También se mostrará la compatibilidad web de cada propiedad para su mejor comprensión dejamos una leyenda para poder observarla en la figura 2.

Propiedad width:100% :

Se empleo width 100% para el ancho de este efecto ya que si bien esta propiedad tiene diversas unidades de medidas como:

- EM
- VH
- PÍXELES
- REM
- PORCENTAJE

Se fue descartando una a una al momento de probarla, ya que la al usar la unidad EM este se distorsionaba por su propia naturaleza, básicamente esta propiedad usa el ancho de la pantalla del dispositivo donde estemos y las va ajustando así que fue descartado.

A continuación, veamos la compatibilidad web al usar esta propiedad en la figura 3.

Propiedad background-color:rgba(0,0,0,0.5)

Esta propiedad se usa para darle un color de fondo para esta ocasión se decidió que sería una tonalidad oscura, así mismo es la encargada de transparencia debido a la poca intensidad de color que le estamos dando, cuando se usa la propiedad rgba este espera los siguientes parámetros rojo, verde y azul, pero cabe recalcar que el cuarto valor es el canal Alpha este es el encargado de la intensidad de transparencia u opacidad del color. Ya que es más comercial y combina con cualquier tonalidad.

A continuación, veamos la compatibilidad web al usar esta propiedad en la figura 5.

Propiedad display: flex

Con esta propiedad se puede aplicar la alineación de los elementos en una sola dirección sea horizontal o vertical, estos valores se puede conjugar con las distintos valores que tiene, se optó por esta propiedad ya que es una forma óptima de maquetar y de que los elementos no se desborden de sus contenedores como solía suceder al maquetar usando position o tables en la antigüedad además flexbox viene para usarlo con la versión 3 de css llamada css3 haciendo más robusta la maquetación de elementos. A continuación, veamos la compatibilidad web al usar esta propiedad en la figura 43.

Propiedad align-items: center

Para usar esta propiedad es un requisito indispensable tener habilitado display:flex ,luego de eso se puede conjugar align-items con los siguientes valores:

- CENTER
- END
- FLEX-END
- FLEX-START
- INHERIT
- INITIAL
- NORMAL
- REVERT
- SELF-END
- START
- STRETCH
- UNSET

En este caso se usó con el valor center ya que lo que ara es centrar en el centro de gravedad a los elementos, pero en dirección vertical. A continuación, veamos la compatibilidad web al usar esta propiedad en la figura 44.

Propiedad justify-content:center

La propiedad justify-content tiene como prerequisites haber declarado display:flex en el mismo elemento o en un elemento padre. Tiene las siguientes conjugaciones:

- CENTER
- END
- FLEX-END
- FLEX-START
- INHERIT
- INITIAL
- LEFT
- NORMAL
- REVERT
- RIGHT
- SPACE-AROUND
- SPACE-BETWEEN
- SPACE-EVENLY
- START
- STRETCH
- UNSET

Se opto por elegir el valor center ya que este centra en horizontal cualquier elemento. A continuación, veamos la compatibilidad web al usar esta propiedad en la figura 45.

Propiedad border:solid 4.4px

En esta propiedad le damos un border sólido, esto quiere decir que el borde será visible, además de que será una línea definida sin interrupciones. Luego la otra característica es que tendrá una anchura de 4.4px, dicho de mejor forma será una línea gruesa. A continuación, veamos la compatibilidad web al usar esta propiedad en la figura 37.

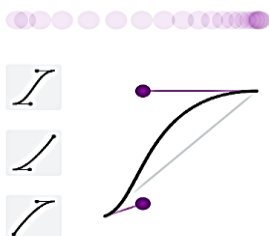
Propiedad transform:rotate(9.9deg)

La función que cumple es rotar la etiqueta por otra parte el prefijo deg hace referencia a los grados de inclinación en esta propiedad en particular es 9.9 lo que quiere decir que se mantendrá en una posición con inclinación de 9.9 grados. A continuación, veamos la compatibilidad web al usar esta propiedad en la figura 101.

Propiedad transition:all 0.5s ease

La propiedad transition tiene muchas conjugaciones en este caso en particular lo que indica es el tiempo de velocidad al cambiar de una propiedad de css a otra. El primer valor que tenemos es all el cual indica que todos los valores de transición el segundo valor es el tiempo expresado en segundos para ello se le agrega el prefijo s para indicarlo, luego tenemos la palabra reservada ease lo cual determina como será reproducida ease significa comenzar lento, acelerar, luego terminar lento nuevamente veamos el gráfico de esta animación:

Figura 342. Gráfico del valor ease en la propiedad transition



A continuación, veamos la compatibilidad web al usar esta propiedad en la figura 47.

Principales dificultades:

Para este efecto en concreto la principal dificultad fue organizar de distintas formas posibles, así como ver las posibles conjugaciones que pueden hacer con el efecto para que el usuario al final no tenga problemas al implementarlo. Esas dificultades surgieron a raíz de las siguientes preguntas:

- ¿Cómo hago para que sea responsive y permanezca el contenido sin quebrarse?

Si bien sirve aplicar las bases de la implementación responsive como usar flexible y valores en porcentaje no era suficiente, por ello se tuvo que aplicar puntos de corte en tres puntos pc, tablet y celular además se tuvo que editar propiedades en cada punto de corte para poder solucionar este problema.

- ¿Cómo hago transparente a los formularios?

Esta idea parecía simple antes de iniciar la codificación, luego al empezar a codificar así como probar distintas formas de hacerlo el tiempo se encargó de decirme que no iba hacer tan sencillo, se probó creando un before y un after para que hagan pantalla al formulario y no funciono, también se probó colocando la propiedad background-transparent ahí también hubo conflictos, otro método muy usado para hacer contenido transparente es usar la propiedad opacity este tampoco funciono ya que se tenía inputs de por medio.

¿Qué usamos para hacer las pruebas?

Se uso el navegador Google Chrome y opera mini, así como la siguiente página web que es un simulador online:

<http://www.responsinator.com>

Esto con motivos de asegurarnos que funcione en dispositivos móviles y tabletas. Ya que en base a lo trabajado no es del todo fiable usar el navegador para verificar la correcta maquetación de los elementos, así como tampoco es fiable depender de un software web, por ello se decidió diversificar en cuanto a las herramientas para probar las clases y efectos trabajados en el framework XRL8.

También usamos como editor de código visual studio code y su extensión live server para visualizar en vivo los efectos realizados en el css en tiempo real, también usamos el autoguardado que trae dicho editor de código para no estar guardando el proyecto a cada rato. Así conseguimos un mejor desempeño y evitamos el tedioso guardar cambios a cada rato, también evitamos el que la memoria cache se llene mostrándonos los mismos resultados.

FASE 3.- Hacerlo responsive

Con este efecto para que tome forma el maquetado se usó lo siguiente:

- Se uso la propiedad flex-box y sus posibles conjugaciones con cada uno de sus valores.
- Use porcentaje en las dimensiones de ancho de los contenedores para asegurar la gradualidad automática de los elementos así estos se adaptan fácilmente a cualquier dispositivo.
- Se trabajo con la propiedad @media_query en las resoluciones de pantalla para celulares y tabletas. Siempre empezando a maquetar por la parte de celulares y luego tabletas especialmente en los siguientes puntos de corte:

❖ Formulario transparente I

Figura 343. Uso de media query en el componente formularios parte I

```
1 @media only screen and (max-width: 1000px) {
2   .form_trans_I {
3     width: 59vh;
4   }
5
6   .form_trans_I h1, .form_trans_I h2, .form_trans_I h3 {
7     font-size: 3.4vh;
8   }
9
10  .form_trans_I:hover {
11    transform: scale(1.03);
12  }
13
14  .form_trans_I .btn-response:active {
15    transform: scale(1.09);
16  }
17 }
```


Figura 344. Uso de media query en el componente formularios parte II

```
1 @media only screen and (max-width: 580px) {
2   .form_trans_I {
3     width: 95%;
4   }
5   .form_trans_I h1, .form_trans_I h2, .form_trans_I h3 {
6     font-size: 3vh;
7   }
8 }
9 .form_trans_I .btn-response:active {
10  transform: scale(1.03);
11 }
12
13 .form_trans_I:hover {
14  transform: scale(1);
15 }
16 }
```

Figura 345. Uso de media query en el componente formularios parte III

```
1 @media only screen and (max-width: 240px) {
2   .form_trans_I {
3     width: 100%;
4   }
5
6   .form_trans_I h1, .form_trans_I h2, .form_trans_I h3 {
7     font-size: 2vh;
8   }
9 }
```

❖ Formulario transparente II

Figura 346. Uso de media query en el componente formularios parte IV

```
1 @media only screen and (max-width: 460px) {
2   .form_trans_II {
3     width: 91%;
4   }
5
6   .form_trans_II input {
7     width: 100%;
8   }
9
10  .form_trans_II button {
11    width: 100%;
12  }
13
14  .form_trans_II h2 {
15    font-size: 5vh;
16  }
17 }
```

Figura 347. Uso de media query en el componente formularios parte V

```
1 @media only screen and (max-width: 400px) {
2   .form_trans_II {
3     width: 80%;
4     padding: 20px;
5   }
6
7   .form_trans_II h2 {
8     font-size: 3.5vh;
9   }
10 }
```

Figura 348. Uso de media query en el componente formularios parte VI

```
1 @media only screen and (max-width: 200px) {
2   .form_trans_II {
3     width: 83%;
4     padding: 15px;
5   }
6
7   .form_trans_II h2 {
8     font-size: 2.8vh;
9   }
10 }
```

- También usamos las unidades de medida vh para texto ya que estas son imprescindibles a la hora de hacer responsive en un sitio web.
- Se redujo el padding de los contenedores para que estos tengan más espacio y no se desborde.

FASE 4.- Testing asegurarse que no tenga conflicto con otros efectos.

En esta parte del desarrollo de la clase se dispuso a probar poniéndolo con las demás clases juntándolas. Para ello lo que se hizo fue colocar dentro de un archivo index.html todas las clases trabajadas y ver como el módulo formulario transparente I y II respondían. En algunos casos había conflicto con los demás efectos para corregir esto se procedió a implementar en el desarrollo dos cosas básicas la primera fue:

- Para detectar que clase era la que causaba el conflicto y ver en que archivo scss estaba. Se procedía a inspeccionar la página para detectar donde estaba el problema. En muchas ocasiones no se daba con el archivo causante del error entonces en ese caso aplicábamos la frase “Divide y vencerás”. Lo que se procedió hacer fue comentar de uno en uno cada efecto en el archivo principal que los llamaba esto reducía las líneas de código de css, además como teníamos en tiempo real los cambios fue mucho más rápido detectar el problema. Ya que no se tenía que recargar a cada rato además otro punto a favor era que se auto compilaba los cambios de sass automáticamente gracias a nodeJS el cual mediante la consola de comandos de Windows cuando detectaba un cambio en las líneas de código se auto compilaba y generando o reemplazando el archivo css en el cual hacíamos las pruebas, esto aumento la productividad para crear los efectos ya que al comentar de uno en uno cada archivo sass llegaba un momento donde desaparecía el error entonces revisábamos y nos dimos cuenta de que era uno o en algunos casos hasta tres efectos que hacían conflicto con la clase escrita.
- Se procedió a escribir mediante comentarios en un archivo llamado `_error_logs.scss` la clase que hace conflicto con la clase escrita, además

de describir una nota breve en caso no se disponía de tiempo para solucionarlo luego, esto permitió tener un reporte más preciso de los incidentes que íbamos detectando.

- En este caso en particular ambos módulos hacen conflicto si se encuentra juntos ya que están codificados para estar en el centro de una pantalla, pero si se le colocan en un contenedor externo no hay problema.

FASE 5.- Aplicar la metodología OOCSS, llevándolo a SASS

- **Código css.** - Esta parte ya estaría completa por lo descrito anteriormente.

- **Código repetido.** - Identificación de código repetido

En este pequeño apartado luego de escribir y debugear el error se procedió a identificar el código css repetido para llevarlo a un archivo SCSS en sass, para su mayor optimización y aplicación de la metodología. Así que se procedió a reciclar el siguiente código:

Figura 349. Código repetido del componente formularios parte I

```
1  .form_trans_I {
2    width: 55%;
3    height: auto;
4    margin: 80px auto;
5    padding: 20px;
6    border-radius: 9px;
7    border: solid 4.4px black;
8    color: white;
9    background-color: rgba(0, 0, 0, 0.5);
10   transition: all 2s ease;
11   text-align: justify;
12   font-family: Courier, monospace;
13   display: flex;
14   flex-direction: column;
15   box-sizing: border-box;
16 }
```

Figura 350. Código repetido del componente formularios parte II

```
1  .form_trans_I input {
2    width: 100%;
3    padding: 5px;
4    margin-bottom: 20px;
5    transition: all 0.3s ease;
6    display: flex;
7    flex-direction: column;
8    box-sizing: border-box;
9  }
```

Figura 351. Código repetido del componente formularios parte III

```
1 .form_trans_I .btn-response {
2   padding: 9px 9%;
3   border-radius: 10px;
4   margin-bottom: 10px;
5   width: 49%;
6   margin: 0 29%;
7   margin-bottom: 9px;
8   background-color: rgba(0, 0, 0, 0.6);
9   color: white;
10  display: flex;
11  flex-direction: column;
12  box-sizing: border-box;
13  justify-content: center;
14  align-items: center;
15 }
```

Figura 352. Código repetido del componente formularios parte IV

```
1 .form_trans_II {
2   border: 1px solid rgba(255, 255, 255, 0.3);
3   border-radius: 5px;
4   padding: 3em;
5   position: absolute;
6   top: 50%;
7   left: 50%;
8   transform: translateX(-50%) translateY(-50%);
9   display: flex;
10  flex-direction: column;
11  box-sizing: border-box;
12 }
```

Figura 353. Código repetido del componente formularios parte V

```
1 .form_trans_II input {
2   display: block;
3   width: 320px;
4   height: 50px;
5   background: rgba(0, 0, 0, 0.3);
6   outline: none;
7   border: 1px solid rgba(0, 0, 0, 0.5);
8   font-family: Source Sans Pro;
9   font-weight: lighter;
10  font-size: 14px;
11  margin-bottom: 10px;
12  padding-left: 10px;
13  border-radius: 5px;
14  color: white;
15  display: flex;
16  flex-direction: column;
17  box-sizing: border-box;
18 }
```

Figura 354. Código repetido del componente formularios parte VI

```
1 .form_trans_II button {
2   width: 321px;
3   height: 50px;
4   font-size: 16px;
5   background: #000;
6   font-weight: lighter;
7   color: white;
8   border: 0px;
9   border-radius: 5px;
10  display: flex;
11  flex-direction: column;
12  box-sizing: border-box;
13  justify-content: center;
14  align-items: center;
15 }
```

- **Crear un objeto.** - Se describe la lógica de la función para luego crear un `mixing`.

Una vez identificado el código `css` repetido se procede hacer una función la cual se ejecuta al llamado del código, mediante esta técnica nuestro código será más puro y limpio. Cabe recalcar que `formulario transparente I`, así como `formulario transparente II` comparte características similares.

Paralelamente en esta parte se determina la lógica de las funciones y ciclos como lo son `for`, `while`, `each`. También determinamos el pase de parámetros, así como la condicional que llevaran dentro, por otra parte, y no menos importante se define el nombre con el cual se llamara a las funciones.

```
function base_formulario_trans($condicion){  
  
  display:flex;  
  flex-direccion:column;  
  box-sizing:border-box;  
  if($condicion == 'centrado'){  
    justify-content:center;  
    align-items:center;  
  }  
}
```

Luego de tener la lógica se procede a llevarlo a `SASS` para ello tenemos dos opciones, la primera es usar `@function` ya que como su nombre ase referencia es para crear una función la otra opción es usar `@mixin`, pero ambas tienen diferencias. La primera retorna un valor, mientras `@mixin` solo llama a los valores en este caso no necesitamos que retorne un valor por ello usaremos `@mixin`. Entonces quedaría de la siguiente forma:

Figura 355. Creación del objeto `base_formulario_trans` en función `mixin`

```
1 @mixin base_formulario_trans($condicion){  
2   display: flex;  
3   flex-direction: column;  
4   box-sizing: border-box;  
5   @if $condicion == 'centrado'{  
6     justify-content: center;  
7     align-items: center;  
8   }  
9 }
```

- **OOCSS**. – Luego de tener la función o la lógica para reciclar el código se procede a integrar la metodología OOCSS para ello debemos llamar al mixin para hacerlo solo necesitamos agregar @include “nombre del mixin”. Luego de integrarlo ya estará aplicada correctamente la metodología OOCSS

❖ Formulario transparente I

Figura 356. Metodología OOCSS en el componente formularios parte I

```

1  .form_trans_I{
2      width:$formulario_trans_I_ancho;
3      height:$formulario_trans_I_alto;
4      margin:80px auto;
5      padding: 20px;
6      border-radius:9px;
7      border:solid 4.4px $color-negro-1;
8      color: $color-blanco-1;
9      background-color: rgba(0, 0, 0, 0.48);
10     transition: all 2s ease;
11     text-align:justify;
12     font-family:$fuente-periodico;
13     @include base_formulario_trans('normal');
14     &:hover{
15         transition: $formulario_trans_I_tiempo_animacion;
16         transform: scale(1.1);
17         transform:rotate(9.9);
18         color: $color-blanco-1;
19     }
20     input{
21         width: 100%;
22         padding: 5px;
23         margin-bottom: 20px;
24         transition: $formulario_trans_I_tiempo_animacion;
25         @include base_formulario_trans('normal');
26         &:hover{
27             transition: $formulario_trans_I_tiempo_animacion;
28             color: $color-blanco-1;
29             background-color: rgba(0,0,0,0.2)
30         }
31     }
32     h1,h2,h3{
33         padding: 0 0 15px 0;
34         text-align: center;
35         color: $color-blanco-1;
36     }
37     .btn-response{
38         @include base_formulario_trans('centrado');
39         padding: 9px 9%;
40         border-radius: 10px;
41         margin-bottom: 10px;
42         width: 49%;
43         margin:0 29%;
44         margin-bottom: 9px;
45         background-color: rgba(0,0,0,0.6);
46         color: $color-blanco-1;
47         &:active{
48             transform: scale(0.9);
49         }
50     }
51     h1:hover, .btn-response:hover{
52         color:$formulario_trans_I_fondo_boton;
53     }
54 }

```

Figura 357 Metodología OOCSS en el componente formularios parte II

```
1 .form_trans_II{
2   border: $formulario_trans_II_grosor_borde $formulario_trans_II_tipo_borde $formulario_trans_II_color_fondo;
3   border-radius: 5px;
4   padding: 3em;
5   position: absolute;
6   top: 50%;
7   left: 50%;
8   transform: translateX(-50%) translateY(-50%);
9   @include base_formulario_trans('normal');
10 }
```

Figura 358 Metodología OOCSS en el componente formularios parte III

```
1 .form_trans_II input{
2   display: block;
3   width:$formulario_trans_II_btn_ancho;
4   height:$formulario_trans_II_btn_alto;
5   background: rgb(0, 0,0,0.3);
6   outline: none;
7   border: 1px solid rgba(0,0, 0, 0.5);
8   font-family: Source Sans Pro;
9   font-weight: lighter;
10  font-size: 14px;
11  margin-bottom: 10px;
12  padding-left: 10px;
13  border-radius: 5px;
14  color:$color-blanco-1;
15  @include base_formulario_trans('normal');
16 }
```

Figura 359 Metodología OOCSS en el componente formularios parte IV

```
1 .form_trans_II button{
2   width: 321px;
3   height: 50px;
4   font-size: 16px;
5   background: #000;
6   font-weight: lighter;
7   color: $color-blanco-1;
8   border: 0px;
9   border-radius: 5px;
10  @include base_formulario_trans('centrado');
11 }
```

Bien veamos el tema de las variables estas variables se encuentran en un archivo sass denominado variables valga la redundancia donde se encuentran las siguientes variables:

❖ Formulario transparente I

Figura 360 Variables sass del componente formularios parte I

```
1 $formulario_trans_I_ancho:55%;
2 $formulario_trans_I_alto:auto;
3 $formulario_trans_I_tiempo_animacion:all .3s ease;
4 $formulario_trans_I_fondo_boton:turquoise;
```

❖ Formulario transparente II

Figura 361 Variables sass del componente formularios parte II

```
1 $formulario_trans_II_grosor_borde:1px;
2 $formulario_trans_II_tipo_borde:solid;
3 $formulario_trans_II_color_fondo:rgba(255,255, 255, 0.3);
4 $formulario_trans_II_btn_ancho:320px;
5 $formulario_trans_II_btn_alto:50px;
```

Como se ha visto SASS es un preprocesador de css por ello se ejecuta antes del css tradicional esto permite añadir variables, así como agregar condicionales, etc. Con la aplicación de la metodología como se observa se reutiliza el código css, además de que se hace escalable el proyecto. Ya que, si en un futuro queremos cambiar o añadir más propiedades css, solo cambiaríamos las variables o añadiríamos más propiedades en el mixin.

