Universidad Nacional Mayor de San Marcos

Universidad del Perú. Decana de América

Dirección General de Estudios de Posgrado
Facultad de Ingeniería de Sistemas e Informática
Unidad de Posgrado

Framework para la automatización del maquetado de páginas web aplicando reconocimiento de voz

TESIS

Para optar el Grado Académico de Magíster en Ingeniería de Sistemas e Informática con mención en Ingeniería de Software

AUTOR

Julio Cesar PRUDENCIO NIEVES

ASESOR

Dra. Lenis Rossi WONG PORTILLO

Lima, Perú

2022

Reconocimiento - No Comercial - Compartir Igual - Sin restricciones adicionales

https://creativecommons.org/licenses/by-nc-sa/4.0/

Usted puede distribuir, remezclar, retocar, y crear a partir del documento original de modo no comercial, siempre y cuando se dé crédito al autor del documento y se licencien las nuevas creaciones bajo las mismas condiciones. No se permite aplicar términos legales o medidas tecnológicas que restrinjan legalmente a otros a hacer cualquier cosa que permita esta licencia.

Referencia bibliográfica

Prudencio, J. (2022). *Framework para la automatización del maquetado de páginas web aplicando reconocimiento de voz*. [Tesis de maestría, Universidad Nacional Mayor de San Marcos, Facultad de Ingeniería de Sistemas e Informática, Unidad de Posgrado]. Repositorio institucional Cybertesis UNMSM.

Metadatos complementarios

Datos de autor	
Nombres y apellidos	Julio Cesar Prudencio Nieves
Tipo de documento de identidad	DNI
Número de documento de identidad	46610089
URL de ORCID	https://orcid.org/0000-0002-2416-9121
Datos de asesor	
Nombres y apellidos	Lenis Rossi Wong Portillo
Tipo de documento de identidad	DNI
Número de documento de identidad	10438282
URL de ORCID	https://orcid.org/0000-0002-5032-3233
Datos del jurado	
Presidente del jurado	
Nombres y apellidos	Cayo Víctor León Fernández
Tipo de documento	DNI
Número de documento de identidad	07001405
Miembro del jurado 1	
Nombres y apellidos	José Alfredo Herrera Quispe
Tipo de documento	DNI
Número de documento de identidad	40362859
Miembro del jurado 2	
Nombres y apellidos	Fany Yexenia Sobero Rodríguez
Tipo de documento	DNI
Número de documento de identidad	20120467
Miembro del jurado 3	
Nombres y apellidos	Lenis Rossi Wong Portillo
Tipo de documento	DNI

Número de documento de identidad	10438282
Datos de investigación	
Línea de investigación	C.0.3.22. Ingeniería de software
Grupo de investigación	No aplica
Agencia de financiamiento	Sin financiamiento
Ubicación geográfica de la investigación	Edificio: Facultad de Ingeniería de Sistemas e Informática, Universidad Nacional Santiago Antúnez de Mayolo País: Perú Departamento: Ancash Provincia: Huaraz Distrito: Independencia Urbanización: Shancayan Calle: Av. Universitaria N°115 Latitud: -9.51806 Longitud: -77.52403
Año o rango de años en que se realizó la investigación	Agosto 2021 - octubre 2021
URL de disciplinas OCDE	Ingeniería de sistemas y comunicaciones https://purl.org/pe-repo/ocde/ford#2.02.04

UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS
Universidad del Perú. Decana de América
Facultad de Ingeniería de Sistemas e Informática
Vicedecanato de Investigación y Posgrado
Unidad de Posgrado

ACTA DE SUSTENTACIÓN VIRTUAL DE TESIS PARA OPTAR EL GRADO ACADÉMICO DE MAGÍSTER EN INGENIERÍA DE SISTEMAS E INFORMÁTICA CON MENCIÓN EN INGENIERÍA DE SOFTWARE

A los veintidós (22) días del mes de setiembre de 2022, siendo las 17:57 horas, se reunieron en la sala virtual ***https://meet.google.com/tkm-ndcd-khf*** *el Jurado de Tesis conformado por los siguientes docentes:*

Dr. Cayo Víctor León Fernández (Presidente)
Dr. José Alfredo Herrera Quispe (Miembro)
Mg. Fany Yexenia Sobero Rodríguez (Miembro)
Dra. Lenis Rossi Wong Portillo (Miembro Asesor)

Se inició la Sustentación invitando al candidato a Magíster ***JULIO CESAR PRUDENCIO NIEVES,*** *para que realice la exposición oral y virtual de la tesis para optar el Grado Académico de Magíster en Ingeniería de Sistemas e Informática con mención en Ingeniería de Software, siendo la Tesis intitulada:*

"FRAMEWORK PARA LA AUTOMATIZACIÓN DEL MAQUETADO DE PÁGINAS WEB APLICANDO RECONOCIMIENTO DE VOZ"

Concluida la exposición, los miembros del Jurado de Tesis procedieron a formular sus preguntas que fueron absueltas por el graduando; acto seguido se procedió a la evaluación correspondiente, habiendo obtenido la siguiente calificación:

…….…… DIECISEIS (16) BUENO……..

Por tanto, el presidente del Jurado, de acuerdo con el Reglamento General de Estudios de Posgrado, otorga al Bachiller ***JULIO CESAR PRUDENCIO NIEVES*** *el Grado Académico de Magíster en Ingeniería de Sistemas e Informática con mención en Ingeniería de Software.*

Siendo las. 19.10 horas, el presidente del Jurado de Tesis, da por concluido el acto académico de Sustentación de Tesis.

Dr. Cayo Víctor León Fernández
(Presidente)

Dr. José Alfredo Herrera Quispe
(Miembro)

Mg. Fany Yexenia Sobero Rodríguez
(Miembro)

Dra. Lenis Rossi Wong Portillo
(Miembro Asesor)

Firmado digitalmente por LEON FERNANDEZ Cayo Victor FAU 20148092282 soft
Motivo: Soy el autor del documento
Fecha: 18.07.2022 14:47:18 -05:00

Universidad del Perú. Decana de América
Facultad de Ingeniería de Sistemas e Informática
Vicedecanato de Investigación y Posgrado
Unidad de Posgrado

Lima, 18 de Julio del 2022

INFORME N° 000008-2022-UPG-VDIP-FISI/UNMSM

INFORME DE EVALUACIÓN DE ORIGINALIDAD

1. Autoridad Académica que emite el Informe de Originalidad:	*Director de la Unidad de Posgrado*
2. Apellidos y Nombres de la autoridad académica:	*Dr. Cayo Víctor León Fernández*
3. Operador del programa informático de similitudes:	*Dr. Cayo Víctor León Fernández*
4. Documento evaluado:	*Tesis para Posgrado* *Título: "Framework para la automatización del maquetado de páginas web aplicando reconocimiento de voz"*
5. Autor del documento:	*Julio Cesar Prudencio Nieves*
6. Fecha de recepción de documento	*18/07/2022*
7. Fecha de aplicación del programa detector de similitudes:	*18/07/2022*
8. Software utilizado:	*Turnitin*
9. Configuración del programa detector de similitudes:	*i. Excluye textos entrecomillados: SI* *ii. Excluye biografías: SI* *iii. Excluye cadenas menores a 40 palabras: SI* *iv. Otro criterio (especificar): NO*
10. Porcentaje de similitudes según programa detector de similitudes	*Siete por ciento (7%)*
11. Fuentes originales de las similitudes encontradas	*Se adjuntan en seis (06) fojas al presente informe*
12. Observaciones:	*Ninguna*
13. Calificación de originalidad *i. Documento cumple criterios de originalidad, sin observaciones.* *ii. Documento cumple criterior de originaldiad, con observaciones.* *iii. Documento no cumple criterios de originalidad.*	*Documento cumple criterio de originalidad, sin observaciones.*
14. Fecha del Informe:	*18/07/2022*

DR. CAYO VICTOR LEON FERNANDEZ
DIRECTOR DE LA UNIDAD DE POSGRADO(e)

CLF/rda

Esta es una copia auténtica imprimible de un documento electrónico archivado por la Universidad Nacional Mayor de San Marcos, aplicando lo dispuesto por el Art. 25 de D.S. 070-2013-PCM y la Tercera Disposición Complementaria Final del D.S. 026- 2016-PCM. Su autenticidad e integridad pueden ser contrastadas a través de la siguiente dirección web: **https://spsgd.unmsm.edu.pe/verifica/inicio.do** e ingresando el siguiente código de verificación: **BKNVQIA**

Universidad del Perú. Decana de América
Facultad de Ingeniería de Sistemas e Informática
Vicedecanato de Investigación y Posgrado
Unidad de Posgrado

Framework para la automatización del maquetado de páginas web aplicando reconocimiento de voz

por Julio Cesar Prudencio Nieves

Fecha de entrega: 18-jul-2022 01:06a.m. (UTC-0500)
Identificador de la entrega: 1872018871
Nombre del archivo: 18_7_Julio_Cesar_PRUDENCIO_NIEVES_PROYECTO_DE_TESIS_8.pdf (8.25M)
Total de palabras: 66959
Total de caracteres: 362009

Esta es una copia auténtica imprimible de un documento electrónico archivado por la Universidad Nacional Mayor de San Marcos, aplicando lo dispuesto por el Art. 25 de D.S. 070-2013-PCM y la Tercera Disposición Complementaria Final del D.S. 026- 2016-PCM. Su autenticidad e integridad pueden ser contrastadas a través de la siguiente dirección web: **https://spsgd.unmsm.edu.pe/verifica/inicio.do** e ingresando el siguiente código de verificación: **BKNVQIA**

DEDICATORIA

Dedico esta tesis a mis padres y seres queridos quienes me enseñaron a amar y buscar constantemente el conocimiento, dedicaron su tiempo y esfuerzo con amor y paciencia, apoyándome para lograr finalizar los estudios de posgrado.

Dedico también este trabajo a los docentes de la Unidad de Posgrado de la Facultad de Ingeniería de Sistemas e Informática de la Universidad Nacional Mayor de San Marcos, UNMSM, por brindarnos sus conocimientos y enseñarnos los valores, habilidades y capacidades propias de un Maestro en Ingeniería de Sistemas e Informática con Mención en Ingeniería del Software.

AGRADECIMIENTO

Agradezco al director de la Escuela de Ingeniería de Sistemas e Informática de la Universidad Nacional Santiago Antúnez de Mayolo (UNASAM), Dr. Luis Ruperto Alvarado Cáceres, por brindarme todas las facilidades para poder desarrollar esta tesis. Asimismo, agradezco a todos los alumnos del noveno ciclo de la carrera de Ingeniería de Sistemas e Informática que participaron en la realización de la presente tesis. Finalmente, mi agradecimiento a la Dr. Lenis Wong Portillo por su asesoría constante.

ÍNDICE GENERAL

ÍNDICE GENERAL ..... **III**

ÍNDICE DE TABLAS ..... **VII**

ÍNDICE DE FIGURAS ..... **IX**

RESUMEN ..... **XII**

ABSTRACT ..... **XIII**

CAPÍTULO I: INTRODUCCIÓN **1**

1.1. SITUACIÓN PROBLEMÁTICA 1
1.2. FORMULACIÓN DEL PROBLEMA 2
1.3. JUSTIFICACIÓN DE LA INVESTIGACIÓN 2
1.4. MOTIVACIÓN 7
1.5. OBJETIVOS 8
1.5.1. Objetivo general *8*
1.5.2. Objetivos específicos *8*

CAPÍTULO II: MARCO TEÓRICO **9**

2.1. ANTECEDENTES DE INVESTIGACIÓN 9
2.1.1. Diseño de asistente de voz basado en web para personas con visión subnormal (Collado & Aparicio, 2017) *10*
2.1.2. Reconocimiento de comandos de voz en español orientado al control de una silla de ruedas (Gil, Castillo, & Flórez, 2016) *12*
2.1.3. Desarrollo de una aplicación para la web utilizando el Web Speech API (Roig, 2019) *13*
2.1.4. Aplicación multimedia para el entrenamiento en la certificación TOEFL mediante reconocimiento de voz (Hernández & Lemuz, 2016) *15*
2.1.5. Editor web visual para HTML, CSS y JavaScript de apoyo a la docencia (Jaimez & Vargas, 2017) *16*
2.1.6. El maquetado a base de scripts y hojas de estilo en cascada (CSS) y su incidencia en la optimización de un sitio web (Pilamunga, 2012) *18*
2.1.7. The challenge of web design guidelines: Investigating issues of awareness, interpretation, and efficacy (Szigeti, 2012) *21*
2.2. BASES TEÓRICAS 25
2.2.1. Framework CSS *25*
2.2.2. Reconocimiento de voz *25*
2.2.3. Control por comandos de voz *26*
2.2.4. Web Speech API *27*
2.2.5. Maquetado o diseño web *28*

2.2.6. Indicadores de maquetado o diseño web *29*
2.2.7. Análisis de validez por juicio de expertos *35*
2.2.8. Muestreo estadístico *37*
2.2.9. Metodología SCRUM *38*
2.2.10. Lenguaje de modelado unificado - UML *39*
2.2.11. Hypertext Markup Language - HTML *43*
2.2.12. Cascading Style Sheets - CSS *43*
2.2.13. JavaScript *44*
2.2.14. Hypertext Preprocessor - PHP *45*
2.2.15. JavaScript Object Notation - JSON *45*

CAPÍTULO III: METODOLOGÍA 46

3.1. TIPO Y DISEÑO DE LA INVESTIGACIÓN 46
3.1.1. Tipo de investigación *46*
3.1.2. Diseño de la investigación *46*
3.2. UNIDAD DE ANÁLISIS 47
3.3. POBLACIÓN DE ESTUDIO 47
3.4. TAMAÑO DE LA MUESTRA 47
3.5. SELECCIÓN DE LA MUESTRA 47
3.5.1. Criterios de inclusión *48*
3.5.2. Criterio de exclusión *48*
3.6. TÉCNICA DE RECOLECCIÓN DE DATOS 49
3.7. ANÁLISIS E INTERPRETACIÓN DE LA INFORMACIÓN 53

CAPÍTULO IV: FRAMEWORK PROPUESTO 54

4.1. INTERFACE MFML 55
4.1.1. Modelo matemático *56*
4.1.2. Teoría de nomenclaturas *66*
4.1.3. Interface MFML: Modelo Físico – Modelo Lógico *83*
4.2. COMANDOS DE VOZ 90
4.2.1. Comandos propiamente *91*
4.2.2. Comandos auxiliares *93*
4.2.3. Comandos de texto *96*
4.2.4. Comandos de selector *97*
4.2.5. Comandos de declaración *99*
4.2.6. Comandos de reglas-AT *99*
4.2.7. Comandos de descriptores *102*
4.3. CONSTRUCCIÓN DEL FRAMEWORK WEB CSS 3 CON SCRUM Y UML 103
4.3.1. Roles de scrum *103*
4.3.2. Historias de usuario *104*
4.3.3. Pila del producto (Product backlog) *109*

4.3.4. Estimación de la pila del producto ... *109*
4.3.5. Primer sprint ... *110*
4.3.6. Segundo sprint ... *154*
4.3.7. Arquitectura general del framework CSS 3 ... *171*
4.4. FRAMEWORK WEB CSS 3 CON RECONOCIMIENTO DE VOZ ... 178
4.4.1. Paso 1 – Cargar .html ... *178*
4.4.2. Paso 2 – Importar .XML o Crear .CSS ... *181*
4.4.3. Ejecutar Código – Ocultar o Mostrar el framework ... *187*
4.4.4. Generar MFML – Exportar archivos .XML y .CSS ... *189*
4.4.6. Código Fuente ... *201*

CAPÍTULO V: VALIDACIÓN Y RESULTADOS ... 205

5.1. VALIDACIÓN ... 205
5.1.1. Escenario 1: obtención de datos del pre-test ... *206*
5.1.2. Escenario 2: capacitación a los participantes ... *207*
5.1.3. Escenario 3: obtención de datos del post-test ... *207*
5.2. RESULTADOS DESCRIPTIVOS ... 208
5.2.1. Maquetado de páginas web ... *208*
5.2.2. Dimensión animación del maquetado de páginas web ... *209*
5.2.3. Dimensión botones del maquetado de páginas web ... *210*
5.2.4. Dimensión contenido del maquetado de páginas web ... *211*
5.2.5. Dimensión fondos del maquetado de páginas web ... *212*
5.2.6. Dimensión imágenes del maquetado de páginas web ... *213*
5.2.7. Dimensión tipografías del maquetado de páginas web ... *214*
5.3. RESULTADOS SEGÚN DIMENSIONES DE MAQUETADO WEB ... 215
5.3.1. Resultados para el objetivo general ... *216*
5.3.2. Resultados para las dimensiones de maquetado web ... *217*

CAPÍTULO VI: DISCUSIÓN ... 226

6.1. DISCUSIÓN PARA EL OBJETIVO GENERAL ... 226
6.2. DISCUSIÓN PARA LAS DIMENSIONES DE MAQUETADO WEB ... 229
6.2.1. Discusión del maquetado de animaciones ... *229*
6.2.2. Discusión del maquetado de botones ... *229*
6.2.3. Discusión del maquetado de contenido ... *230*
6.2.4. Discusión del maquetado de fondos ... *231*
6.2.5. Discusión del maquetado de imágenes ... *231*
6.2.6. Discusión del maquetado de tipografías ... *232*

CAPÍTULO VII: CONCLUSIONES Y TRABAJOS FUTUROS ... 233

7.1. CONCLUSIONES ... 233
7.1.1. Conclusión general ... *233*

7.1.2. Conclusiones específicas *234*
7.2. TRABAJOS FUTUROS 236

REFERENCIAS BIBLIOGRÁFICAS 237

ANEXOS 241

ÍNDICE DE TABLAS

TABLA 1 **CATEGORÍAS PRINCIPALES DE SITIOS WEB** 3
TABLA 2 **CAPÍTULOS Y CANTIDAD DE INDICADORES PARA EL DESARROLLO DE UN SITIO WEB** 34
TABLA 3 **COEFICIENTES V DE AIKEN PARA 10 EVALUADORES** 36
TABLA 4 **LISTADO DE DIMENSIONES E INDICADORES DEL CUESTIONARIO** 49
TABLA 5 **VALORES DE LA ESCALA DE LIKERT UTILIZADOS EN EL CUESTIONARIO** 52
TABLA 6 **ANÁLISIS DE VALIDEZ POR JUICIO DE EXPERTOS** 53
TABLA 7 **RECONOCIMIENTO DE VOZ - COMANDOS PROPIAMENTE** 91
TABLA 8 **RECONOCIMIENTO DE VOZ - COMANDOS AUXILIARES** 94
TABLA 9 **RECONOCIMIENTO DE VOZ - COMANDOS DE SELECTOR** 97
TABLA 10 **RECONOCIMIENTO DE VOZ - COMANDOS DE DECLARACIÓN** 99
TABLA 11 **RECONOCIMIENTO DE VOZ - COMANDOS DE REGLAS-AT** 100
TABLA 12 **RECONOCIMIENTO DE VOZ - COMANDOS DE DESCRIPTORES** 102
TABLA 13 **SCRUM - ROLES DE SCRUM DESEMPEÑADOS POR EL AUTOR** 103
TABLA 14 **SCRUM - HISTORIA DE USUARIO CARGAR ARCHIVO HTML** 104
TABLA 15 **SCRUM - HISTORIA DE USUARIO IMPORTAR ARCHIVO XML** 105
TABLA 16 **SCRUM - HISTORIA DE USUARIO CREAR ARCHIVO CSS** 105
TABLA 17 **SCRUM - HISTORIA DE USUARIO PRONUNCIACIÓN DE COMANDOS DE VOZ** 106
TABLA 18 **SCRUM - HISTORIA DE USUARIO OCULTAR FRAMEWORK** 107
TABLA 19 **SCRUM - HISTORIA DE USUARIO MOSTRAR FRAMEWORK** 107
TABLA 20 **SCRUM - HISTORIA DE USUARIO EXPORTAR ARCHIVOS .XML Y .CSS** 108
TABLA 21 **SCRUM - HISTORIA DE USUARIO DESCARGAR CÓDIGO FUENTE** 109
TABLA 22 **SCRUM - PILA DEL PRODUCTO (PRODUCT BACKLOG)** 109
TABLA 23 **SCRUM - ESTIMACIÓN DE LA PILA DEL PRODUCTO** 110
TABLA 24 **SCRUM - PLANIFICACIÓN DEL PRIMER SPRINT** 111
TABLA 25 **UML - CASOS DE USO DEL PRIMER SPRINT** 112
TABLA 26 **UML - CLASES DEL PRIMER SPRINT** 113
TABLA 27 **UML - ESPECIFICACIÓN CASO DE USO CARGAR HTML** 116
TABLA 28 **UML - ESPECIFICACIÓN CASO DE USO IMPORTAR XML** 118
TABLA 29 **UML - ESPECIFICACIÓN CASO DE USO CREAR CSS** 121
TABLA 30 **UML - ESPECIFICACIÓN CASO DE USO PRONUNCIAR COMANDOS DE VOZ** 124
TABLA 31 **UML - ESPECIFICACIÓN CASO DE USO GESTIONAR EDICION TEXTO** 129
TABLA 32 **UML - ESPECIFICACIÓN CASO DE USO GESTIONAR SELECTOR CSS** 133
TABLA 33 **UML - ESPECIFICACIÓN CASO DE USO GESTIONAR DECLARACION CSS** 136
TABLA 34 **UML - ESPECIFICACIÓN CASO DE USO GESTIONAR REGLAAT CSS** 140
TABLA 35 **UML - ESPECIFICACIÓN CASO DE USO GESTIONAR DESCRIPTOR CSS** 144
TABLA 36 **UML - ESPECIFICACIÓN CASO DE USO GESTIONAR COMENTARIO CSS** 148
TABLA 37 **SCRUM - PILA DEL PRIMER SPRINT** 151
TABLA 38 **SCRUM - PLANIFICACIÓN DEL SEGUNDO SPRINT** 155