FASE 6.- Documentar en la web site www.phooldx.com

Finalmente, si el efecto paso por todos los de más filtros se procede a su documentación respectiva, para documentar todos los efectos se creó una página web de documentación usando clases y elementos del mismo framework. A continuación, se describirá como se procede a la documentación de la misma. Siguiendo estos pasos:

- Se copia las etiquetas de HTML5 junto con la clase a un txt temporal para no perder la sintaxis
- Se guarda los cambios en el compilador de SASS
- Como es la parte final se procedió hacer copias de seguridad para no perder el avance, en archivos RAR. Estos archivos estaban por versiones enumerados además se agregaba un pequeño texto describiendo el avance. Este guardado se hacía de manera local en un USB.
- Para una mayor seguridad del avance del código se procedió luego a copiar la versión del archivo RAR junto a la descripción del mismo.

Guardado de código en GitHub. - Luego de tener la copia del proyecto a nivel físico en un USB, se procedió a guardarlo en un repositorio de GitHub para lo cual se usó Git, para ello se procedió de la siguiente forma:

- Se guardaba todo el proyecto con el siguiente comando que se observa en la figura 12.

- Luego se verificaba los cambios guardados con el siguiente comando que se observa a continuación en la figura 13.
- Después se hacía el commit, es decir se les asignaba un nombre a los cambios realizados con el siguiente comando que se puede apreciar en la figura 14.
- Luego se procede a empaquetar los datos para que se suban al repositorio de GitHub con el siguiente código que se observa en la figura 15
- Finalmente, luego de haber enviados el proyecto empaquetado hacia GitHub como se puede apreciar en la imagen, este queda guardado en dicho repositorio.

Figura 362. Visualización en github del componente formularios



Se guarda en una escala de tiempo, con todos los cambios hechos desde la creación del proyecto, esto ayuda a tener un mejor control de versiones. Asimismo, también ayuda a que más personas se sumen al proyecto y puedan clonar el proyecto como trabajar por separado el proyecto, entre otras muchas ventajas.

- Así mismo se manejó un archivo llamado bitácora.txt donde aquí se registró todos los incidentes ocurridos, así como los cambios hechos, se registraba también las metas cumplidas, como las tareas nuevas para desarrollar al día siguiente. Al finalizar con un efecto por completo se quitaba de la lista de prioridades así dábamos pase al siguiente efecto o problema ha resolver. Esto servía para tener un mejor orden y saber ejecutar prioridades.
- Luego de tener guardado el avance y de registrar el avanzase, se procedió a copiar del archivo documentación Etiquetas.txt el fragmento de código que se copió anteriormente para luego pegarlo en www.nosetup.org esta

página web se encargaba de convertir html en texto plano para poder visualizar html

- Antes de finalizar se procedió a crear un archivo llamado modulo_trans_I.html así como otro archivo por separado llamado modulo_trans_II.html donde se dividía en tres partes:
 - En la primera se describía un poco al efecto que era y que hacia
 - En la segunda parte se procedía a copiar el código ya formateado obtenido de www.nosetup.com para luego pegarlo.
 - En la tercera parte lo que se hacía era mostrar un apartado llamado estructura donde se describía como se llamaban las variables para que sea modificado en SASS.
- Finalmente, el ejemplo listo para usarse queda de la siguiente forma:

❖ Modulo formulario transparente I

Figura 363. Código html5 final del componente formularios parte I

```
<body style="background-image: url(multimedia/pollo.png)">
<div class="form-trans">
  <h1>Formulario</h1>

  <label for="">Nombre:</label>
  <input type="text" name="" id="">

  <label for="">Apellidos:</label>
  <input type="text" name="" id="">

  <label for="">Edad:</label>
  <input type="text" name="" id="">

  <button type="submit" class="btn-response">Enviar</button>
</div>
</body>
```

❖ Modulo formulario transparente II

Figura 364. Código html5 final del componente formularios parte II

```
<body style="background-color: teal;">
<form class="form_trans_II" >
  <h2>INICIAR</h2>
  <input type="text" placeholder="Ingresar Usuario" required>
  <input type="password" placeholder="Ingresar Contraseña" required>
  <button class="" >Ingresar</button>
</form>
</body>
```

CENTRADO HORIZONTAL Y VERTICAL

FASE 1.- Investigación del efecto y su importancia

Como primer paso lo primero que se hizo fue investigar e indagar sobre el efecto, para iniciar en muchas ocasiones es común querer centrar elementos HTML en la mayoría de casos es difícil centrar este elemento porque existen distintas propiedades para hacerlo, además existen otros factores que lo hacen laborioso uno de ellos es saber que propiedades usando al maquetar el elemento padre. En XRL8 creemos firmemente que mientras menos clases y menos código se tenga que hacer vamos por buen camino así que se decidió poner punto final al centrado de elementos. Así mismo una de las palabras frecuentes buscadas según Google trends como se observa el grafico es como centrar un div:

Figura 365. Grafico de barras de Google trends sobre como centrar un div



Si bien existe framework el cual logra centrar div en el centro de gravedad de una pantalla dicha de otra forma centrado vertical y horizontal estos desde mi punto de vista son muy largos de implementar y hasta un poco engorroso ya que en muchos casos debe tener varias capas o niveles en la maquetación para que haga efecto. Lo que se propone en esta sección es hacer lo mismo que otros frameworks pero en una o dos clases a lo mucho, evitando esas capas de etiquetas una sobre otras, así ayudando al usuario final quienes son los desarrolladores web.

Otro del motivo el cual fue elegido el efecto para formar parte del framework XRL8 fue que el proyecto presente debe estar a la altura de los

frameworks que están en la actualidad siendo usados en la industria del desarrollo web como son:

- **Bootstrap.** - El presente framework cuenta con las clases row,vh-100,justify-content-around,align-items-center las cuales todas juntas ayudan a centrar un elemento o etiqueta HTML en el centro de la pantalla, si desean saber más sobre las clases mencionadas puede visitar la web oficial de documentación de dicho framework mediante el siguiente enlace: <https://getbootstrap.com/docs/4.0/utilities/flex/>
- **Bulma.**- Este framework cuenta con distintas clases para centrar un elemento HTML y la unión de todas estas clases hace que sea posible el centrar un elemento ya que individualmente no se puede estas clases son columns,is-centered,columna,is-half estas son empleadas en distintas capas para que hagan efecto, para ver cómo se emplea y formas de uso revisar el siguiente enlace que los llevara a la documentación oficial de dicho framework: <https://bulma.io/documentation/columns/options/#centering-columns>
- **TailwindCSS.**- Quisa sea este el framework de css más usado en los últimos 3 años sobre pasando a los demás por sus formas de integración con NodeJS y react.Este framework al igual que los demás para poder centrar se necesita recurrir a varias clases para poder centrar adecuadamente un elemento HTML estas clases son grid,place-items-center,h-screen cabe recalcar que este framework se caracteriza por no tener clases predefinidas para efectos sino todo lo contrario tiene un conjunto de clases en el cual el usuario puede elegir los efectos más simples para colocar a su diseño. Para revisar su documentación ingresar al siguiente enlace: <https://tailwindcss.com/docs/justify-content>
- **Materialize.**- Otro framework del medio es materialize el cual cuenta con una sección llamada helpers el cual tiene un sección donde muestran como centrar elementos usando las clases valign-wraper y center-align. Lo podremos observar más detalladamente accediendo mediante el siguiente link: <https://materializecss.com/helpers.html>

- **UIKIT.-** También este framework cuenta con una clase para centrar elementos HTML en el centro de gravedad la clase empleada en este caso tiene por nombre `.uk-position-center`, para ver más sobre este framework recomiendo revisar la documentación en el siguiente enlace:

<https://getuikit.com/docs/position>

- **Semantic UI.-** A continuación tenemos otro framework de css muy completo la verdad el cual tiene soporte y componentes para ser usado con react, la clase la cual ayuda a centrar un elemento no es solo una clase ya que se tiene que combinar varias para poder centrarla esas clases son `ui,grid,middle,aligned,row,column,text,container, segment, inverted` con todas ellas en diferentes capas se puede centrar un div o etiqueta HTML, podemos ver una gran cantidad de ejemplos mediante el siguiente enlace:

<https://semantic-ui.com/collections/grid.html>

Modulo: Un módulo según definición propia es una sección de trabajo principal a comparación de un submódulo este puede ser denominado la estructura padre y los submódulos estructuras hijas. En este efecto en particular el módulo vendría a ser la siguiente clase:

- **Centrador pro.** - Para este efecto nos basamos en el uso de flex-box así como en sus conjugaciones para una centrada poderosa y a su vez efectiva.

Submódulos: Considere llamarle submódulo, a los efectos o clases que están dentro de un módulo en este caso el módulo es centrador pro, pero el submódulo comparte atributos similares. Un submódulo según denominación propia vendría hacer un efecto similar al original, pero con características diferentes, hagamos un ejemplo para comprender esto de una manera mejor.

- **Centrador_pro_alternativo.** - Este sería un sub modulo ya que tendría propiedades compartidas con el módulo original, pero discreparían en funcionalidades mientras el módulo original tiene flex-box este tiene un posicionamiento absoluto.

Estos submódulos son muy útiles ya que desde mi punto de vista ayuda a diversificar los efectos y las clases. Beneficiando al usuario final en la construcción de sitios web.

FASE 2.- Codificar el efecto

Antes de ver esta sección a fondo quisiera comentar que la codificación fue sencilla mas no lo fue el saber que propiedad emplear.

Bien luego de tener claro que efecto queríamos construir y porque lo íbamos a construir como se vio en el capítulo anterior se procedió a escribir código como se muestra a continuación.

Código completo

❖ Código del módulo centrado pro

Figura 366. Código de centrado horizontal y vertical parte I

```
1  .centrador_pro {
2    width: 100%;
3    height: 100%;
4    display: flex;
5    justify-content: center;
6    align-items: center;
7  }
```

❖ Código del submódulo centrado pro alternativo

Figura 367. Código de centrado horizontal y vertical parte II

```
1  .centrador_pro_alterno {
2    width: 100%;
3    height: 100%;
4    display: flex;
5    justify-content: center;
6    align-items: center;
7    position: absolute;
8  }
```

Descripción de las propiedades más significativas:

A continuación, se procederá a detallar las propiedades más usadas y su aplicación. También se mostrará la compatibilidad web de cada propiedad para su mejor comprensión dejamos una leyenda la cual se observa en la figura 2.

Propiedad width:100% :

Se empleo width 100% para el ancho de este efecto ya que si bien esta propiedad tiene diversas unidades de medidas como:

- EM
- VH
- PIXELES
- REM
- PORCENTAJE

Se fue descartando una a una al momento de probarla, ya que la al usar la unidad EM este se distorsionaba por su propia naturaleza, básicamente esta propiedad usa el ancho de la pantalla del dispositivo donde estemos y las va ajustando así que fue descartado.

A continuación, veamos la compatibilidad web al usar esta propiedad en la figura 3.

Propiedad height:100%:

Luego de ver las unidades de medida en la propiedad anterior se decidió trabajar en porcentaje, ya que se necesitaba un valor que responsive, en ese sentido se descartó las propiedades que toman el ancho del dispositivos como son REM y EM por otra parte también se descartó a los que toman el alto del dispositivo como son VH .Luego de reducir nuestro marco de trabajo se tenía PORCENTAJE o PIXELES así que descartamos a PIXELES porque tampoco necesitábamos algo estático sino más bien algo que sea ajustable al alto de la pantalla.

A continuación, veamos la compatibilidad web al usar esta propiedad en la figura 4.

Propiedad display: flex

Con esta propiedad se puede aplicar la alineación de los elementos en una sola dirección sea horizontal o vertical, estos valores se puede conjugar con las distintos valores que tiene, se optó por esta propiedad ya que es una forma óptima de maquetar y de que los elementos no se desborden de sus contenedores como solía suceder al maquetar usando position o tables en la antigüedad además flexbox viene para usarlo con la versión 3 de css llamada

css3 haciendo más robusta la maquetación de elementos. A continuación, veamos la compatibilidad web al usar esta propiedad en la figura 43.

Propiedad align-items: center

Para usar esta propiedad es un requisito indispensable tener habilitado `display:flex`, luego de eso se puede conjugar `align-items` con los siguientes valores:

- CENTER
- END
- FLEX-END
- FLEX-START
- INHERIT
- INITIAL
- NORMAL
- REVERT
- SELF-END
- START
- STRETCH
- UNSET

En este caso se usó con el valor `center` ya que lo que ara es centrar en el centro de gravedad a los elementos, pero en dirección vertical. A continuación, veamos la compatibilidad web al usar esta propiedad en la figura 44.

Propiedad justify-content:center

La propiedad `justify-content` tiene como prerequisites haber declarado `display:flex` en el mismo elemento o en un elemento padre. Tiene las siguientes conjugaciones:

- CENTER
- END
- FLEX-END
- FLEX-START
- INHERIT
- INITIAL
- LEFT
- NORMAL
- REVERT
- RIGHT
- SPACE-AROUND

- SPACE-BETWEEN
- SPACE-EVENLY
- START
- STRETCH
- UNSET

Se opto por elegir el valor center ya que este centra en horizontal cualquier elemento. A continuación, veamos la compatibilidad web observando la figura 45.

Propiedad position: absolute

La propiedad ABSOLUTE lo que hace es sobreponerse sobre cualquier elemento que tenga static o relative. Bien en la actualidad se considera mala práctica a la hora de maquetar usar la propiedad position para maquetar, ya que no es favorable para hacerlo responsive, en vez de eso ahora se usa flexbox, y en algunos casos grid. Dicho de paso ambas propiedades formaron parte en la actualización llamada css3.

La propiedad position tiene varios valores los cuales son:

- STATIC
- RELATIVE
- ABSOLUTE
- STICKY
- FIXED

A continuación, veamos la compatibilidad web al usar esta propiedad en los principales navegadores.

Figura 368. Compatibilidad web, propiedad position

CSS property: position

Usage % of all users Global 94,13%

Current aligned Usage relative Date relative Filtered All

IE	Edge	Firefox	Chrome	Safari	Opera	Safari on iOS	Opera Mini	Android Browser	Opera Mobile	Chrome for Android	Firefox for Android	UC Browser for Android	Samsung Internet	QQ Browser	Baidu Browser	KaiOS Browser
6-10	12-99	2-98	4-100	3.1-15.3	10-85	3.2-15.3		2.1-4.4.4	12-12.1				4-15.0			
11	100	99	101	15.4	86	15.4	all	100	64	100	99	12.12	16.0	10.4	7.12	2.5
		100-101	102-104	TP	87											

Principales dificultades:

Para este efecto en concreto la principal dificultad fue organizar de distintas formas posibles, así como ver las posibles conjugaciones que pueden hacer con el efecto para que el usuario al final no tenga problemas al implementarlo. Esas dificultades surgieron a raíz de las siguientes preguntas:

- ¿Qué propiedad uso para centrar elementos?

Bien parece una pregunta simple, pero no es así para centrar elementos existen diversas formas, tenemos las siguientes propiedades para hacerlo:

- Margin. - Esta propiedad hubiera servido, pero se hubiera requerido de mucho más código. Otro punto en contra es que es la más usada si se hubiera implementado esta propiedad hubiera sido difícil para el usuario redefinirla.
- Position. - Esta propiedad trae consigo varios valores para conjugar uno de esos valores es absolute, el cual fue elegido por las propiedades descrita anteriormente.
- Flex-box. - Esta propiedad se usó porque en la actualidad es una buena práctica además de que el framework XRL8 esta maquetado usando flexbox y usando las buenas prácticas que esta propiedad trae.

¿Qué usamos para hacer las pruebas?

Se uso el navegador Google Chrome y opera mini, así como la siguiente página web que es un simulador online:

<http://www.responsinator.com>

Esto con motivos de asegurarnos que funcione en dispositivos móviles y tabletas. Ya que en base a lo trabajado no es del todo fiable usar el navegador para verificar la correcta maquetación de los elementos, así como tampoco es fiable depender de un software web, por ello se decidió diversificar en cuanto a las herramientas para probar las clases y efectos trabajados en el framework XRL8.

También usamos como editor de código visual studio code y su extensión live server para visualizar en vivo los efectos realizados en el css en tiempo real, también usamos el autoguardado que trae dicho editor de código para no estar guardando el proyecto a cada rato. Así conseguimos un mejor desempeño y evitamos el tedioso guardar cambios a cada rato, también evitamos el que la memoria cache se llene mostrándonos los mismos resultados.

FASE 3.- Hacerlo responsive

Con este efecto en particular fue sencillo ya que la propiedad width estaba en % también se usó flexbox que garantiza que no se desborde el contenido en la maquetación, además de que no se necesitó una maquetación extra como veremos más adelante en el presente informe. En caso se hubiera requerido el uso de otras propiedades para hacerlo responsive lo que hubiéramos usado sería:

- Hubiéramos trabajado con la propiedad @media_query en las resoluciones de pantalla para celulares y tabletas. Siempre empezando a maquetar por la parte de celulares y luego tabletas.
- También hubiéramos usado las unidades de medida vh o rem para texto y contenedores ya que estas son imprescindibles a la hora de hacer responsive un sitio web.

FASE 4.- Testing asegurarse que no tenga conflicto con otros efectos.

En esta parte del desarrollo de la clase se dispuso a probar poniéndolo con las demás clases juntándolas. Para ello lo que se hizo fue colocar dentro de un archivo index.html todas las clases trabajadas y ver como el módulo y submódulo respondían. En algunos casos había conflicto con los demás efectos para corregir esto se procedió a implementar en el desarrollo dos cosas básicas la primera fue:

- Para detectar que clase era la que causaba el conflicto y ver en que archivo scss estaba. Se procedía a inspeccionar la página para detectar donde estaba el problema. En muchas ocasiones no se daba con el archivo causante del error entonces en ese caso aplicábamos la frase “Divide y vencerás”. Lo que se procedió hacer fue comentar de uno en uno cada

efecto en el archivo principal que los llamaba esto reducía las líneas de código de css, además como teníamos en tiempo real los cambios fue mucho más rápido detectar el problema. Ya que no se tenía que recargar a cada rato además otro punto a favor era que se auto compilaba los cambios de sass automáticamente gracias a nodeJS el cual mediante la consola de comandos de Windows cuando detectaba un cambio en las líneas de código se auto compilaba y generando o reemplazando el archivo css en el cual hacíamos las pruebas, esto aumento la productividad para crear los efectos ya que al comentar de uno en uno cada archivo sass llegaba un momento donde desaparecía el error entonces revisábamos y nos dimos cuenta de que era uno o en algunos casos hasta tres efectos que hacían conflicto con la clase escrita.

- Se procedió a escribir mediante comentarios en un archivo llamado `_error_logs.scss` la clase que hace conflicto con la clase escrita, además de describir una nota breve en caso no se disponía de tiempo para solucionarlo luego, esto permitió tener un reporte más preciso de los incidentes que íbamos detectando.

FASE 5.- Aplicar la metodología OOCSS, llevándolo a SASS

- **Código css.** - Esta parte ya estaría completa por lo descrito anteriormente.
- **Código repetido.** - Identificación de código repetido

En este pequeño apartado luego de escribir y debugear el error se procedió a identificar el código css repetido para llevarlo a un archivo SCSS en sass, para su mayor optimización y aplicación de la metodología. Así que se procedió a reciclar el siguiente código:

❖ Modulo centrado pro

Figura 369. Código repetido del componente centrado horizontal y vertical parte I

```
1 .centrador_pro {
2   width: 100%;
3   height: 100%;
4   display: flex;
5   justify-content: center;
6   align-items: center;
7 }
```

❖ Submódulo centrado pro alternativo

Figura 370. Código repetido del componente centrado horizontal y vertical parte II

```
1 .centrador_pro_alterno {
2   width: 100%;
3   height: 100%;
4   display: flex;
5   justify-content: center;
6   align-items: center;
7   position: absolute;
8 }
```

- **Crear un objeto.** - Se describe la lógica de la función para luego crear un mixing.

Una vez identificado el código css repetido se procede hacer una función la cual se ejecuta al llamado del código, mediante esta técnica nuestro código será más puro y limpio. Cabe recalcar que centrador pro, así como el submódulo comparte características similares.