TABLA 39 **UML - CASOS DE USO DEL SEGUNDO SPRINT** 155
TABLA 40 **UML - CLASES DEL SEGUNDO SPRINT** 156
TABLA 41 **UML - ESPECIFICACIÓN CASO DE USO OCULTAR FRAMEWORK** 158
TABLA 42 **UML - DIAGRAMA DE ACTIVIDAD DEL CASO DE USO OCULTAR FRAMEWORK.** 160
TABLA 43 **UML - DIAGRAMA DE SECUENCIA DEL CASO DE USO OCULTAR FRAMEWORK.** 161
TABLA 44 **UML - ESPECIFICACIÓN CASO DE USO MOSTRAR FRAMEWORK** 161
TABLA 45 **UML - DIAGRAMA DE SECUENCIA DEL CASO DE USO MOSTRAR FRAMEWORK.** 163
TABLA 46 **UML - ESPECIFICACIÓN CASO DE USO EXPORTAR ARCHIVOS .XML Y .CSS** 163
TABLA 47 **UML - ESPECIFICACIÓN CASO DE USO DESCARGAR CÓDIGO FUENTE** 166
TABLA 48 **SCRUM - PILA DEL SEGUNDO SPRINT** 168
TABLA 49 **ESTADÍSTICAS DESCRIPTIVAS DE LOS PUNTAJES DEL MAQUETADO WEB EN EL PRE-TEST Y POST-TEST** 216
TABLA 50 **PRUEBA T DE IGUALDAD DE MEDIAS DE LOS PUNTAJES DEL MAQUETADO WEB EN EL PRE-TEST Y POST-TEST** 217
TABLA 51 **ESTADÍSTICAS DESCRIPTIVAS DE LOS PUNTAJES DE LA DIMENSIÓN ANIMACIÓN DEL MAQUETADO WEB EN EL PRE-TEST Y POST-TEST** 218
TABLA 52 **PRUEBA T DE IGUALDAD DE MEDIAS DE LOS PUNTAJES DE LA DIMENSIÓN ANIMACIÓN DEL MAQUETADO WEB EN EL PRE-TEST Y POST-TEST** 218
TABLA 53 **ESTADÍSTICAS DESCRIPTIVAS DE LOS PUNTAJES DE LA DIMENSIÓN BOTONES DEL MAQUETADO WEB EN EL PRE-TEST Y POST-TEST** 219
TABLA 54 **PRUEBA T DE IGUALDAD DE MEDIAS DE LOS PUNTAJES DE LA DIMENSIÓN BOTONES DEL MAQUETADO WEB EN EL PRE-TEST Y POST-TEST** 219
TABLA 55 **ESTADÍSTICAS DESCRIPTIVAS DE LOS PUNTAJES DE LA DIMENSIÓN CONTENIDO DEL MAQUETADO WEB EN EL PRE-TEST Y POST-TEST** 220
TABLA 56 **PRUEBA T DE IGUALDAD DE MEDIAS DE LOS PUNTAJES DE LA DIMENSIÓN CONTENIDO DEL MAQUETADO WEB EN EL PRE-TEST Y POST-TEST** 220
TABLA 57 **ESTADÍSTICAS DESCRIPTIVAS DE LOS PUNTAJES DE LA DIMENSIÓN FONDO DEL MAQUETADO WEB EN EL PRE-TEST Y POST-TEST** 221
TABLA 58 **PRUEBA T DE IGUALDAD DE MEDIAS DE LOS PUNTAJES DE LA DIMENSIÓN FONDO DEL MAQUETADO WEB EN EL PRE-TEST Y POST-TEST** 222
TABLA 59 **ESTADÍSTICAS DESCRIPTIVAS DE LOS PUNTAJES DE LA DIMENSIÓN IMÁGENES DEL MAQUETADO WEB EN EL PRE-TEST Y POST-TEST** 222
TABLA 60 **PRUEBA T DE IGUALDAD DE MEDIAS DE LOS PUNTAJES DE LA DIMENSIÓN IMÁGENES DEL MAQUETADO WEB EN EL PRE-TEST Y POST-TEST** 223
TABLA 61 **ESTADÍSTICAS DESCRIPTIVAS DE LOS PUNTAJES DE LA DIMENSIÓN TIPOGRAFÍAS DEL MAQUETADO WEB EN EL PRE-TEST Y POST-TEST** 224
TABLA 62 **PRUEBA T DE IGUALDAD DE MEDIAS DE LOS PUNTAJES DE LA DIMENSIÓN TIPOGRAFÍAS DEL MAQUETADO WEB EN EL PRE-TEST Y POST-TEST** 224

ÍNDICE DE FIGURAS

FIGURA 1 **ESTADÍSTICAS DE USO DEL SERVIDOR WEB APACHE.** 4
FIGURA 2 **ESTADÍSTICAS DE USO DEL LENGUAJE DE PROGRAMACIÓN WEB PHP.** 5
FIGURA 3 **CRECIMIENTO DEL USO DE LA TECNOLOGÍA DE RECONOCIMIENTO DE VOZ.** 6
FIGURA 4 **PORCENTAJE DE APLICACIÓN DE INDICADORES DE DISEÑO WEB.** 22
FIGURA 5 **PROMEDIO DE APLICACIÓN DE INDICADORES DE DISEÑO WEB EN SITIOS WEB DE SALUD.** 23
FIGURA 6 **PORCENTAJES DE LA FRECUENCIA DE APLICACIÓN DE INDICADORES DE DISEÑO WEB.** 23
FIGURA 7 **MEDIOS DE TRANSMISIÓN PRINCIPALES DE LOS INDICADORES DE DISEÑO WEB.** 24
FIGURA 8 **WSDM: WEB SITE DESIGN METHOD - AUDIENCE-DRIVEN APPROACH.** 30
FIGURA 9 **FASE DE IMPLEMENTACIÓN DEL DISEÑO - METODOLOGÍA WSDM.** 32
FIGURA 10 **INDICADORES DE LA CATEGORÍA INTERFACE Y ESTILO.** 33
FIGURA 11 **COEFICIENTE V DE AIKEN.** 36
FIGURA 12 **MODELO MATEMÁTICO - INTERFACE MFML.** 58
FIGURA 13 **NOMENCLATURA REGLA-AT LINEAL (J = 1).** 69
FIGURA 14 **NOMENCLATURA REGLA-AT ANIDADA (J > 1).** 75
FIGURA 15 **NOMENCLATURA SELECTOR LINEAL.** FUENTE: ELABORACIÓN PROPIA 80
FIGURA 16 **NOMENCLATURA COMENTARIO.** 82
FIGURA 17 **INTERFACE MFML: MODELO FÍSICO - MODELO LÓGICO.** 84
FIGURA 18 **RECONOCIMIENTO DE VOZ - COMANDOS DE TEXTO.** 96
FIGURA 19 **UML - DIAGRAMA DE CASOS DE USO DEL PRIMER SPRINT.** 113
FIGURA 20 **UML - DIAGRAMA DE CLASES DEL PRIMER SPRINT.** FUENTE: ELABORACIÓN PROPIA ... 115
FIGURA 21 **UML - DIAGRAMA DE ACTIVIDAD DEL CASO DE USO CARGAR HTML.** 117
FIGURA 22 **UML - DIAGRAMA DE SECUENCIA DEL CASO DE USO CARGAR HTML.** 118
FIGURA 23 **UML - DIAGRAMA DE ACTIVIDAD DEL CASO DE USO IMPORTAR XML.** 120
FIGURA 24 **UML - DIAGRAMA DE SECUENCIA DEL CASO DE USO IMPORTAR XML.** 121
FIGURA 25 **UML - DIAGRAMA DE ACTIVIDAD DEL CASO DE USO CREAR CSS.** 123
FIGURA 26 **UML - DIAGRAMA DE SECUENCIA DEL CASO DE USO CREAR CSS.** 124
FIGURA 27 **UML - DIAGRAMA DE ACTIVIDAD DEL CASO DE USO PRONUNCIAR COMANDOS DE VOZ.** 127
FIGURA 28 **UML - DIAGRAMA DE SECUENCIA DEL CASO DE USO PRONUNCIAR COMANDOS DE VOZ.** FUENTE: ELABORACIÓN PROPIA 128
FIGURA 29 **UML - DIAGRAMA DE ACTIVIDAD DEL CASO DE USO GESTIONAR EDICION TEXTO.** 132
FIGURA 30 **UML - DIAGRAMA DE SECUENCIA DEL CASO DE USO GESTIONAR EDICION TEXTO.** 133
FIGURA 31 **UML - DIAGRAMA DE SECUENCIA DEL CASO DE USO GESTIONAR SELECTOR CSS.** 136
FIGURA 32 **UML - DIAGRAMA DE SECUENCIA DEL CASO DE USO GESTIONAR DECLARACION CSS.** 140
FIGURA 33 **UML - DIAGRAMA DE ACTIVIDAD DEL CASO DE USO GESTIONAR REGLAAT CSS.** 143
FIGURA 34 **UML - DIAGRAMA DE SECUENCIA DEL CASO DE USO GESTIONAR REGLAAT CSS.** 144
FIGURA 35 **UML - DIAGRAMA DE SECUENCIA DEL CASO DE USO GESTIONAR DESCRIPTOR CSS.** .. 148
FIGURA 36 **UML - DIAGRAMA DE ACTIVIDAD DEL CASO DE USO GESTIONAR COMENTARIO CSS.** . 150

FIGURA 37 **UML - DIAGRAMA DE SECUENCIA DEL CASO DE USO GESTIONAR COMENTARIO CSS.** 151
FIGURA 38 **SCRUM - SEGUIMIENTO DE HISTORIAS DE USUARIO DEL PRIMER SPRINT.** 153
FIGURA 39 **SCRUM - GRÁFICO BURNDOWN DEL PRIMER SPRINT.** 154
FIGURA 40 **UML - DIAGRAMA DE CASOS DE USO DEL SEGUNDO SPRINT.** 156
FIGURA 41 **UML - DIAGRAMA DE CLASES DEL SEGUNDO SPRINT.** 157
FIGURA 42 **UML - DIAGRAMA DE SECUENCIA DEL CASO DE USO EXPORTAR ARCHIVOS .XML Y .CSS.** 166
FIGURA 43 **UML - DIAGRAMA DE ACTIVIDAD DEL CASO DE USO DESCARGAR CODIGO FUENTE.** .. 167
FIGURA 44 **UML - DIAGRAMA DE SECUENCIA DEL CASO DE USO DESCARGAR CÓDIGO FUENTE.** .. 168
FIGURA 45 **SCRUM - SEGUIMIENTO DE HISTORIAS DE USUARIO DEL SEGUNDO SPRINT.** 170
FIGURA 46 **SCRUM - GRÁFICO BURNDOWN DEL SEGUNDO SPRINT.** 170
FIGURA 47 **UML - DIAGRAMA DE PAQUETES.** 171
FIGURA 48 **UML - DIAGRAMA DE CASOS DE USO.** 172
FIGURA 49 **UML - DIAGRAMA DE CLASES.** FUENTE: ELABORACIÓN PROPIA 173
FIGURA 50 **UML - DIAGRAMA DE COMPONENTES.** FUENTE: ELABORACIÓN PROPIA 175
FIGURA 51 **UML - DIAGRAMA DE DESPLIEGUE.** FUENTE: ELABORACIÓN PROPIA 177
FIGURA 52 **FRAMEWORK WEB CSS 3 - IMPORTAR ARCHIVO .CSS A GENERAR.** 179
FIGURA 53 **FRAMEWORK WEB CSS 3 - SECCIÓN HEAD ARCHIVO .HTML.** 179
FIGURA 54 **FRAMEWORK WEB CSS 3 - BOTÓN SELECCIONAR ARCHIVO .HTML.** 180
FIGURA 55 **FRAMEWORK WEB CSS 3 - SELECCIONAR ARCHIVO .HTML.** 180
FIGURA 56 **FRAMEWORK WEB CSS 3 - ARCHIVO .HTML CARGADO.** 181
FIGURA 57 **FRAMEWORK WEB CSS 3 - BOTÓN SELECCIONAR ARCHIVO .XML.** 182
FIGURA 58 **FRANEWORK WEB CSS 3 - SELECCIONAR ARCHIVO .XML.** 183
FIGURA 59 **FRAMEWORK WEB CSS 3 - MODELO FÍSICO GENERADO - ARCHIVO .XML CARGADO.** . 184
FIGURA 60 **FRAMEWORK WEB CSS 3 - ASIGNAR NOMBRE AL ARCHIVO .CSS.** 185
FIGURA 61 **FRAMEWORK WEB CSS 3 - INSERCIÓN DE ELEMENTOS POR COMANDOS DE VOZ.** 186
FIGURA 62 **FRAMEWORK WEB CSS 3 - BOTÓN OCULTAR FRAMEWORK.** 187
FIGURA 63 **FRAMEWORK WEB CSS 3 - BOTÓN MOSTRAR FRAMEWORK.** 188
FIGURA 64 **FRAMEWORK WEB CSS 3 - BARRA DE COMANDOS DE VOZ.** 189
FIGURA 65 **FRAMEWORK WEB CSS 3 - EXPORTACIÓN MODELO LÓGICO (ARCHIVOS .XML Y .CSS).** 190
FIGURA 66 **FRAMEWORK WEB CSS 3 - ARCHIVOS DESCARGADOS .XML Y .CSS.** 190
FIGURA 67 **FRAMEWORK WEB CSS 3 - ARCHIVO DESCARGADO .CSS.** 191
FIGURA 68 **FRAMEWORK WEB CSS 3 - ARCHIVO DESCARGADO .XML.** 192
FIGURA 69 **FRAMEWORK WEB CSS 3 - ARCHIVO EJEMPLODISENIO.HTML.** 193
FIGURA 70 **FRAMEWORK WEB CSS 3 - SECCIÓN HEAD DEL ARCHIVO EJEMPLODISENIO.HTML.** 194
FIGURA 71 **FRAMEWORK WEB CSS 3 - SELECCIONAR ARCHIVO EJEMPLODISENIO.HTML.** 194
FIGURA 72 **FRAMEWORK WEB CSS 3 - ARCHIVO EJEMPLODISENIO.HTML CARGADO.** 195
FIGURA 73 **FRAMEWORK WEB CSS 3 - ESCRIBIR NOMBRE EJEMPLODISENIO DE ARCHIVO .CSS.** ... 196

FIGURA 74 **FRAMEWORK WEB CSS 3 - MODELO FÍSICO FINAL ARCHIVO EJEMPLODISENIO.HTML.** 197
FIGURA 75 **FRAMEWORK WEB CSS 3 - ARCHIVO EJEMPLODISENIO.HTML MAQUETADO.** 197
FIGURA 76 **FRAMEWORK WEB CSS 3 - MODELO LÓGICO EJEMPLODISENIO DESCARGADO.** 198
FIGURA 77 **FRAMEWORK WEB CSS 3 - ARCHIVOS EJEMPLODISENIO .CSS / .XML DESCARGADOS.** . 198
FIGURA 78 **FRAMEWORK WEB CSS 3 - ARCHIVO EJEMPLODISENIO.CSS INTEGRADO EN EL PROYECTO.** 199
FIGURA 79 **FRAMEWORK WEB CSS 3 - CONTENIDO ARCHIVO EJEMPLODISENIO.XML.** 200
FIGURA 80 **FRAMEWORK WEB CSS 3 - CONTENIDO ARCHIVO EJEMPLODISENIO.CSS.** 201
FIGURA 81 **FRAMEWORK WEB CSS 3 - EXTRACCIÓN DEL ARCHIVO FRAMEWORKCSS3.RAR.** 202
FIGURA 82 **FRAMEWORK WEB CSS 3 - ARCHIVOS DENTRO DE LA CARPETA PUBLIC_HTML DE FRAMEWORKCSS3.** 203
FIGURA 83 **FRAMEWORK WEB CSS 3 - SUBIR CARPETA FRAMEWORKCSS3 AL SERVIDOR WEB.** 203
FIGURA 84 **FRAMEWORK WEB CSS 3 - RUTA HTTP LOCAL PARA EJECUCIÓN DEL FRAMEWORK.** .. 204
FIGURA 85 **PROCEDIMIENTO DE VALIDACIÓN DEL FRAMEWORK PROPUESTO.** 206
FIGURA 86 **NIVELES DEL MAQUETADO WEB EN EL PRE-TEST Y POST-TEST.** 208
FIGURA 87 **NIVELES DE LA DIMENSIÓN ANIMACIÓN DEL MAQUETADO WEB EN EL PRE-TEST Y POST-TEST.** 209
FIGURA 88 **NIVELES DE LA DIMENSIÓN BOTONES DEL MAQUETADO WEB EN EL PRE-TEST Y POST-TEST.** 210
FIGURA 89 **NIVELES DE LA DIMENSIÓN CONTENIDO DEL MAQUETADO WEB EN EL PRE-TEST Y POST-TEST.** 211
FIGURA 90 **NIVELES DE LA DIMENSIÓN FONDO DEL MAQUETADO WEB EN EL PRE-TEST Y POST-TEST.** 212
FIGURA 91 **NIVELES DE LA DIMENSIÓN IMÁGENES DEL MAQUETADO WEB EN EL PRE-TEST Y POST-TEST.** 213
FIGURA 92 **NIVELES DE LA DIMENSIÓN TIPOGRAFÍAS DEL MAQUETADO WEB EN EL PRE-TEST Y POST-TEST.** 214

RESUMEN

La presente tesis de maestría aplica la teoría de reconocimiento de voz sobre el maquetado web para desarrollar un framework que permita automatizar el maquetado web mediante el uso de las tecnologías CSS 3 y reconocimiento de voz. La tecnología de reconocimiento de voz se encarga de procesar y convertir una expresión hablada, captada como audio, a formato texto y para ello se ha usado la Web Speech API. Las expresiones convertidas a texto son llamadas comandos de voz y se encuentran definidas dentro de un vocabulario reconocido por el framework como la lista de acciones posibles a realizar. Por otro lado, CSS 3 es la tecnología que nos permite estilizar profesionalmente páginas o sistemas web en poco tiempo y con pocas líneas de código para desplegar un sitio web en cualquier tipo de dispositivo, reducir el tamaño del sitio y el tráfico entre el servidor y el dispositivo cliente.

El framework propuesto tiene la ventaja de usar comandos de voz CSS 3 dentro de un entorno web intuitivo, fácil, moderno e innovador que permite realizar vistas previas del maquetado según el avance, generar y descargar automáticamente las hojas de estilo CSS gestionadas mediante el uso del mouse y del teclado, así como de la voz, lo que representa una innovación del método tradicional de diseño web.

El objetivo de la tesis es determinar de qué manera el framework basado en reconocimiento de voz influye en la automatización del maquetado de páginas web. El framework ha sido validado con los alumnos del noveno ciclo de la Escuela de Ingeniería de Sistemas e Informática de la UNASAM. Los resultados demuestran que el framework CSS 3 mejora en un 38.59% el maquetado de páginas web en general, en un 41.46% el maquetado de animaciones, en un 39.22% el maquetado de botones, en un 32.33% el maquetado de contenido, en un 39.94% el maquetado de fondos, en un 41.04% el maquetado de imágenes y en un 39.28% el maquetado de tipografías de páginas web.

PALABRAS CLAVE: Reconocimiento de voz, Automatización de maquetado web, Web Speech API.

ABSTRACT

This master thesis applies the speech voice recognition theory to web design to develop a framework that allows automating the web layout through the use of CSS 3 and speech recognition technologies. The speech recognition technology is responsible for processing and converting a spoken expression, captured as audio, to text format and for this the Web Speech API has been used. Expressions converted to text are called voice commands and are defined within a vocabulary recognized by the framework as the list of possible actions to perform. On the other hand, CSS 3 is the technology that allow us to professionally stylize web pages or systems in a short time and with few code lines to display a web site on any type of device, reduce the site size and the traffic between the server and the client device.

The proposed framework has the advantage of use CSS 3 voice commands within a web intuitive, easy, modern and innovator environment that allow make design previews according the advance, automatically generate and download CSS styles sheets managed throught the use of mouse and keyboard, as well as the voice, which represent an innovation of the traditional method of design web.

This thesis objective is to determine how the framework based on speech voice recognition influences the automation of web page layout. The framework has been validated with the students of the ninth cycle of the School of Systems Engineering and Informatics of UNASAM. The results show that the CSS 3 framework improves the layout of web pages in general by 38.59%, the layout of animations by 41.46%, the layout of buttons by 39.22%, the layout of content by 32.33%, the layout of backgrounds by 39.94%, the layout of images by 41.04% and 39.28% the layout of web page fonts.

KEY WORDS: Speech voice recognition, Web design automatization, Web Speech API.

CAPÍTULO I: INTRODUCCIÓN

1.1. Situación Problemática

Hoy en día resulta imposible pensar en un mundo ajeno a la tecnología. La tecnología ha evolucionado hasta convertirse en parte esencial de nuestras vidas y absorbe cada vez más todas las actividades cotidianas: académicas, sociales, familiares, salud, negocios, gobierno, entre otros (Dentzel, 2013). La tecnología web, cumple un papel protagónico en el avance y expansión tecnológica del mundo, propiciando la integración digital a través de la World Wide Web (WWW[1]).

La World Wide Web (WWW) y la Internet son, posiblemente, uno de los inventos más importantes del siglo XX, pues permiten la integración entre las personas, acortando distancias sociales, reduciendo brechas culturales, permitiendo compartir conocimientos a nivel mundial, y hasta incluso promover el comercio internacional (Malone, 2013). Todo esto es posible gracias a la tecnología web que permite generar, difundir, gestionar y administrar la información a través de las páginas o sistemas web.

Las páginas web, permiten mostrar contenido estático, contenido que no es modificado por el usuario final, pero muestra información detallada sobre algún tópico en particular. Los sistemas web, en cambio, sí permiten interactuar con el sistema con fines de presentación o generación de nueva información sobre algún tópico en particular: académico, social, familiar, salud, negocio, gobierno, entre otros (Nagilla, 2012). Una de las características esenciales de una página o sistema web es el diseño estético, también llamado diseño o maquetado web, el cual es la expresión artística que dota de atractivo a la web, causando en los usuarios finales emociones diversas y consolidando el propósito de su construcción (Jarrar, 2002).

[1] La red informática mundial es un sistema de ordenadores interconectados entre sí con la finalidad de intercambiar información de interés para el usuario a través de una conexión a internet.

Existen muchos entornos de desarrollo web, que permiten realizar el maquetado de páginas o la construcción de sistemas en la web, todas ellas consolidadas a lo largo de los años. Sin embargo, todas ellas requieren del uso de un teclado de computadora para escribir las líneas de código necesarias para su funcionamiento (Mihalcin, 2007). Además, los entornos de programación donde se escriben dichos códigos, son tradicionales y no motivan a los profesionales o futuros profesionales, practicantes o curiosos a utilizar y expandir más sus conocimientos en diseño web. Pese a que el diseño web está en constante evolución y cada vez más se adhieren nuevos profesionales en la materia, las herramientas y entornos de desarrollo parecen haberse quedado rezagadas a lo tradicional: simple uso de teclado para codificar y poca motivación e innovación en el uso de otros medios de codificación, como la voz, que permitan mejorar y expandir su uso, enseñanza y aprendizaje (Delgado, Sandoval, & Arteaga, 2018).

Todo lo expuesto anteriormente nos motiva a construir un framework de diseño web el cual sea intuitivo, fácil, moderno e innovador, basado en la tecnología de reconocimiento de voz que permita el uso de la voz como mecanismo de interacción entre el usuario y el computador para indicar al framework la realización de alguna acción en particular (Roig, 2019), que rompa el esquema tradicional de codificación y automatice el proceso de maquetado web propiciando su uso, enseñanza y aprendizaje.

1.2. Formulación del Problema

¿De qué manera el desarrollo de un framework basado en reconocimiento de voz influye en la automatización del maquetado de páginas web?

1.3. Justificación de la Investigación

El crecimiento exponencial del uso de las páginas y sistemas web a nivel mundial hace obligatorio el aprendizaje y aplicación de las metodologías y de los lenguajes de desarrollo y maquetado web. La creciente demanda de profesionales en la materia

propicia la necesidad de innovar los mecanismos tradicionales de codificación, contando con un entorno de desarrollo innovador y práctico como soporte profesional a procesos propios de la tecnología web (García, 2019).

En la Tabla 1 mostramos las principales categorías en las que se clasifican los distintos sitios web. Se dice entonces que existe una amplia variedad de sitios en los cuales se requiere contar con profesionales expertos en diseño o desarrollo web.

Tabla 1 **Categorías principales de sitios web**

Categoría	Descripción	Ejemplo
Archivo	Usados para preservar contenido a lo largo del tiempo	Google Groups
Blog	Usados para postear ideas propias sobre algún tópico en particular	Blogger
Donativo	Permite realizar donaciones de caridad	Freerice
E-Commerce	Realiza ventas y transacciones económicas en línea	Amazon
Foro	Las personas tienen discusiones sobre algún tópico en particular	Theforumsite
Gobierno	Hechos por el gobierno o alguna entidad gubernamental	Richmond.com
Media	Los usuarios suben y bajan contenido multimedia	YouTube, Flickr
Noticias	Similar a sitios de información	CNN, BBC
Búsqueda	Indexa material en internet	Google search
Red social	Los usuarios se comunican y comparten imágenes, vídeos, blogs	Facebook
Académica	Docentes o estudiantes postean información académica o institucional	Vub.ac.be
Wiki	Los usuarios componen o editan contenido colaborativamente	Wikipedia

Fuente. (Sajjadi, 2012)

La construcción de un framework para automatizar el maquetado de páginas web aplicando reconocimiento de voz producirá una herramienta de desarrollo intuitiva, fácil, moderna e innovadora en el campo del maquetado o diseño web. Su construcción se realizará completamente usando los lenguajes de programación utilizados a nivel mundial los cuales son, del lado del cliente: HTML 5, CSS 3 y JavaScript (Calle & Trujillo, 2019), y del lado del servidor: PHP (Nagilla, 2012). De esta forma, se reducirán los esfuerzos y tiempos de desarrollo del framework. La comprensión de su funcionamiento y su posterior adaptación y mejora serán totalmente posibles debido al uso de estándares de desarrollo utilizados en su creación y a la posibilidad de descarga total del código fuente.

En la Figura 1 se muestra el servidor web Apache sobre el cual corren los archivos, del lado del cliente: HTML, CSS y JavaScript, propios del framework desarrollado. Como se aprecia este servidor es el más usado y es también el preferido en el área de desarrollo web del lado del cliente.

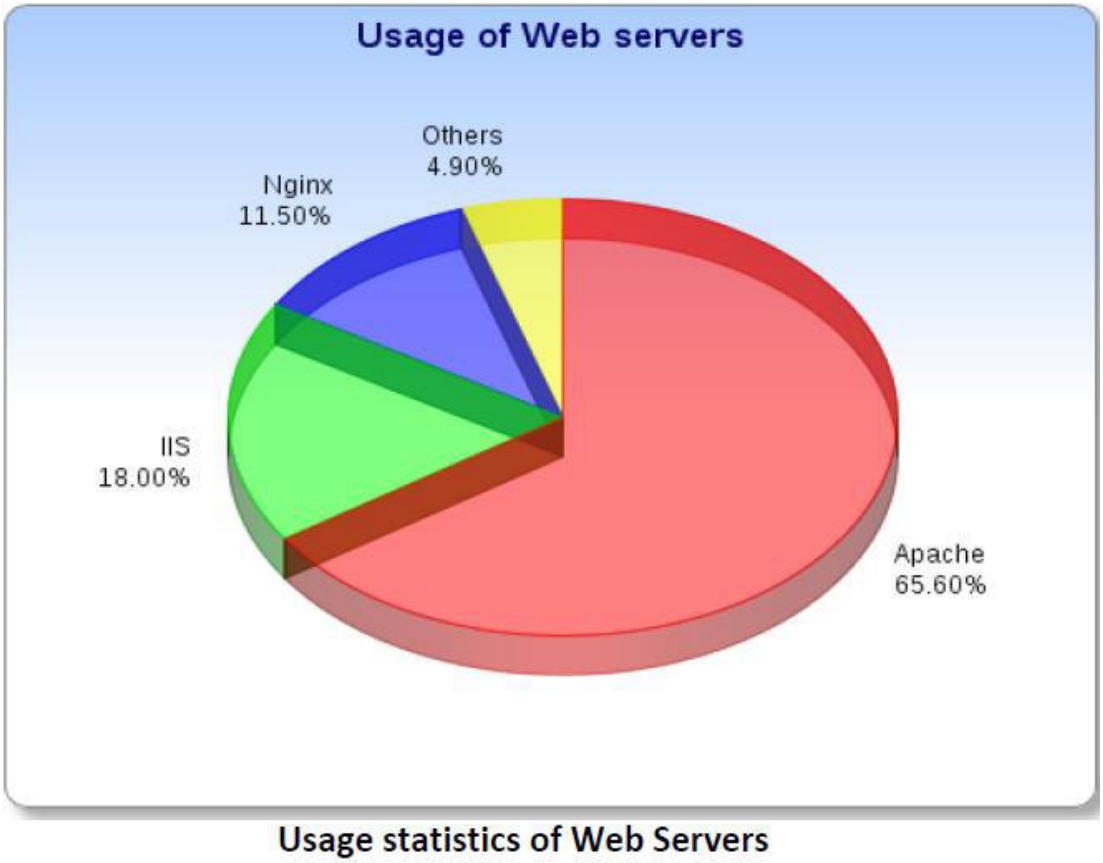


Figura 1 **Estadísticas de uso del servidor web Apache.**

Fuente: *(Nagilla, 2012)*

En la Figura 2 se muestra el lenguaje de programación del lado del servidor PHP sobre el cual corren los archivos, del lado del servidor, propios del framework desarrollado. Como se aprecia este servidor es el más usado y es también el preferido en el área de desarrollo web del lado del servidor.

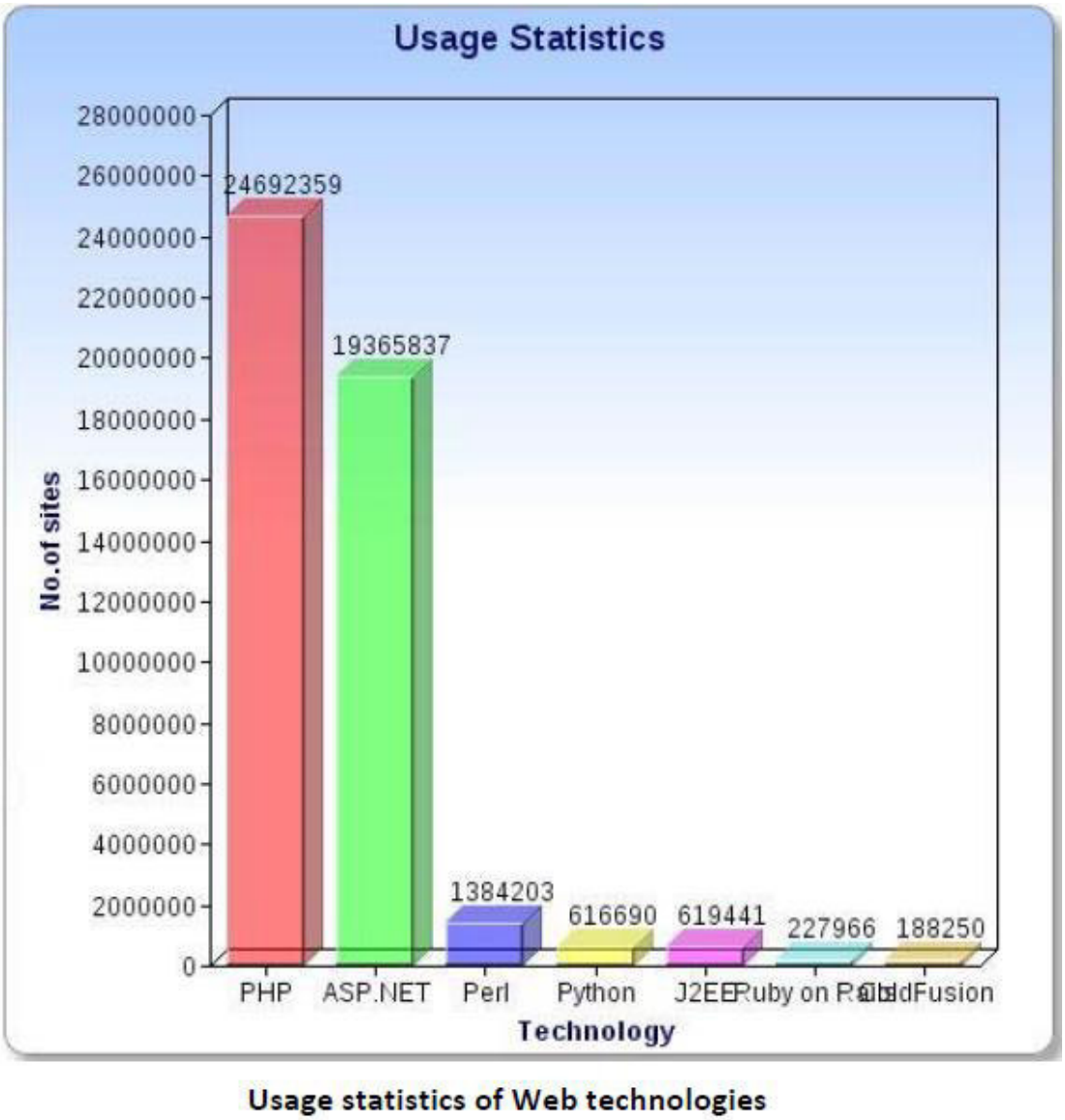


Figura 2 **Estadísticas de uso del lenguaje de programación web PHP.**

Fuente: *(Nagilla, 2012)*

Por otro lado, existen sistemas que permiten el reconocimiento de voz para realizar funciones de diversa índole: traducción, asistentes de configuración, conversores de texto a voz, entre otros. Muchos de ellos, son sistemas que necesitan contar con una licencia para su funcionamiento, la cual requiere de un pago económico a la empresa

que brinda dichos servicios (De la calle, 2014). No obstante, el framework propuesto para la automatización del maquetado web aplicando reconocimiento de voz, no requerirá de ninguna licencia para su uso pues será de código abierto o software libre (GNU, 2021). El código fuente podrá ser descargado completamente para poder ser mejorado o adaptado por profesionales o empresas que así lo dispongan, incluso con la finalidad de distribuirlo comercialmente.

En la Figura 3 se muestra el nivel de incremento en el uso de la tecnología de reconocimiento de voz, específicamente de su aplicación llamada reconocimiento de voz. Como se aprecia el uso de esta tecnología empieza a incrementarse a partir del 2016 en los Estados Unidos.

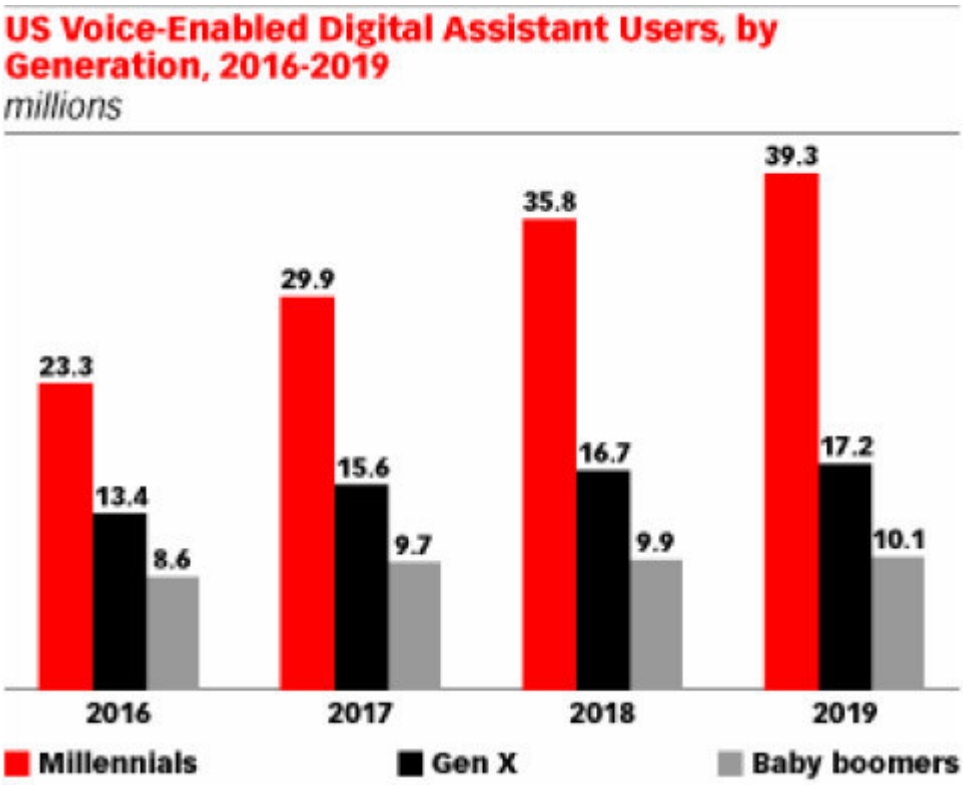


Figura 3 **Crecimiento del uso de la tecnología de reconocimiento de voz.**

Fuente: *(Roig, 2019)*

Con la construcción del framework que automatice el maquetado de páginas web aplicando reconocimiento de voz se pretende romper el esquema tradicional de codificación en diseño web, prescindiendo en cierta parte del uso del teclado como mecanismo de desarrollo de código fuente, propiciando la innovación y expansión de

las herramientas tecnológicas de maquetado web (Vivancos, 2016). Además, se pretende abrir las puertas a una nueva forma de programación que permita reducir el tiempo de desarrollo. El framework propuesto será construido con las herramientas tecnológicas más populares y comúnmente usadas con el propósito de hacerlo accesible desde todas partes del mundo y desde todos los dispositivos móviles inteligentes, sin la condición de instalar ningún programa adicional, si no tan sólo contar con un navegador web y una simple conexión a internet.

1.4. Motivación

El reconocimiento de voz ha sido aplicado con éxito en muchas investigaciones. (Collado & Aparicio, 2017) aplicaron la tecnología de reconocimiento y síntesis de voz, utilizando la Web Speech API, como apoyo a personas con problemas visuales para adherirlos al mundo de la tecnología mediante la búsqueda de información a través de un framework desarrollado con HTML, CSS y JavaScript que asocia las expresiones habladas por el usuario como los criterios de búsqueda y, luego de finalizar dicha búsqueda los resultados son presentados en formato lista y son leídos al usuario para que éste pueda elegir uno en concreto. Por otro lado, (Gil, Castillo, & Flórez, 2016), emplean el reconocimiento automático del habla para permitir a personas con discapacidad de movimiento interactuar con una silla de ruedas mediante el dictado de comandos de voz los cuales posibilitan acciones como el desplazamiento y el control de signos vitales. Por su parte, (Jaimez & Vargas, 2017), desarrollaron un framework WYSYWIG que permite automatizar la creación de sitios web permitiendo a docentes y alumnos de los cursos de diseño web seleccionar elementos HTML y colocarlos sobre un espacio de trabajo que simula un navegador web para luego permitir la aplicación de estilos CSS a cada elemento o, si se desea, realizar validaciones sobre los elementos de entrada utilizando JavaScript para finalmente generar, de forma automática, archivos con el código fuente de cada lenguaje de programación mencionado.

Se entiende entonces que el uso de la tecnología de reconocimiento y síntesis de voz mediante la API Web Speech y la automatización de la creación de sitios web han sido aplicados anteriormente, pero ninguna de estas investigaciones ha aplicado el

reconocimiento de voz específicamente para gestionar propiamente el maquetado de sitios web, permitiendo luego la generación y descarga de forma automática de las hojas de estilo bajo los estándares establecidos por la W3C. Por ello, en la presente tesis se desarrolla un framework web CSS 3, que permita al usuario gestionar gráficamente el maquetado web a través del uso de comandos de voz para crear, previsualizar el avance, descargar las hojas de estilo generadas automáticamente e incorporar dichos archivos CSS en la carpeta de estilos del proyecto de maquetado web sobre la que esté trabajando el equipo de desarrollo.

1.5. Objetivos

1.5.1. Objetivo general

Determinar de qué manera el desarrollo de un framework basado en reconocimiento de voz influye en la automatización del maquetado de páginas web permitiendo el uso de comandos de voz CSS 3 dentro de un entorno web intuitivo, fácil, moderno e innovador que permite generar y descargar automáticamente las hojas de estilo CSS gestionadas mediante el uso del mouse y el teclado, así como de la voz, lo que representa una innovación del método tradicional de diseño web.

1.5.2. Objetivos específicos

- OE1: Recolectar, analizar y obtener trabajos de investigación sobre frameworks que aplican reconocimiento de voz.

- OE2: Recolectar, procesar y obtener indicadores de maquetado de páginas o sistemas web que aseguran su aceptación.

- OE3: Diseñar los comandos de voz reconocidos y gestionados por el framework para la realización de una acción en particular.

- OE4: Diseñar un caso de estudio para evaluar el framework propuesto mediante la aplicación de los indicadores de maquetado web.

CAPÍTULO II: MARCO TEÓRICO

En este capítulo se ha documentado la revisión del estado del arte sobre frameworks que aplican reconocimiento de voz como medio para permitir la interacción humano – máquina mediante las tecnologías de síntesis o reconocimiento de voz, principalmente haciendo uso de la Web Speech API. Se han documentado además los trabajos de investigación sobre framework que permite la automatización del diseño o maquetado de sistemas o páginas web, sobre la importancia del uso de CSS en el maquetado o diseño web y sobre el uso de indicadores posibles de ser aplicados como guías para asegurar que el maquetado de un sitio web será ampliamente aceptado por los usuarios finales. Por último, se han detallado los conceptos teóricos como soporte al presente trabajo de investigación.

2.1. Antecedentes de investigación

Para la obtención de los antecedentes de investigación hemos aplicado una revisión del estado del arte considerando las siguientes interrogantes de investigación:

- ¿Qué frameworks aplican reconocimiento de voz como medio de interacción humano – máquina mediante las tecnologías de síntesis de voz o reconocimiento de voz?

- ¿Qué investigaciones existen sobre frameworks que permiten la automatización del diseño o maquetado de sistemas o páginas web?

- ¿Qué investigaciones existen sobre la importancia del maquetado o diseño web aplicando CSS?

- ¿Qué investigaciones existen sobre el uso de indicadores que aseguran que el maquetado de una página o sistema web será ampliamente aceptado?

Los años de publicación de los trabajos de investigación obtenidos y documentados van desde el 2004 al 2020, según la aparición y expansión de la Web 2.0, que ha sido el motor fundamental para el uso y evolución de los sistemas y páginas web.

Las palabras principales para la búsqueda de los trabajos de investigación fueron:

1. "Speech Voice Recognition Framework"
2. "Speech Synthesis Framework"
3. "Web Page Design Framework" and automated web page design
4. "Web Page Design Framework" and automated web system design
5. "Web Page Design" and CSS 3 importance
6. "Web Page Design" and use of guidelines in web layout

Los trabajos de investigación obtenidos luego de aplicar los criterios de búsqueda fueron veinte de los cuales escogimos documentar sólo siete. Se documentaron sólo siete trabajos de investigación debido a que éstos son considerados como los más adecuados y sirven como sustento a nuestro tema de investigación. Las restantes trece investigaciones fueron tomadas como referencia para la comprensión de conceptos necesarios y no fueron documentadas para evitar redundancias.

Los siete trabajos de investigación escogidos son mostrados en el orden siguiente: los cuatro primeros son referentes a frameworks con capacidad de síntesis y reconocimiento de voz, el quinto trabajo de investigación es referente a un framework que automatiza el diseño o maquetado de páginas o sistemas web, la sexta investigación corresponde a la importancia del uso de la tecnología CSS como medio de optimización del maquetado o diseño web y, la última investigación, corresponde al estudio sobre el uso de indicadores como guías para asegurar que el maquetado de un sitio web será aceptado.

2.1.1. Diseño de asistente de voz basado en web para personas con visión subnormal (Collado & Aparicio, 2017)

La investigación aplicada, perteneciente al área de investigación de la Ingeniería de Software que sigue la línea de investigación en Ingeniería en Tecnología Web, tiene

como objetivo el desarrollo de un asistente de voz basado en web que incluya a personas con problemas de visión y personas con problemas motores al mundo de la tecnología orientada a buscar temas de interés en Internet, para informarse sobre estos temas y realizar diversas consultas a través de comandos de voz.

El análisis de requerimientos y funcionalidades deseadas por los usuarios se realizó siguiendo las indicaciones y sugerencias de las personas con habilidades diferentes los cuales sirvieron para desarrollar los módulos del asistente de voz. En la investigación se determinó que el propósito principal de los usuarios era la realización de búsquedas de información. Los resultados de las búsquedas se presentan al usuario a través de una lista mostrada en pantalla y mediante reproducción por audio. Cuando el usuario elige un resultado en particular el framework se encarga de mostrar la información completa.

Los requerimientos funcionales se clasificaron considerando cuatro categorías, las que son: *reconocimiento de voz* referente al vocabulario de palabras reconocidas por el asistente, *sintetizador de voz* referente a la reproducción por audio de los resultados de la búsqueda, *consulta de búsquedas* referente a mostrar en pantalla los resultados obtenidos y *secuencia de palabras clave* que son las palabras reservadas que realizan las acciones *parar*, *volver a buscar* o *cerrar la aplicación*. Los requerimientos no funcionales fueron clasificados según: *rendimiento* que especifica que las búsquedas tienen que ser finalizadas con un máximo de 30sec.; *disponibilidad* que indica que el asistente de voz deberá funcionar las 24 horas del día durante todo el año; *usabilidad* referente a la simplicidad en el uso fundamentada en una interfaz amigable; *concurrencia* que especifica que la aplicación web deberá soportar un máximo de 1000/m de usuarios simultáneamente y, en caso de sobrepasar este límite, se rechazarán las peticiones sin afectar a los usuarios actuales; *portabilidad* que hace posible que el asistente de voz sea accedido desde cualquier equipo que cuente con un navegador web Chrome y con una conexión a internet.

La metodología aplicada en la investigación consistió en realizar capacitaciones sobre el asistente de voz en tres instituciones que atienden a personas con problemas de visón y con problemas motores, para que posteriormente se realice una encuesta para evaluar la funcionalidad implementada del asistente. Los encuestados han sido clasificados por

edades. Para ello, se consideró la clasificación siguiente: niños cuya edad estuvo entre los 8 a 12 años, adultos jóvenes con edad entre los 24 a 29 años, adultos con edad de 32 a 55 años y adultos mayores cuyas edades se encuentran en los 56 a 78 años.

Los resultados han demostrado que el tiempo de carga es de 26.3sec. con una disponibilidad del 99%, de tal manera que el 63.3% de encuestados indican que se sienten satisfechos con la interacción con la aplicación, el 100% indicó que se requería previamente una capacitación en el uso de la aplicación y el 66.7% indicó que resultó fácil interactuar con la aplicación. Se concluyó que el asistente de voz ayudó a que las personas adquieran una mayor confianza en la búsqueda de información y navegación por la web.