Paralelamente en esta parte se determina la lógica de las funciones y ciclos como lo son for, while, each. También determinamos el pase de parámetros, así como la condicional que llevaran dentro, por otra parte, y no menos importante se define el nombre con el cual se llamara a las funciones.

```
function centrador_pro_base(condicion){
  width:100%;
  height:100%;
  display:flex;
  justify-content:center;
  align-items:center;
  if(condicion == 'absolute'){
    position:absolute;
  }
}
```

Luego de tener la lógica se procede a llevarlo a SASS para ello tenemos dos opciones, la primera es usar @function ya que como su nombre ase referencia es para crear una función la otra opción es usar @mixin, pero ambas tienen diferencias. La primera retorna un valor, mientras @mixin solo llama a los valores en este caso no necesitamos que retorne un valor por ello usaremos @mixin. Entonces quedaría de la siguiente forma:

Figura 371. Creación del objeto centrador_pro_base en función mixin

```
1 @mixin centrador_pro_base($condicion){
2     width: 100%;
3     height: 100%;
4     display: flex;
5     justify-content: center;
6     align-items: center;
7
8     @if($condicion == 'absolute'){
9         position: absolute;
10    }
11 }
```

- **OOCSS**. – Luego de tener la función o la lógica para reciclar el código se procede a integrar la metodología OOCSS para ello debemos llamar al mixin para hacerlo solo necesitamos agregar @include “nombre del mixin”. Luego de integrarlo ya estará aplicada correctamente la metodología OOCSS

❖ **Modulo centrador pro**

Figura 372. Metodología OOCSS en el componente centrado horizontal y vertical parte I

```
1 .centrador_pro{
2     @include centrador_pro_base('normal');
3 }
```

❖ **Submódulo centrador alternativo**

Figura 373. Metodología OOCSS en el componente centrado horizontal y vertical parte II

```
1 .centrador_pro_alterno{
2     @include centrador_pro_base('absolute');
3 }
```

Bien volviendo al tema de las variables estas variables se encuentran en un archivo sass denominado variables valga la redundancia donde se encuentran las siguientes variables:

Figura 374. Variables sass del componente centrado vertical y horizontal

```
1 $centrador_pro_ancho:100%;
2 $centrador_pro_alto:100%;
3 $centrador_pro_position:absolute;
```

Como se ha visto SASS es un preprocesador de css por ello se ejecuta antes del css tradicional esto permite añadir variables, así como agregar condicionales, etc. Con la aplicación de la metodología como se observa se reutiliza el código css, además de que se hace escalable el proyecto. Ya que, si en un futuro queremos cambiar o añadir más propiedades css, solo cambiaríamos las variables o añadiríamos más propiedades en el mixin.

FASE 6.- Documentar en la web site www.phooldx.com

Finalmente, si el efecto paso por todos los de más filtros se procede a su documentación respectiva, para documentar todos los efectos se creó una página web de documentación usando clases y elementos del mismo framework. A continuación, se describirá como se procede a la documentación de la misma. Siguiendo estos pasos:

- Se copia las etiquetas de HTML5 junto con la clase a un txt temporal para no perder la sintaxis
- Se guarda los cambios en el compilador de SASS
- Como es la parte final se procedió hacer copias de seguridad para no perder el avance, en archivos RAR. Estos archivos estaban por versiones enumerados además se agregaba un pequeño texto describiendo el avance. Este guardado se hacía de manera local en un USB.
- Para una mayor seguridad del avance del código se procedió luego a copiar la versión del archivo RAR junto a la descripción del mismo.

Guardado de código en GitHub. - Luego de tener la copia del proyecto a nivel físico en un USB, se procedió a guardarlo en un repositorio de GitHub para lo cual se usó Git, para ello se procedió de la siguiente forma:

- Se guardaba todo el proyecto con el siguiente comando que se observa en la figura 12.
- Luego se verificaba los cambios guardados con el siguiente comando que se observa en la figura 13.

- Después se hacía el commit, es decir se les asignaba un nombre a los cambios realizados con el siguiente comando que se observa en la figura 14.
- Luego se procede a empaquetar los datos para que se suban al repositorio de GitHub con el siguiente código que se observa en la figura 15.
- Finalmente, luego de haber enviados el proyecto empaquetado hacia GitHub como se puede apreciar en la imagen, este queda guardado en dicho repositorio.

Figura 375. Visualización en github del componente centrado horizontal y vertical



Se guarda en una escala de tiempo, con todos los cambios hechos desde la creación del proyecto, esto ayuda a tener un mejor control de versiones. Asimismo, también ayuda a que más personas se sumen al proyecto y puedan clonar el proyecto como trabajar por separado el proyecto, entre otras muchas ventajas.

- Así mismo se manejó un archivo llamado bitácora.txt donde aquí se registró todos los incidentes ocurridos, así como los cambios hechos, se registraba también las metas cumplidas, como las tareas nuevas para desarrollar al día siguiente. Al finalizar con un efecto por completo se quitaba de la lista de prioridades así dábamos pase al siguiente efecto o problema ha resolver. Esto servía para tener un mejor orden y saber ejecutar prioridades.
- Luego de tener guardado el avance y de registrar el avanzase, se procedió a copiar del archivo documentación Etiquetas.txt el fragmento de código que se copió anteriormente para luego pegarlo en www.nosetup.org esta página web se encargaba de convertir html en texto plano para poder visualizar html

- Antes de finalizar se procedió a crear un archivo llamado modulo_centrado_V_&_H.html donde se dividía en tres partes:
 - En la primera se describía un poco al efecto que era y que hacía
 - En la segunda parte se procedía a copiar el código ya formateado obtenido de www.nosetup.com para luego pegarlo.
 - En la tercera parte lo que se hacía era mostrar un apartado llamado estructura donde se describía como se llamaban las variables para que sea modificado en SASS.
- Finalmente, el ejemplo listo para usarse queda de la siguiente forma:

❖ Modulo centrado pro

Figura 376. Código html5 final del componente centrado horizontal y vertical parte I

```
<div class="centrado_pro">  
</div>
```

❖ Submódulo centrado alternativo

Figura 377. Código html5 final del componente centrado horizontal y vertical parte II

```
<div class="centrado_pro_alternativo">  
</div>
```

Menú acordeón

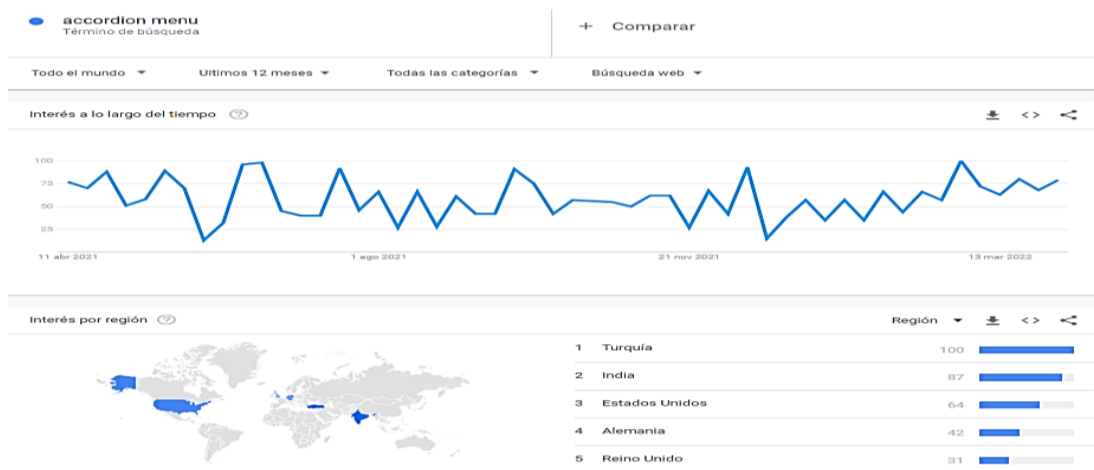
FASE 1.- Investigación del efecto y su importancia

Como primer paso lo primero que se hizo fue investigar e indagar sobre el efecto, para empezar un menú acordeón es un menú pero se caracteriza en ser uno vertical, ya que lo más frecuente es ver menús horizontal en muchas páginas web, este menú vertical suele ser usado en foros o en la sección de preguntas y respuestas debido a lo beneficioso que es poder expandirlo de ahí proviene la terminología acordeón que es desplegable esto al haber una suma considerable de usuarios interactuando, es muy útil la verdad de esta forma se logra optimizar la velocidad de carga ya que no

cargara de golpe todos los comentarios, sino los que se vayan dando clic en desplegar.

Así mismo una de las palabras frecuentes buscadas según Google trends como se observa en el grafico es acordeón menú:

Figura 378. Gráfico de barras de Google trends sobre accordion menu



Otro del motivo el cual fue elegido el efecto para formar parte del framework XRL8 fue que el proyecto presente debe estar a la altura de los frameworks que están en la actualidad siendo usados en la industria del desarrollo web como son:

- **Bootstrap.** - El presente framework cuenta con la clase `.accordion` para poder crear menú con estilo de un acordeón, si desean saber más sobre ello puede visitar la web oficial de documentación de dicho framework mediante el siguiente enlace:

<https://getbootstrap.com/docs/5.0/components/accordion/>

- **Bulma.**- Este framework también cuenta con una clase para crear un menú acordeón aunque como mencionan en la página de su documentación es una extensión del framework y esta denominado con la clase `.accordions`, para ver cómo se emplea y formas de uso revisar el siguiente enlace que los llevara a las extensiones que posee dicho framework:

<https://bulma.io/extensions/>

- **TailwindCSS.**- Quisa sea este el framework de css más usado en los últimos 3 años sobre pasando a los demás por sus formas de integración con NodeJS y react. Este framework dentro de sus múltiples componentes tiene un componente llamado accordion el cual se arma colocando muchas clases juntas en diversos niveles del HTML interno haciendo un poco engorroso la comprensión, principalmente este framework se diferencia de los demás por no tener clases predefinidas para efectos sino todo lo contrario tiene un conjunto de clases en el cual el usuario puede elegir los efectos más simples para colocar a su diseño. Para revisar su documentación ingresar al siguiente enlace:

<https://tailwind-elements.com/docs/standard/components/accordion/>

- **Foundation.**- Uno de los primeros frameworks en salir al mercado sin duda será foundation aunque con el pasar de los años a perdido popularidad este framework también dentro de sus clases tiene una sección para hacer menús verticalmente en forma de acordeón usando las clases vertical, menú y accordion-menu. Por lo general se suele usar verticalmente en foros que contendrán comentarios de los usuarios. Para ver el uso de este se puede recurrir al siguiente link:

<https://get.foundation/sites/docs/accordion-menu.html>

- **Materialize.**- Otro framework del medio es materialize el cual también cuenta con una sección llamada collapsible el cual ayuda a construir un menú acordeón deslizable. Lo podremos observar más detalladamente accediendo mediante el siguiente link:

<https://materializecss.com/collapsible.html>

- **UIKIT.**- También este framework cuenta con una clase para hacer un menú acordeon, la clase empleada en este caso tiene por nombre .uk-accordion para ver más sobre este framework recomiendo revisar la documentación en el siguiente enlace:

<https://getuikit.com/docs/accordion>

- **Semantic UI.**- A continuación tenemos otro framework de CSS muy completo la verdad el cual tiene soporte y componentes para ser usado con React, las clases las cuales ayudan a crear un menú acordeón son `ui,styled` y `accordion`. Podemos ver una gran cantidad de ejemplos mediante el siguiente enlace:

<https://semantic-ui.com/modules/accordion.html>

Modulo: Un módulo según definición propia es una sección de trabajo principal a comparación de un submódulo este puede ser denominado la estructura padre y los submódulos estructuras hijas. En este efecto en particular el módulo vendría a ser la siguiente clase:

- **Menú acordeón.** - Para este efecto nos basamos en la comodidad de los usuarios en usar un menú vertical en vez de uno horizontal, está conformado por tres clases las cuales unidas en cascada hacen que funcione correctamente el efecto, también se usó `inputs` tipo `checkbox` para poder hacer darle funcionalidad ya que XRL8 es un framework construido exclusivamente en CSS.

Submódulos: Considere llamarle submódulo, a los efectos o clases que están dentro de un módulo en este caso el módulo es menú acordeón, si tendríamos un efecto similar con atributos diferentes en ese caso sería un submódulo, como también podría ser un submódulo clases o elementos que hereden del módulo atributos.

Estos submódulos son muy útiles ya que desde mi punto de vista ayuda a diversificar los efectos y las clases. Beneficiando al usuario final en la construcción de sitios web. Actualmente el framework XRL8 no tiene pensado sacar sub módulos para el efecto menú acordeón.

FASE 2.- Codificar el efecto

Antes de ver esta sección a fondo quisiera hacer énfasis que este efecto en particular no fue de mucha dificultad en cuando a la escritura de código.

Bien luego de tener claro que efecto queríamos construir y porque lo íbamos a construir como se vio en el capítulo anterior se procedió a escribir código como se muestra a continuación.

Código completo

❖ Código del módulo acordeón

[parte 01 – menú acordeón]

Figura 379. Código de menu acordeon parte I

```
1  .acordeon input {
2    position: absolute;
3    opacity: 0;
4    z-index: -1;
5  }
6
7  .contenedor_acordeon {
8    border-radius: 8px;
9    overflow: hidden;
10   box-shadow: 0 4px 4px -2px rgba(0, 0, 0, 0.5);
11  }
12
13  .acordeon {
14    width: 100%;
15    color: white;
16    overflow: hidden;
17  }
18
19  .acordeon_opcion {
20    display: -webkit-box;
21    display: flex;
22    justify-content: space-between;
23    -webkit-box-pack: justify;
24    padding: 1em;
25    background: #2c3e50;
26    font-weight: bold;
27    cursor: pointer;
28  }
```

[parte 02 – menú acordeón]

Figura 380. Código de menu acordeon parte II

```
1  .acordeon_opcion:hover {
2    background: #1a252f;
3  }
4
5  .acordeon_opcion::after {
6    content: ">";
7    width: 1em;
8    height: 1em;
9    text-align: center;
10   -webkit-transition: all 0.35s;
11   transition: all 0.35s;
12 }
13
14 .acordeon_contenido {
15   max-height: 0;
16   padding: 0 1em;
17   color: #2c3e50;
18   background: white;
19   -webkit-transition: all 0.35s;
20   transition: all 0.35s;
21   display: -webkit-box;
22   display: flex;
23   justify-content: space-between;
24   -webkit-box-pack: justify;
25 }
```

[parte 03 – menú acordeón]

Figura 381. Código de menú acordeón parte III

```
1  input:checked + .acordeon_opcion {
2    background: #1a252f;
3  }
4
5  input:checked + .acordeon_opcion::after {
6    -webkit-transform: rotate(90deg);
7    transform: rotate(90deg);
8  }
9
10 input:checked ~ .acordeon_contenido {
11   max-height: 100vh;
12   padding: 1em;
13 }
```

Descripción de las propiedades más significativas:

A continuación, se procederá a detallar las propiedades más usadas y su aplicación. También se mostrará la compatibilidad web de cada propiedad para su mejor comprensión dejamos una leyenda la cual se puede observar en la figura 2.

Propiedad width:100% :

Se empleo width 100% para el ancho de este efecto ya que si bien esta propiedad tiene diversas unidades de medidas como:

- EM
- VH
- PIXELES
- REM
- PORCENTAJE

Se fue descartando una a una al momento de probarla, ya que al usar la unidad EM este se distorsionaba por su propia naturaleza, básicamente esta propiedad usa el ancho de la pantalla del dispositivo donde estemos y las va ajustando así que fue descartado.

A continuación, veamos la compatibilidad web al usar esta propiedad en la figura 3.

Propiedad height:1em:

Luego de ver las unidades de medida en la propiedad anterior se decidió trabar con em ya que al ser un menú desplegable verticalmente se tendrá que expandir la altura por ello la mejor opción es que este agarre la altura del dispositivo para una mayor expansión de la misma.

A continuación, veamos la compatibilidad web al usar esta propiedad en la figura 4.

Propiedad display: flex

Con esta propiedad se puede aplicar la alineación de los elementos en una sola dirección sea horizontal o vertical, estos valores se puede conjugar con los distintos valores que tiene, se optó por esta propiedad ya que es una forma óptima de maquetar y de que los elementos no se desborden de sus contenedores como solía suceder al maquetar usando position o tables en la antigüedad además flexbox viene para usarlo con la versión 3 de css llamada css3 haciendo más robusta la maquetación de elementos.

A continuación, veamos la compatibilidad web al usar esta propiedad en la figura 43.

Propiedad justify-content: space-between

La propiedad justify-content tiene como prerrequisitos haber declarado display:flex en el mismo elemento o en un elemento padre. Tiene las siguientes conjugaciones:

- CENTER
- END
- FLEX-END
- FLEX-START
- INHERIT
- INITIAL
- LEFT
- NORMAL
- REVERT
- RIGHT
- SPACE-AROUND
- SPACE-BETWEEN
- SPACE-EVENLY
- START
- STRETCH
- UNSET

Se opto por elegir el valor space-between porque este brinda una distribución uniforme entre los elementos que se encuentre dentro de este haciendo que no sobre ni falte ningún espacio es decir es como si usaría el 100% del espacio, pero también genera un margen entre los elementos que se encuentra menores a 30% de la capacidad máxima del contenedor si es mayor solo la propiedad generara bordes alrededor de sí mismo similar a un borde exterior.

A continuación, veamos la compatibilidad web al usar esta propiedad en la figura 45.

Propiedad transform:rotate(90deg)

La función que cumple es rotar la etiqueta sobre donde se aplicó la propiedad, esta se aplicara en el eje X como en el de Y, el prefijo deg hace referencia a los grados de inclinación en esta propiedad en particular es 90 lo que quiere decir que se mantendrá en una posición con inclinación 90 grados dicho de otra forma mantendrá medio giro. A continuación, veamos la compatibilidad web al usar esta propiedad en la figura 101.

Propiedad transition:all 0.35s

La propiedad transition tiene muchas conjugaciones en este caso en particular lo que indica es el tiempo de velocidad al cambiar de una propiedad de css a otra. Dichos valores están expresados en segundos y se le agrega

el prefijo `s` para indicarlo. A continuación, veamos la compatibilidad web al usar esta propiedad en la figura 47.

Propiedad `color:white`

Como `color` por default se eligió el blanco ya que es combinaría con cualquier tonalidad sea oscura o clara salvo con el mismo color blanco claro está, además de esta puede ser modificada mediante otra clase que le puede agregar el usuario o el desarrollador web.

A continuación, veamos la compatibilidad web al usar esta propiedad en la figura 21.

Propiedad `border-radius:8px`

Esta propiedad lo que hace es darle milésimas de bordes redondeados, esta propiedad nos brinda la ventaja de no depender de la resolución de pantalla del usuario siempre estará fija.

A continuación, veamos la compatibilidad web al usar esta propiedad en la figura 39.

Propiedad `content:">"`

Para usar esta propiedad es requisito primordial usar un pseudo-elemento como:

- **Before:** Lo que hace esta propiedad es insertar elementos antes de la clase.
- **After:** Lo que hace esta propiedad es insertar elementos después de la clase.

Estos pseudo-elementos son usados para añadir contenido a un elemento con la propiedad `content`, aunque también en `content` puede ir palabras y estas se insertaran. Normalmente va entre comilla que indica que estaría vacía en este caso se le inserto la flecha derecha esto indica que se insertara ese elemento, esta decisión se tomó para poder representar gráficamente el deslizar de nuestro menú acordeón.

A continuación, veamos la compatibilidad web al usar esta propiedad en la figura 144.

Propiedad overflow:hidden

Esta propiedad lo que hace es ocultar los elementos que salgan de su contenedor mostrando solo una parte del contenido que encaje exacto en dicho contenedor al mismo tiempo lo que hace es eliminar el scroll horizontalmente.

A continuación, veamos la compatibilidad web al usar esta propiedad en la figura 181.

Propiedad max-height:0:

Esta propiedad tiene la particularidad de que como máximo llegara al 0 de su capacidad a diferencia de la propiedad height, que solo es un tipo de alto específico mientras max-height impide que el alto en height se más largo que max-height, dicho de una forma más sencilla max-height siempre será el valor más alto tanto que sobrescribirá al valor de height si este es mayor al de max-height. Recapitulando es como usar dos valores en un rango siendo "a" el punto inicial denominado height y "b" siendo el rango más alto denominado max-height, su uso es principalmente para no tener que chancar o reemplazar valores, en la clase donde se aplicó se requería que el menú acordeón este sin altura para que dé la impresión que desapareció, luego al dar clic sobre el elemento correspondiente este desplegara tomando inmediatamente el valor de max-height.

A continuación, veamos la compatibilidad web al usar esta propiedad en la figura 270.

Principales dificultades:

Para este efecto en concreto la principal dificultad fue organizar de distintas formas posibles, así como ver las posibles conjugaciones que pueden hacer con el efecto para que el usuario al final no tenga problemas al implementarlo. Esas dificultades surgieron a raíz de las siguientes preguntas:

- ¿Cómo hacemos para insertar un icono el cual cambie al desplegar el menú acordeón?

Al inicio esta pregunta resulto ser incomoda ya que había demasiadas formas de hacerla la más fácil era usar font-awesome, la más difícil era

maquetar un icono especial y darle funcionalidad, es por ello que con el pasar del tiempo se decidió insertarlo mediante la propiedad content y darle funcionalidad con un input checkox.

¿Qué usamos para hacer las pruebas?

Se uso el navegador Google Chrome y opera mini, así como la siguiente página web que es un simulador online:

<http://www.responsinator.com>

Esto con motivos de asegurarnos que funcione en dispositivos móviles y tabletas. Ya que en base a lo trabajado no es del todo fiable usar el navegador para verificar la correcta maquetación de los elementos, así como tampoco es fiable depender de un software web, por ello se decidió diversificar en cuanto a las herramientas para probar las clases y efectos trabajados en el framework XRL8.

También usamos como editor de código visual studio code y su extensión live server para visualizar en vivo los efectos realizados en el css en tiempo real, también usamos el autoguardado que trae dicho editor de código para no estar guardando el proyecto a cada rato. Así conseguimos un mejor desempeño y evitamos el tedioso guardar cambios a cada rato, también evitamos el que la memoria cache se llene mostrándonos los mismos resultados.