2.1.2. Reconocimiento de comandos de voz en español orientado al control de una silla de ruedas (Gil, Castillo, & Flórez, 2016)

La investigación realizada tuvo como finalidad implementar una aplicación de computadora para el *reconocimiento de instrucciones de voz* dictadas en idioma español. Estas instrucciones de voz, también llamados *comandos de voz*, se encontraron contenidas dentro de un vocabulario reducido compuesto por palabras reservadas y por ello se le dio la denominación de *vocabulario cerrado*. Las palabras clave que conforman este vocabulario son identificadas por la silla de ruedas como la lista de instrucciones que controlan, entre otras funciones, su desplazamiento. La tecnología de *Interfaz de Programación de Aplicaciones de Voz* (SAPI) de Microsoft, con *System.Speech.Recognition* como espacio de nombres, fue utilizada en la aplicación para reconocer las palabras reservadas del vocabulario, las cuales están relacionadas con las funcionalidades ofrecidas en cuanto a: desplazamiento de izquierda a derecha o de adelante y atrás; funcionalidades de domótica adecuadas al entorno donde se desplazará el usuario y, finalmente, funcionalidades de salud en cuanto a la medición de presión, temperatura, pulso y oxígeno.

Las herramientas de desarrollo utilizadas fueron: C# como lenguaje de programación, Microsoft SAPI, disponible en el sistema operativo Windows 7 y Microsoft .NET Framework para el reconocimiento del habla en idioma español el cual permitió el reconocimiento de palabras clave sin la necesidad de entrenar previamente a la

aplicación con la voz del hablante, proporcionando las funcionalidades para: *gestionar la entrada de voz*, *crear gramáticas* de reconocimiento de voz, *capturar información* sobre eventos generados por el reconocedor de voz y *configurar y administrar* los motores de reconocimiento de voz. El vocabulario cerrado del que consta el framework hizo posible que el módulo de reconocimiento de voz interpretara sólo las palabras clave permitiendo las ventajas siguientes: *aumento de la precisión y rendimiento* del módulo reconocedor, aseguraron que cada resultado tenga un *significado coherente* y ejecuten las *acciones programadas* según la frase identificada, *disminución de la carga de procesamiento*, eliminaron la necesidad de entrenar al módulo reconocedor por cada usuario que vaya a utilizar la silla de ruedas.

La metodología utilizada para la obtención de resultados en cuanto a la fiabilidad del reconocimiento de los comandos de voz en español, contenidos dentro del vocabulario cerrado, constó de la realización de tres pruebas con la participación de 10 hombres y 10 mujeres, siendo un total de 20 participantes. Se configuró, además, los siguientes tres rangos de ruido ambiental medidos en decibeles: [35-55 dB], [60-72 dB] y [73-85 dB], respectivamente.

Los resultados demostraron que para el primer rango de ruido se obtuvo un reconocimiento exitoso del 100% de los comandos pronunciados. Los resultados de la prueba con el segundo rango fueron también exitosos con excepción de un error de sustitución en el caso de las mujeres y en el de los hombres se dio un error de omisión. Los resultados de la prueba para el tercer rango, en el caso de las mujeres mostraron que 28 de los 35 comandos pronunciados fueron reconocidos exitosamente y los 07 comandos restantes presentaron errores de omisión; en el caso de los hombres 21 de los 35 comandos pronunciados fueron exitosos y los 14 comandos restantes mostraron errores de omisión o de sustitución.

2.1.3. Desarrollo de una aplicación para la web utilizando el Web Speech API (Roig, 2019)

La investigación realizada tuvo como finalidad el desarrollo de un sistema web dividido claramente en componentes que van desde el lado del cliente (*frontend*) hacia el lado del servidor (*backend*). El frontend dio al usuario la posibilidad de *configurar*

la voz y la *velocidad de respuesta* del sintetizador de voz. Del lado del servidor, el sistema web reconoció los siguientes comandos de voz: *apertura del navegador* mediante el comando "abrir navegador", *apertura de la aplicación YouTube* con el comando "abrir YouTube", *apertura de la aplicación Facebook* utilizando el comando "abrir Facebook", *apertura de la aplicación Twitter* a través del comando "abrir Twitter", *apertura de la aplicación Galería de imágenes* pronunciando el comando "abrir galería", *información de la hora actual* usando el comando "que hora es", *conocer el clima actual* con el comando de voz "que tiempo hace en + ciudad", *información de la presión y humedad de la atmósfera* a través del comando "cuál es la presión y humedad en + ciudad"; entre otras funciones.

El sistema web constó de las capas siguientes: dispositivo, datos de audio y aplicación. La capa llamada *dispositivo* se dividió en *dispositivo de entrada de audio* conformada por el micrófono y la tarjeta de audio encargados de la captura de voz, *dispositivo de salida de audio* conformada por los altavoces encargados de la salida del audio y *dispositivo de salida de texto* conformada por la pantalla del computador que permitió mostrar texto informativo. La capa llamada *datos de audio* se encargó de procesar el audio recibido y enviar los datos de salida al sintetizador de voz para ser reproducidos. La capa llamada *aplicación* fue la responsable de la gestión de los diferentes métodos del reconocimiento y síntesis de voz del framework.

El sistema web fue construido utilizando las tecnologías: HTML como lenguaje usado para estructurar y definir el contenido de las páginas a través de etiquetas que representan texto, imágenes, vídeos, entre otros; CSS como lenguaje usado para separar el diseño del contenido dentro de las página web y utilizado junto a HTML y JavaScript para dar estilo a las páginas del framework haciéndolas más atractivas mediante la aplicación de propiedades a los elementos de las páginas; JavaScript como lenguaje usado para mejorar la interacción entre el sistema web y el usuario pues dotó de funcionalidad a los elementos de las páginas. La Web Speech API, fue la encargada de gestionar el reconocimiento del habla y la síntesis de voz; esta API fue gestionada mediante el lenguaje JavaScript y expone las interfaces llamadas: *webkitSpeechSynthesis* y *webkitSpeechRecognition*, encargadas de activar los métodos y propiedades de síntesis y reconocimiento de voz, respectivamente.

La metodología usada para la obtención de resultados consta de pruebas realizadas en el navegador web Chrome. En cuanto al reconocimiento de voz se evaluaron todos los comandos de voz establecidos en la gramática del sistema, compuesta por los comandos mencionados anteriormente, correspondiente al módulo de reconocimiento del habla. Por otro lado, la evaluación de la síntesis de voz correspondió a la evaluación de las respuestas devueltas en audio por el módulo síntesis de voz. Se concluye que el uso de los tres lenguajes usados en el desarrollo de sistemas web hizo posible la incorporación de las tecnologías de síntesis y reconocimiento de voz a través de los diferentes métodos y propiedades proporcionados por la Web Speech API.

2.1.4. Aplicación multimedia para el entrenamiento en la certificación TOEFL mediante reconocimiento de voz (Hernández & Lemuz, 2016)

La investigación realizada tuvo como finalidad crear un sistema web para la práctica del idioma inglés con el objetivo de mejorar el nivel de capacidad de los interesados en rendir los exámenes de certificación TOEFL. Las características que presentó la aplicación web fueron: *utilidad para el examen de certificación TOEFL* pues fortaleció los conocimientos que los alumnos tenían sobre el idioma inglés, *síntesis de voz* para convertir a audio las respuestas de las evaluaciones para cada sesión, *lecciones de la vida diaria* académica.

La aplicación se compuso de los módulos siguientes: *Reading*, *Listening*, *Writing* y *Speaking*. Centró la gran mayoría de sus actividades dentro del módulo Speaking debido al uso de la tecnología de reconocimiento de voz. El proceso de enseñanza pudo ser realizado por un profesor con dominio del inglés y por el aplicativo solamente y, en cuanto el nivel de aprendizaje, el sistema tuvo la capacidad de conservar el último nivel logrado por el alumno. Los grados de dificultad de las lecciones académicas fueron regulados por los niveles alcanzados y variaron su dificultad según los criterios de pronunciación, longitud de las respuestas o el cambio del contexto de los temas académicos.

La aplicación del reconocimiento de voz correspondió a la modalidad de reconocimiento automático del habla por medio de comandos de voz y obedeció a un

vocabulario cerrado cuya gramática estuvo relacionada a cada una de las respuestas en una determina evaluación. Se utilizó la *API Web Speech* la cual proporcionó los métodos y atributos de gestión para la síntesis y reconocimiento de voz a través de las interfaces *webkitSpeechSynthesis* y *webkitSpeechRecognition*, respectivamente. Esta API es desarrollada por el equipo W3C y es impulsada por Google y openStream, se requirió del lenguaje de programación JavaScript para su uso y gestión.

En el diseño de la aplicación se puso especial consideración en un diseño sencillo, estético y adaptable para diferentes dispositivos como son el computador, la tablet o el smartphone, de tal manera que se pudo acceder desde estos dispositivos que contaron con un navegador web y una conexión a internet. Como se ha dicho, la API Web Speech se gestionó con el uso de JavaScript, es así que el sistema web utilizó AJAX para recuperar las respuestas correctas desde una base de datos y las cotejó con las palabras reconocidas por la API, las cuales fueron dictadas por el alumno durante una evaluación. De esta forma, para poder mostrar el resultado al estudiante con respecto a la evaluación realizada se verificó la cantidad de errores y la cantidad de aciertos resultantes. Así, si la cantidad de respuestas correctas era igual al total de preguntas evaluadas, el alumno pasaba satisfactoriamente la evaluación, en caso contrario, si el 50% o más de las palabras dictadas eran incorrectas, el alumno tenía que repetir la evaluación completamente.

2.1.5. Editor web visual para HTML, CSS y JavaScript de apoyo a la docencia (Jaimez & Vargas, 2017)

La investigación realizada tuvo como finalidad el desarrollo de un sistema web para *generar de forma automática código HTML, CSS y JavaScript*, separados en archivos independientes, mediante la construcción de páginas y sistemas web dentro de un entorno gráfico que redujo el tiempo de desarrollo. Así, el framework web fue diseñado y construido con la finalidad de apoyar las actividades de docencia en los talleres de programación, diseño web estático, programación web estático y dinámico, entre otros, de la Universidad Autónoma Metropolitana de México.

El framework web perteneció a la categoría WYSYWIG y fue accesible a través de un navegador web que contó con una conexión a internet. La arquitectura del framework

fue cliente-servidor. El lado del cliente se compuso de un *conjunto de componentes para la interfaz gráfica de usuario* y de los *módulos para la generación de código*. Así, el módulo para la generación de código del primer lenguaje, que es HTML, funcionó del lado del cliente a quien se le permitió seleccionar y colocar un conjunto de elementos sobre el espacio de trabajo pudiendo configurar todos los atributos de dicho elemento. Por su parte, el módulo para la generación de código CSS permitió la creación de hojas de estilo aplicadas, a nivel de clase, de elemento e ID, a cada uno de los elementos HTML, almacenando una estructura de datos con los estilos creados, utilizados luego para la generación de código. El módulo para la generación de código JavaScript se ocupó de realizar las validaciones para los elementos HTML de entrada de datos.

Por el lado del servidor, se compuso del *generador de código para los lenguajes HTML, CSS y JavaScript,* y del *módulo encargado de la generación de archivos*. El módulo encargado de la generación de código, se compuso de tres generadores correspondientes a cada uno de los lenguajes gestionados por el framework, los que después fueron pasados al módulo de generación de archivos. El generador de código CSS utilizó la estructura de datos creada desde la capa cliente guardando en memoria los estilos CSS aplicados a cada elemento con el objetivo de gestionarlos. Por otro lado, la funcionalidad de este módulo se compuso de dos pasos: recuperar la información del generador de código de los tres lenguajes gestionados y construir todos los archivos necesarios, aplicando el formato correspondiente a cada lenguaje y comprimiéndolos en un archivo ZIP, para devolver al cliente un link de descarga.

La interfaz gráfica del framework web constó de los componentes: *barra de elementos HTML* que mostró todos éstos elementos posibles de ser utilizados; *espacio de trabajo* que mostró el avance de la construcción de la página web; *barra de configuración* donde se mostró la pestaña atributos con los atributos de la etiqueta HTML seleccionada y la pestaña estilo mostrando todos los estilos CSS posibles de ser aplicados a la etiqueta; *barra de elementos en árbol* que mostró información sobre etiquetas HTML dentro del espacio de trabajo, además fue utilizada como un selector de elementos pues se pudo acceder a ellas mediante su selección; *barra de configuración* que estuvo conformada por todos los botones con las funcionalidades de: ejecución de código presentando una vista del avance, ocultar o mostrar la barra

de configuración y generar los archivos con los códigos HTML, CSS y JavaScript, respectivamente.

La metodología usada para la obtención de resultados constó de la creación de tres páginas web con el framework, utilizando diversas etiquetas HTML, sobre las cuales se aplicaron estilos CSS y validaciones de entrada de datos con JavaScript como medio de comprobación del correcto funcionamiento del framework. Así, por ejemplo, se creó una página web con las etiquetas HTML siguientes: table, div, img, h1, form, input y button. Estos elementos sirvieron para configurar la estructura del contenido de la página web la cual contenía imágenes, texto descriptivo, encabezados y un formulario de entrada de datos. La aplicación de estilos CSS permitió estilizar la página y sus elementos haciéndola más atractiva a la vista del usuario pues permitió asignar colores a los elementos, cambiar el tamaño de los textos, configurar los bordes, espacios y otras configuraciones de la etiqueta table. Mediante JavaScript se hizo posible validar la entrada de datos del formulario verificando que el correo electrónico tenga un formato correcto.

Los resultados obtenidos demuestran que el framework web desarrollado como apoyo a las actividades docente, el cual es de código fuente libre desarrollado en la UAM-C, hace posible los siguientes objetivos: la *construcción de páginas web de manera gráfica*, la *generación automatizada de código* bien estructurado y configurable, para que aquellos usuarios avanzados en el desarrollo web puedan extender las funcionalidades dentro de éstos archivos generados, el *apoyo al estudiante a entender el código generado* a partir del desarrollo gráfico de una página web.

2.1.6. El maquetado a base de scripts y hojas de estilo en cascada (CSS) y su incidencia en la optimización de un sitio web (Pilamunga, 2012)

La investigación realizada tuvo como finalidad determinar el nivel de incidencia sobre la optimización del tamaño y funcionalidad de un sitio web al ser maquetado mediante CSS y JavaScript. La investigación bibliográfica y de campo realizada identificó las causas siguientes motivos del tamaño excesivo de un sitio web maquetado incorrectamente: la *falta de conocimiento sobre la tecnología CSS* para el maquetado,

poca publicidad sobre los beneficios de CSS, una *formación regular o baja* durante el estudio de posgrado con respecto al diseño web, el *poco interés de los estudiantes por el aprendizaje de CSS*, la *existencia de sitios web construidos sin tomar en cuenta las normas de usabilidad, SEO y W3C*, *aplicación nula de normas de maquetado* al usarse programas de diseño web.

El diseño actual de los sitios web está regido por la abundancia de imágenes, íconos, botones, animaciones, videos, entre otros, lo que ocasiona un tiempo excesivo de carga y presentación ante el usuario pues se torna muy pesado y esto causa incomodidad, en la mayoría de las veces, provoca también el abandono del sitio web por parte del usuario. Ante esta realidad, se determinó que una correcta maquetación, que use las normas establecidas por la W3C junto con el uso de scripts y CSS resultan primordiales para la optimización del tamaño y tiempo de carga de un sitio web pues redujeron en un 60% el tamaño del mismo. Otras de las ventajas de realizar una adecuada maquetación vino a ser: la *centralización del código fuente* de maquetado lo que reduce el tiempo y la cantidad de recursos al momento de realizar una actualización del sitio, la *clara división entre los elementos del contenido y los elementos de la presentación*, la *modificación de la visualización del documento alterando sólo el contenido del CSS* o del script, la *optimización del ancho de banda de la conexión* debido a la reducción de archivos de maquetado pues mediante la aplicación de clases CSS se puede importar una única hoja de estilos en varios archivos HTML de tal manera que uno o más elementos dentro de un documento HTML puede ser estilizado mediante una sola hoja de estilos.

La tecnología CSS nos permite definir las siguientes formas para aplicar las propiedades y valores de estilo a cada elemento dentro de un documento HTML: el archivo CSS se aplicará a todo un documento HTML mediante la *importación del CSS mediante un pequeño trozo de código en la cabecera del HTML* o se aplicará a una etiqueta en particular mediante la *adición del código CSS en la definición de la etiqueta*. Las ventajas de usar CSS son: la *separación de forma y contenido* pues por lo general el código CSS se encuentra en un archivo diferente al HTML, *reduce el tráfico entre el servidor web y la máquina usuario* en un 40% y 60%, *reduce los tiempos de carga* presentando con mayor rapidez el sitio web al usuario, permite *controlar absolutamente todos los elementos* de la página pues permite configurar

todas las propiedades de un determinado elemento, *reduce el tiempo de mantenimiento* al momento de realizar una actualización pues se modifican sólo los archivos CSS, *permite una actualización flexible del código fuente de maquetado* mediante el cambio completo o parcial del aspecto de un sitio web con sólo modificar la hoja de estilo, *mejora el posicionamiento web* pues crea archivos HTML semánticamente más correctos.

La metodología usada para la obtención de resultados constó de la aplicación de encuestas a 3 docentes especialistas en diseño o maquetado web, 25 egresados que mantienen vínculo con la institución y 101 alumnos que se encontraron cursando estudios entre el octavo y décimo ciclo de la carrera de diseño gráfico los cuales se encontraron con edades de entre 23 a 30 años. Las encuestas se realizaron mediante un cuestionario que se componía de 11 preguntas clara y concisamente establecidas enfocadas a conocer los temas siguientes: diseño previo de páginas web, herramientas utilizadas en el maquetado web, inconvenientes al utilizar programas de maquetado, conocimiento sobre la utilidad de CSS, razones para el no uso de CSS, función de CSS en el diseño de páginas web, uso de JavaScript como medio de validación de datos de entrada, uso de CSS y JavaScript como medio de optimización del tamaño de sitios web, uso de CSS y JavaScript como medio de mejora de la funcionalidad de sitios web, reducción de la cantidad de archivos de maquetado innecesarios en un sitio web.

Los resultados demostraron que CSS era escasamente utilizada con un 9% de uso frente a un 78% de Dreamweaver y un 99% de Flash. No obstante, se demostró con un 100% que el uso de estas herramientas incrementa innecesariamente la cantidad de imágenes durante el maquetado web causando que el tamaño de un sitio web sea excesivo. Se demostró también que la principal razón para no utilizar CSS fue que el total de los participantes encuestados tenía conocimientos insuficientes o nulos sobre dicha tecnología. Se pudo concluir que la realización del diseño o maquetado a base de CSS y JavaScript influye de manera significativa a la hora de mejorar y optimizar un sitio web, lo cual se pudo verificar con la rapidez y velocidad en la carga, navegación y accesibilidad con poca latencia.

2.1.7. The challenge of web design guidelines: Investigating issues of awareness, interpretation, and efficacy (Szigeti, 2012)

La investigación realizada se enfoca en determinar cuál es la relación existente entre los indicadores establecidos para el maquetado web y los profesionales correspondientes para lograr determinar si estos indicadores son usados o no, los motivos del uso o no uso y el impacto de diseñar interfaces web aplicando estos indicadores, específicamente extraídos de la publicación Research-Based Web Design & Usability Guidelines (U.S. Department of Health and Human Services, 2006), que establece un conjunto de directrices de maquetado web altamente recomendadas como ayuda al profesional en diseño para determinar si el trabajo que está realizando será aceptado o no por la gran mayoría de usuarios finales del sitio web.

La investigación completa gira en torno a seis interrogantes establecidas las que abordan los temas siguientes: la postura de los diseñadores de interfaces web hacia el uso de indicadores, el grado de aplicación y uso de los indicadores por parte de los diseñadores, el enfoque que dan los maquetadores web a estos indicadores, la frecuencia con la que los indicadores son usados en el proceso de diseño web, que indicadores son los más utilizados por los diseñadores y cuál es la manera en que estos indicadores son transmitidos entre los profesionales de maquetado web.

La metodología usada para la obtención de resultados constó de la realización de cuatro casos de estudio. El primer caso de estudio se realizó con la participación de 16 estudiantes de ingeniería a los cuales se les solicitó diseñar una Wiki aplicando 25 indicadores extraídos de la publicación Research-Based Web Design & Usability Guidelines (U.S. Department of Health and Human Services, 2006) con el objetivo de analizar el grado de aplicación de los indicadores y su utilidad a la hora de realizar el diseño de las interfaces web, encontrándose que el 51.8% de los indicadores fueron aplicados en el diseño web de la Wiki.

La Figura 4 muestra la clasificación en categorías de los indicadores evaluados en el diseño de la Wiki junto a su porcentaje de aplicación.

Table 3.2: Guideline usage by category

Categories	% of instances guidelines in a category were used		
	Applied	Partially applied	Not applied
Page Layout	54%	30%	16%
Text Appearance	74%	16%	10%
Graphics, Images and Multimedia	35%	21%	44%

Figura 4 **Porcentaje de aplicación de indicadores de diseño web.**

Fuente: *(Szigeti, 2012)*

El segundo caso de estudio se enfocó en analizar la importancia de la aplicación de indicadores de diseño en la evaluación de sitios médicos que brindan información sobre el cáncer. Para ello se encargó a 8 maquetadores web con alrededor de seis años de experiencia evaluar 10 de los sitios web mencionados aplicando un total de 67 indicadores extraídos de la publicación Research-Based Web Design & Usability Guidelines (U.S. Department of Health and Human Services, 2006), encontrándose que el promedio de cumplimiento de la aplicación de los indicadores se encuentra en un rango de entre 52.2% a 73.7%, que representan un total de 35 y 49.4 indicadores aplicados, respectivamente.

La Figura 5 muestra los promedios de aplicación de los indicadores evaluados por cada uno de los diseñadores web para cada uno de los sitios web médicos.

El tercer caso de estudio tuvo como finalidad comprender el enfoque que dan los maquetadores web a los indicadores de diseño. Para ello se aplicó un cuestionario vía web, posteado a través de www.surveymonkey.com, el cual constó de 37 preguntas categorizadas en cuatro grupos para determinar datos sobre la experiencia en diseño web del encuestado, su grado de conocimiento y comprensión acerca de los indicadores, el nivel de aplicación de éstos y conocer los datos demográficos del encuestado, respectivamente. El total de encuestados fue de 116 de los cuales 59 fueron de Canadá, 40 de U.S., 5 de India, 4 de U.K., 2 de Australia, 2 de Alemania, 1 de Austria, 1 de Croatia-Herzegoviana, 1 de Francia y un encuestado que no quiso indicar su nacionalidad. Se obtuvo como resultado que la mayoría de los encuestados usó indicadores en el desarrollo del maquetado y que alrededor del 80% considera que

los indicadores son generalmente muy útiles demostrando una concepción positiva sobre su uso.

Table 4.2: Assessments by individual participants. Sites ranked by the average number of guidelines with which they complied (maximum possible was 67).

		Participant designers								
Rank	Site	A	B	C	D	E	F	G	H	Average
1	Mayoclinic.com	54	57	45	47	54	54	39	45	49.4
2	Cancer.ca	46	49	53	49	48	50	40	48	47.9
3	Revolutionhealth.com	48	51	53	49	43	45	44	44	47.1
4	iVillagehealth.com	51	40	44	45	51	51	32	43	44.6
5	Cancer.gov	44	40	49	40	51	40	37	48	43.6
6	Everydayhealth.com	39	36	42	47	40	52	44	44	43.0
7	Medicinenet.com	34	55	32	51	50	46	27	43	42.3
8	Netdoctor.co.uk	38	34	37	45	43	54	37	41	41.1
9	Medhelp.org	34	50	28	37	44	45	43	47	41.0
10	About.com Health	26	27	38	47	41	36	25	40	35.0

Figura 5 **Promedio de aplicación de indicadores de diseño web en sitios web de salud.**

Fuente: *(Szigeti, 2012)*

La Figura 6 muestra los porcentajes de aplicación de los indicadores de diseño web por parte de los encuestados como ayuda en el maquetado de sitios web.

Table 5.3 Frequency with which guidelines help address design problems

Response	# of respondents	% of respondents
Often	32	27.6%
Sometimes	48	41.4%
Rarely	12	10.3%
Never	0	0.0%
Don't Know	4	3.4%
Skipped question	20	17.2%
TOTAL	116	99.9%

Figura 6 **Porcentajes de la frecuencia de aplicación de indicadores de diseño web.**

Fuente: *(Szigeti, 2012)*

El cuarto caso de estudio tuvo como finalidad analizar la manera en que los indicadores de diseño web son transmitidos entre los profesionales de diseño. Para ello se entrevistó a 20 diseñadores de interfaces web, los cuales participaron en el tercer estudio siendo estos de Canadá, U.S. e India. La realización de estas entrevistas se llevó a cabo vía Skype, llamadas telefónicas y en forma presencial. Los entrevistados respondieron 23 preguntas de las cuales 11 fueron acerca de los indicadores de diseño, 5 fueron sobre los beneficios de usar estos indicadores, 4 fueron sobre los medios de transmisión del conocimiento acerca de los indicadores y 2 preguntas cerradas para que el usuario pueda expresar sus ideas propiamente. Los resultados indican que los indicadores son transmitidos mayormente entre colegas en conversiones de trabajo, a través de emails y sitios web de especialización.

La Figura 7 muestra los medios de transmisión principales de los indicadores de diseño web.

Table 6.3: References to knowledge sharing between designers

Method of knowledge sharing	References in interviews	Percentage of total
Face-to-face	11	42.3%
Email	4	15.4%
Blog posting	3	11.6%
Digital services (wikis, listservs, content management systems	3	11.6%
None	3	11.6%
Phone	2	7.7%
Total	**26**	100.2%

Figura 7 **Medios de transmisión principales de los indicadores de diseño web.**

Fuente: *(Szigeti, 2012)*

Del estudio se concluye que el uso de indicadores de diseño web resulta adecuado como guía o directriz al profesional en maquetado para ayudarle a incrementar las probabilidades de aceptación, por parte del usuario final, del sitio web sobre el cual aplique indicadores de diseño.

2.2. Bases teóricas

2.2.1. Framework CSS

Un framework o marco o entorno de trabajo, en el desarrollo de software, es un entorno de trabajo donde se construyen los módulos de un software, cuenta con funcionalidades claramente definidas para guiar el desarrollo y crear componentes correctamente estructurados (Torres, 2013). Técnicamente un framework provee funcionalidades que guían el desarrollo de un software en particular bajo uno o más lenguajes de programación cuya sintaxis es soportada por el framework. Se sabe claramente que hoy en día los sistemas software que se desarrollan con mayor demanda son los sistemas orientados a la web, lo que quiere decir que la construcción de sistemas web o la construcción de páginas web tienen una alta demanda en el mercado. Para cubrir esta alta demanda se cuenta con tecnologías web que permiten: estructurar, estilizar y dotar de funcionalidad a los sistemas o páginas a construir o desarrollar.

Entre estas tecnologías tenemos a los lenguajes de programación: HTML que se ocupa de la estructuración de la página, CSS para estilizar los elementos de la página y JavaScript que dota de funcionalidad a éstos elementos. Estos lenguajes de programación son usados en el desarrollo web y junto a una metodología que guíe el desarrollo permiten diseñar, construir e implementar sistemas web de alta calidad. Es así que, al hablar específicamente de un framework de CSS, nos referimos a un conjunto de propiedades y valores llamados estilos que son usadas para el maquetado o diseño web, los cuales contribuyen con las ventajas de: *facilitar y agilizar* el diseño web, *asegurar el funcionamiento* del diseño en una amplia gama de navegadores, *cumplir con normas estándar* de desarrollo web y asegurar que lo que se use para el diseño funcionará sin errores (Wikipedia, 2020).

2.2.2. Reconocimiento de voz

El reconocimiento automático de voz es una disciplina de la inteligencia artificial cuyo objetivo es hacer posible la comunicación entre seres humanos y computadoras por medio del habla. El reconocimiento de voz se utiliza generalmente en los sistemas de

telefonía para atender a la persona que realiza la llamada, con una voz artificial o leyendo un mensaje pregrabado, ofreciéndole un conjunto de opciones a realizar según elija el hablante. Otra de las áreas donde se emplea el reconocimiento de voz es en la medicina para transcribir los diagnósticos e informes médicos con una elevada precisión. Actualmente el reconocimiento de voz se ha extendido ampliamente gracias a la masificación del uso de los smartphone que incorporan esta tecnología para realizar acciones diversas (Vivancos, 2016).

Una de las clasificaciones de los sistemas de reconocimiento de voz es llamada reconocimiento de voz de gramática restringida la cual es usada para permitir al hablante dar órdenes breves al sistema con la intención de que esta realice tareas concretas pues se basan en un *vocabulario de expresiones reducidas y reconocidas* lo que permite al sistema de reconocimiento de voz trabajar mejor pues tanto la sintaxis como la semántica de tales expresiones están integradas. Las clasificaciones de los sistemas de reconocimiento del habla están sujetas a los criterios de: *entrenabilidad* que verifica si es necesario un entrenamiento previo del sistema antes de su uso, este parámetro de entrenamiento es determinado por otro criterio llamado *dependencia del hablante* el cual categoriza al hablante según su idioma, su dialecto, entre otros. Otro parámetro de categorización es llamado *continuidad* y sirve para indicar si el sistema es capaz de reconocer el habla continuamente o es necesario realizar pausas entre cada palabra. Por su parte, el criterio *robustez* indica si el sistema es capaz de interpretar las palabras habladas en ambientes con condiciones ruidosas o no. El parámetro llamado *tamaño del dominio* es el que determina si el sistema trabajará con una gramática restringida o reconocerá un enorme número de expresiones. Una de las principales aplicaciones del parámetro tamaño del dominio es llamado *control por comandos de voz* los cuales son aquellos sistemas de reconocimiento de voz diseñados para dar órdenes a un computador y reconocer un vocabulario de expresiones reducidas lo que incrementa su capacidad y rendimiento (Gil, Castillo, & Flórez, 2016).

2.2.3. Control por comandos de voz

Los sistemas de reconocimiento voz caracterizados por el tamaño reducido y predefinido de su gramática son llamados control por comandos. La gramática de estos sistemas contiene todas las órdenes que el usuario puede pronunciar con la intención

de ordenar al sistema realizar una acción en particular definida previamente. Si el usuario expresa un comando no válido entonces no se realizará nada pues no se ha encontrado una acción asociada a dicha orden (Vivancos, 2016). En la actualidad existen muchos sistemas que permiten aplicar control por comandos, reconociendo la expresión hablada y devolviéndola como texto, una de ellas es llamada Web Speech API, la cual funciona en la web y ha sido creado por W3C.

2.2.4. Web Speech API

Esta API permite a los desarrolladores web, bajo una forma totalmente transparente, hacer uso de la tecnología de síntesis y reconocimiento de voz a través de funciones de entrada, procesamiento y salida de voz para hacer posible la conversión de voz a texto y de texto a voz. Una de las ventajas de esta API es la independencia de la implementación del reconocimiento de voz, esto es, permite reconocer la entrada de voz, especificando si será de manera continua o no, para que luego de realizar el procesamiento de reconocimiento sea devuelta al usuario en formato texto, dándole la posibilidad de realizar con ella lo que mejor le parezca (Roig, 2019).

Web Speech API es una librería para JavaScript que da la posibilidad de realizar: búsquedas por voz en la web, construir una interfaz de comandos de voz, reconocer la voz de forma continua en un diálogo abierto como en el caso de dictados de voz. El funcionamiento de esta API consta de una interfaz creada con diferentes métodos y eventos llamada *webkitSpeechRecognition* con los atributos: *grammars* para especificar las gramáticas activas, *lang* para establecer el idioma, *continuos* que indica si se devolverán uno o más resultados, *interimResults* para devolver resultados provisionales, *maxAlternatives* para especificar el número máximo de alternativas como candidatos a resultados (Roig, 2019).

Los métodos expuestos por la interfaz *webkitSpeechRecognition* son: *start* para determinar el inicio del servicio de reconocimiento de voz, *stop* sirve para indicar al servicio de voz que se debe detener la recopilación de información vía audio, *abort* para detener de inmediato el servicio de escucha en ejecución. Los eventos de la API son: *audiostart* activado cuando se empieza a capturar audio, *soundstart* activado cuando se recibe o detecta algún sonido, *speechstart* activado cuando se detecta que el

evento *audiostart* ha sido activado, *speechend* activado al finalizar el habla, *soundend* activado cuando no se detecta ningún sonido, *audioend* activado cuando finaliza la entrada de audio, *result* activado cuando reconocedor devuelve un resultado, *nomatch* activado cuando se devuelve un resultado que no cumple el umbral de confianza, *error* activado cuando se produzca un error en el reconocimiento, *start* activado cuando el servicio de reconocimiento de voz comienza a escuchar el audio a ser reconocido, *end* activado cuando el servicio se desconecta. La interface *SpeechRecognitionAlternative* es la interfaz que representa la respuesta del reconocimiento y tiene los atributos: *transcript* representa las palabras identificadas y, cuando se especifique que se desea reconocer continuadamente las palabras, abarcan también los espacios en blanco para que se realce una correcta transcripción, *confidence* representa la estimación numérica entre el rango de valores de 0 a 1 de la precisión del reconocimiento (Roig, 2019).

El framework CSS propuesto usa la función de reconocimiento de voz de la API Web Speech mediante una de sus aplicaciones llamada, *control por comandos*, lo que nos ha permitido crear y utilizar una gramática reducida perteneciente a un vocabulario reducido de expresiones para dar órdenes reconocidas por el framework. El atributo *grammars* se ha definido dentro del framework propuesto con los comandos de control propios para la gestión de una hoja de estilo (CSS), el atributo *lang* se ha establecido a español, el atributo llamado *continuos* ha sido especificado para indicar que se reconocerá un habla continua, y los atributos *interimResults* y *maxAlternatives* se han establecido en falso y en uno (01). Hemos usado los métodos *start* para el inicio de la escucha del audio por voz a reconocer y el método *error* para manejar los errores de reconocimiento dentro del framework.

2.2.5. Maquetado o diseño web

Usualmente se piensa que la maquetación web es simplemente estilizar elementos HTML de la página, pero en realidad va mucho más allá, pues se define como la actividad que requiere de pasos como son: la planificación, diseño e implementación de sitios y páginas web teniendo en cuenta los aspectos de navegabilidad, usabilidad, multimedia y la arquitectura de la información. La información estructurada se enlazada con otros documentos a través de hiperenlaces posibles de ser accedidos y

mostrados en diferentes dispositivos como son computadores, celulares, tabletas, entre otros (Pilamunga, 2012).

Podemos identificar entonces tres etapas en el maquetado o diseño web: la primera es la organización de la información contenida dentro de la página web a maquetar realizando prototipos previamente de la distribución y organización de los elementos de la página a maquetar; la segunda etapa es el análisis y la navegación por el sitio web de tal manera que el usuario pueda desplazarse por todas las páginas del sitio con normalidad en busca de la información relevante para el usuario; la tercera etapa es el posicionamiento web referente a la aparición en los primeros resultados de las búsquedas que el usuario realice sobre un tema en particular (Pilamunga, 2012).

2.2.6. Indicadores de maquetado o diseño web

Los indicadores de diseño o maquetado web son considerados como guías altamente recomendadas a los profesionales de diseño para garantizar que la organización, contenido, navegabilidad, estilo y accesibilidad de cualquier sitio web tendrán altas probabilidades de ser ampliamente aceptados por los usuarios finales. Los indicadores de maquetado web han sido establecidos en base al consenso de criterios de expertos en la materia. A continuación, se presenta un trabajo de investigación donde se listan una serie de indicadores de diseño propuestos para la fase de implementación del diseño, la cual forma parte de la metodología WSDM (De Troyer, 2001), y se presenta también un estudio que expone puramente un conjunto de indicadores o directrices categorizados en capítulos que van desde el análisis y diseño y finaliza con el capítulo de pruebas de usabilidad del sitio web.

2.2.6.1. Web Design Guidelines for WSDM (Jarrar, 2002)

En este estudio se desarrolla la fase de implementación del diseño, perteneciente a la metodología de diseño web WSDM (De Troyer, 2001), la cual se basa en un enfoque guiado por usuarios. Esta metodología se muestra en la Figura 8 donde son mencionadas las cinco fases de desarrollo de un sitio web.

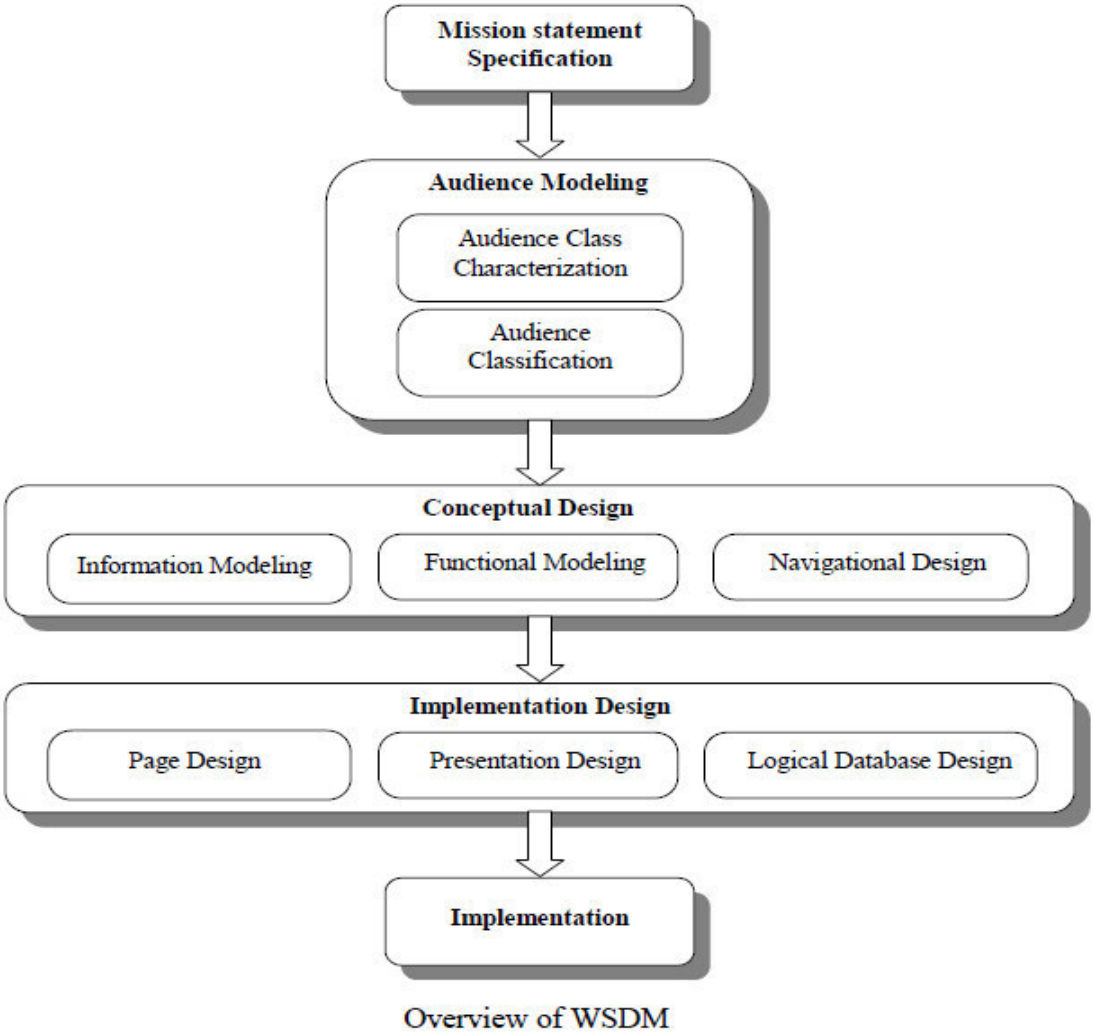


Figura 8 **WSDM: Web Site Design Method - Audience-driven approach.**

Fuente: *(De Troyer, 2001)*

La primera fase de desarrollo fundamenta los requerimientos de información y los tipos de usuario más importantes para el sitio web. La segunda fase clasifica precisamente estos tipos de usuarios identificando sus preferencias y actividades principales las que serán plasmadas dentro del sitio. La tercera fase realiza el modelado de la información a presentar en el sitio web, el modelado de las funciones y de los enlaces de navegación disponibles para cada tipo de usuario identificado. La cuarta fase, implementación del diseño, encierra la actividad llamada diseño de página la cual es la actividad relevante para este trabajo de investigación. La quinta fase analiza las herramientas tecnológicas para construir el sitio web y analiza también los entornos de producción posibles para su despliegue en la web.

La actividad diseño de página, mostrada en la Figura 9, correspondiente a la fase implementación del diseño, establece una serie de indicadores clasificados en 10 grupos o categorías principales: navegación, usabilidad, interface y estilo, legalidad, comunicación e infraestructura, semántica, marketing, error, prueba y mantenibilidad. Las categorías de navegación, usabilidad, interface y estilo son referentes a la estructura del sitio web pues se encargan de tópicos como la accesibilidad, la organización de la navegabilidad y maquetado del sitio web propiamente.

Las categorías de legalidad y seguridad, comunicación e infraestructura, semántica y marketing se encargan de tópicos relacionados con la funcionalidad del sitio web analizando los criterios siguientes: protocolos de seguridad, protocolos de comunicación entre los usuarios y el servidor o servidores web donde se encuentran alojados los módulos del sitio y criterios de publicidad promocional.

Las categorías de error, prueba y mantenibilidad se encargan de evaluar los tópicos de confiabilidad y calidad para el sitio web que se desea construir contemplando las actividades de aseguramiento de la calidad, como son: la realización de pruebas para cumplir con los requerimientos solicitados por cada tipo de usuario y la realización del análisis de actividades a seguir cuando se requiera una actualización o mantenimiento posterior del sitio.

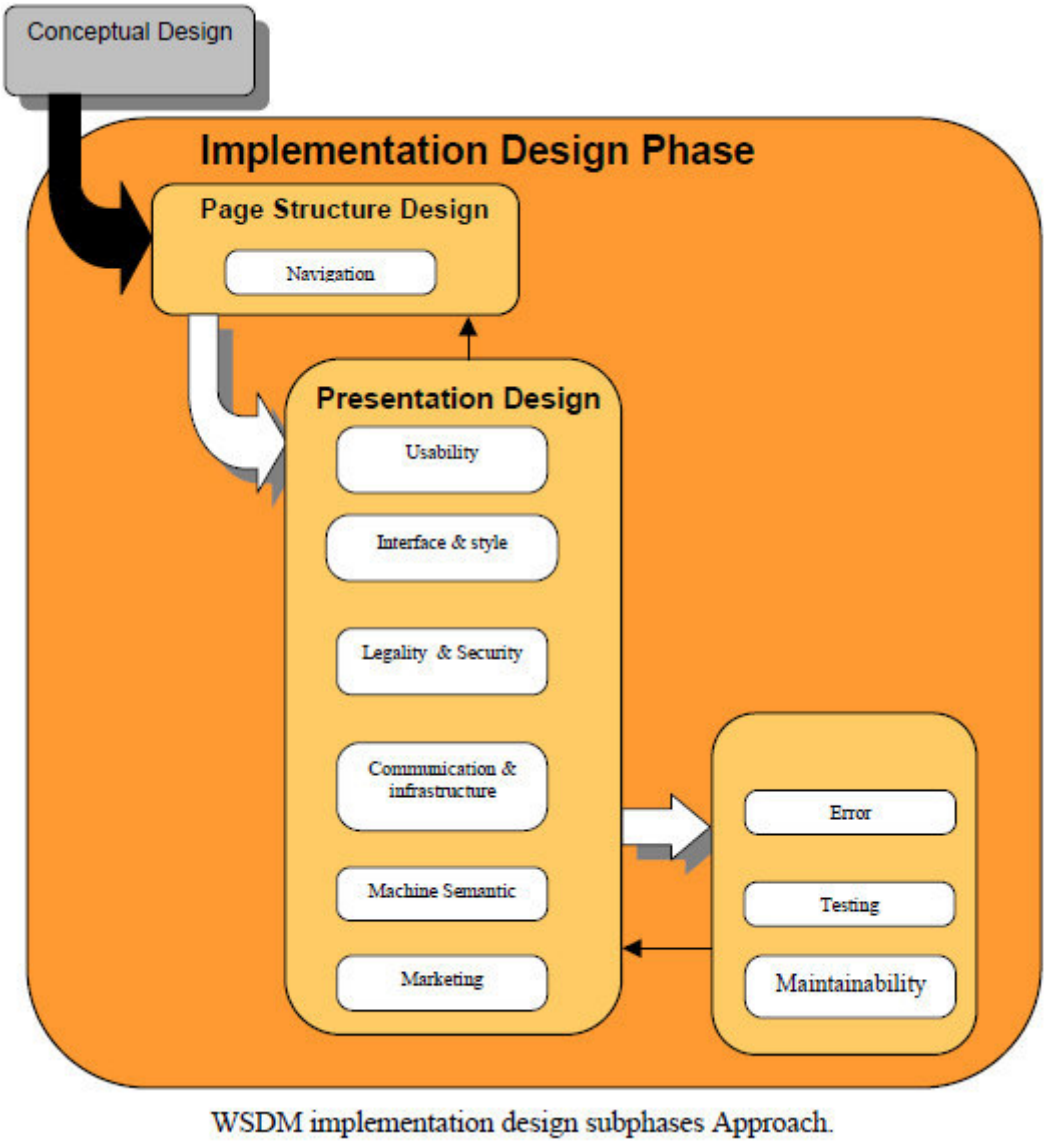


Figura 9 **Fase de implementación del diseño - Metodología WSDM.**

Fuente: *(Jarrar, 2002)*

Cabe mencionar que para este trabajo de investigación nos hemos enfocado únicamente en los indicadores de la categoría interface y estilo, perteneciente a la actividad diseño de página, propia de la fase implementación del diseño. El motivo se sustenta en el hecho de que cada uno de estos indicadores se encuentran estrechamente relacionados con el maquetado de cualquier sitio web.