FASE 3.- Hacerlo responsive

Se uso las siguientes técnicas para hacerlo responsive:

- Usamos la propiedad width en porcentaje para evitar desbordamientos en la maquetación.
- Se uso la propiedad flex-box y sus posibles conjugaciones con cada uno de sus valores según sea el caso.
- También usamos las unidades de medida vh o rem para texto y contenedores ya que estas son imprescindibles a la hora de hacer responsive un sitio web.

FASE 4.- Testing asegurarse que no tenga conflicto con otros efectos.

En esta parte del desarrollo de la clase se dispuso a probar poniéndolo con las demás clases juntándolas. Para ello lo que se hizo fue colocar dentro de un archivo index.html todas las clases trabajadas y ver como el efecto menú acordeón respondía. En algunos casos había conflicto con los demás efectos para corregir esto se procedió a implementar en el desarrollo dos cosas básicas la primera fue:

- Para detectar que clase era la que causaba el conflicto y ver en que archivo scss estaba. Se procedía a inspeccionar la página para detectar donde estaba el problema. En muchas ocasiones no se daba con el archivo causante del error entonces en ese caso aplicábamos la frase “Divide y vencerás”. Lo que se procedió hacer fue comentar de uno en uno cada efecto en el archivo principal que los llamaba esto reducía las líneas de código de css, además como teníamos en tiempo real los cambios fue mucho más rápido detectar el problema. Ya que no se tenía que recargar a cada rato además otro punto a favor era que se auto compilaba los cambios de sass automáticamente gracias a nodeJS el cual mediante la consola de comandos de Windows cuando detectaba un cambio en las líneas de código se auto compilaba y generando o reemplazando el archivo css en el cual hacíamos las pruebas, esto aumento la productividad para crear los efectos ya que al comentar de uno en uno cada archivo sass llegaba un momento donde desaparecía el error entonces revisábamos y nos dimos cuenta de que era uno o en algunos casos hasta tres efectos que hacían conflicto con la clase escrita.
- Se procedió a escribir mediante comentarios en un archivo llamado `_error_logs.scss` la clase que hace conflicto con la clase escrita, además de describir una nota breve en caso no se disponía de tiempo para solucionarlo luego, esto permitió tener un reporte más preciso de los incidentes que íbamos detectando.
- En el desarrollo había conflicto entre el contenido del acordeón con el icono de deslizamiento maquetado exclusivamente para desplegar un menú, ya que este tenía posición absoluta y el contenido del acordeón tenía posicionamiento relativo.

- Hubo conflicto al usar flex-box en el icono que se maquetó con el contenido del acordeón ya que el icono se desbordaba constantemente y ocupaba espacio entre el contenido, impidiendo el correcto desplazamiento del contenido en el menú acordeón.

FASE 5.- Aplicar la metodología OOCSS, llevándolo a SASS

- **Código css.** - Esta parte ya estaría completa por lo descrito anteriormente.

- **Código repetido.** - Identificación de código repetido

En este pequeño apartado luego de escribir y debugear el error se procedió a identificar el código css repetido para llevarlo a un archivo SCSS en sass, para su mayor optimización y aplicación de la metodología. Así que se procedió a reciclar el siguiente código:

Figura 382. Código repetido del componente menú acordeón

```
1  .acordeon_contenido {
2    max-height: 0;
3    padding: 0 1em;
4    color: #2c3e50;
5    background: white;
6    -webkit-transition: all 0.35s;
7    transition: all 0.35s;
8    display: -webkit-box;
9    display: flex;
10   justify-content: space-between;
11   -webkit-box-pack: justify;
12 }
13
14 .acordeon_opcion::after {
15   content: ">";
16   width: 1em;
17   height: 1em;
18   text-align: center;
19   -webkit-transition: all 0.35s;
20   transition: all 0.35s;
21 }
22
23 .acordeon_opcion {
24   display: -webkit-box;
25   display: flex;
26   justify-content: space-between;
27   -webkit-box-pack: justify;
28   padding: 1em;
29   background: #2c3e50;
30   font-weight: bold;
31   cursor: pointer;
32 }
```

- **Crear un objeto.** - Se describe la lógica de la función para luego crear un `mixing`.

Una vez identificado el código `css` repetido se procede hacer una función la cual se ejecuta al llamado del código, mediante esta técnica nuestro código será más puro y limpio.

Paralelamente en esta parte se determina la lógica de las funciones y ciclos como lo son `for`, `while`, `each`. También determinamos el pase de parámetros, así como la condicional que llevaran dentro, por otra parte, y no menos importante se define el nombre con el cual se llamara a las funciones.

```
function menu_acordeon(){
  display:-webkit-box;
  display:flex;
  justify-content:space-between;
  -webkit-box-pack:justify;
}
```

```
function menu_acordeon_tiempo_animacion(){
  -webkit-transition:all 0.35s;
  transition:all 0.35s
}
```

Luego de tener la lógica se procede a llevarlo a `SASS` para ello tenemos dos opciones, la primera es usar `@function` ya que como su nombre ase referencia es para crear una función la otra opción es usar `@mixin`, pero ambas tienen diferencias. La primera retorna un valor, mientras `@mixin` solo llama a los valores en este caso no necesitamos que retorne un valor por ello usaremos `@mixin`. Entonces quedaría de la siguiente forma:

Figura 383. Creación del objeto `menú_acordeon` en función `mixin`

```
1  @mixin menu_acordeon{
2    display: -webkit-box;
3    display: flex;
4    justify-content: space-between;
5    -webkit-box-pack: justify;
6  }
7
8  @mixin menu_acordeon_tiempo_animacion{
9    -webkit-transition: all $menu_acordeon_duracion_animacion;
10   transition: all $menu_acordeon_duracion_animacion;
11 }
```

- **OOCSS**. – Luego de tener la función o la lógica para reciclar el código se procede a integrar la metodología OOCSS para ello debemos llamar al mixin para hacerlo solo necesitamos agregar @include “nombre del mixin”. Luego de integrarlo ya estará aplicada correctamente la metodología OOCSS

Figura 384. Metodología OOCSS en el componente menú acordeon

```
1  .acordeon_opcion {
2    @include menu_acordeon;
3    padding: 1em;
4    background: #2c3e50;
5    font-weight: bold;
6    cursor: pointer;
7  }
8
9  .acordeon_opcion::after {
10   content: "\276F";
11   width: 1em;
12   height: 1em;
13   text-align: center;
14   @include menu_acordeon_tiempo_animacion;
15 }
16
17 .acordeon_contenido {
18   max-height: 0;
19   padding: 0 1em;
20   color: #2c3e50;
21   background: white;
22   @include menu_acordeon_tiempo_animacion;
23   @include menu_acordeon;
24 }
```

Bien volviendo al tema de las variables estas variables se encuentran en un archivo sass denominado variables valga la redundancia donde se encuentran las siguientes variables:

Figura 385. Variables sass del componente menú acordeón

```
$menu_acordeon_duracion_animacion:.35s;
$menu_acordeon_fondo_deslizamiento:white;
$menu_acordeon_fondo_estatico:#2c3e50;
$menu_acordeon_color_premisa_activo:#1a252f;
$menu_acordeon_color_hover:#1a252f;
```

Como se ha visto SASS es un preprocesador de css por ello se ejecuta antes del css tradicional esto permite añadir variables así como agregar condicionales, etc. Con la aplicación de la metodología como se observa se reutiliza el código css, además de que se hace escalable el proyecto. Ya que,

si en un futuro queremos cambiar o añadir más propiedades css, solo cambiaríamos las variables o añadiríamos más propiedades en el mixin.

FASE 6.- Documentar en la web site www.phooldx.com

Finalmente, si el efecto paso por todos los de más filtros se procede a su documentación respectiva, para documentar todos los efectos se creó una página web de documentación usando clases y elementos del mismo framework. A continuación, se describirá como se procede a la documentación de la misma. Siguiendo estos pasos:

- Se copia las etiquetas de HTML5 junto con la clase a un txt temporal para no perder la sintaxis
- Se guarda los cambios en el compilador de SASS
- Como es la parte final se procedió hacer copias de seguridad para no perder el avance, en archivos RAR. Estos archivos estaban por versiones enumerados además se agregaba un pequeño texto describiendo el avance. Este guardado se hacía de manera local en un USB.
- Para una mayor seguridad del avance del código se procedió luego a copiar la versión del archivo RAR junto a la descripción del mismo.

Guardado de código en GitHub. - Luego de tener la copia del proyecto a nivel físico en un USB, se procedió a guardarlo en un repositorio de GitHub para lo cual se usó Git, para ello se procedió de la siguiente forma:

- Se guardaba todo el proyecto con el siguiente comando que se aprecia en la figura 12.
- Luego se verificaba los cambios guardados con el siguiente comando que se aprecia en la figura
- Después se hacía el commit, es decir se les asignaba un nombre a los cambios realizados con el siguiente comando que se aprecia en la figura 13.
- Luego se procede a empaquetar los datos para que se suban al repositorio de GitHub con el siguiente código que se aprecia en la figura 14.

- Finalmente, luego de haber enviados el proyecto empaquetado hacia GitHub como se puede apreciar en la imagen, este queda guardado en dicho repositorio.

Figura 386 Visualización en github del componente menú acordeón



Se guarda en una escala de tiempo, con todos los cambios hechos desde la creación del proyecto, esto ayuda a tener un mejor control de versiones. Asimismo, también ayuda a que más personas se sumen al proyecto y puedan clonar el proyecto como trabajar por separado el proyecto, entre otras muchas ventajas.

- Así mismo se manejó un archivo llamado bitácora.txt donde aquí se registró todos los incidentes ocurridos, así como los cambios hechos, se registraba también las metas cumplidas, como las tareas nuevas para desarrollar al día siguiente. Al finalizar con un efecto por completo se quitaba de la lista de prioridades así dábamos pase al siguiente efecto o problema a resolver. Esto servía para tener un mejor orden y saber ejecutar prioridades.
- Luego de tener guardado el avance y de registrar el avanzase, se procedió a copiar del archivo documentación Etiquetas.txt el fragmento de código que se copió anteriormente para luego pegarlo en www.nosetup.org esta página web se encargaba de convertir html en texto plano para poder visualizar html
- Antes de finalizar se procedió a crear un archivo llamado modulo_menu_acordeon.html donde se dividía en tres partes:
 - En la primera se describía un poco al efecto que era y que hacia
 - En la segunda parte se procedía a copiar el código ya formateado obtenido de www.nosetup.com para luego pegarlo.

- En la tercera parte lo que se hacía era mostrar un apartado llamado estructura donde se describía como se llamaban las variables para que sea modificado en SASS.
- Finalmente, el ejemplo listo para usarse queda de la siguiente forma:

Figura 387 Código html5 final del componente menú acordeón

```

<div class="contenedor_acordeon">
  <div class="acordeon">
    <input type="checkbox" id="chck1">
    <label class="acordeon_opcion" for="chck1">Que es XRL8?</label>
    <div class="acordeon_contenido">
      XRL8 es un framework hecho en css, lo cual es super veloz,cuenta con efectos clasicos y con efectos nuevos ideal para desarrolladores
    </div>
  </div>

  <div class="acordeon">
    <input type="checkbox" id="chck2">
    <label class="acordeon_opcion" for="chck2">Lo que ofrecemos?</label>
    <div class="acordeon_contenido">
      XRL8 proporciona una excelente colección de HTML5, CSS3, Javascript, JQuery y muchas más creaciones para ayudar a los desarrolladores
    </div>
  </div>

  <div class="acordeon">
    <input type="checkbox" id="chck3">
    <label class="acordeon_opcion" for="chck3">Puedo unirme?</label>
    <div class="acordeon_contenido">
      Siempre estamos buscando personas con mentes creativas y talentosas para unirse a nosotros. Queremos que la gente escriba para nosotros
    </div>
  </div>
</div><!-- .contenedor_acordeon-->

```

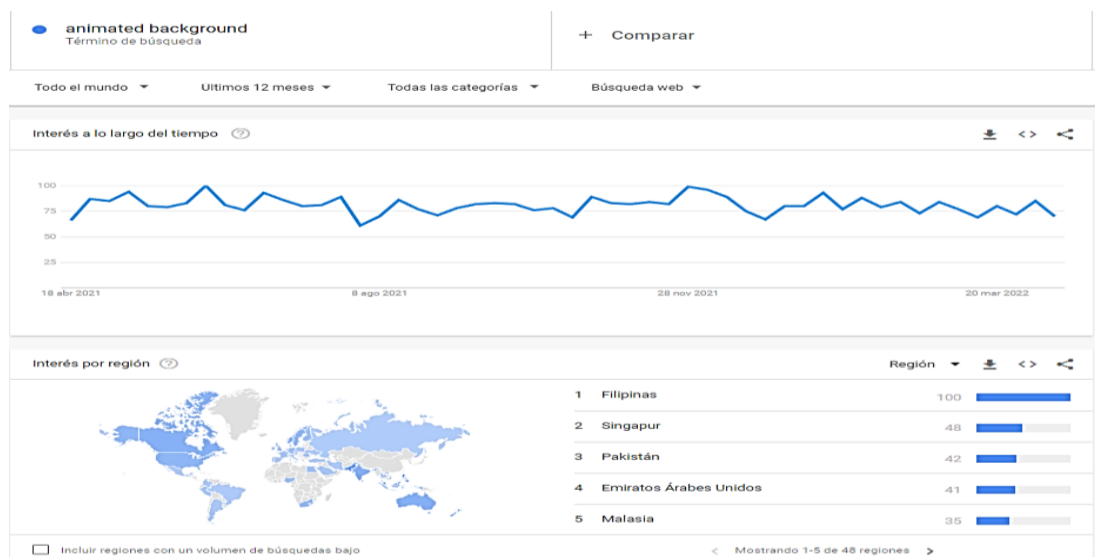
IMAGEN EN MOVIMIENTO

FASE 1.- Investigación del efecto y su importancia

Como primer paso lo primero que se iso fue investigar e indagar sobre el efecto, imagen en movimiento es un efecto creado exclusivamente con ese nombre para XRL8 ya que no existe algo similar o parecido al menos con el mismo nombre o sinónimo cabe recalcar que con esto me refiero solo al nombre del efecto porque en si el efecto ya existía hace tiempo pero no tiene un nombre en espesifico,el efecto sirve para hacer más llamativa un sitio web ya que al ser un efecto no muy usado suele ser algo exótico, así mismo no requiere convertir la imagen a jpg o tener un video clip para mostrarlo, cabe aclara que el usar clip de video en una página web prolonga la carga del sitio siendo el más ligero los textos y las imágenes. Así que uno de sus beneficios es que el efecto en si es una alternativa para agregar contenido multimedia ligero en vez de video para así poder obtener el mejor rendimiento en nuestro sitio web.

Una de las palabras frecuentes buscadas según Google trends como se observa el grafico es como animar un background el cual eventualmente lleva a crear una imagen en movimiento:

Figura 388. Gráfico de barras de Google trends sobre animated background



Otro del motivo el cual fue elegido el efecto para formar parte del framework XRL8 fue que el proyecto presente debe estar a la altura de las librerías que están en la actualidad siendo usados en la industria del desarrollo web como son:

- **Vanta.js.** – La presente librería nos presenta imagen en movimiento como si se tratase de imágenes gif estas están basadas en el lenguaje de programación JavaScript que junto a css llegan a formar estos increíbles efectos, si desean saber más sobre ello puede visitar la web oficial de la librería mediante el siguiente enlace:

<https://www.vantajs.com/?effect=rings>

- **Starback.JS.** - La presente librería cuenta con la particularidad de hacer caer meteoritos con movimiento en un fondo oscuro infinitamente, para ver cómo se emplea y formas de uso revisar el siguiente enlace que los llevara a la documentación oficial de dicha librería:

<https://www.cssscript.com/demo/star-falling-background/>

- **Slizer.js.** – Esta librería es una de las pocas en este rubro, su funcionalidad es hacer de una imagen un recorrida de canto a canto en diferentes ángulos, para que el resultado sea posible poder visualizarlo como si fuera una imagen en formato gif. Para revisar su documentación ingresar al siguiente enlace:

<https://www.cssscript.com/demo/auto-moving-backgrounds-slizer/>

Modulo: Un módulo según definición propia es una sección de trabajo principal a comparación de un submódulo este puede ser denominado la estructura padre y los submódulos estructuras hijas. En este efecto en particular el módulo vendría a ser la siguiente clase:

- **Imagen en movimiento.** - Para este efecto nos basamos en imágenes con una resolución en pixeles mínimo de 200KB para garantizar una adecuada visualización, este efecto funciona excelente en cualquier contenedor o espacio que tengamos dentro de nuestra maquetación por sus propiedades responsive con las que se construyó.

Submódulos: Considere llamarle submódulo, a los efectos o clases que están dentro de un módulo en este caso el módulo es imagen en movimiento, si tendríamos un efecto similar con atributos diferentes en ese caso sería un submódulo, como también podría ser un submódulo clases o elementos que hereden del módulo atributos.

Estos submódulos son muy útiles ya que desde mi punto de vista ayuda a diversificar los efectos y las clases. Beneficiando al usuario final en la construcción de sitios web. Actualmente el framework XRL8 no tiene pensado sacar sub módulos para el efecto imagen en movimiento.

FASE 2.- Codificar el efecto

Antes de ver esta sección a fondo, quería comentar que la codificación fue sencilla fue una de los pocos efectos desarrollados en XRL8 los cuales fueron sencillos y que al mismo tiempo se cumplió con las expectativas tenidas.

Bien luego de tener claro que efecto queríamos construir y porque lo íbamos a construir como se vio en el capítulo anterior se procedió a escribir código como se muestra a continuación.

Código completo

Figura 389. Código de la imagen en movimiento

```
1  .movimiento_1 {
2    width: 100%;
3    height: 100%;
4    margin: 0 auto;
5    padding: 0;
6    box-sizing: border-box;
7    z-index: 1;
8    background-image: url("../multimedia/moverse1.jpg");
9  }
10
11 .movimiento_2 {
12   width: 100%;
13   height: 100%;
14   margin: 0 auto;
15   padding: 0;
16   box-sizing: border-box;
17   z-index: 1;
18   background-image: url("../multimedia/moverse2.jpg");
19 }
20
21 .movimiento_3 {
22   width: 100%;
23   height: 100%;
24   margin: 0 auto;
25   padding: 0;
26   box-sizing: border-box;
27   z-index: 1;
28   background-image: url("../multimedia/moverse3.jpg");
29 }
30
31 .contenedor_img_movi {
32   border: solid 6px gold;
33   width: 100%;
34   height: 400px;
35   background-size: 200%;
36   animation: movimiento 20s infinite linear alternate;
37 }
38
39 @keyframes movimiento {
40   from {
41     background-position: bottom left;
42   }
43   to {
44     background-position: top right;
45   }
46 }
```

Descripción de las propiedades más significativas:

A continuación, se procederá a detallar las propiedades más usadas y su aplicación. También se mostrará la compatibilidad web de cada propiedad para su mejor comprensión dejamos una leyenda la cual se puede observar en la figura 2.

Propiedad width:100% :

Se empleo width 100% para el ancho de este efecto ya que si bien esta propiedad tiene diversas unidades de medidas como:

- EM
- VH
- PIXELES
- REM
- PORCENTAJE

Se fue descartando una a una al momento de probarla, ya que la al usar la unidad EM este se distorsionaba por su propia naturaleza, básicamente esta propiedad usa el ancho de la pantalla del dispositivo donde estemos y las va ajustando así que fue descartado.

A continuación, veamos la compatibilidad web al usar esta propiedad en la figura 3.

Propiedad height:100%:

Luego de ver las unidades de medida en la propiedad anterior se decidió trabar en porcentaje, ya que se necesitaba un valor dinámico poniendo que único límite sea las dimensiones del dispositivo del usuario, en ese sentido se descartó las propiedades que toman el ancho del dispositivo como son REM y EM por otra parte también se descartó a los que toman el alto del dispositivo como son VH. Luego de reducir nuestro marco de trabajo se tenía PORCENTAJE o PIXELES así que descartamos a PIXELES.

Se empleo PORCENTAJE por su propiedad para hacer responsive los elementos que se encuentren dentro del contenedor o sean este un elemento padre.

A continuación, veamos la compatibilidad web al usar esta propiedad en la figura 4.

Propiedad padding:0px

Por ser una imagen que debe moverse, este no debe de tener relleno interno como es la propiedad padding por ello se le especifica al navegador que no queremos espacios.

A continuación, veamos la compatibilidad web al usar esta propiedad en la figura 20.