No obstante, se menciona claramente que en la investigación se han desarrollado indicadores para cada una de las 10 categorías mencionadas. La Figura 10 muestra la lista de indicadores en el idioma original en el que fueron escritos.

76.Every non-text element shall be provided via "alt" (alternative text attribute), "longdesc" (long description tag) or in element content.
77.Maintain a standard page layout throughout the web site.
78.Use plain backgrounds and simple layouts to improve the readability of text
79.Make sure that all copy is easy to read and that background images do not distract or make the copy impossible to read.
80.Keep site consistent, don't have different color backgrounds for every page.
81.Remember yellow copy and white copy are hard to read and not printable.
82.Do not use all CAPITALs.
83.Use only a limited number of carefully chosen fonts
84.Avoid defining very specific fonts as some users may not have them loaded on their machines.
85.Always provide the pixel size for graphics and the width of columns
86.Choose text and background colors to provide maximum contrast.
87.Provide information of row and column headers.
88.Ensure that text can always be clearly read at any location against the background.
89.Avoid cheap graphics.
90.Avoid animated gifs.
91.Do not use interlaced or progressive images.
92.Avoid Dropshadows.
93.Avoid blinking text.
94.Avoid the unnecessary use of icons, graphics and photographs.
95.Contrast the foreground and background colors.
96.Do not abbreviate dates; for example.
97.Keep the number of colors in your images to a minimum.
98.Design your background image at the lowest color depth and resolution you can.
99.Avoid/Limit using image maps.
100.Provide text transcriptions of all video clips.
101.Give a written description of any critical information that is contained in audio files contained on your website.
102.Only white, or light colors, for background.
103.Dark backgrounds can be used on home page only.
104.Avoid red and blue togather.
105.use color to show relationships and differences.
106.avoid peige or brown pictures.
107.use the same color(s) to in group related elements.

Figura 10 **Indicadores de la categoría interface y estilo.**

Fuente: *(Jarrar, 2002)*

La lista de indicadores mostrada fue tomada como referencia para elaborar y categorizar nuestros propios indicadores los cuales fueron utilizados como instrumento de recolección de datos para esta tesis.

2.2.6.2. Research-Based Web Design and Usability Guidelines (U.S. Department of Health and Human Services, 2006)

En este estudio se establecen directrices o indicadores guía para el desarrollo de cualquier sitio web, en especial, para aquellos sitios web de salud médica. La investigación se organiza en capítulos los cuales van desde el análisis y diseño de un sitio web, pasando por los capítulos de experiencia del usuario, accesibilidad del sitio, consideraciones de software y hardware para su despliegue, criterios de navegabilidad y características de maquetado que debiera tener todo sitio web. Finaliza con el capítulo de pruebas de usabilidad de un sitio web.

La Tabla 2 muestra los 18 capítulos del libro y la cantidad de indicadores por cada capítulo. Los capítulos han sido traducidos para su mejor compresión.

Tabla 2 **Capítulos y cantidad de indicadores para el desarrollo de un sitio web**

Capítulo	N° indicadores
Diseño y evaluación del proceso	11
Optimización de la experiencia del usuario	16
Accesibilidad	13
Hardware y software	5
Página de inicio	9
Maquetado de la página	13
Navegación	12
Paginación y desplazamiento	5
Encabezados, títulos y etiquetas	8
Enlaces	14
Apariencia del texto	11
Listas	9
Controles basados en pantalla	25
Gráficos, imágenes y multimedia	16
Escritura del contenido web	11
Organización del contenido	9
Búsqueda	9

Pruebas de usabilidad	18

Fuente. (U.S. Department of Health and Human Services, 2006)

De este estudio hemos adaptado los indicadores de los siguientes cuatro capítulos: maquetado de la página; encabezados, títulos y etiquetas; apariencia del texto; gráficos, imágenes y multimedia, por considerarlos relevantes para nuestro tema de investigación.

2.2.7. Análisis de validez por juicio de expertos

El análisis de validez por juicio de expertos es sumamente útil cuando se desea determinar la validez de cualquier instrumento de recolección de datos el cual se desea aplicar en una determinada investigación. A continuación, presentamos un estudio que analiza la validez de un instrumento de medición mediante el coeficiente V de Aiken aplicando el criterio de jueces.

2.2.7.1. Cuantificación de la validez de contenido por criterio de jueces (Escurra, 1988)

En esta investigación se aplica los métodos: Índice de acuerdo (IA), la Prueba binomial (PB) y el coeficiente v de Aiken (V). Estos métodos permiten certificar que el contenido de un determinado instrumento de recolección de datos es adecuado o válido para garantizar que los datos que se obtendrán, producto de aplicar el instrumento de recolección de datos sobre la muestra representativa en un trabajo de investigación, son aptos para comprobar la hipótesis de investigación.

La validez del contenido de un instrumento de recolección de datos es determinada por el grado de consenso que existe entre un grupo de expertos en el tema de investigación. Este grupo de expertos evalúan cada uno de los indicadores del instrumento de medición usando una escala de evaluación que puede ser cualitativa o cuantitativa. Es así que, al proceso de evaluación de la validez del contenido de un determinado instrumento de medición, por parte de un grupo de expertos reconocidos en el tema de investigación, es llamado juicio de expertos.

Para determinar el grado de consenso de un grupo de expertos sobre la validez de un determinado instrumento de medición se aplica el coeficiente v de Aiken por ser fácil de calcular matemáticamente mediante el Índice de acuerdo (IA) y la posibilidad de contrastar estadísticamente sus resultados mediante la Prueba binomial (PB). Este método se calcula como se muestra en la Figura 11. Cabe indicar que los resultados de aplicar la fórmula mostrada caerán entre valores que van desde 0 a 1 y que mientras más se acerquen a la unidad mayor será el grado de validez.

$$V = \frac{S}{(n\,(c\,-1))}$$

Donde:
S = la suma de si
si = valor asignado por el juez i
n = número de jueces
c = número de valores de la escala de valoración

Figura 11 **Coeficiente V de Aiken.**

Fuente: *(Escurra, 1988)*

En la Tabla 3 se muestran los resultados de aplicar el método Índice de acuerdo (IA), la Prueba binomial (PB) y el Coeficiente V de Aiken (V) junto con el nivel de significancia estadística **p,** según el número de acuerdos de 5 a 10 jueces. En ella se interpreta que los valores a ser aceptados son preferentemente aquellos que tienen un coeficiente de entre 0.80 a 1.00.

Tabla 3 **Coeficientes V de Aiken para 10 evaluadores**

Jueces	Acuerdos	IA	PB	V	p
5	3	0.60	.312	0.60	
	4	0.80	.156	0.80	
	5	1.00	.031	1.00	.032
6	4	0.67	.234	0.67	
	5	0.83	.094	0.83	

	6	1.00	.016	1.00	0.16
7	5	0.71	.164	0.71	
	6	0.86	.054	0.86	
	7	1.00	.008	1.00	.008
8	6	0.75	.109	0.75	
	7	0.88	.031	0.88	.035
	8	1.00	.004	1.00	.004
9	7	0.77	.070	0.77	
	8	0.89	.018	0.89	.020
	9	1.00	.002	1.00	.002
10	8	0.80	.043	0.80	.049
	9	0.90	.009	0.90	.001
	10	1.00	.000	1.00	.001

Fuente. (Escurra, 1988)

El estudio concluye que si se desea evaluar si el contenido de un instrumento de medición es válido o no, aplicando el criterio de jueces, se recomienda altamente la aplicación del coeficiente V de Aiken. Además, notamos que, para un grupo de cinco jueces, lograr un coeficiente de 0.80 de aprobación, que representa el consenso total de cuatro jueces, es un signo válido de aceptación del instrumento de medición evaluado.

2.2.8. Muestreo estadístico

Al realizar una investigación científica es importante tener en cuenta el tamaño de la población a estudiar. Así, en el caso de poblaciones demasiado grandes se recomienda sólo la selección de un grupo representativo perteneciente a esta población, la cual recibe el nombre de muestra, y este proceso de selección es llamado muestreo. El muestreo se clasifica en el muestreo probabilístico el cual utiliza métodos estadísticos para asignar a cada elemento de la población iguales probabilidades de formar parte de la muestra, y en el muestreo no probabilístico, que selecciona a los elementos de la muestra según un rasgo o característica que el investigador considere conveniente para la investigación (Hernandez & Carpio, 2019).

Entre las técnicas utilizadas para el muestreo no probabilístico se encuentra la técnica de muestreo por conveniencia, la cual permite seleccionar a elementos de la muestra cuando la población es demasiado grande para ser evaluada y las muestras se encuentran convenientemente disponibles y accesibles para el investigador. De esta forma, se concibe un método práctico, más rápido y rentable para el investigador, pues los participantes se encuentran dispuestos y motivos a participar en la investigación (Otzen & Manterola, 2017).

2.2.9. Metodología SCRUM

Los conceptos a continuación explicados han sido extraídos de (Palacio, 2020), donde se detallan los roles, artefactos, eventos y la herramienta de medición de avance de la metodología de desarrollo ágil Scrum.

Scrum es una metodología ágil que considera como factores esenciales de la calidad al equipo humano y al desarrollo iterativo e incremental confiando en la capacidad y creatividad del equipo. Sus fundamentos son: pronta entrega del sistema solicitado, reflexión constante sobre el proceso de trabajo y un ritmo continuo y constante de trabajo. Sus valores son: apreciación del talento humano, confianza del trabajo en equipo, personas trabajando bajo una meta en común. No obstante, se hace necesario el apoyo de la gerencia de la empresa y de su compromiso y apoyo al equipo supliendo todas sus necesidades y atendiendo todos sus requerimientos.

Los roles de Scrum son: propietario del producto quien gestiona la pila del producto, interactúa constantemente con el cliente y hace llegar sus requerimientos al equipo; desarrolladores quienes son los responsables de desarrollar el incremento solicitado manteniendo un trabajo constante; scrum master quien es el que modera las reuniones diarias y busca soluciones a las dificultades identificadas para contribuir con el avance del equipo y del trabajo.

Los artefactos producidos por Scrum son: pila del producto o llamada también historias de usuario, son las que representan las funcionalidades que deben incorporarse al sistema en cada sprint; pila del sprint que es la descomposición en tareas para desarrollar cada historia de usuario en un determinado sprint; incremento que es el

entregable desarrollado en el sprint y que se encuentra terminado, probado, operativo y cumple con la calidad especificada para el sistema.

Los eventos de Scrum son: sprint que es conocido como iteración y representa cada fase de trabajo con una duración fija y constante; reunión de planificación del sprint donde empieza el sprint, se determina su objetivo y las tareas a realizar; scrum diario que es una reunión cotidiana para verificar el ritmo de avance o identificar si existe algún problema que impida el avance del trabajo, en este evento cada persona es responsable de actualizar el tiempo o esfuerzo pendiente de sus tareas dentro de la pila del sprint. Al revisar un sprint se analiza e inspecciona el incremento desarrollado y luego se procede a actualizar la pila del producto en caso fuese necesario; la retrospectiva del sprint es una reunión llevada a cabo al final del sprint donde se analiza los aspectos operativos del trabajo del equipo y se gestiona el plan de mejoras, en caso sea necesario, para la siguiente iteración.

El gráfico burndown también llamado gráfico de avance sirve para monitorizar el ritmo de avance y es actualizado por los desarrolladores durante el sprint, si fuese el caso, diariamente. Es representado en el plano cartesiano considerando al eje Y como los puntos de historia no completados y al eje X como la cantidad de días de duración del sprint, la diagonal representa el avance ideal planificado. Si trazamos una línea según el avance del trabajo y ésta se mantiene constantemente sobre la diagonal la mayor cantidad de días, entonces indica que se requerirá más tiempo para finalizar el sprint, en caso contrario si desciende rápidamente es señal de que el sprint se terminará antes de lo planificado.

2.2.10. Lenguaje de modelado unificado - UML

Los conceptos teóricos a continuación explicados han sido extraídos de (No Magic, Inc., 2010) donde se describe el diagrama de caso de uso, el diagrama de clases, el diagrama de actividad, el diagrama de secuencia, el diagrama de componentes y el diagrama de despliegue pertenecientes al Lenguaje de modelado unificado (UML).

UML proporciona figuras estandarizadas para representar componentes, operaciones, propiedades y relaciones entre los elementos que conforman las aplicaciones software

y permite al equipo de desarrollo entender rápidamente las funcionalidades contenidas dentro de éstas. Provee un modelado visual para diseñar, en primer lugar, el sistema mediante el dibujo de diagramas y luego emplear herramientas para convertir estos diagramas en código fuente. Se entiende entonces que un diagrama es equivalente a un "plano" sobre el cual se plasman los requerimientos de los usuarios y en muchas ocasiones se requieren de diferentes tipos de diagramas o "planos" para un mismo sistema.

La versión 2.5 de UML define 13 diagramas que son agrupados en 4 + 1 vistas de arquitectura de software. Estas vistas de arquitectura son: vista de caso de uso que representa la funcionalidad y el comportamiento de un sistema o subsistema y está destinada principalmente a mostrar éstas funcionalidades a los clientes, diseñadores, desarrolladores y evaluadores del software; la vista de estructura representa la estructura de los elementos que hacen posible implementar la solución dada a los requerimientos de los usuarios, identificando todas las entidades de negocio y su relación; la vista de comportamiento muestra el comportamiento constante del sistema y se centra principalmente en las interacciones entre sus objetos internos así como las actividades y trabajos realizadas por las partes del sistema.

La vista de implementación describe los artefactos que hacen posible la implementación de los subsistemas lógicos definidos en la vista de estructura (pueden incluir archivos de código, librerías, archivos de datos u otros) y define además las dependencias entre los componentes de implementación y sus conexiones a través de las interfaces requeridas y proveídas; la vista de entorno representa la disposición física de los módulos de un sistema instalados en los dispositivos físicos de interconexión llamados nodos (servidores, computadores, impresoras, módems u otros requeridos).

El diagrama de casos de uso muestra las relaciones entre los casos de uso donde un caso de uso es la descripción de una funcionalidad que provee el sistema. Los casos de uso son descritos sólo en términos del comportamiento externo que muestra el sistema a los usuarios que interactúen con él, mas no describe cómo son realizadas éstas funcionalidades. Los elementos principales de este diagrama son: actores que representan un rol realizado por humanos, hardware independiente del sistema u otros elementos; los casos de uso que representa un tipo de conducta del sistema; la relación

“include” que indica que la instancia de un caso de uso A puede contener la funcionalidad especificada por el caso de uso B; la relación “extend” que define cómo y cuándo la funcionalidad del caso de uso A es extendida al caso de uso B; la relación “association” indica la participación de un actor sobre un caso de uso; la relación “generalization” que representa la relación entre un elemento general y sus elementos específicos.

El diagrama de clases es una representación gráfica de la estructura interna del sistema donde se muestran las clases, sus atributos o características, sus operaciones o funciones y sus interfaces que permiten la interconexión unas con otras. Las clases representan los tipos de objetos manejados por el sistema y no muestra información dinámica, sólo estática. Los elementos principales del diagrama de clases son: la clase que es la descripción que se da al elemento que representa a un conjunto de objetos con estructura, comportamiento y relaciones similares; la interface que es el nombre que recibe un conjunto de funciones visibles expuestas por la clase; la relación “generalization” que representa la relación entre una clase general y sus clases específicas; la relación “association” indica la conexión entre clases, así como de sus objetos; la relación “n-ary association” que representa la asociación entre dos o más clases; la relación “association class” es una declaración de la relación semántica entre clases; la relación “aggregation” que indica la conexión entre la clase agregada y una clase componente; la relación “composition” que indica que las clases componentes no pueden existir sin la clase compuesta.

El diagrama de actividad se centra en el manejo de las actividades del flujo interno de procesamiento del sistema y es clasificada dentro de los diagramas de interacción enfocándose en el trabajo realizado por el sistema en lugar de las interacciones entre sus objetos. Cualquier diagrama de actividad recibe como entrada acciones y muestra como salidas los resultados del procesamiento de estas acciones. Los elementos principales del diagrama de actividad son: “acciones” que es un elemento que representa una unidad de funcionalidad ejecutable cuya ejecución representa alguna transformación o procesamiento en el modelado del sistema; el elemento “initial node” representa el punto de inicio de todos los flujos dentro del diagrama de actividad; el elemento “activity final” representa el final de todos los flujos dentro del diagrama de actividad; el elemento “decision” es un nodo de control que permite elegir la actividad

a realizar en base a la evaluación de una condición dada; el elemento “swimlanes” permite organizar y asignar acciones separadas por clases.

El diagrama de secuencia muestra la interacción entre objetos del sistema en torno a una secuencia de tiempo y posee dos ejes: el vertical simboliza el tiempo y el horizontal los objetos participantes. Los elementos principales del diagrama de secuencia son: el “lifeline” que representa la existencia de un objeto durante un tiempo en particular; el elemento “activation bar” representa el período durante el cual un objeto está realizando alguna acción; el elemento “message” representa el flujo de comunicación entre objetos para realizar una acción; el elemento “message to self” permite a un objeto enviarse mensajes hacia sí mismo.

El diagrama de componentes describe componentes software y sus dependencias donde un componente es una parte del software encapsulada, reutilizable, reemplazable, tiene mayor responsabilidad que una clase e interactúa con otros a través de interfaces requeridas y proveídas. Los elementos principales del diagrama de componentes son: componentes que deben de ser usados para definir los requisitos de cada elemento software físico; las interfaces que son todas las funcionalidades realizadas por un componente en particular; la relación “interface realization” que representa que un componente implementa la interface; la relación “usage” que indica que un componente requiere de otro para su completa implementación u operación; el puerto representa los puntos de interacción entre los componentes y su entorno especificando que cualquier petición que requerida por este puerto será atendida.

El diagrama de despliegue representa la arquitectura física del sistema en tiempo de ejecución donde se muestran los procesadores, los dispositivos (nodos) y los artefactos software que se ejecutan en dicha arquitectura. Los nodos son objetos físicos (dispositivos) y los artefactos son ficheros físicos, librerías, ficheros fuente u otros que el software usa. Los elementos principales del diagrama de despliegue son: nodos que representan un objeto físico con capacidad de procesamiento; el entorno de ejecución que es usado para indicar que un nodo es un entorno de ejecución; el dispositivo representa el hardware con capacidad de procesamiento sobre el cual son desplegados los artefactos; los artefactos representan una porción de información que es usada o producida durante el desarrollo del sistema.

2.2.11. Hypertext Markup Language - HTML

Es un lenguaje de desarrollo web que usa etiquetas para definir el diseño de una página Eeb y, mediante el uso de hipertexto, organiza la información para poder enlazarla y compartirla entre fuentes diversas por medio de enlaces (Apaza, 2017).

El código fuente escrito a través de etiquetas HTML es interpretado por el navegador web, el cual lo interpreta y transforma en una página web, proporcionando información en cuanto a contenido de la página. HTML hace uso de las metaetiquetas que brindan información relevante a los motores de búsqueda y no son mostrados a través del navegador pero indican a las SEO datos como: palabras clave usadas en la página, autor de la página, una breve descripción de la página, entre otros (Pilamunga, 2012).

La anatomía de un documento HTML está compuesta por las etiquetas: *<DOCTYPE html>* es una etiqueta que indica que el archivo con el que se está trabajando es un documento bajo la tecnología HTML, *<html>* el cual engloba el contenido total de la página incluyendo aquellos contenidos que no son mostrados al usuario como las metaetiquetas, *<head>* es el contenedor de todos los parámetros a incluir o importar en la página los cuales no son visibles para los visitantes pero son necesarios para el posicionamiento de la página por los motores de búsqueda, *<meta charset="utf-8">* el cual establece la codificación utf-8 para incluir la mayoría de caracteres de todos los idiomas del mundo, *<title>* para establecer el título de la página que aparece en la pestaña del navegador web, *<body>* contiene todo el contenido visible a los usuarios que visiten la página web pudiendo ser estos texto, videos, entre otros (MDN Web Docs, 2021).

2.2.12. Cascading Style Sheets - CSS

Es un lenguaje de desarrollo web basado en reglas usado para controlar la interfaz de los documentos creados con HTML pues definen estilos utilizados para controlar la presentación, el formato y la apariencia de un documento de marcado. Las reglas que usa generalmente se componen de selectores, los cuales representa a las etiquetas HTML correspondientes a elementos dentro de una página web, y un grupo de

declaraciones compuestas de propiedades y sus valores correspondientes (Apaza, 2017).

Para que la tecnología CSS se aplique sobre los elementos HTML existen las maneras siguientes de realizarlo: importar la hoja de estilo dentro de la etiqueta <*head*> del archivo HTML, escribir un fragmento de código CSS dentro del propio documento HTML contenido dentro de la etiqueta <*style*>, escribir un fragmento de código CSS dentro de la etiqueta HTML de un elemento de la página como parte del atributo *style* de dicho elemento. Las ventajas de maquetar páginas web con CSS son: nos permite una clara separación entre el contenido y los estilos de la página, reducción del tráfico en el servidor, mejora del tiempo de carga de una página web, mejora del mantenimiento de la página pues con tan sólo cambiar parcial o totalmente la hoja de estilo se puede cambiar también el estilo de la página web (Pilamunga, 2012).

Como se mencionó una *propiedad* y su *valor* conforman una declaración y el navegador web interpreta que declaraciones aplican a cada elemento de la página web, mediante el *selector* que lo precede el cual representa algunos elementos de la página, para estilizarlos y mostrarlos adecuadamente. Las propiedades y los valores distinguen entre mayúsculas y minúsculas y se separan mediante el carácter *dos puntos* ":", de izquierda a derecha, respectivamente. Existen más de 100 propiedades CSS diferentes con sus respectivos valores, pero no todos los valores son aceptados por una propiedad, sino que es la propiedad quien define que valores son válidos o posibles de ser asignados a esta. Estas declaraciones se agrupan en bloques llamados *bloques de declaraciones*, el cual es una estructura definida por una *llave de apertura* "}", y una *llave de cierre* "{", dentro del cual van las declaraciones separadas por un *punto y coma* ";" (MDN Web Docs, 2021).

2.2.13. JavaScript

Es un lenguaje interpretado de desarrollo web que no necesita ser compilado y se ejecuta directamente en el navegador web de la maquina cliente. Es utilizado principalmente para el desarrollo de sistemas web pues dota de funcionalidad a las páginas web creadas con HTML (Apaza, 2017).

JavaScript se utiliza principalmente del lado del cliente dotando de funcionalidad y permitiendo la interacción del sistema web con el usuario. Todos los navegadores modernos interpretan el lenguaje JavaScript el cual se integra a las páginas web construidas con HTML mediante la importación del archivo que contiene el script dentro de la etiqueta <*head*> o mediante la inserción de trozos de código de JavaScript dentro del propio documento HTML mediante la etiqueta <*script*> (Pilamunga, 2012).