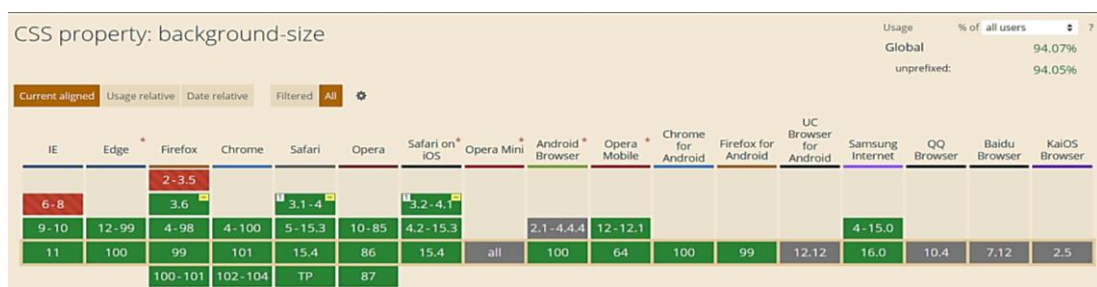
Propiedad margin:0 auto

Esta propiedad como esta en el contenedor de input clásico lo que se busco era centrar todo el contenido en el centro para ello la mejor opción era usar los valores 0 arriba y debajo luego automático centrar a la derecha como a la izquierda. Ambos valores para hacerlo de forma abreviada solo se dispusieron a poner 0 y auto. A continuación, veamos la compatibilidad web al usar esta propiedad en la figura 38.

Propiedad background-size:200%:

Esta propiedad tiene la particularidad de hacer zoom a las imágenes en este caso esta expresado en porcentaje.

Figura 390. Compatibilidad web propiedad background-size



Propiedad @keyframe movimiento

Con esta propiedad se pueden crear animaciones más sofisticadas ya que se pueden mezclar propiedades de css en su interior además de combinarlas con la propiedad animation y transform. Esta se declara usando la palabra reservada @keyframe seguido de un nombre en este caso el nombre que le asignamos es movimiento. La estructura es la siguiente:

- ❖ Usando la estructura from y to de la siguiente manera como se observa en la figura 183.

A continuación, veamos la compatibilidad web al usar esta propiedad en la figura 185.

Principales dificultades:

Para este efecto en concreto la principal dificultad fue organizar de distintas formas posibles, así como ver las posibles conjugaciones que pueden hacer con el efecto para que el usuario al final no tenga problemas al implementarlo. Esas dificultades surgieron a raíz de las siguientes preguntas:

- ¿Cómo hacemos zoom a la imagen para poder mostrarlo como si fuera un recorrido de un video o un gif?

Se desconocía la propiedad que hacia eso, así que se tuvo que indagar por un prologando tiempo, luego se encontró este fue la propiedad background-size

- ¿Cómo realizamos el movimiento de estas imágenes?

Se uso la propiedad @keyframes en el cual se especificó que empiece de abajo el zoom y que termine en la parte de arriba de la imagen en un lapso de tiempo indeterminado.

¿Qué usamos para hacer las pruebas?

Se uso el navegador Google Chrome y opera mini, así como la siguiente página web que es un simulador online:

<http://www.responsinator.com>

Esto con motivos de asegurarnos que funcione en dispositivos móviles y tabletas. Ya que en base a lo trabajado no es del todo fiable usar el navegador para verificar la correcta maquetación de los elementos, así como tampoco es fiable depender de un software web, por ello se decidió diversificar en cuanto a las herramientas para probar las clases y efectos trabajados en el framework XRL8.

También usamos como editor de código visual studio code y su extensión live server para visualizar en vivo los efectos realizados en el css en tiempo real, también usamos el autoguardado que trae dicho editor de código para no estar guardando el proyecto a cada rato. Así conseguimos un mejor desempeño y evitamos el tedioso guardar cambios a cada rato, también evitamos el que la memoria cache se llene mostrándonos los mismos resultados.

FASE 3.- Hacerlo responsive

En esta propiedad en específico solo se utilizó porcentaje para definir el ancho ya que al ser una imagen este adquiere el ancho y alto del contenedor padre. No hubo mucha maquetación de por medio como para usar otras propiedades responsive.

FASE 4.- Testing asegurarse que no tenga conflicto con otros efectos.

En esta parte del desarrollo de la clase se dispuso a probar poniéndolo con las demás clases juntándolas. Para ello lo que se hizo fue colocar dentro de un archivo index.html todas las clases trabajadas y ver como el efecto de imagen en movimiento respondía. En algunos casos había conflicto con los demás efectos para corregir esto se procedió a implementar en el desarrollo dos cosas básicas la primera fue:

- Para detectar que clase era la que causaba el conflicto y ver en que archivo scss estaba. Se procedía a inspeccionar la página para detectar donde estaba el problema. En muchas ocasiones no se daba con el archivo causante del error entonces en ese caso aplicábamos la frase "Divide y vencerás". Lo que se procedió hacer fue comentar de uno en uno cada efecto en el archivo principal que los llamaba esto reducía las líneas de código de css, además como teníamos en tiempo real los cambios fue mucho más rápido detectar el problema. Ya que no se tenía que recargar a cada rato además otro punto a favor era que se auto compilaba los cambios de sass automáticamente gracias a nodeJS el cual mediante la consola de

comandos de Windows cuando detectaba un cambio en las líneas de código se auto compilaba y generando o reemplazando el archivo css en el cual hacíamos las pruebas, esto aumento la productividad para crear los efectos ya que al comentar de uno en uno cada archivo sass llegaba un momento donde desaparecía el error entonces revisábamos y nos dimos cuenta de que era uno o en algunos casos hasta tres efectos que hacían conflicto con la clase escrita.

- Se procedió a escribir mediante comentarios en un archivo llamado `_error_logs.scss` la clase que hace conflicto con la clase escrita, además de describir una nota breve en caso no se disponía de tiempo para solucionarlo luego, esto permitió tener un reporte más preciso de los incidentes que íbamos detectando.

FASE 5.- Aplicar la metodología OOCSS, llevándolo a SASS

- **Código css.** - Esta parte ya estaría completa por lo descrito anteriormente.
- **Código repetido.** - Identificación de código repetido

En este pequeño apartado luego de escribir y debugear el error se procedió a identificar el código css repetido para llevarlo a un archivo SCSS en sass, para su mayor optimización y aplicación de la metodología. Así que se procedió a reciclar el siguiente código:

Figura 391. Código repetido del componente imagen en movimiento parte I

```
1  .movimiento_1 {
2    width: 100%;
3    height: 100%;
4    margin: 0 auto;
5    padding: 0;
6    box-sizing: border-box;
7    z-index: 1;
8    background-image: url("../multimedia/moverse1.jpg");
9  }
10
11 .movimiento_2 {
12   width: 100%;
13   height: 100%;
14   margin: 0 auto;
15   padding: 0;
16   box-sizing: border-box;
17   z-index: 1;
18   background-image: url("../multimedia/moverse2.jpg");
19 }
```

Figura 392. Código repetido del componente imagen en movimiento parte II

```
1  .movimiento_3 {
2    width: 100%;
3    height: 100%;
4    margin: 0 auto;
5    padding: 0;
6    box-sizing: border-box;
7    z-index: 1;
8    background-image: url("../multimedia/moverse3.jpg");
9  }
10
11 .contenedor_img_movi {
12   border: solid 6px gold;
13   width: 100%;
14   height: 400px;
15   background-size: 200%;
16   animation: movimiento 20s infinite linear alternate;
17 }
```

- **Crear un objeto.** - Se describe la lógica de la función para luego crear un mixing.

Una vez identificado el código css repetido se procede hacer una función la cual se ejecuta al llamado del código, mediante esta técnica nuestro código será más puro y limpio.

Paralelamente en esta parte se determina la lógica de las funciones y ciclos como lo son for, while, each. También determinamos el pase de parámetros, así como la condicional que llevaran dentro, por otra parte, y no menos importante se define el nombre con el cual se llamara a las funciones.

```
function imagen_movimiento_base(){
width:100%;
height:100%;
margin:0 auto;
padding: 0;
box-sizing: border-box;
z-index:1;
}
```

Luego de tener la lógica se procede a llevarlo a SASS para ello tenemos dos opciones, la primera es usar @function ya que como su nombre ase referencia es para crear una función la otra opción es usar @mixin, pero ambas tienen diferencias. La primera retorna un valor, mientras @mixin solo llama a los valores en este caso no necesitamos que retorne un valor por ello usaremos @mixin. Entonces quedaría de la siguiente forma:

Figura 393. Creación del objeto imagen_movimiento_base en función mixin

```
1 @mixin imagen_movimiento_base{
2     width: 100%;
3     height: 100%;
4     margin:0 auto;
5     padding:0;
6     box-sizing: border-box;
7     z-index:1;
8 }
```

- **OOCSS.** – Luego de tener la función o la lógica para reciclar el código se procede a integrar la metodología OOCSS para ello debemos llamar al mixin para hacerlo solo necesitamos agregar @include “nombre del mixin”. Luego de integrarlo ya estará aplicada correctamente la metodología OOCSS.

Figura 394. Creación del objeto en un ciclo para la clase movimiento

```
1 @for $i from 1 through $num_img_movimiendose {
2     .movimiento_#{ $i }{
3         @include imagen_movimiento_base();
4         background-image:url("../multimedia/moverse#{ $i }.jpg");
5     }
6 }
```

Bien volviendo al tema de las variables estas variables se encuentran en un archivo sass denominado variables valga la redundancia donde se encuentran las siguientes variables:

Figura 395. Variables sass del componente imagen en movimiento

```
$imagen-movindose-ancho: 100%;
$imagen-movindose-alto: 400px;
$imagen-movindose-dimencion-fondo: 200%;
$num_img_movimiendose:3;
```

Como se ha visto SASS es un preprocesador de css por ello se ejecuta antes del css tradicional esto permite añadir variables así como agregar condicionales, etc. Con la aplicación de la metodología como se observa se reutiliza el código css, además de que se hace escalable el proyecto. Ya que, si en un futuro queremos cambiar o añadir más propiedades css, solo cambiaríamos las variables o añadiríamos más propiedades en el mixin.

FASE 6.- Documentar en la web site www.phooldx.com

Finalmente, si el efecto paso por todos los de más filtros se procede a su documentación respectiva, para documentar todos los efectos se creó una página web de documentación usando clases y elementos del mismo framework. A continuación, se describirá como se procede a la documentación de la misma. Siguiendo estos pasos:

- Se copia las etiquetas de HTML5 junto con la clase a un txt temporal para no perder la sintaxis
- Se guarda los cambios en el compilador de SASS
- Como es la parte final se procedió hacer copias de seguridad para no perder el avance, en archivos RAR. Estos archivos estaban por versiones enumerados además se agregaba un pequeño texto describiendo el avance. Este guardado se hacía de manera local en un USB.
- Para una mayor seguridad del avance del código se procedió luego a copiar la versión del archivo RAR junto a la descripción del mismo.

Guardado de código en GitHub. - Luego de tener la copia del proyecto a nivel físico en un USB, se procedió a guardarlo en un repositorio de GitHub para lo cual se usó Git, para ello se procedió de la siguiente forma:

- Se guardaba todo el proyecto con el siguiente comando que se observa en la figura 12.
- Luego se verificaba los cambios guardados con el siguiente comando que se observa en la figura 13.
- Después se hacía el commit, es decir se le asignaba un nombre a los cambios realizados con el siguiente comando que se observa en la figura 14.
- Luego se procede a empaquetar los datos para que se suban al repositorio de GitHub con el siguiente código que se observa en la figura 15.
- Finalmente, luego de haber enviados el proyecto empaquetado hacia GitHub como se puede apreciar en la imagen, este queda guardado en dicho repositorio.

Figura 396. Visualización en github del componente imagen en movimiento



Se guarda en una escala de tiempo, con todos los cambios hechos desde la creación del proyecto, esto ayuda a tener un mejor control de versiones. Asimismo, también ayuda a que más personas se sumen al proyecto y puedan clonar el proyecto como trabajar por separado el proyecto, entre otras muchas ventajas.

- Así mismo se manejó un archivo llamado bitácora.txt donde aquí se registró todos los incidentes ocurridos, así como los cambios hechos, se registraba también las metas cumplidas, como las tareas nuevas para desarrollar al día siguiente. Al finalizar con un efecto por completo se quitaba de la lista de prioridades así dábamos pase al siguiente efecto o problema a resolver. Esto servía para tener un mejor orden y saber ejecutar prioridades.
- Luego de tener guardado el avance y de registrar el avanzase, se procedió a copiar del archivo documentación Etiquetas.txt el fragmento de código que se copió anteriormente para luego pegarlo en www.nosetup.org esta página web se encargaba de convertir html en texto plano para poder visualizar html
- Antes de finalizar se procedió a crear un archivo llamado modulo_imagen_en_movimiento.html donde se dividía en tres partes:
 - En la primera se describía un poco al efecto que era y que hacia
 - En la segunda parte se procedía a copiar el código ya formateado obtenido de www.nosetup.com para luego pegarlo.
 - En la tercera parte lo que se hacía era mostrar un apartado llamado estructura donde se describía como se llamaban las variables para que sea modificado en SASS.
- Finalmente, el ejemplo listo para usarse queda de la siguiente forma:

Figura 397 Código html5 final del componente imagen en movimiento.

```
<div class="contenedor_img_movi movimiento_1">
</div>
```

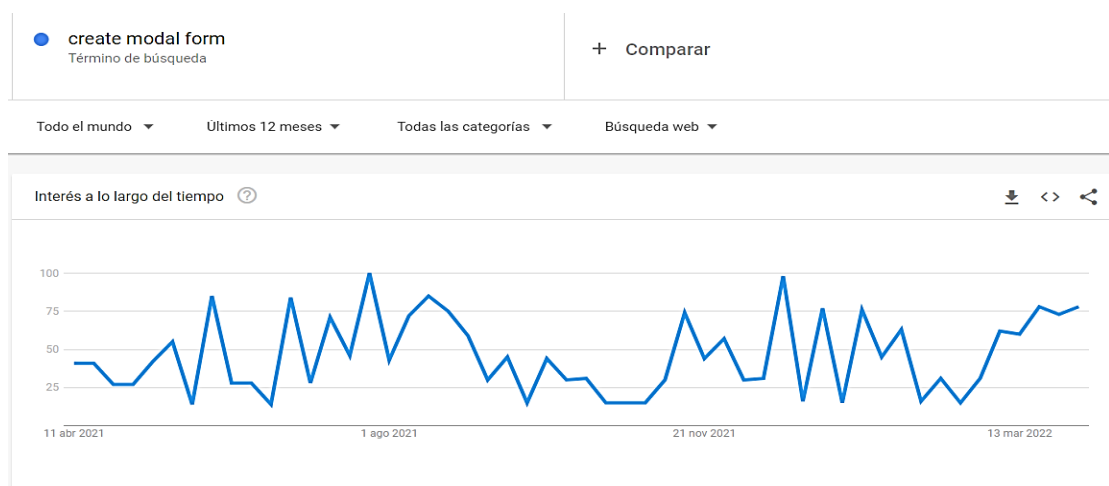
MODAL FORMULARIOS

FASE 1.- Investigación del efecto y su importancia

Como primer paso lo primero que se iso fue investigar e indagar sobre el efecto, una de las muchas cosas llamativas de este efecto es que son activables y desactivables lo que lo hace llamativo su uso, un modal es una ventana emergente en el cual sirve para mostrar alguna información relevante para el usuario como también para un usuario que es desarrollador web para el cual le será útil que el modal acepte introducir formularios, los formularios se caracterizan por poner contenido exclusivo ya que ocupan un espacio considerable, estos pueden venir por tamaños como L,M,S siendo L el tamaña más grande, algunos frameworks suele trabajar con una medida fija como se verá más adelante. Recapitulando tiene dos grandes ventajas la primera que son reutilizables ya que se pueden abrir y cerrar con un clic y la otra ventaja es que suele ocupar poco espacio por lo cual se suele agregar contenido exclusivo, otro factor es que ayuda al usuario a interactuar de manera dinámica con el sitio web.

Así mismo una de las palabras más frecuentes buscadas según Google trends como se observa en el grafico es como crear un formulario modal:

Figura 398 Gráfico de barras de Google trends sobre modal formularios



Otro del motivo el cual fue elegido el efecto para formar parte del framework XRL8 fue que el proyecto presente debe estar a la altura de los frameworks que están en la actualidad siendo usados en la industria del desarrollo web como son:

- **Bootstrap.** - El presente framework cuenta con la clase `.modal` para poder crear ventanas modales emergentes además estos modales usan jquery para su correcto funcionamiento también cuenta con distintos tamaños, estos modales esta armados o construido con un contenedor llamado modal en la cual tiene tres secciones el header body y footer, si desean saber más sobre ello puede visitar la web oficial de documentación de dicho framework mediante el siguiente enlace:

<https://getbootstrap.com/docs/4.3/components/modal/>

- **Bulma.**- Este framework también cuenta con una clase reservada para ventanas modales denominado `.modal`, este framework clasifico su estructura modal en tres básicamente un contenedor denominado modal,un fondo denominado modal-background luego tenemos donde se colocara el contenido denominado modal-content y finalmente el botón el cual cierra el modal denominado modal-close, para ver cómo se emplea y formas de uso revisar el siguiente enlace que los llevara a la documentación oficial de dicho framework:

<https://bulma.io/documentation/components/modal/>

- **TailwindCSS.**- Quisa sea este el framework de css más usado en los últimos 3 años sobre pasando a los demás por sus formas de integración con NodeJS y react.Este framework dentro de su contenido mezcla con otras clases se puede construir una ventana modal emergente gracias a sus componentes, estos a diferencia de los demás se caracteriza por no tener clases predefinidas para efectos sino todo lo contrario tiene un conjunto de clases en el cual el usuario puede elegir los efectos más simples para colocar a su diseño. Para revisar su documentación ingresar al siguiente enlace:

<https://tailwindui.com/components/application-ui/overlays/modals>

- **Foundation.-** Uno de los primeros frameworks en salir al mercado sin duda será foundation aunque con el pasar de los años a perdido popularidad este framework también dentro de sus clases tiene una sección para hacer ventanas modales usando la clase `.reveal` esta es combinada con la clase `lead` la cual hace énfasis en las letras que se mostrara en el modal. Para ver el uso de este se puede recurrir al siguiente link:

<https://get.foundation/sites/docs/reveal.html>

- **Materialize.-** Otro framework del medio es materialize el cual también cuenta con una sección llamada modal el cual ayuda a construir modales emergentes la estructura se divide en tres partes la primera es el contenedor en el cual se aplica la clase `modal`, luego tenemos la segunda parte en la cual va el contenido con la clase `modal-content` por último la parte final es denominada `modal-footer` el cual es el pie de página. Lo podremos observar más detalladamente accediendo mediante el siguiente link:

<https://materializecss.com/modals.html>

- **UIKIT.-** También este framework cuenta con una clase para hacer ventanas modales la clase empleada en este caso tiene por nombre `.uk-modal` la cual combinada junto a otras clases se logra construir un modal, para ver más sobre este framework y sus usos recomiendo revisar la documentación en el siguiente enlace:

<https://getuikit.com/docs/modal>

- **Semantic UI.-** A continuación tenemos otro framework de css muy completo la verdad el cual tiene soporte y componentes para ser usado con react, la clase la cual ayuda a crear ventanas modales es denominada `.ui modal` esta es la clase principal que sirve como contenedor esta puede ser personalizada una vez construida asi mismo requiere de jquery para su correcto funcionamiento, podemos ver una gran cantidad de ejemplos mediante el siguiente enlace:

<https://semantic-ui.com/modules/modal.html>

Modulo: Un módulo según definición propia es una sección de trabajo principal a comparación de un submódulo este puede ser denominado la estructura padre y los submódulos estructuras hijas. En este efecto en particular el módulo vendría a ser la siguiente clase:

- **Modal formulario.** - Para este efecto nos basamos en tener un formulario dentro del modal como su propio nombre lo indica, estos formularios están elaborados con la redacción del documento input formulario propiedad de XRL8 donde se muestra los tipos de inputs que tenemos para formularios, así como los tamaños de estos y sus formas de uso. Todas las clases de formularios descritos en el documento mencionado están reestructuradas para funcionar con la clase modal formulario.

Submódulos: Considere llamarle submódulo, a los efectos o clases que están dentro de un módulo en este caso el módulo es modal formulario porque comparten atributos similares. Un submódulo según denominación propia vendría hacer un efecto similar al original, pero con características diferentes, hagamos un ejemplo para comprender esto de una manera mejor.

- Modal publicidad. - Este sería un sub modulo ya que tendría propiedades compartidas con el módulo original, pero discreparían en funcionalidades mientras el módulo original está enfocado a recibir datos en sus inputs como lo son formularios, este submódulo está orientado a ser usado para notificaciones, también puede ser empleado para alertas y por ultimo como su propio nombre lo indica publicidad en los sitios web ya que está configurado para aparecer de forma automática cada 4 segundos y este permanecerá abierto hasta que el usuario decida cerrar dando clic en la "X".

Estos submódulos son muy útiles ya que desde mi punto de vista ayuda a diversificar los efectos y las clases. Beneficiando al usuario final en la construcción de sitios web.

FASE 2.- Codificar el efecto

Antes de ver esta sección a fondo quisiera hacer énfasis en que la codificación no estuvo complicada, además se facilitó demasiado el haber tenido desarrollado previamente el módulo input formulario ya que solo se tuvo que adaptar algunos puntos para su correcto funcionamiento.

Bien luego de tener claro que efecto queríamos construir y porque lo íbamos a construir como se vio en el capítulo anterior se procedió a escribir código como se muestra a continuación.

Código completo

❖ Código del módulo modal formulario

[parte 01 – modal form]

Figura 399. Código de la modal formularios parte I

```
1  .habilitar_form_modal {
2    max-width:unset !important;
3    width:unset !important;
4    padding-left: 0 !important;
5    padding-right: 0 !important;
6    border: none !important;
7  }
8
9  /***** TIPOS DE MODAL *****/
10 .modal_peque {
11   width: 22.5rem !important;
12 }
13
14 .modal_grande {
15   width: 48rem !important;
16 }
17
18 div#contenedor_modal {
19   display: flex;
20   justify-content: center;
21   align-items: center;
22   margin: 0 auto;
23   padding: 0;
24   box-sizing: border-box;
25 }
```

[parte 02 – modal form]

Figura 400. Código de la modal formularios parte II

```
1  .modal:before {
2    content: "";
3    display: none;
4    background: rgba(0, 0, 0, 0.6);
5    position: fixed;
6    top: 0;
7    left: 0;
8    right: 0;
9    bottom: 0;
10   z-index: 10;
11   margin: 0 auto;
12   padding: 0;
13   box-sizing: border-box;
14 }
15
16 .modal:target:before {
17   display: block;
18   margin: 0 auto;
19   padding: 0;
20   box-sizing: border-box;
21 }
22
23 .modal:target .modal_form {
24   -webkit-transform: translate(0, 0);
25   -ms-transform: translate(0, 0);
26   transform: translate(0, 0);
27   top: 10vh;
28   margin: 0 auto;
29   padding: 0;
30   box-sizing: border-box;
31 }
```

[parte 03 – modal form]

Figura 401. Código de la modal formularios parte III

```
1  .modal_form {
2    width: 31rem;
3    background: #fefefe;
4    border: #333 solid 1px;
5    border-radius: 5px;
6    position: fixed;
7    top: -100%;
8    z-index: 11;
9    -webkit-transform: translate(0, -500%);
10   -ms-transform: translate(0, -500%);
11   transform: translate(0, -500%);
12   -webkit-transition: -webkit-transform 0.3s ease-out;
13   -moz-transition: -moz-transform 0.3s ease-out;
14   -o-transition: -o-transform 0.3s ease-out;
15   transition: transform 0.3s ease-out;
16   left: 0;
17   right: 0;
18   margin: 0 auto;
19   padding: 0;
20   box-sizing: border-box;
21 }
22
23 .modal_resistencia,
24 .modal_contenido {
25   padding: 10px 20px;
26 }
27
28 .modal_resistencia {
29   border-bottom: #eee solid 1px;
30 }
```

[parte 04 – modal form]

Figura 402. Código de la modal formularios parte IV

```
1  .modal_resistencia h2 {
2    font-size: 20px;
3  }
4
5  .modal_contenido {
6    border-top: #eee solid 1px;
7  }
8
9  /** LOS BOTONES EN 1 ****/
10 .btn_modal {
11   background: #428bca;
12   border: #357ebd solid 1px;
13   border-radius: 3px;
14   color: #fff;
15   display: inline-block;
16   font-size: 14px;
17   padding: 0.3rem 1rem !important;
18   text-decoration: none;
19   text-align: center;
20   position: relative;
21   transition: color 0.1s ease;
22 }
23
24 .btn_modal:hover {
25   background: #357ebd;
26 }
27
28 .modal_btn_salir_modal {
29   color: #aaa;
30   font-size: 30px;
31   text-decoration: none;
32   position: absolute;
33   right: 5px;
34   top: 0.7rem;
35 }
```

❖ Código del submódulo modal publicidad

[parte 01 – modal publicidad]

Figura 403. Código de la modal formularios parte V

```
1  .contenedor-modal-publi {
2    text-align: center;
3    font-family: Verdana, Geneva, sans-serif;
4    margin: 0 auto;
5    padding: 0;
6    box-sizing: border-box;
7  }
8
9  .modal-publicidad {
10   width: 100%;
11   height: 100vh;
12   background: rgba(0, 0, 0, 0.8);
13   position: absolute;
14   top: 0;
15   left: 0;
16   display: flex;
17   animation: modal 2s 3s forwards;
18   visibility: hidden;
19   opacity: 0;
20   margin: 0 auto;
21   padding: 0;
22   box-sizing: border-box;
23 }
```

[parte 02 – modal publicidad]

Figura 404. Código de la modal formularios parte VI

```
1  .contenido-publicidad {
2    margin: auto;
3    width: 50%;
4    height: 50%;
5    background: white;
6    border-radius: 10px;
7  }
8
9  #cerrar-publicidad {
10   display: none;
11 }
12
13 #cerrar-publicidad + label {
14   position: fixed;
15   color: white;
16   font-size: 25px;
17   z-index: 50;
18   background: darkred;
19   height: 40px;
20   width: 40px;
21   line-height: 40px;
22   border-radius: 50%;
23   right: 10px;
24   cursor: pointer;
25   animation: modal 2s 3s forwards;
26   visibility: hidden;
27   opacity: 0;
28   margin: 0 auto;
29   padding: 0;
30   box-sizing: border-box;
31 }
```

[parte 03 – modal publicidad]

Figura 405. Código de la modal formularios parte VII

```
1  #cerrar-publicidad:checked + label, #cerrar-publicidad:checked ~ .modal-publicidad {
2    display: none;
3  }
4
5  @keyframes modal {
6    100% {
7      visibility: visible;
8      opacity: 1;
9    }
10 }
```

Descripción de las propiedades más significativas:

A continuación, se procederá a detallar las propiedades más usadas y su aplicación. También se mostrará la compatibilidad web de cada propiedad para su mejor comprensión dejamos una leyenda la cual se puede observar en la figura 2.

Propiedad width: unset

Se empleo width unset para el ancho de este efecto ya que si bien esta propiedad tiene diversas unidades de medidas como:

- EM
- VH
- PIXELES
- REM
- PORCENTAJE

Se fue descartando una a una al momento de probarla, ya que la al usar la unidad con su valor unset lo que hace es que toma el valor de la clase padre, caso contrario toma a su valor inicial.

A continuación, veamos la compatibilidad web al usar esta propiedad en la figura 3.

Propiedad display: flex

Con esta propiedad se puede aplicar la alineación de los elementos en una sola dirección sea horizontal o vertical, estos valores se puede conjugar con las distintos valores que tiene, se optó por esta propiedad ya que es una forma óptima de maquetar y de que los elementos no se desborden de sus contenedores como solía suceder al maquetar usando position o tables en la antigüedad además flexbox viene para usarlo con la versión 3 de css llamada css3 haciendo más robusta la maquetación de elementos. A continuación, veamos la compatibilidad web al usar esta propiedad en la figura 43.

Propiedad align-items: center

Para usar esta propiedad es un requisito indispensable tener habilitado display: flex, luego de eso se puede conjugar align-items con los siguientes valores:

- CENTER
- END
- FLEX-END
- FLEX-START
- INHERIT
- INITIAL
- NORMAL
- REVERT
- SELF-END

- START
- STRETCH
- UNSET

En este caso se usó con el valor center ya que lo que ara es centrar en el centro de gravedad a los elementos, pero en dirección vertical.

A continuación, veamos la compatibilidad web al usar esta propiedad en la figura 44.

Propiedad justify-content: center

La propiedad justify-content tiene como prerequisites haber declarado display: flex en el mismo elemento o en un elemento padre. Tiene las siguientes conjugaciones:

- CENTER
- END
- FLEX-END
- FLEX-START
- INHERIT
- INITIAL
- LEFT
- NORMAL
- REVERT
- RIGHT
- SPACE-AROUND
- SPACE-BETWEEN
- SPACE-EVENLY
- START
- STRETCH
- UNSET

Se opto por elegir el valor center ya que este centra en horizontal cualquier elemento. A continuación, veamos la compatibilidad web al usar esta propiedad en la figura 45.

Propiedad border: solid 1px #333

En esta propiedad le damos un borde sólido, esto quiere decir que el borde será visible, además de que será una línea definida sin interrupciones. Luego la otra característica es que tendrá una anchura de 1px, el color que llevara dicha línea es plomo, dada por el numeral y expresada en hexadecimal. A continuación, veamos la compatibilidad web al usar esta propiedad en la figura 37.

Propiedad border-radius:5px

Esta propiedad lo que hace es darle milésimas de bordes redondeados, esta propiedad nos brinda la ventaja de no depender de la resolución de pantalla del usuario siempre estará fija.

A continuación, veamos la compatibilidad web al usar esta propiedad en la figura 39.

Propiedad position: fixed

Esta propiedad en particular lo que hace independientemente del valor que seleccionemos es mantener la posición del elemento HTML5 en este caso, mantendrá la posición en nuestra barra de navegación. Para comprender un poco esta propiedad aremos analogía a un escuadrón de soldados que van hacer un operativo. En este caso el escuadrón de soldados sería la clase padre el cual dentro tendrá clase hijos que haciendo analogía sería cada uno de los soldados que forman parte del escuadrón. Continuando con la analogía un escuadrón mantendrá su posición cuando este en terreno desconocido y estén en fase de exploración, mantener la posición no es más que estar en un estado, este estado puede ser estático, como también puede ser estar en movimiento.

La propiedad position tiene varios valores los cuales son:

- STATIC
- RELATIVE
- ABSOLUTE
- STICKY
- FIXED

La posición STATIC, RELATIVE son más de conservadoras y tienen un comportamiento similar ya que ambas se combinan para que ocupe un espacio definido, dicho de otra forma ambas propiedades ocupan espacio en el DOM y no pueden superponerse uno encima del otro. Pero por otra parte la propiedad ABSOLUTE lo que hace es superponerse sobre cualquier elemento que tenga static o relative. Bien en la actualidad se considera mala práctica a la hora de maquetar usar la propiedad position para maquetar, ya que no es favorable para hacerlo responsive, en vez de eso ahora se usa flexbox, y en algunos casos grid. Dicho de paso ambas propiedades formaron parte en la actualización llamada css3.

Bien ahora conozcamos más a detalle que hacen los valores de STICKY y FIXED bien ambos valores comparten los mismos comportamientos una de las diferencias que tienen es que con FIXED es un valor estático pero absoluto a la vez, mientras que STICKY comparte las mismas propiedades que FIXED, pero la diferencia es que puede volver al lugar de origen y mientras esté haciendo scroll el usuario este se moverá como si fuera una propiedad FIXED.

La razón por la que uso FIXED en lugar de STICKY es porque quiero que los modales sean estática y absoluta a la vez para así poder jugar con los valores y efectos que le pongamos a un futuro.

A continuación, veamos la compatibilidad web al usar esta propiedad en la figura 359.

Principales dificultades:

Para este efecto en concreto la principal dificultad fue organizar de distintas formas posibles, así como ver las posibles conjugaciones que pueden hacer con el efecto para que el usuario al final no tenga problemas al implementarlo. Esas dificultades surgieron a raíz de las siguientes preguntas:

- ¿Qué sucede si el usuario necesita usar un modal que ocupe la mitad de la pantalla o una modal pequeño que se use de alerta?

Al inicio se tenía de una sola medida el modal un modal estándar, luego surgió esta duda, para ello se decidió adaptar los modales para que estos soporten formatos en tabletas y celulares.

¿Qué usamos para hacer las pruebas?

Se uso el navegador Google Chrome y opera mini, así como la siguiente página web que es un simulador online:

<http://www.responsinator.com>

Esto con motivos de asegurarnos que funcione en dispositivos móviles y tabletas. Ya que en base a lo trabajado no es del todo fiable usar el navegador para verificar la correcta maquetación de los elementos, así como

tampoco es fiable depender de un software web, por ello se decidió diversificar en cuanto a las herramientas para probar las clases y efectos trabajados en el framework XRL8.

También usamos como editor de código visual studio code y su extensión live server para visualizar en vivo los efectos realizados en el css en tiempo real, también usamos el autoguardado que trae dicho editor de código para no estar guardando el proyecto a cada rato. Así conseguimos un mejor desempeño y evitamos el tedioso guardar cambios a cada rato, también evitamos el que la memoria cache se llene mostrándonos los mismos resultados.

FASE 3.- Hacerlo responsive

Este módulo lo que se empleo fue los siguientes métodos:

- Se uso la propiedad flex-box y sus posibles conjugaciones con cada uno de sus valores según sea el caso.
- También se usó las unidades de medida vh o rem para texto y contenedores ya que estas son imprescindibles a la hora de hacer responsive un sitio web.
- Se trabajo con la propiedad @media_query en las resoluciones de pantalla para celulares y tabletas. Siempre empezando a maquetar por la parte de celulares y luego tabletas. Los siguientes valores fueron los puntos de corte:

❖ Modulo modal con formulario

Figura 406. Uso de media query en el componente modal formularios parte I

```
1  /*Responsive chico*/
2  @media screen and (max-width: 420px) {
3    .modal_peque {
4      width: 86% !important;
5    }
6  }
7
8  /*Responsive Normal*/
9  @media screen and (max-width: 544px) {
10   .modal_form {
11     width: 92%;
12   }
13 }
14
15 /*Responsive Grande*/
16 @media screen and (max-width: 800px) {
17   .modal_grande {
18     width: 96% !important;
19   }
20 }
```

❖ Submódulo modal publicidad

Figura 407. Uso de media query en el componente modal formularios parte II

```
1 @media (max-width: 688px) {  
2   .contenido-publicidad {  
3     width: 90%;  
4     max-width: none;  
5   }  
6 }
```

FASE 4.- Testing asegurarse que no tenga conflicto con otros efectos.

En esta parte del desarrollo de la clase se dispuso a probar poniéndolo con las demás clases juntándolas. Para ello lo que se hizo fue colocar dentro de un archivo index.html todas las clases trabajadas y ver como el efecto de modal formulario así como el submódulo modal publicidad respondían. En algunos casos había conflicto con los demás efectos para corregir esto se procedió a implementar en el desarrollo dos cosas básicas la primera fue:

- Para detectar que clase era la que causaba el conflicto y ver en que archivo scss estaba. Se procedía a inspeccionar la página para detectar donde estaba el problema. En muchas ocasiones no se daba con el archivo causante del error entonces en ese caso aplicábamos la frase “Divide y vencerás”. Lo que se procedió hacer fue comentar de uno en uno cada efecto en el archivo principal que los llamaba esto reducía las líneas de código de css, además como teníamos en tiempo real los cambios fue mucho más rápido detectar el problema. Ya que no se tenía que recargar a cada rato además otro punto a favor era que se auto compilaba los cambios de sass automáticamente gracias a nodeJS el cual mediante la consola de comandos de Windows cuando detectaba un cambio en las líneas de código se auto compilaba y generando o reemplazando el archivo css en el cual hacíamos las pruebas, esto aumento la productividad para crear los efectos ya que al comentar de uno en uno cada archivo sass llegaba un momento donde desaparecía el error entonces revisábamos y nos dimos cuenta de que era uno o en algunos casos hasta tres efectos que hacían conflicto con la clase escrita.
- Se procedió a escribir mediante comentarios en un archivo llamado `_error_logs.scss` la clase que hace conflicto con la clase escrita, además

de describir una nota breve en caso no se disponía de tiempo para solucionarlo luego, esto permitió tener un reporte más preciso de los incidentes que íbamos detectando.

- Se tenía problemas con el módulo input formulario ya que estos iban dentro del modal formularios los contenedores de ambos módulos se repelían haciendo un espacio demasiado grande como para ser un marco de trabajo esto a su vez hacía que al llegar a los puntos de corte de los media query se distorsione los casilleros de los inputs.

FASE 5.- Aplicar la metodología OOCSS, llevándolo a SASS

- **Código css.** - Esta parte ya estaría completa por lo descrito anteriormente.
- **Código repetido.** - Identificación de código repetido

En este pequeño apartado luego de escribir y debugear el error se procedió a identificar el código css repetido para llevarlo a un archivo SCSS en sass, para su mayor optimización y aplicación de la metodología. Así que se procedió a reciclar el siguiente código:

❖ Modulo modal formulario

Figura 408. Código repetido del componente modal formularios parte I

```
1  div#contenedor_modal {
2    display: flex;
3    justify-content: center;
4    align-items: center;
5    margin: 0 auto;
6    padding: 0;
7    box-sizing: border-box;
8  }
9
10 .modal:before {
11   content: "";
12   display: none;
13   background: rgba(0, 0, 0, 0.6);
14   position: fixed;
15   top: 0;
16   left: 0;
17   right: 0;
18   bottom: 0;
19   z-index: 10;
20   margin: 0 auto;
21   padding: 0;
22   box-sizing: border-box;
23 }
```

Figura 409. Código repetido del componente modal formularios parte II

```
1  .modal:target:before {
2    display: block;
3    margin: 0 auto;
4    padding: 0;
5    box-sizing: border-box;
6  }
7
8  .modal:target .modal_form {
9    -webkit-transform: translate(0, 0);
10   -ms-transform: translate(0, 0);
11   transform: translate(0, 0);
12   top: 10vh;
13   margin: 0 auto;
14   padding: 0;
15   box-sizing: border-box;
16 }
```

Figura 410. Código repetido del componente modal formularios parte III

```
1  .modal_form {
2    width: 31rem;
3    background: #fefefe;
4    border: #333 solid 1px;
5    border-radius: 5px;
6    position: fixed;
7    top: -100%;
8    z-index: 11;
9    -webkit-transform: translate(0, -500%);
10   -ms-transform: translate(0, -500%);
11   transform: translate(0, -500%);
12   -webkit-transition: -webkit-transform 0.3s ease-out;
13   -moz-transition: -moz-transform 0.3s ease-out;
14   -o-transition: -o-transform 0.3s ease-out;
15   transition: transform 0.3s ease-out;
16   left: 0;
17   right: 0;
18   margin: 0 auto;
19   padding: 0;
20   box-sizing: border-box;
21 }
22
```

❖ Submódulo modal publicidad

Figura 411. Código repetido del componente modal formularios parte IV

```
1  .contenedor-modal-publi {
2    text-align: center;
3    font-family: Verdana, Geneva, sans-serif;
4    margin: 0 auto;
5    padding: 0;
6    box-sizing: border-box;
7  }
8
9  .modal-publicidad {
10   width: 100%;
11   height: 100vh;
12   background: rgba(0, 0, 0, 0.8);
13   position: absolute;
14   top: 0;
15   left: 0;
16   display: flex;
17   animation: modal 2s 3s forwards;
18   visibility: hidden;
19   opacity: 0;
20   margin: 0 auto;
21   padding: 0;
22   box-sizing: border-box;
23 }
```

Figura 412. Código repetido del componente modal formularios parte V

```
1 #cerrar-publicidad + label {
2   position: fixed;
3   color: white;
4   font-size: 25px;
5   z-index: 50;
6   background: darkred;
7   height: 40px;
8   width: 40px;
9   line-height: 40px;
10  border-radius: 50%;
11  right: 10px;
12  cursor: pointer;
13  animation: modal 2s 3s forwards;
14  visibility: hidden;
15  opacity: 0;
16  margin: 0 auto;
17  padding: 0;
18  box-sizing: border-box;
19 }
```

- **Crear un objeto.** - Se describe la lógica de la función para luego crear un mixing.

Una vez identificado el código css repetido se procede hacer una función la cual se ejecuta al llamado del código, mediante esta técnica nuestro código será más puro y limpio. Cabe recalcar que modal formulario, así como modal publicidad comparte características similares.