2.2.14. Hypertext Preprocessor - PHP

Es uno de los lenguajes de desarrollo web del lado del servidor ampliamente usado a nivel mundial que se ocupa de contestar las peticiones que el usuario realiza a través de los protocolos de comunicación: http, ftp, smtp, entre otros para permitir la interacción con los clientes principalmente mediante el control de sesiones, cookies y tokens. Entre sus funciones podemos destacar la creación, modificación y eliminación de diferentes tipos de archivos, la gestión de la lógica de negocio mediante la ejecución de todo tipo de algoritmos codificados con su sintaxis la cual es una de las más fáciles de manejar, la realización de conexiones con los motores de base de datos: MySQL, SQL, Oracle y otros para almacenar, actualizar o eliminar datos permanentemente (Nagilla, 2012).

2.2.15. JavaScript Object Notation - JSON

Es un formato para el intercambio de datos de texto plano ampliamente usado en el desarrollo de sistemas basados en la web el cual es sumamente ligero lo cual es una ventaja que hace que el envío de datos desde el servidor hacia el cliente sea rápida y simple. Los tipos de datos que maneja son: cadenas de texto, números, objetos, arreglos, booleanos y nulos. La cantidad de datos que puede almacenar no tiene límite y la organización de su contenido va desde una simple estructura clave – valor hasta evolucionar a estructuras de objetos complejos (JSON, 2021).

CAPÍTULO III: METODOLOGÍA

En este capítulo explicaremos la metodología seguida para el desarrollo de la presente tesis de maestría. Iniciaremos indicando el tipo y diseño de investigación, continuando con la unidad de análisis, la población y muestra del estudio, describiremos la técnica y el instrumento de recolección de datos utilizado y, finalmente, describiremos el mecanismo de análisis e interpretación de la información.

3.1. Tipo y diseño de la investigación

3.1.1. Tipo de investigación

La investigación realizada es de tipo correlacional, debido a que se estableció la influencia del framework basado en reconocimiento de voz (variable independiente) sobre la automatización del maquetado de páginas web (variable dependiente), en la cual se determinó que la presencia de la variable independiente afecta positivamente a la variable dependiente mediante la evaluación estadística de los indicadores de maquetado web analizados. Usando estadística inferencial pudimos realizar generalizaciones a partir de las muestras analizadas (Murillo, 2013).

3.1.2. Diseño de la investigación

El diseño de la investigación realizada es de carácter pre-experimental y se utilizó el sistema de representación universal (Murillo, 2013).

$$\mathbf{G} \quad \boldsymbol{O_1} \quad \mathbf{X} \quad \boldsymbol{O_2}$$

Donde:

$\boldsymbol{O_1}$**:** Medición previa de la variable dependiente o, antes del diseño e implementación del framework, basado en reconocimiento de voz (pre-experimento)

$\boldsymbol{X}$**:** Framework basado en reconocimiento de voz (variable independiente)

$\boldsymbol{O_2}$: Medición posterior de la variable dependiente o, después del diseño e implementación del framework, basado en reconocimiento de voz (post-experimento)

3.2. Unidad de análisis

La unidad de análisis en esta investigación viene a ser aquel estudiante dedicado al diseño de páginas web que pertenece al noveno ciclo de la Escuela de Ingeniería de Sistemas e Informática de la UNASAM.

3.3. Población de estudio

La población de estudio está conformada por todas aquellas personas dedicadas al diseño web que pertenecen a instituciones públicas o privadas en general, sin distinción de su ubicación, tamaño o rubro.

3.4. Tamaño de la muestra

La muestra, en este caso, está conformada por una institución pública peruana, la cual es la Universidad Nacional Santiago Antúnez de Mayolo (UNASAM). Dentro de esta institución se ha elegido particularmente a los alumnos del noveno ciclo de la Escuela de Ing. de Sistemas e Informática, los cuales son un grupo de quince estudiantes dedicados al diseño de páginas web.

3.5. Selección de la muestra

La muestra, para este trabajo de investigación, es no probabilística utilizándose la técnica de muestreo por conveniencia (Hernandez & Carpio, 2019) para su selección.

Esta decisión se vuelve realmente obvia al considerar el gran tamaño de la población que abarca instituciones que se encuentran fuera del alcance del investigador y por ende no son posibles de evaluar. Considerar a la UNASAM como institución de muestra, por ser una universidad ampliamente conocida por el investigador y convenientemente disponible para el trabajo de investigación, hace que el desarrollo de la investigación se facilite y rentabilice pues los participantes se encuentran motivados a participar en el aprendizaje e implementación del framework.

El perfil de los participantes seleccionados cumple los criterios de inclusión y exclusión, (Arias-Gomez, Villasís-Keever, & Miranda-Novales, 2016), siguientes:

3.5.1. Criterios de inclusión

- Alumnos que pertenecen a la Escuela de Ingeniería de Sistemas e Informática de la UNASAM.

- Alumnos que pertenecen o se encuentren matriculados en el ciclo académico 2021-1.

- Alumnos que acepten ser parte de la investigación voluntariamente.

3.5.2. Criterio de exclusión

- Alumnos que no pertenecen a la Escuela de Ingeniería de Sistemas e Informática de la UNASAM.

- Alumnos que no pertenecen o no se encuentren matriculados en el ciclo académico 2021-1.

- Alumnos que no acepten ser parte de la investigación de manera voluntaria.

3.6. Técnica de recolección de datos

La técnica de recolección de datos aplicada en esta tesis ha sido un cuestionario, que representa el instrumento de recolección de datos, conformado por 6 dimensiones y 31 indicadores de maquetado web, mostrados en la Tabla 4. El cuestionario, mostrado en los Anexos 10 y 11, ha sido elaborado con el programa Google Forms y contiene los 31 indicadores de maquetado web en forma de preguntas cerradas con 5 alternativas valorizadas del 1 al 5 en función a la escala de Likert, siendo posible elegir sólo una de ellas por indicador. En la Tabla 5 se muestran las escalas con las que se evalúan las respuestas del cuestionario.

Tabla 4 **Listado de dimensiones e indicadores del cuestionario**

N°	Dimensiones	Muy Deficiente	Deficiente	Regular	Eficiente	Muy Eficiente
	Animación					
1	El maquetado mantiene un diseño de animación y movimiento estándar					
2	El maquetado evita el uso de gifs animados					
3	El maquetado usa animaciones creadas con CSS, evitando el uso de tecnologías desfasadas de animación (flash)					
4	El maquetado mantiene la coherencia entre las animaciones y el contenido para su mejor entendimiento					
	Botones					
5	El maquetado mantiene un diseño de botones e íconos estándar					
6	El maquetado utiliza colores para indicar la acción a realizar (verde para grabar, rojo para eliminar, entre otros)					

7	El maquetado utiliza iconos según la funcionalidad a realiza (basurero para eliminar, flecha izquierda para retroceder, entre otros)					
8	El maquetado evita el uso innecesario de iconos y gráficos					
9	El maquetado cambia de color o forma para enfatizar la interacción entre el elemento y el usuario					
Contenido						
10	El maquetado mantiene un diseño de contenido estándar					
11	El maquetado permite el movimiento vertical dinámico del encabezado de las columnas de una tabla con más de dos filas					
12	El maquetado evita el uso innecesario de fotografías					
13	El maquetado evita o limita el uso de mapas de imágenes					
14	El maquetado brinda una descripción por escrito de cualquier información crítica contenida en los archivos de audio publicados					
15	El maquetado utiliza los mismos colores para agrupar elementos relacionados (menús de navegación, íconos, datos relacionados, entre otros)					
Fondo						
16	El maquetado mantiene un diseño de fondo estándar, evitando tener fondos de colores diferentes para cada página					
17	El maquetado asegura diseños de fondo sencillos para mejorar la legibilidad del texto					

18	El maquetado asegura que las imágenes de fondo no hagan imposible la lectura y comprensión de la página					
19	El maquetado cuenta con colores de fondo para proporcionar el máximo contraste, evitando las sombras					
20	El maquetado cuenta con el color blanco o colores claros, para el fondo					
	Imágenes					
21	El maquetado mantiene un diseño de imágenes estándar					
22	El maquetado asegura que todas las imágenes sean nítidas y fáciles de visualizar					
23	El maquetado mantiene la cantidad de colores en sus imágenes al mínimo					
24	El maquetado evita las imágenes *peige* o marrones					
25	El maquetado utiliza imágenes entrelazadas o progresivas					
	Tipografía					
26	El maquetado mantiene un diseño de tipografía estándar					
27	El maquetado cuenta con textos fáciles de leer e interpretar por el usuario					
28	El maquetado evita texto de colores amarillo o blanco, ya que son difíciles de leer e imprimir, y evita la combinación de rojo y azul					
29	El maquetado evita el uso de letras mayúsculas en todo el texto					
30	El maquetado evita el texto parpadeante					
31	El maquetado utiliza solo un número limitado de fuentes cuidadosamente seleccionadas					

Fuente. Elaboración propia

Tabla 5 **Valores de la escala de Likert utilizados en el cuestionario**

Escala	Respuesta
1	Muy Deficiente
2	Deficiente
3	Regular
4	Eficiente
5	Muy Eficiente

Fuente. Elaboración propia

Los indicadores de maquetado web, según el estado del arte, sirven como directrices altamente recomendadas durante el proceso de maquetado para asegurar su aceptación por parte de los usuarios finales (Szigeti, 2012). La obtención de indicadores inició con la búsqueda de investigaciones sobre estos. Durante la actividad de búsqueda encontramos dos investigaciones resaltantes para esta tesis. El primer trabajo se titula: Web Design Guidelines for WSDM (Jarrar, 2002), que propone la aplicación de un conjunto de indicadores como aporte a la fase de implementación del diseño, la cual es parte de la metodología de diseño web WSDM (De Troyer, 2001). El segundo trabajo de investigación es titulado como: Research-Based Web Design & Usability Guidelines (U.S. Department of Health and Human Services, 2006), que establece una metodología completa de diseño web abarcando las fases de análisis, maquetado propiamente y pruebas de usabilidad de un sitio web.

Una vez obtenidos los indicadores que forman parte del cuestionario, proseguimos con su validación mediante el análisis de validez por juicio de expertos en diseño web, a quienes se les envió el instrumento mencionado. Estos expertos en diseño web fueron elegidos mediante un muestreo no probabilístico usando la técnica muestreo por conveniencia y fueron contactados mediante correo electrónico. En el ANEXO 1 se muestra el archivo excel enviado a cada experto conteniendo los indicadores y sus dimensiones.

La Tabla 6 muestra el resultado del procesamiento estadístico aplicando la V de Aiken (Aiken, 1985) para la validación, mediante juicio de expertos, de las dimensiones e indicadores del cuestionario. Se analizaron las respuestas de 5 expertos, de los cuales

4 concordaron con los indicadores y sus dimensiones, obteniendo un coeficiente V de 0.8 de concordancia entre jueces, un nivel en el cual se puede concluir que los jueces coinciden en la correcta redacción y consistencia del instrumento de recolección de datos (Escurra, 1988).

Tabla 6 **Análisis de validez por juicio de expertos**

Jueces	Acuerdo	IA	PB	V	P
5	4	0.8	0.156	0.8	0.032

Fuente. Elaboración propia

3.7. Análisis e interpretación de la información

La medición de la variable dependiente, maquetado de páginas web, inició con la aplicación del cuestionario para obtener los datos del pre-test, la cual fue realizada por el autor para garantizar que los resultados obtenidos reflejen el grado de conocimiento de los participantes con respecto a los indicadores evaluados mediante el maquetado de una página web de prueba. Posteriormente, procedimos a explicar y aplicar el framework propuesto mediante el maquetado de la misma página web de prueba para que, finalizada la capacitación y diseño de la página, se entregue a los participantes el link del cuestionario del post-test, el cual fue llenado según sus criterios. Luego de obtener los cuestionarios llenados del pre-test y post-test se procedió a realizar el análisis de datos mediante la comprobación de la hipótesis usando las técnicas estadísticas medidas de correlación (Vinuesa, 2016) y T de Student.

Una vez obtenidos los resultados de aplicar las medidas de correlación y T de Student, para cada indicador del cuestionario, procedimos con la interpretación de los resultados estadísticos obtenidos, considerando dos enfoques: el primero fue el enfoque de ingeniería, como sustento técnico – operativo del motivo del resultado y, el segundo enfoque, considera el marco teórico como sustento bibliográfico de dicho resultado. Finalmente, hemos documentado las conclusiones y recomendaciones producto de la presente tesis de maestría realizada.

CAPÍTULO IV: FRAMEWORK PROPUESTO

Esta tesis de maestría investiga la relación existente entre el reconocimiento de voz y el maquetado web, aplicando la tecnología Web Speech API (MDN Web Dcos, 2021) y CSS 3 (MDN Web Docs, 2021), respectivamente, para generar de forma automática hojas de estilo gestionadas a través de comandos de voz. Para ello desarrolla la Interface MFML: Modelo físico – Modelo lógico, considerándola como el puente que hace posible la integración entre ambas teorías.

Para poder entender la relación anterior partiremos del análisis de la estructura de una hoja de estilo CSS 3. Este archivo es el resultado de desarrollar código CSS 3 para el maquetado de páginas o sistemas web y, por ende, sobre él se escriben bloques de código que constan de selectores, dentro de los cuales van las declaraciones compuestas por propiedades y sus respectivos valores. Además de los selectores y declaraciones, una hoja de estilo CSS 3 contiene también descriptores (declaraciones descendientes directas de reglas-AT), comentarios y reglas-AT: lineales y anidadas. Las reglas-AT anidadas son aquellas que permiten anidar bloques de código y otras reglas-AT, también anidadas, dentro de sí mismas. Las reglas-AT lineales en cambio no permiten ningún nivel de anidamiento; es decir, no tienen descendientes.

Para poder automatizar la generación de hojas de estilo a través de comandos de voz se tiene que analizar la estructura que define tales hojas y cuál es su proceso de construcción. Este proceso de construcción se rige por la operación de inserción y eliminación de elementos CSS. La operación de inserción permite al usuario añadir nuevos selectores, declaraciones, reglas-AT, descriptores y comentarios teniendo en cuenta que las declaraciones van dentro de los selectores y que los descriptores definen a las reglas-AT y, por ende, son contenidos por éstas. La operación eliminar elementos permite al usuario quitar selectores, declaraciones, reglas-AT, descriptores y comentarios considerando que al eliminar todas las declaraciones o descriptores de un selector o una regla-AT, respectivamente, se tienen que eliminar también estos elementos contenedores.

Por otro lado, conocer las operaciones del proceso de construcción normal de cualquier hoja de estilo CSS no es suficiente si se desea automatizar la generación de éstas hojas, pues se tiene que considerar también: el orden de inserción, el número de elemento y la jerarquía de cualquier elemento CSS dentro de una hoja de estilo. Conocer el número de elemento resulta crucial para gestionar la estructura interna, el orden de inserción determina el número de elemento y define también la jerarquía de los elementos categorizándolos como elementos padre o hijos.

Existe una tercera operación posible de realizar la cual es llamada actualización. La actualización ordena de manera automática el número de cada elemento y es invocada implícitamente por la operación eliminar luego de quitar algún elemento de la hoja pues se necesita actualizar el orden de inserción, la numeración y, si fuera el caso, actualizar también las jerarquías de los elementos restantes.

En respuesta a la necesidad de gestionar la estructura y las operaciones propias del proceso de construcción de una hoja de estilo CSS se ha desarrollado un modelo matemático que permite automatizar las operaciones de inserción, actualización y eliminación y una teoría de nomenclaturas que permite administrar el orden de inserción, el número y la jerarquía de cualquier elemento CSS.

4.1. Interface MFML

La interface MFML: modelo físico – modelo lógico, es la encargada de gestionar la automatización del maquetado web mediante el uso de comandos de voz. Las teorías desarrolladas como soporte a esta interface son el modelo matemático y la teoría de nomenclaturas. El modelo matemático permite automatizar las operaciones normales de construcción de una hoja de estilo como lo son las operaciones de: inserción, actualización, eliminación. La teoría de nomenclaturas, es desarrollada como apoyo al modelo matemático y sirve para administrar el orden de inserción, el número y la jerarquía de cualquier elemento CSS dentro de una hoja de estilo.

El modelo matemático, la teoría de nomenclaturas y la interface MFML son desarrolladas a continuación.

4.1.1. Modelo matemático

Como se mencionó anteriormente el modelo matemático permite automatizar las operaciones de inserción, actualización y eliminación que forman parte del proceso de construcción de una hoja de estilo CSS (MDN Web Docs, 2021). Es así que la estructura de éstas hojas sigue un patrón de desarrollo incremental en cuanto al número de elementos y al número de descendientes. Esto es, cada elemento contenido en dichas hojas se define por las dos características siguientes: cantidad de elementos del mismo tipo y cantidad de elementos descendientes clasificados en niveles de profundidad o anidamiento.

Se sabe entonces que si se desea automatizar la generación de una hoja de estilo CSS se tiene que satisfacer la necesidad de gestionar eficientemente las características o variables de: cantidad de elementos y nivel de anidamiento. Resulta apropiado entonces representar la estructura de la hoja de estilo CSS 3 en el plano cartesiano asumiendo al eje X como la cantidad de reglas-AT anidadas padre o la cantidad de selectores o comentarios y, al eje Y, como el nivel de anidamiento o número de descendientes de una regla-AT anidada padre. Se interpretará a la unidad (01) de dos formas: la primera será como el nivel inicial de anidamiento a partir del cual se empezarán a anidar los elementos contenidos y la segunda forma será como el nivel único de profundidad de los elementos que no contienen descendientes. La primera forma representará a las reglas-AT anidadas y la segunda forma será usada para gestionar reglas-AT lineales, selectores o comentarios.

Se entiende entonces que, si trazamos una recta diagonal de izquierda a derecha a la cual llamaremos “L”, tomando como origen el punto (0, 0) del plano, representará la cantidad infinita de elementos descendientes soportados por el modelo matemático de tal manera que para localizar y administrar un elemento del modelo sólo será necesario especificar las coordenadas (x, y): número de elemento y nivel de anidamiento en el cual se encuentra. Resulta evidente que el universo del modelo son los enteros positivos; es decir, el modelo matemático se encuentra definido en el segundo cuadrante del plano cartesiano bidimensional y sólo acepta cantidades positivas.

Denominaremos a la variable “n” como el n-ésimo elemento dentro de la hoja de estilo clasificado según su tipo: regla-AT, selector o comentario. Y consideraremos a la variable “j” como el j-ésimo nivel del elemento anidado o nivel de anidamiento total del elemento padre, en el caso de reglas-AT anidadas. Cabe aclarar que el nivel de anidamiento se refiere a la cantidad de bloques de código encerrados dentro de otro bloque superior, siendo éstos sólo reglas-AT anidadas, que por definición pueden contener selectores u otras reglas-AT anidadas, los cuales inician un nuevo bloque de código.

Los selectores que no se encuentren anidados dentro de ninguna regla-AT son elementos de nivel (j = 1); es decir, no contienen descendientes. Se hace especial énfasis en este criterio pues se desea aclarar que para que exista un nivel de anidamiento se tienen que anidar reglas-AT contenedoras de bloques de código y no asumir que los elementos dentro de un simple bloque de código, como en el caso de declaraciones pertenecientes a selectores lineales, puedan generar anidamiento. Lo mismo sucede con las reglas-AT lineales, las cuales contienen dentro de sí descriptores, cuya estructura es similar al de las declaraciones pues contienen propiedades y sus respectivos valores, pero no por eso se puede asumir que los elementos descriptores contenidos formen niveles de anidamiento, sino que sólo componen un bloque de código simple.

En la Figura 12 se presenta el modelo matemático representado en el plano cartesiano bidimensional considerando el eje X como el número de selectores, comentarios o reglas-AT lineales o anidadas, los cuales se encuentran en el primer nivel de anidamiento. El eje Y representa el nivel de anidamiento originado como resultado de anidar una regla-AT hija dentro de una regla-AT padre, considerando que una regla-AT anidada es aquella que contiene, entre otros elementos, a otra regla-AT con bloques de código compuestos por descriptores o selectores. Como se puede notar los elementos del modelo matemático se encuentran definidos en el primer cuadrante y sólo aceptan valores positivos los cuales son incrementados en función a la cantidad de elementos que formen parte de una hoja de estilo.

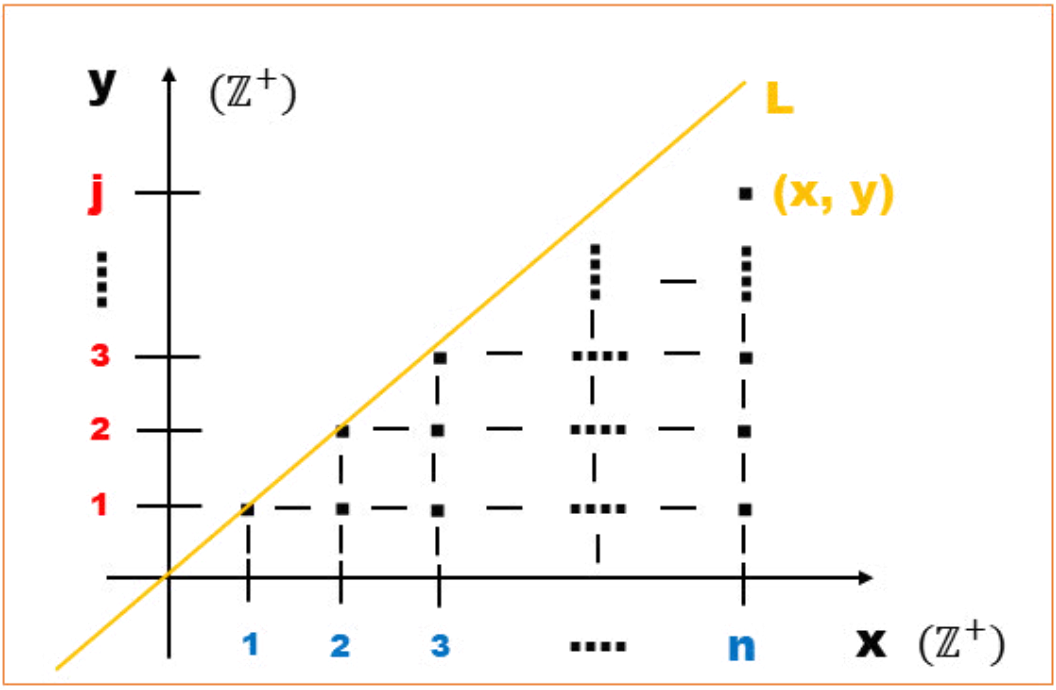


Figura 12 **Modelo matemático - Interface MFML.**

Fuente: Elaboración propia

Donde:

X: Número de reglas-AT anidadas padre, es decir, reglas-AT de nivel 1

Y: Nivel de anidamiento, es decir, el número de descendientes de una regla-AT anidada padre

L: Recta definida por puntos (x, y) que representan la n-ésima regla-AT de nivel j-ésimo

Z+: Los valores n-ésimos y j-ésimos pertenecen a enteros positivos

El modelo matemático mostrado permite realizar las operaciones: insertar, actualizar y eliminar a continuación detalladas.