Paralelamente en esta parte se determina la lógica de las funciones y ciclos como lo son for, while, each. También determinamos el pase de parámetros, así como la condicional que llevaran dentro, por otra parte, y no menos importante se define el nombre con el cual se llamara a las funciones.

```
function modal_formulario_y_publicidad_base(){
  margin: 0 auto;
  padding: 0;
  box-sizing: border-box;
}
```

Luego de tener la lógica se procede a llevarlo a SASS para ello tenemos dos opciones, la primera es usar @function ya que como su nombre ase referencia es para crear una función la otra opción es usar @mixin, pero ambas tienen diferencias. La primera retorna un valor, mientras @mixin solo llama a los valores en este caso no necesitamos que retorne un valor por ello usaremos @mixin. Entonces quedaría de la siguiente forma:

Figura 413. Creación del objeto modal_formulario_y_publicidad_base en función mixin

```
1 @mixin modal_formulario_y_publicidad_base{
2     margin: 0 auto;
3     padding: 0;
4     box-sizing: border-box;
5 }
```

- **OOCSS**. – Luego de tener la función o la lógica para reciclar el código se procede a integrar la metodología OOCSS para ello debemos llamar al mixin para hacerlo solo necesitamos agregar @include “nombre del mixin”. Luego de integrarlo ya estará aplicada correctamente la metodología OOCSS como se muestra a continuación:

❖ Modulo modal formulario

Figura 414. Metodología OOCSS en el componente modal formulario parte I

```
1 div#contenedor_modal {
2     display: flex;
3     justify-content: center;
4     align-items: center;
5     @include modal_formulario_y_publicidad_base();
6 }
7
8 .modal:before {
9     content: "";
10    display: none;
11    background:$modal_formulario_transparencia_fondo;
12    position: fixed;
13    top: 0;
14    left: 0;
15    right: 0;
16    bottom: 0;
17    z-index: 10;
18    @include modal_formulario_y_publicidad_base();
19 }
```

Figura 415. Metodología OOCSS en el componente modal formulario parte II

```
1 .modal:target:before {
2     display: block;
3     @include modal_formulario_y_publicidad_base();
4 }
5
6 .modal:target .modal_form {
7     -webkit-transform: translate(0, 0);
8     -ms-transform: translate(0, 0);
9     transform: translate(0, 0);
10    top:10vh;
11    @include modal_formulario_y_publicidad_base();
12 }
```


Figura 416, Metodología OOCSS en el componente modal formulario parte III

```
1 .modal_form {
2   width: 31rem;
3   background: #fefefe;
4   border: #333 solid 1px;
5   border-radius: 5px;
6   position: fixed;
7   top: -100%;
8   z-index: 11;
9   -webkit-transform: translate(0, -500%);
10  -ms-transform: translate(0, -500%);
11  transform: translate(0, -500%);
12  -webkit-transition: -webkit-transform 0.3s ease-out;
13  -moz-transition: -moz-transform 0.3s ease-out;
14  -o-transition: -o-transform 0.3s ease-out;
15  transition: transform 0.3s ease-out;
16  left: 0;
17  right: 0;
18  @include modal_formulario_y_publicidad_base();
19 }
```

❖ Submódulo modal publicidad

Figura 417. Metodología OOCSS en el componente modal formulario parte IV

```
1 .contenedor-modal-publi{
2   text-align: center;
3   font-family:$fuente_formal;
4   @include modal_formulario_y_publicidad_base();
5 }
6
7 .modal-publicidad{
8   width: 100%;
9   height: 100vh;
10  background: rgba(0,0,0,$modal_publicidad_fondo_transparencia);
11  position: absolute;
12  top:0;
13  left: 0;
14  display: flex;
15  animation: modal $modal_publicidad_tiempo_aparecion forwards;
16  visibility: hidden;
17  opacity:0;
18  @include modal_formulario_y_publicidad_base();
19 }
```

Figura 418. Metodología OOCSS en el componente modal formulario parte V

```
1 #cerrar-publicidad + label{
2   position: fixed;
3   color:$color-blanco-1;
4   font-size: 25px;
5   z-index:$modal_publicidad_nivel_visibilidad;
6   background: darkred;
7   height: 40px;
8   width: 40px;
9   line-height: 40px;
10  border-radius: 50%;
11  right: 10px;
12  cursor: pointer;
13  animation: modal $modal_publicidad_tiempo_aparecion forwards;
14  visibility: hidden;
15  opacity:0;
16  @include modal_formulario_y_publicidad_base();
17 }
```

Bien volviendo al tema de las variables estas variables se encuentran en un archivo sass denominado variables valga la redundancia donde se encuentran las siguientes variables:

❖ Modulo modal formulario

Figura 419. Variables sass del componente modal formulario parte I

```
$modal_formulario_ancho_peque:22.5rem!important;  
$modal_formulario_ancho_grande:48rem!important;  
$modal_formulario_transparencia_fondo:rgba(0, 0, 0, 0.6);
```

❖ Submódulo modal publicidad

Figura 420. Variables sass del componente modal formulario parte II

```
$modal_publicidad_fondo_transparencia:0.8;  
$modal_publicidad_tiempo_aparecion:2s 3s;  
$modal_publicidad_nivel_visibilidad:50;
```

Como se ha visto SASS es un preprocesador de css por ello se ejecuta antes del css tradicional esto permite añadir variables, así como agregar condicionales, etc. Con la aplicación de la metodología como se observa se reutiliza el código css, además de que se hace escalable el proyecto. Ya que, si en un futuro queremos cambiar o añadir más propiedades css, solo cambiaríamos las variables o añadiríamos más propiedades en el mixin.

FASE 6.- Documentar en la web site www.phooldx.com

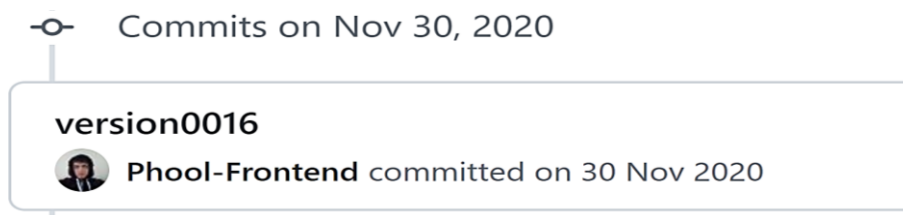
Finalmente, si el efecto paso por todos los de más filtros se procede a su documentación respectiva, para documentar todos los efectos se creó una página web de documentación usando clases y elementos del mismo framework. A continuación, se describirá como se procede a la documentación de la misma. Siguiendo estos pasos:

- Se copia las etiquetas de HTML5 junto con la clase a un txt temporal para no perder la sintaxis
- Se guarda los cambios en el compilador de SASS
- Como es la parte final se procedió hacer copias de seguridad para no perder el avance, en archivos RAR. Estos archivos estaban por versiones enumerados además se agregaba un pequeño texto describiendo el avance. Este guardado se hacía de manera local en un USB.
- Para una mayor seguridad del avance del código se procedió luego a copiar la versión del archivo RAR junto a la descripción del mismo.

Guardado de código en GitHub. - Luego de tener la copia del proyecto a nivel físico en un USB, se procedió a guardarlo en un repositorio de GitHub para lo cual se usó Git, para ello se procedió de la siguiente forma:

- Se guardaba todo el proyecto con el siguiente comando que se observa en la figura 12.
- Luego se verificaba los cambios guardados con el siguiente comando que se observa en la figura 13.
- Después se hacía el commit, es decir se le asignaba un nombre a los cambios realizados con el siguiente comando que se observa en la figura 14.
- Luego se procede a empaquetar los datos para que se suban al repositorio de GitHub con el siguiente código que se observa en la figura 15.
- Finalmente, luego de haber enviados el proyecto empaquetado hacia GitHub como se puede apreciar en la imagen, este queda guardado en dicho repositorio.

Figura 421 Visualización en github del componente modal formulario



Se guarda en una escala de tiempo, con todos los cambios hechos desde la creación del proyecto, esto ayuda a tener un mejor control de versiones. Asimismo, también ayuda a que más personas se sumen al proyecto y puedan clonar el proyecto como trabajar por separado el proyecto, entre otras muchas ventajas.

- Así mismo se manejó un archivo llamado bitácora.txt donde aquí se registró todos los incidentes ocurridos, así como los cambios hechos, se registraba también las metas cumplidas, como las tareas nuevas para

desarrollar al día siguiente. Al finalizar con un efecto por completo se quitaba de la lista de prioridades así dábamos pase al siguiente efecto o problema ha resolver. Esto servía para tener un mejor orden y saber ejecutar prioridades.

- Luego de tener guardado el avance y de registrar el avanzase, se procedió a copiar del archivo documentación Etiquetas.txt el fragmento de código que se copió anteriormente para luego pegarlo en www.nosetup.org esta página web se encargaba de convertir html en texto plano para poder visualizar html
- Antes de finalizar se procedió a crear un archivo llamado `modulo_modal.html` donde se dividía en tres partes:
 - En la primera se describía un poco al efecto que era y que hacia
 - En la segunda parte se procedía a copiar el código ya formateado obtenido de www.nosetup.com para luego pegarlo.
 - En la tercera parte lo que se hacía era mostrar un apartado llamado estructura donde se describía como se llamaban las variables para que sea modificado en SASS.
- Finalmente, el ejemplo listo para usarse queda de la siguiente forma:

❖ Modulo modal formulario

- Modal grande

Figura 422. Código html5 final del componente modal formulario parte I

```
<a href="#contenedor_modal_grande" class="btn_modal">Abrir modal</a>

<div class="modal" id="contenedor_modal_grande" aria-hidden="true">
  <div class="modal_form modal_grande">
    <div class="modal_resistencia">
      <h2>Formulario</h2>
      <a href="#" class="modal_btn_salir_modal" aria-hidden="true">x</a>
    </div>
    <div class="modal_contenido">
      <div class="form_base habilitar_form_modal">
        <h1>Buenas tardes</h1>
        <div class="maquetado">

          <div class="campo_form col-largo-2 col-medi-3 col-peque-3 col-chiki-6">
            <label>Nombres:</label>
            <input class="max_clasi" type="text" name="" id="">
          </div>

          <div class="campo_form col-largo-2 col-medi-3 col-peque-3 col-chiki-6">
            <label>Apellidos Paterno:</label>
            <input class="max_clasi" type="text" name="" id="">
          </div>

          <div class="campo_form col-largo-2 col-medi-3 col-peque-3 col-chiki-6">
            <label>Apellido Materno:</label>
            <input class="max_clasi" type="text" name="" id="">
          </div>

          <div class="campo_form col-largo-2 col-medi-3 col-peque-3 col-chiki-6">
            <label>Direccion:</label>
            <input class="max_clasi" type="text" name="" id="">
          </div>

          <div class="campo_form col-largo-2 col-medi-3 col-peque-3 col-chiki-6">
            <label>Correo:</label>
            <input class="max_clasi" type="text" name="" id="">
          </div>

          <div class="campo_form col-largo-2 col-medi-3 col-peque-3 col-chiki-6">
            <label>Telefono:</label>
            <input class="max_clasi" type="text" name="" id="">
          </div>

          <div class="campo_form col-largo-2 col-medi-2 col-peque-2 col-chiki-6">
            <label>Email or phone:</label>
            <input class="max_clasi" type="text" name="" id="">
          </div>

        </div>
      </div>
    </div>
  </div>
  <div class="modal_contenido">
    <a href="#" class="btn_modal">Enviar</a>
  </div>
</div>
```

- Modal estándar

Figura 423. Código html5 final del componente modal formulario parte II

```
<a href="#contenedor_modal_normal" class="btn_modal">Abrir modal</a>

<div class="modal" id="contenedor_modal_normal" aria-hidden="true">
<div class="modal_form modal_grande">
<div class="modal_resistencia">
<h2>Formulario Normal</h2>
<a href="#" class="modal_btn_salir_modal" aria-hidden="true"></a>
</div>
<div class="modal_contenido">
<div class="form_base habilitar_form_modal">
<div class="maquetado">

<div class="campo_form col-largo-2 col-medi-3 col-peque-3 col-chiki-6">
<label>Nombres:</label>
<input type="text" name="" id="">
</div>

<div class="campo_form col-largo-2 col-medi-3 col-peque-3 col-chiki-6">
<label>Apellidos Paterno:</label>
<input type="text" name="" id="">
</div>

<div class="campo_form col-largo-2 col-medi-3 col-peque-3 col-chiki-6">
<label>Apellido Materno:</label>
<input type="text" name="" id="">
</div>

<div class="campo_form col-largo-2 col-medi-3 col-peque-3 col-chiki-6">
<label>Direccion:</label>
<input type="text" name="" id="">
</div>

<div class="campo_form col-largo-2 col-medi-3 col-peque-3 col-chiki-6">
<label>Correo:</label>
<input type="text" name="" id="">
</div>

<div class="campo_form col-largo-2 col-medi-3 col-peque-3 col-chiki-6">
<label>Telefono:</label>
<input type="text" name="" id="">
</div>

</div>
</div>
</div>
<div class="modal_contenido">
<a href="#" class="btn_modal">Enviar</a>
</div>
</div>
</div>
```

- Modal pequeño

Figura 424. Código html5 final del componente modal formulario parte III

```
<a href="#contenedor_modal_peque" class="btn_modal">Abrir modal</a>

<div class="modal" id="contenedor_modal_peque" aria-hidden="true">
<div class="modal_form modal_grande">
<div class="modal_resistencia">
<h2>Formulario Pequeño</h2>
<a href="#" class="modal_btn_salir_modal" aria-hidden="true">x</a>
</div>
<div class="modal_contenido">
<div class="form_base habilitar_form_modal">
<div class="maquetado">

<div class="campo_form col-largo-2 col-medi-3 col-peque-3 col-chiki-6">
<label>Nombres:</label>
<input class="min_clasi" type="text" name="" id="">
</div>

<div class="campo_form col-largo-2 col-medi-3 col-peque-3 col-chiki-6">
<label>Apellidos Paterno:</label>
<input class="min_clasi" type="text" name="" id="">
</div>

<div class="campo_form col-largo-2 col-medi-3 col-peque-3 col-chiki-6">
<label>Apellido Materno:</label>
<input class="min_clasi" type="text" name="" id="">
</div>

<div class="campo_form col-largo-2 col-medi-3 col-peque-3 col-chiki-6">
<label>Direccion:</label>
<input class="min_clasi" type="text" name="" id="">
</div>

<div class="campo_form col-largo-2 col-medi-3 col-peque-3 col-chiki-6">
<label>Correo:</label>
<input class="min_clasi" type="text" name="" id="">
</div>

<div class="campo_form col-largo-2 col-medi-3 col-peque-3 col-chiki-6">
<label>Telefono:</label>
<input class="min_clasi" type="text" name="" id="">
</div>

</div>

</div>
</div>
<div class="modal_contenido">
<a href="#" class="btn_modal">Enviar</a>
</div>
</div>
</div>
```

❖ Submódulo modal publicidad

Figura 425. Código html5 final del componente modal formulario parte IV

```
<div class="contenedor-modal-publi">
  <input type="checkbox" id="cerrar-publicidad">
  <label for="cerrar-publicidad" id="btn-cerrar">X</label>
  <div class="modal-publicidad">
    <div class="contenido-publicidad">
      <h2>Visita nuestro blog</h2>
      <label>Nombre:</label><br>
      <input type="text" name="" id=""><br>

      <label for="">Edad:</label><br>
      <input type="text" name="" id=""><br>
    </div>
  </div>
  <main>
    <h1>Ventana modal con HTML y CSS</h1>
    <p>Veremos una ventana modal de publicidad hecha usando css y html</p>
  </main>
</div>
```

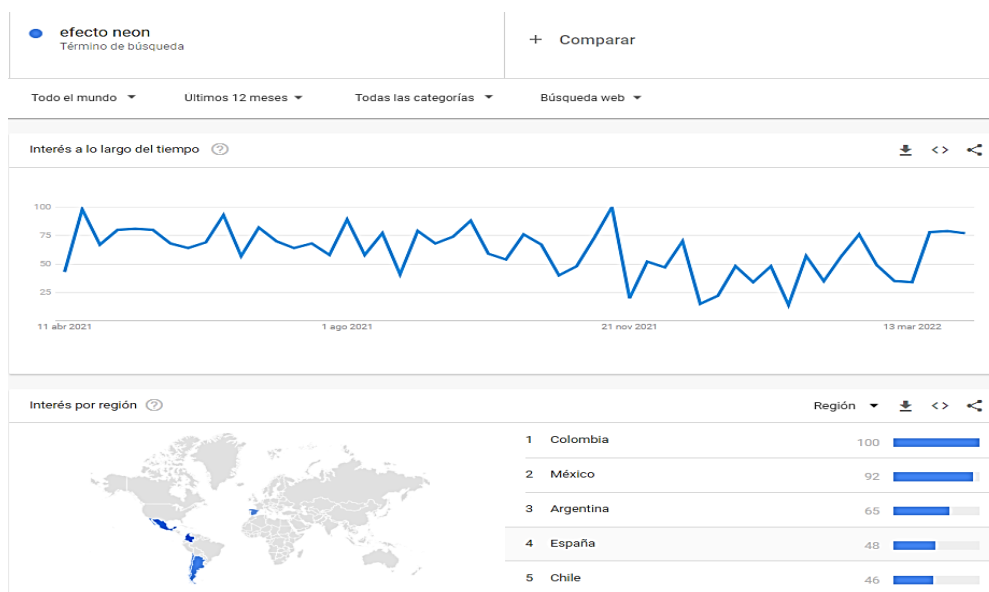

NEON

FASE 1.- Investigación del efecto y su importancia

Como primer paso lo primero que se hizo fue investigar e indagar sobre el efecto, en primera instancia el efecto neón no es más que la fosforescencia como la intensidad mezclada con las sombras que se le asigna a un texto todas estas combinadas produce una luz muy llamativa ante el ojo humano, ya que es inusual verla es por ello que en las señales de tránsito son implementadas, ya que se activa con la luz de los automóviles pero en desarrollo web no necesita activarse con nada simplemente es cuestión de añadirle varias capas de código. Este fue elegido porque beneficia enormemente a la experiencia de usuario, es muy usado en sitios web que ofrecen servicios de consumo como suelen ser bares, clubs nocturnos, salas de billar, club de juegos, comida rápida, karaokes, etc. En resumen, sitios que suelen abrir en la noche, este efecto fue elegido para ser enfocado o aplicado a este tipo de sector de la industria que pasa desapercibido en algunos casos asíéndolos común debido a varios factores siendo uno de ellos la falta de percepción en los efectos nocturnos, así como la implementación en estos.

Por otra parte, una de las palabras frecuentes buscadas según Google trends como se observa el gráfico es el efecto neón:

Figura 426 Gráfico de barras de Google trends sobre efecto neón



Otro del motivo el cual fue elegido el efecto para formar parte del framework XRL8 fue que el proyecto presente debe estar a la altura de los frameworks y librerías o inclusive superar a los que están en la actualidad siendo usados en la industria del desarrollo web, aunque para este efecto en particular solo se encontró repositorios en github y librerías como lo son:

- **ToxelCSS.** – La presente librería es pequeña y recién está en fase Alpha de su desarrollo esta cuenta con una clase denominada luz-neón en el cual el texto se transforma en fosforescente, si desean saber más sobre ello puede visitar la web oficial de la librería mediante el siguiente enlace:

<https://www.luisangelmaciell.com/Toxel/>

- **Choreographet.**- Esta librería se caracteriza por tener animación en las letras como un efecto fosforescente característico del neón , para ver cómo se emplea y formas de uso revisar el siguiente enlace que los lleve a la documentación oficial de dicha librería:

<https://christinecha.github.io/choreographer-js/>

- **Glowbutton.**- Esta librería es propiamente dicha para el uso en Android studio dicha propiedad está enfocada a la implantación de los botones. Para revisar su documentación ingresar al siguiente enlace:

<https://github.com/SanojPunchihewa/GlowButton>

- **GlowNeonButton.**- La presente librería también está enfocada para el uso de Android studio la diferencia de sus antecesores es que esta trae iconos en cada botón así como una gama más amplia de colores que podemos usar en nuestro diseño. Para ver el uso de este se puede recurrir al siguiente link:

<https://github.com/SMehranB/GlowNeonButton>

- **JqueryGlow.**- Finalmente tenemos a la presente librería la cual está apoyada en los métodos de la librería de JavaScript conocida como jQuery, la librería trabaja con las propiedades de sobras que trae css para crear los

efectos neón. Lo podremos observar más detalladamente accediendo mediante el siguiente link:

<https://github.com/nakajima/jquery-glow>

Modulo: Un módulo según definición propia es una sección de trabajo principal a comparación de un submódulo este puede ser denominado la estructura padre y los submódulos estructuras hijas. En este efecto en particular el módulo vendría a ser la siguiente clase:

- **Neón.** - Para este efecto nos basamos en tres capas, las cuales fueron fundamentales para el funcionamiento de este módulo, la primera capa fue el núcleo de las letras por así llamarlo, la segunda capa fueron el borde de las letras y la tercera capa fue la extensión de sombra todas ellas juntas están como variables en el framework XRL8.

Submódulos: Considere llamarle submódulo, a los efectos o clases que están dentro de un módulo en este caso el módulo es neón, si tendríamos un efecto similar con atributos diferentes en ese caso sería un submódulo, como también podría ser un submódulo clases o elementos que hereden del módulo atributos.

Estos submódulos son muy útiles ya que desde mi punto de vista ayuda a diversificar los efectos y las clases. Beneficiando al usuario final en la construcción de sitios web. Actualmente el framework XRL8 no tiene pensado sacar sub módulos para el efecto Neón.

FASE 2.- Codificar el efecto

Antes de ver esta sección a fondo, quisiera comentar que fue sencilla la codificación, lo duro realmente fue saber el número de capas para el neón y los colores que esten sincronizados armoniosamente.

Bien luego de tener claro que efecto queríamos construir y porque lo íbamos a construir como se vio en el capítulo anterior se procedió a escribir código como se muestra a continuación.

Código completo

Figura 427. Código de neón

```
1  .neon {
2    position: absolute;
3    top: 50%;
4    left: 50%;
5    transform: translate(-50%, -50%);
6    margin: 0;
7    padding: 0 20px;
8    font-size: 6em;
9    color: white;
10   text-shadow: 0 0 20px #ff005b;
11 }
12
13 .neon:after {
14   content: attr(data-text);
15   position: absolute;
16   top: 0;
17   left: 0;
18   padding: 0 20px;
19   z-index: -1;
20   color: #ff0059fa;
21   filter: blur(15px);
22 }
23
24 .neon:before {
25   content: "";
26   position: absolute;
27   top: 0;
28   left: 0;
29   width: 100%;
30   height: 100%;
31   background: #fe3a80;
32   z-index: -2;
33   opacity: 0.6;
34   filter: blur(40px);
35 }
```

Descripción de las propiedades más significativas:

A continuación, se procederá a detallar las propiedades más usadas y su aplicación. También se mostrará la compatibilidad web de cada propiedad para su mejor comprensión dejamos una leyenda la cual se puede observar en la figura 2.

Propiedad width:100% :

Se empleo width 100% para el ancho de este efecto ya que si bien esta propiedad tiene diversas unidades de medidas como:

- EM
- VH
- PÍXELES
- REM
- PORCENTAJE

Se fue descartando una a una al momento de probarla, ya que al usar la unidad EM este se distorsionaba por su propia naturaleza, básicamente esta propiedad usa el ancho de la pantalla del dispositivo donde estemos y las va ajustando así que fue descartado.

A continuación, veamos la compatibilidad web al usar esta propiedad en la figura 3.

Propiedad height:100%:

Luego de ver las unidades de medida en la propiedad anterior se decidió trabar en porcentaje, ya que se necesitaba un valor dinámico, poniendo como único límite sea las dimensiones del dispositivo del usuario, en ese sentido se descartó las propiedades que toman el ancho del dispositivo como son REM y EM por otra parte también se descartó a los que toman el alto del dispositivo como son VH. Luego de reducir nuestro marco de trabajo se tenía PORCENTAJE o PÍXELES así que descartamos a PÍXELES.

Se empleo PORCENTAJE por su propiedad para hacer responsive los elementos que se encuentren dentro del contenedor o sean este un elemento padre.

A continuación, veamos la compatibilidad web al usar esta propiedad en la figura 4.

Propiedad position: absolute

La propiedad ABSOLUTE lo que hace es sobreponerse sobre cualquier elemento que tenga static o relative. Bien en la actualidad se considera mala práctica a la hora de maquetar usar la propiedad position para maquetar, ya que no es favorable para hacerlo responsive, en vez de eso ahora se usa flexbox, y en algunos casos grid. Dicho de paso ambas propiedades formaron parte en la actualización llamada css3.

A continuación, veamos la compatibilidad web al usar esta propiedad en la figura 359.

Propiedad filter: blur(15px)

La propiedad filter hace un desenfocado y satura los efectos visuales para un elemento, el valor blur es un desenfocado gaussiano hace que se mezclen los pixeles recibe como parámetro el valor o la cantidad a mezclar además no se acepta porcentaje.

A continuación, veamos la compatibilidad web al usar esta propiedad en la figura 147.

Propiedad content: attr(data-text)

Para usar esta propiedad es requisito primordial usar un pseudo-elemento como:

- Before: Lo que hace esta propiedad es insertar elementos antes de la clase.
- After: Lo que hace esta propiedad es insertar elementos después de la clase.

Estos pseudo-elementos son usados para añadir contenido a un elemento con la propiedad content, aunque también en content puede ir palabras y estas se insertaran, en esta clase en particular donde se usó, se observa attr(data-text) esto quiere decir que se insertara en el atributo que tenga data-text toda la propiedad que este pueda tener.

A continuación, veamos la compatibilidad web al usar esta propiedad en la figura 144.

Principales dificultades:

Para este efecto en concreto la principal dificultad fue organizar de distintas formas posibles, así como ver las posibles conjugaciones que pueden hacer con el efecto para que el usuario al final no tenga problemas al implementarlo. Esas dificultades surgieron a raíz de las siguientes preguntas:

- ¿Cómo le damos la tonalidad neón a una letra?

Sin duda este fue el primer obstáculo con el que nos topamos, se intentó usar sombras con tex-shadow pero no fue suficiente, luego se probó

usar bordes tampoco se logró el resultado esperado, sentí que faltaba el ingrediente secreto, ese ingrediente era una fina capa de sombras realizada con padding pero luego de lograrlo no me convenció como se veía así que se decidió unir todo lo mencionado sombras,bordes, fina sobra hecha con padding, hasta que se logró el resultado esperado.

Pero no quedo tampoco como un neón le faltaba la cereza al pastel, así que decidí ver imágenes neón y avisos en neón por un tiempo prolongado hasta que "Eureka" me di cuenta que la esencia era tener un color base y a partir de ahí ir bajando el grado de intensidad del color como si esta emanara iluminación, hasta que al final salió el efecto.

- ¿Qué pasa si tenemos textos demasiado pequeños?

Al empezar a escribir código no se tuvo en cuenta ese pequeño detalle se pensó que era como declarar la propiedad color en una clase y esta afecta directamente a todo el texto o elementos hijos que tenga. Pero me equivoque porque el efecto neón para se tiene que mezclar varios elementos y se necesita tener la cantidad de letras exactas para que este se aplique correctamente así que se decidió maquetar de una forma diferente a lo usual, haciendo esto se logró reducir esto a 3 elementos como mínimo, pero al final tuvimos que volver a redefinir toda la clase para usar un atributo en el HTML y este mediante CSS hacer que lo reconozca esa propiedad fue `content: attr(data-text)`.

- ¿Qué pasa si tenemos textos demasiado largos?

Al inicio no se tomó en cuenta en cuenta la longitud del texto sea pequeño o largo y cuando se solucionaba la parte de letras mínimas a 3 elementos la parte de longitud larga se redujo a 7 pero no era una solución óptima porque esto podía variar así que al final tuvimos que volver a redefinir toda la clase para usar atributo en el HTML y este mediante CSS hacer que lo reconozca esa propiedad fue `content:attr(data-text)`.

¿Qué usamos para hacer las pruebas?

Se uso el navegador Google Chrome y opera mini, así como la siguiente página web que es un simulador online:

<http://www.responsinator.com>

Esto con motivos de asegurarnos que funcione en dispositivos móviles y tabletas. Ya que en base a lo trabajado no es del todo fiable usar el navegador para verificar la correcta maquetación de los elementos, así como tampoco es fiable depender de un software web, por ello se decidió diversificar en cuanto a las herramientas para probar las clases y efectos trabajados en el framework XRL8.

También usamos como editor de código visual studio code y su extensión live server para visualizar en vivo los efectos realizados en el css en tiempo real, también usamos el autoguardado que trae dicho editor de código para no estar guardando el proyecto a cada rato. Así conseguimos un mejor desempeño y evitamos el tedioso guardar cambios a cada rato, también evitamos el que la memoria cache se llene mostrándonos los mismos resultados.

FASE 3.- Hacerlo responsive

Se empleo las siguientes técnicas para hacerlo responsive:

- Se uso en porcentaje al maquetar el ancho del efecto.
- También se uso las unidades de medida rem para texto y contenedores ya que estas son imprescindibles a la hora de hacer responsive un sitio web.

FASE 4.- Testing asegurarse que no tenga conflicto con otros efectos.

En esta parte del desarrollo de la clase se dispuso a probar poniéndolo con las demás clases juntándolas. Para ello lo que se iso fue colocar dentro de un archivo index.html todas las clases trabajadas y ver como el efecto neón respondía. En algunos casos había conflicto con los demás efectos para corregir esto se procedió a implementar en el desarrollo dos cosas básicas la primera fue:

- Para detectar que clase era la que causaba el conflicto y ver en que archivo scss estaba. Se procedía a inspeccionar la página para detectar donde estaba el problema. En muchas ocasiones no se daba con el archivo causante del error entonces en ese caso aplicábamos la frase “Divide y vencerás”. Lo que se procedió hacer fue comentar de uno en uno cada efecto en el archivo principal que los llamaba esto reducía las líneas de código de css, además como teníamos en tiempo real los cambios fue mucho más rápido detectar el problema. Ya que no se tenía que recargar a cada rato además otro punto a favor era que se auto compilaba los cambios de sass automáticamente gracias a nodeJS el cual mediante la consola de comandos de Windows cuando detectaba un cambio en las líneas de código se auto compilaba y generando o reemplazando el archivo css en el cual hacíamos las pruebas, esto aumento la productividad para crear los efectos ya que al comentar de uno en uno cada archivo sass llegaba un momento donde desaparecía el error entonces revisábamos y nos dimos cuenta de que era uno o en algunos casos hasta tres efectos que hacían conflicto con la clase escrita.
- Se procedió a escribir mediante comentarios en un archivo llamado `_error_logs.scss` la clase que hace conflicto con la clase escrita, además de describir una nota breve en caso no se disponía de tiempo para solucionarlo luego, esto permitió tener un reporte más preciso de los incidentes que íbamos detectando.
- El mayor conflicto que se obtuvo al desarrollar este efecto fue los márgenes y padding que hacían conflicto con otras clases ya que no se encontraba la porción exacta según se ha de aplicar sobre una oración o palabra, pero esto se solucionó usando un atributo de datos el cual fue denominado como `data-text` que es un identificador, este es fácil de usar a decir verdad es como usar una clase en una etiqueta pero este contendrá datos en este caso se escribirá el nombre o la palabra que quiere el usuario hacer neón.

FASE 5.- Aplicar la metodología OOCSS, llevándolo a SASS

- **Código css.** - Esta parte ya estaría completa por lo descrito anteriormente.

- **Código repetido.** - Identificación de código repetido

En este pequeño apartado luego de escribir y debugear el error se procedió a identificar el código css repetido para llevarlo a un archivo SCSS en sass, para su mayor optimización y aplicación de la metodología. Así que se procedió a reciclar el siguiente código:

Figura 428. Código repetido del componente neón

```
1  .neon:after {
2  content: attr(data-text);
3  position: absolute;
4  top: 0;
5  left: 0;
6  padding: 0 20px;
7  z-index: -1;
8  color: #ff0059fa;
9  filter: blur(15px);
10 }
11
12 .neon:before {
13 content: "";
14 position: absolute;
15 top: 0;
16 left: 0;
17 width: 100%;
18 height: 100%;
19 background: #fe3a80;
20 z-index: -2;
21 opacity: 0.6;
22 filter: blur(40px);
23 }
```

- **Crear un objeto.** - Se describe la lógica de la función para luego crear un mixing.

Una vez identificado el código css repetido se procede hacer una función la cual se ejecuta al llamado del código, mediante esta técnica nuestro código será más puro y limpio.

Paralelamente en esta parte se determina la lógica de las funciones y ciclos como lo son for, while, each. También determinamos el pase de parámetros, así como la condicional que llevaran dentro, por otra parte, y no menos importante se define el nombre con el cual se llamara a las funciones.

```
function neon_base(){
  position: absolute;
  top: 0;
  left: 0;
}
```

Luego de tener la lógica se procede a llevarlo a SASS para ello tenemos dos opciones, la primera es usar @function ya que como su nombre ase referencia es para crear una función la otra opción es usar @mixin, pero ambas tienen diferencias. La primera retorna un valor, mientras @mixin solo llama a los valores en este caso no necesitamos que retorne un valor por ello usaremos @mixin. Entonces quedaría de la siguiente forma:

Figura 429. Creación del objeto neon_base en función mixin

```
@mixin neon_base{
  position: absolute;
  top: 0;
  left: 0;
}
```

- **OOCSS.** – Luego de tener la función o la lógica para reciclar el código se procede a integrar la metodología OOCSS para ello debemos llamar al mixin para hacerlo solo necesitamos agregar @include “nombre del mixin”. Luego de integrarlo ya estará aplicada correctamente la metodología OOCSS

Figura 430. Metodología OOCSS en el componente neón

```
1 .neon:after{
2   content: attr(data-text);
3   @include neon_base;
4   padding: 0 20px;
5   z-index: -1;
6   color:$neon_fondo_segunda_capa;
7   filter:blur(15px);
8 }
9
10 .neon:before{
11  content: '';
12  @include neon_base;
13  width: 100%;
14  height: 100%;
15  background:$neon_fondo_tercera_capa;
16  z-index: -2;
17  opacity:.6;
18  filter: blur(40px);
19 }
```

Bien volviendo al tema de las variables estas variables se encuentran en un archivo sass denominado variables valga la redundancia donde se encuentran las siguientes variables:

Figura 431. Variables sass del componente neon

```
$neon_fondo_primera_capa:#ff005b;  
$neon_fondo_segunda_capa:#ff0059fa;  
$neon_fondo_tercera_capa:#fe3a80;
```

Como se ha visto SASS es un preprocesador de css por ello se ejecuta antes del css tradicional esto permite añadir variables así como agregar condicionales, etc. Con la aplicación de la metodología como se observa se reutiliza el código css, además de que se hace escalable el proyecto. Ya que, si en un futuro queremos cambiar o añadir más propiedades css, solo cambiaríamos las variables o añadiríamos más propiedades en el mixin.

FASE 6.- Documentar en la web site www.phooldx.com

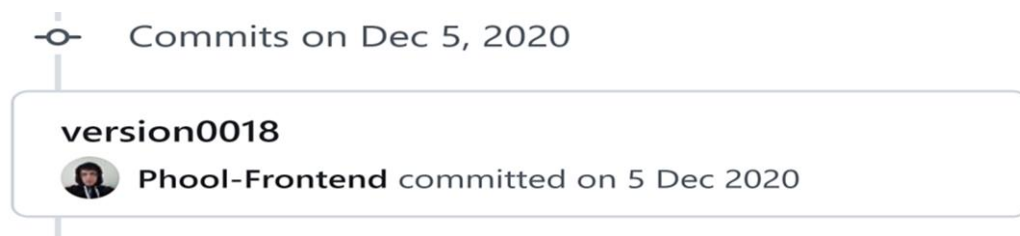
Finalmente, si el efecto paso por todos los de más filtros se procede a su documentación respectiva, para documentar todos los efectos se creó una página web de documentación usando clases y elementos del mismo framework. A continuación, se describirá como se procede a la documentación de la misma. Siguiendo estos pasos:

- Se copia las etiquetas de HTML5 junto con la clase a un txt temporal para no perder la sintaxis
- Se guarda los cambios en el compilador de SASS
- Como es la parte final se procedió hacer copias de seguridad para no perder el avance, en archivos RAR. Estos archivos estaban por versiones enumerados además se agregaba un pequeño texto describiendo el avance. Este guardado se hacía de manera local en un USB.
- Para una mayor seguridad del avance del código se procedió luego a copiar la versión del archivo RAR junto a la descripción del mismo.

Guardado de código en GitHub. - Luego de tener la copia del proyecto a nivel físico en un USB, se procedió a guardarlo en un repositorio de GitHub para lo cual se usó Git, para ello se procedió de la siguiente forma:

- Se guardaba todo el proyecto con el siguiente comando que se observa en la figura 12.
- Luego se verificaba los cambios guardados con el siguiente comando que se observa en la figura 13.
- Después se hacía el commit, es decir se le asignaba un nombre a los cambios realizados con el siguiente comando que se observa en la figura 14.
- Luego se procede a empaquetar los datos para que se suban al repositorio de GitHub con el siguiente código que se observa en la figura 15.
- Finalmente, luego de haber enviados el proyecto empaquetado hacia GitHub como se puede apreciar en la imagen, este queda guardado en dicho repositorio.

Figura 432 Visualización en github del componente neón



Se guarda en una escala de tiempo, con todos los cambios hechos desde la creación del proyecto, esto ayuda a tener un mejor control de versiones. Asimismo, también ayuda a que más personas se sumen al proyecto y puedan clonar el proyecto como trabajar por separado el proyecto, entre otras muchas ventajas.

- Así mismo se manejó un archivo llamado bitácora.txt donde aquí se registró todos los incidentes ocurridos, así como los cambios hechos, se registraba también las metas cumplidas, como las tareas nuevas para desarrollar al día siguiente. Al finalizar con un efecto por completo se

quitaba de la lista de prioridades así dábamos pase al siguiente efecto o problema ha resolver. Esto servía para tener un mejor orden y saber ejecutar prioridades.

- Luego de tener guardado el avance y de registrar el avanzase, se procedió a copiar del archivo documentación Etiquetas.txt el fragmento de código que se copió anteriormente para luego pegarlo en www.nosetup.org esta página web se encargaba de convertir html en texto plano para poder visualizar html
- Antes de finalizar se procedió a crear un archivo llamado `modulo_neon.html` donde se dividía en tres partes:
 - En la primera se describía un poco al efecto que era y que hacia
 - En la segunda parte se procedía a copiar el código ya formateado obtenido de www.nosetup.com para luego pegarlo.
 - En la tercera parte lo que se hacía era mostrar un apartado llamado estructura donde se describía como se llamaban las variables para que sea modificado en SASS.
- Finalmente, el ejemplo listo para usarse queda de la siguiente forma:

Figura 433. Código html5 final del componente neón

```
<h1 class="neon" data-text="Tesis <3">Tesis <3</h1>
```

CONCLUSIONES

- Se ha desarrollado un framework CSS para diseño web responsivo en español, utilizando las herramientas descritas en la parte de desarrollo, logrando un conjunto de módulos de formato de páginas web usando el diseño responsivo y todas las características de las reglas asociadas a la tecnología CSS.
- Se ha diseñado el framework “XRL8” como una estructura básica para agilizar y optimizar la velocidad en el usuario final para que el usuario pueda insertar los elementos de maquetación en una página web. La implementación de las clases básicas en el desarrollo del framework ha mejorado su compatibilidad en Google Chrome por ser uno de los navegadores con mayor demanda en búsquedas.
- Se crearon los componentes: Barra de navegación, pie de página, input de formulario, responsive design, libro 3D, pre carga, slider automática, paginación, texturas, parallax, tablas, imagen redonda, botones, formulario, centrado vertical y horizontal, menú acordeón, imagen en movimiento, modal de formularios y neón.
- Para la creación de cada componente, se aplicaron las 6 fases probetas de la metodología: siendo la Fase 1: la investigación del efecto y su importancia, utilizando las herramientas Bootstrap, Bulma, TailwindCSS, Foundation, Materialize, UIKIFT, y Semantic UI.
- Para la fase 2 se realizó la codificación correspondiente, la descripción de las propiedades más importantes, las dificultades encontradas, y se aplicaron las pruebas correspondientes para esta fase.
- En la fase 03, se aplicó la característica de diseño responsivo para cada elemento, y en la 04 la prueba correspondiente para no generar conflictos o errores con los elementos del framework, en la fase 05 se aplica la metodología OOCSS,
- Finalmente, en la fase 6 se realizó la documentación, en el sitio www.phooldx.com y también en la página de GitHub

REFERENCIAS

- Briceño, W. (2021). *Implementación de nueva tecnología front-end para mejorar el rendimiento de sitios web*. Universidad Tecnológica del Perú.
- García, X. (2015). *Framework J2EE capa de presentación*. Revisado el 11 de marzo del 2019. Disponible en:
<http://openaccess.uoc.edu/webapps/o2/bitstream/10609/40303/8/xaviergarsoTFC0115memoria.pdf#page=10&zoom=100,0,96>
- Grajales, C. (2016). *Implementación de un prototipo web para un plan turístico en la vereda oriente del municipio de Cartago*. Disponible en:
<http://repositorio.utp.edu.co/dspace/bitstream/handle/11059/6362/00676G743.pdf?sequence=1&isAllowed=y>
- Regatto, F. (2015). *diseño e implementación de un sistema hipermedia, que guíe a conductores y peatones sobre el correcto uso de las señales de tránsito en el Ecuador. propuesta basada en tecnología html5 y css3*. Revisado el 05 de marzo del 2019. Disponible en:
<http://repositorio.ug.edu.ec/bitstream/redug/20336/1/TESIS%20FINAL%20LEMA-REGATTO.pdf>
- Rivera, S. (2019). *La calidad de servicio y la satisfacción de los clientes de la empresa Greenandes Ecuador*. Universidad Católica de Ecuador.
- Sandoval, A. (2015). *Análisis y diseño de un Framework JavaScript basado en los estándares de la W3C para la implementación en Front-End de Juliaca.com*. Revisado el 11 de febrero del 2019. Disponible en:
<http://repositorio.uancv.edu.pe/bitstream/handle/UANCV/747/TESIS%2041440198%20%26%2002549288.pdf?sequence=1&isAllowed=y>
- Valbuena, M. (2014). *Guía comparativa de Frameworks para los lenguajes HTML 5, CSS y JavaScript para el desarrollo de aplicaciones Web*. Revisado el 25 de marzo del 2018. Disponible en:
<http://repositorio.utp.edu.co/dspace/handle/11059/4577>
- Valdivia J. (2016). *Modelo de procesos para el desarrollo del front-end de aplicaciones web*. Revisado el 14 de diciembre del 2018. Disponible en:
<https://dialnet.unirioja.es/descarga/articulo/6043088.pdf>

ANEXO

Anexo 1
MATRIZ DE CONSISTENCIA

PROBLEMA GENERAL	OBJETIVO GENERAL	HIPOTESIS GENERAL	VARIABLES	INDICADORES	METODOLOGIA
¿De qué manera la creación del framework "XRL8 bajo la tecnología css, agilizará el desarrollo frontend"?	Crear un framework bajo la tecnología OOCSS para la construcción en cascadas los estilos CSS	Mi presente trabajo no cuenta con una hipótesis porque es de tipo aplicativo, asimismo es será de un uso global, por ende no respeta una población de estudio	Framework	Librería de código CSS	OCCS es una metodología orientada a objetos
Carencia de significado, poco intuitivo, escasos efectos css, clases 3D no implementadas	Diseñar el framework "XRL8" como una estructura básica para agilizar y optimizar su velocidad en el usuario final. Implementar Clases básicas en el desarrollo del framework para mejorar su compatibilidad en los distintos navegadores.		Agilizar Desarrollo frontend	Agilizar la interfaz gráfica, así como una mejor experiencia de usuario	Se reducirá el tiempo de procesamiento de los navegadores usando pre-procesadores como sass, less o stylus