4.1.1.1. Operación insertar

La operación insertar es la primera de las operaciones posibles de realizar dentro de cualquier hoja de estilo CSS 3, y en el modelo matemático presentado, también es posible. Ésta operación se descompone en cuatro (04) posibles tipos de inserción correspondientes a los elementos: reglas-AT, descriptores, selectores y declaraciones.

Así, por ejemplo, para hacer referencia a la operación insertar una nueva regla-AT se invocaría a la función: f (ins AT) y, continuando con la secuencia anterior, las funciones: f (ins Ds), f (ins S) y f (ins D) representarían el llamado de inserción de los descriptores, selectores y declaraciones respectivamente.

$$\mathbf{f(insert)} = \mathrm{f(ins\ AT)}\ \mathbf{or}\ \mathrm{f(ins\ Ds)}\ \mathbf{or}\ \mathrm{f(ins\ S)}\ \mathbf{or}\ \mathrm{f(ins\ D)}$$

Definición:

$$\mathbf{f(ins\ AT)} = \mathrm{AT}\begin{pmatrix} \mathrm{j} \\ (\mathrm{n}+1) \end{pmatrix} \quad \text{or} \quad \mathrm{AT}\begin{pmatrix} (\mathrm{j}+1) \\ \mathrm{n}=1 \end{pmatrix}$$

$$\boldsymbol{f(ins\ Ds)} = AT\begin{pmatrix} j \\ n \end{pmatrix} \ (Ps(h+1) : Vs(h+1))$$

$$\mathbf{f(ins\ S)} = \mathrm{AT}\begin{pmatrix} \mathrm{j} \\ \mathrm{n} \end{pmatrix} \ \mathrm{S(m+1)} \ \ \mathbf{or} \ \ \mathrm{S(m+1)}$$

$$\mathbf{f(ins\ D)} = \mathrm{AT}\begin{pmatrix} \mathrm{j} \\ \mathrm{n} \end{pmatrix} \ \mathrm{S(m+1)} \ \big(\mathrm{P(k+1) : V(k+1)}\big) \ \ \mathbf{or}$$

$$\mathrm{S(m+1)} \ \ (\mathrm{P(k+1) : V(k+1)})$$

Donde:

$\boldsymbol{AT}\begin{pmatrix} \boldsymbol{j} \\ \boldsymbol{n} \end{pmatrix}$	**:** n-ésima regla-AT anidada padre de nivel j-ésimo
$\boldsymbol{Ds}$	**:** descriptor perteneciente a la n-ésima regla-AT anidada padre de nivel j-ésimo
$\boldsymbol{Ps}$**(h + 1)**	**:** (h-ésima + 1) propiedad perteneciente al nuevo descriptor Ds(h + 1)
V$\boldsymbol{s}$**(h + 1)**	**:** (h-ésimo + 1) valor perteneciente al nuevo descriptor Ds(h + 1)
S(m + 1)	**:** (m-ésimo + 1) nuevo selector perteneciente, o no, a la regla-AT anidada padre
$\boldsymbol{D}$	**:** declaración perteneciente al m-ésimo selector S perteneciente, o no, a la regla-AT padre
$\boldsymbol{P}$**(k + 1)**	**:** (k-ésima + 1) propiedad perteneciente a la nueva declaración D(k + 1)
V(k + 1)	**:** (k-ésimo + 1) valor perteneciente a la nueva declaración D(k + 1)

La operación de inserción de una nueva regla-AT: $f(ins\ AT)$, conlleva a realizar una de las dos acciones siguientes. La primera es la adición de la nueva regla-AT en el mismo nivel j-ésimo de la regla-AT padre desde donde fue invocada la operación, provocando sólo el incremento (n + 1) del número de reglas-AT anidadas en dicho nivel, a lo cual llamaremos: generación de un nuevo AT hermano: $AT\binom{j}{(n+1)}$. La segunda acción que puede implicar la inserción de la nueva regla-AT es su adición en el siguiente nivel j-ésimo de la regla-AT desde donde fue invocada la operación, provocando sólo el incremento (j + 1) del nivel de anidamiento, a lo cual llamaremos: generación de un nuevo AT descendiente o hijo: $AT\binom{(j+1)}{n=1}$.

La operación de inserción de un nuevo descriptor: $f(ins\ Ds)$, conlleva a adicionar su propiedad (Ps) en el mismo nivel j-ésimo de la n-ésima regla-AT desde donde fue invocada la operación, provocando sólo el incremento (h + 1) del número de propiedades de dicho descriptor: $AT\binom{j}{n}\ Ps(h+1)$ y, conlleva también a adicionar el valor (Vs) respectivo de la propiedad también en el mismo nivel j-ésimo de la n-ésima regla-AT desde donde fue invocada la operación, provocando sólo el incremento (h + 1) del número de valores de dicho descriptor : $AT\binom{j}{n}\ Vs(h+1)$.

La operación de inserción de un nuevo selector: $f(ins\ S)$, conlleva a realizar una de las dos acciones siguientes. La primera es la adición del nuevo selector en el mismo nivel j-ésimo de la n-ésima regla-AT padre desde donde fue invocada la operación, provocando sólo el incremento (m + 1) del número de selectores anidados en dicho nivel: $AT\binom{j}{n}\ S(m+1)$. La segunda acción que puede implicar la inserción del nuevo selector es su adición en el primer nivel (j = 1) de la hoja de estilo, provocando sólo el incremento (m + 1) del número de selectores: $S(m+1)$.

La operación de inserción de una nueva declaración: $f(ins\ D)$, perteneciente a un selector anidado, conlleva a adicionar su propiedad (P) en el m-ésimo selector anidado perteneciente a la n-ésima regla-AT de nivel j-ésimo desde donde fue invocada la operación, provocando sólo el incremento (k + 1) del número de propiedades de dicho selector anidado: $AT\binom{j}{n}\ S(m+1)\ P(k+1)$ y, conlleva también a adicionar el valor (V) respectivo de la propiedad también en el m-ésimo selector anidado perteneciente

a la n-ésima regla-AT de nivel j-ésimo desde donde fue invocada la operación, provocando sólo el incremento (k + 1) del número de valores de dicho selector anidado: $AT\binom{j}{n}\ S(m+1)\ V(k+1)$.

Por otro lado, la operación de inserción de una nueva declaración: $f(ins\ D)$, perteneciente a un selector no anidado de nivel (j = 1), conlleva a adicionar su propiedad (P) en el m-ésimo selector desde donde fue invocada la operación, provocando sólo el incremento (k + 1) del número de propiedades de dicho selector: $S(m+1)\ P(k+1)$ y, conlleva también a adicionar el valor (V) respectivo de la propiedad también en el m-ésimo selector desde donde fue invocada la operación, provocando sólo el incremento (k + 1) del número de valores de dicho selector: $S(m+1)\ V(k+1)$.

4.1.1.2. Operación actualizar

La operación actualizar es la segunda de las operaciones posibles de realizar dentro de cualquier hoja de estilo CSS 3, y en el modelo matemático presentado, también es posible. Ésta operación se descompone en cuatro (04) posibles tipos de actualización correspondientes a los elementos: reglas-AT, descriptores, selectores y declaraciones. Así, por ejemplo, para hacer referencia a la operación actualizar una regla-AT se invocaría a la función: f (upd AT) y, continuando con la secuencia anterior, las funciones: f (upd Ds), f (upd S) y f (upd D) representarían el llamado de actualización de los descriptores, selectores y declaraciones respectivamente.

Es preciso mencionar que la actualización es una operación que se ejecuta de forma automática inmediatamente después de finalizada la operación de eliminación. Es decir, la actualización del orden, número y jerarquía de cualquier elemento dentro de la hoja de estilo CSS no es invocada explícitamente, sino que se lleva a cabo como respuesta automática al eliminar cualquier elemento de la hoja, siendo éstos: reglas-AT, descriptores, selectores, declaraciones o comentarios.

Se ha considerado conveniente usar como símbolo una flecha que rota en sentido antihorario sobre el eje vertical Y del plano cartesiano bidimensional, para indicar la ejecución de la actualización, y una fecha horizontal de izquierda a derecha para

indicar que se asignará una nueva numeración al elemento actualizado y a todos sus elementos descendientes en cada uno de sus niveles de anidamiento en caso los tuviera, la cual será calculada restando una unidad (01) a su numeración actual. Esta nueva numeración es representada con el símbolo prima (‘).

$$\mathbf{f(update)} = \mathrm{f(upd\ AT)}\ \mathbf{or}\ \mathrm{f(upd\ Ds)}\ \mathbf{or}\ \mathrm{f(upd\ S)}\ \mathbf{or}\ \mathrm{f(upd\ D)}$$

Definición:

$$\mathbf{f(upd\ AT)} = \mathrm{AT}\binom{j}{n} \Rightarrow \mathrm{AT}\binom{j}{n'}\ (\mathrm{Ds(h)})\ \mathbf{and}\ \mathrm{AT}\binom{j}{n'}\ (\mathrm{S(m)})$$

$$\mathbf{f(upd\ Ds)} = \mathrm{Ds(h)} \Rightarrow \mathrm{Ps(h')}\ \mathbf{:}\ \mathrm{Vs(h')}$$

$$\mathbf{f(upd\ S)} = \mathrm{S(m)} \Rightarrow \mathrm{S(m')}\ \mathbf{and}\ \mathrm{S(m')}\ (\mathrm{D(k)})$$

$$\mathbf{f(upd\ D)} = \mathrm{D(k)} \Rightarrow \mathrm{P(k')}\ \mathbf{:}\ \mathrm{V(k')}$$

Donde:

$\boldsymbol{AT}\binom{\boldsymbol{j}}{\boldsymbol{n'}}$	**:** (n-ésima - 1) regla-AT anidada padre de nivel j-ésimo
$\boldsymbol{Ds(h)}$	**:** h-ésimo descriptor perteneciente a la regla-AT anidada padre
S(m)	**:** m-ésimo selector perteneciente, o no, a la regla-AT anidada padre
$\boldsymbol{Ps}$**(h’)**	**:** (h-ésima - 1) propiedad perteneciente al descriptor Ds(h’)
V$\boldsymbol{s}$**(h’)**	**:** (h-ésimo - 1) valor perteneciente al descriptor Ds(h’)
S(m’)	**:** (m-ésimo - 1) selector perteneciente, o no, a la regla-AT anidada padre
$\boldsymbol{D}$**(k)**	**:** k-ésima declaración perteneciente al selector S(m’)
$\boldsymbol{P}$**(k’)**	**:** (k-ésima - 1) propiedad perteneciente a la declaración D(k’)
V(k’)	**:** (k-ésimo - 1) valor perteneciente a la declaración D(k’)

La operación de actualización de una regla-AT: $f(upd\ AT)$, conlleva a realizar las tres acciones siguientes. La primera es la disminución de la numeración de las reglas-AT que estén en el mismo nivel j-ésimo de la n-ésima regla-AT eliminada desde donde fue invocada la operación, provocando el decremento (n - 1) del número de reglas-AT en dicho nivel: $AT\binom{j}{n'}$. La segunda acción que implica la actualización de una regla-AT es la disminución (n - 1) de la numeración de los descriptores hijos de cada una de las reglas-AT actualizadas en la primera acción: $AT\binom{j}{n'}\ (Ds(h))$. La tercera acción

que implica la actualización de una regla-AT es la disminución (n - 1) de la numeración de los selectores hijos de cada una de las reglas-AT actualizadas en la primera acción: $AT\binom{j}{n'}$ $(S(m))$.

La operación de actualización de un descriptor: $f(upd\ Ds)$, conlleva a disminuir la numeración de cada propiedad (Ps) perteneciente a cada descriptor hermano del descriptor eliminado desde donde fue invocada la operación, provocando el decremento (h - 1) del número de propiedades: $Ps(h')$ y, conlleva también a disminuir la numeración de cada valor (Vs) perteneciente a cada descriptor hermano del descriptor eliminado desde donde fue invocada la operación, provocando el decremento (h - 1) del número de valores: $Vs(h')$.

La operación de actualización de un selector: $f(upd\ S)$, conlleva a realizar las dos acciones siguientes. La primera es la disminución de la numeración de los selectores que estén en el mismo nivel j-ésimo del selector eliminado desde donde fue invocada la operación, provocando el decremento (m - 1) del número de selectores en dicho nivel: $S(m')$. La segunda acción que implica la actualización de un selector es la disminución (m - 1) de la numeración de las declaraciones hijas de cada uno de los selectores actualizados en la primera acción: $\mathrm{S(m')}$ $(D(k))$.

La operación de actualización de una declaración: $f(upd\ D)$, conlleva a disminuir la numeración de cada propiedad (P) perteneciente a cada declaración hermana de la declaración eliminada desde donde fue invocada la operación, provocando el decremento (k - 1) del número de propiedades: $P(k')$ y, conlleva también a disminuir la numeración de cada valor (V) perteneciente a cada declaración hermana de la declaración eliminada desde donde fue invocada la operación, provocando el decremento (k - 1) del número de valores: $V(k')$.

4.1.1.3. Operación eliminar

La operación eliminar es la tercera de las operaciones posibles de realizar dentro de cualquier hoja de estilo CSS 3, y en el modelo matemático presentado, también es posible. Ésta operación se descompone en cuatro (04) posibles tipos de eliminación correspondientes a los elementos: reglas-AT, descriptores, selectores y declaraciones.

Así, por ejemplo, para hacer referencia a la operación eliminar una regla-AT se invocaría a la función: f (del AT) y, continuando con la secuencia anterior, las funciones: f (del Ds), f (del S) y f (del D) representarían el llamado de eliminación de los descriptores, selectores y declaraciones respectivamente.

Cuando se ejecuta la operación eliminar implícitamente también se ejecuta la operación actualizar pues se tiene que reestablecer el orden, número y jerarquía de los hermanos a partir del elemento eliminado. Esto quiere decir que al eliminar un elemento de la hoja de estilo siempre se actualizará la numeración de los elementos del mismo tipo y nivel de anidamiento a partir del elemento eliminado, denominándose a estos elementos como elementos hermanos. Adicionalmente, se verifica también si el elemento eliminado forma parte de un elemento contenedor y si es el último de los elementos contenidos, pues si tal fuese el caso, entonces se eliminará también al elemento contenedor.

Así, por ejemplo, al eliminar la última declaración de un selector m-ésimo, se eliminará también a dicho selector pues no tendría ninguna declaración contenida. O en el caso de eliminar al último descriptor perteneciente a una n-ésima regla-AT se tendrá que eliminar también a dicha regla-AT. Evidentemente esta lógica también es aplicable cuando se elimina al elemento contenedor, sea éste un selector o una regla-AT lineal o anidada. Por ejemplo, al eliminar a un selector también se tendría que eliminar a todas sus declaraciones o al eliminar una n-ésima regla-AT padre de nivel j-ésimo, se tendría que eliminar también a todos sus elementos descendientes.

Se ha considerado conveniente usar como símbolo una flecha vertical de arriba hacia abajo para indicar que la eliminación de una regla-AT anidada conllevará implícitamente a eliminar también a todos sus elementos descendientes en cada uno de sus niveles de anidamiento en caso los tuviera, aplicándose la misma lógica para los selectores. Sin embargo, al eliminar un descriptor perteneciente a una regla-AT lineal o una declaración perteneciente a un selector sólo se eliminará a dicho elemento, sin la necesidad de eliminar a ningún otro elemento adicional, esto es representado con una flecha horizontal de izquierda a derecha.

$$\boldsymbol{f(delete)} = f(del\ AT)\ \boldsymbol{or}\ f(del\ Ds)\ \boldsymbol{or}\ f(del\ S)\ \boldsymbol{or}\ f(del\ D)$$

Definición:

$$\mathbf{f(del\ AT)} = \mathbf{AT}\binom{\mathbf{j}}{\mathbf{n}} \Rightarrow \mathbf{f(upd\ AT)}$$

$$\mathbf{f(del\ Ds)} = \mathbf{Ds(h)} \Rightarrow \mathbf{f(upd\ Ds)}$$

$$\mathbf{f(del\ S)} = \mathbf{S(m)} \Rightarrow \mathbf{f(upd\ S)}$$

$$\mathbf{f(del\ D)} = \mathbf{D(k)} \Rightarrow \mathbf{f(upd\ D)}$$

Donde:

$\boldsymbol{AT}\binom{\boldsymbol{j}}{\boldsymbol{n}}$	**:** n-ésima regla-AT anidada padre de nivel j-ésimo
$\boldsymbol{Ds(h)}$	**:** h-ésimo descriptor perteneciente a la regla-AT anidada padre
S(m)	**:** m-ésimo selector perteneciente, o no, a la regla-AT anidada padre
$\boldsymbol{D}$**(k)**	**:** k-ésima declaración perteneciente al selector S(m)
$\boldsymbol{f(upd\ AT)}$	**:** función actualizar regla-AT
$\boldsymbol{f(upd\ Ds)}$	**:** función actualizar descriptor
$\boldsymbol{f(upd\ S)}$	**:** función actualizar selector
$\boldsymbol{f(upd\ D)}$	**:** función actualizar declaración

La operación de eliminación de una regla-AT: $f(del\ AT)$, conlleva a realizar las dos acciones siguientes. La primera es el borrado descendente de todas las reglas-AT hijas y sus respectivos descriptores, selectores y declaraciones, a partir del j-ésimo nivel de la regla-AT padre eliminada desde donde fue invocada la operación. La segunda acción que implica la eliminación de una regla-AT es la invocación de la operación actualizar; es decir, disminuir la numeración: (n - 1) = (n'), de las reglas-AT hermanas que se encuentran en el mismo nivel j-ésimo de la regla-AT eliminada y de sus respectivos elementos descriptores, selectores y declaraciones en forma descendente.
La operación de eliminación de un descriptor: $f(del\ Ds)$, conlleva a realizar las dos acciones siguientes. La primera es el borrado del descriptor desde donde fue invocada la operación. La segunda acción que implica la eliminación del descriptor es la invocación de la operación actualizar; es decir, disminuir la numeración: (h - 1) = (h'), de todos los descriptores hermanos que se encuentran en el mismo nivel j-ésimo del descriptor eliminado.

La operación de eliminación de un selector: $f(del\,S)$, conlleva a realizar las dos acciones siguientes. La primera es el borrado descendente de todas las declaraciones pertenecientes al selector eliminado desde donde fue invocada la operación. La segunda acción que implica la eliminación de un selector es la invocación de la operación actualizar; es decir, disminuir la numeración: (m - 1) = (m'), de los selectores hermanos que se encuentran en el mismo nivel j-ésimo del selector eliminado y de sus respectivas declaraciones en forma descendente.

La operación de eliminación de una declaración: $f(del\,D)$, conlleva a realizar las dos acciones siguientes. La primera es el borrado de la declaración desde donde fue invocada la operación. La segunda acción que implica la eliminación de la declaración es la invocación de la operación actualizar; es decir, disminuir la numeración: (k - 1) = (k'), de todas las declaraciones hermanas que se encuentran en el mismo nivel j-ésimo de la declaración eliminada.

4.1.2. Teoría de nomenclaturas

Como se mencionó anteriormente la teoría de nomenclaturas, desarrollada como apoyo al modelo matemático, permite administrar el orden de inserción, el número y la jerarquía de cualquier elemento dentro de una hoja de estilo CSS. Entonces partiremos del hecho de que los elementos contenidos en dichas hojas son: reglas-AT anidadas o lineales, descriptores, propiedades de descriptores, valores de descriptores, selectores, declaraciones, propiedades de declaraciones, valores de declaraciones y comentarios.

A diferencia del modelo matemático, que modela las operaciones de inserción, actualización y eliminación que forman parte del proceso de construcción de una hoja de estilo CSS, la teoría de nomenclaturas implementa estas operaciones estableciendo un mecanismo de identificación, un orden de inserción, un número correlativo de elemento según su tipo y una jerarquía para cada elemento CSS dentro de la hoja de estilo. Para ello se considera el uso de una tabla html tradicional la cual consta de filas, columnas y elementos contenedores de texto a los cuales llamaremos elementos textarea. Por definición se sabe que las filas contienen columnas y éstas, a su vez, contienen otros elementos. No obstante, la teoría de nomenclaturas considera que una

fila contendrá como máximo dos columnas y que una columna sólo contendrá un elemento textarea.

Los lineamientos descritos en el párrafo anterior son las bases establecidas por la teoría de nomenclaturas: se usará una tabla html tradicional para representar a cualquier elemento CSS dentro de una hoja de estilo, cuyas filas sólo podrán contener como máximo dos columnas (en el caso de que la fila represente a una declaración o un descriptor los cuales constan de propiedades y valores) y éstas, a su vez, contendrán solamente un elemento contenedor de texto llamado textarea, el cual representará el valor de dicho elemento. Siendo éste el caso, la abreviación establecida para referirnos a las reglas-AT es “AT”, para hacer referencia a los descriptores usaremos la abreviación “ds”, la abreviación para las propiedades de los descriptores es “ps” y para sus respectivos valores usaremos “vs”, la abreviación para los selectores es “s”, los elementos declaraciones se abreviarán con “d” y sus respectivas propiedades y valores serán “p” y “v” respectivamente. Por otro lado, la representación de las filas será “tr”, las columnas se representarán con “td” y los elementos textarea con “txta”.

Una vez establecido el mecanismo de identificación el cual está compuesto de ocho (08) abreviaciones para los elementos CSS: “at”, “ds”, “ps”, “vs”, “s”, “d”, “p”, “v” para referirse a: reglas-AT, descriptores, propiedades de descriptor, valores de descriptor, selectores, declaraciones, propiedades de declaración y valores de declaración respectivamente, y de tres (03) abreviaciones para los elementos de la tabla html: “tr”, “td”, “txta” para referirse a: filas, columnas y elementos contenedores de texto llamados textarea, es momento de explicar también como se gestionarán el orden de inserción y el número, según tipo, de cada elemento CSS dentro de la hoja de estilo. Para ello la teoría de nomenclaturas establece cuatro (04) variables globales. La variable “n” representa el número correlativo de reglas-AT que se encuentran dentro de la hoja de estilo, la variable “h” en cambio representa el número correlativo de descriptores pertenecientes a una determinada regla-AT, por su parte la variable “m” representa el número correlativo de selectores que pertenecen a una regla-AT y la variable “k” representa el número correlativo de declaraciones que forman parte de un selector perteneciente, a su vez, a una regla-AT.

En el párrafo anterior, donde se expone la existencia de cuatro (04) variables globales: “n” para representar el número correlativo de reglas-AT, “h” para representar el número correlativo de descriptores, “m” para representar el número correlativo de selectores y “k” para representar el número correlativo de declaraciones, se evidencia claramente la pertenencia o jerarquía entre elementos CSS. Precisamente, la teoría de nomenclaturas relaciona el mecanismo de identificación compuesto por ocho (08) abreviaciones para los elementos CSS y de tres (03) abreviaciones para los elementos de la tabla html con las cuatro (04) variables globales que permiten gestionar el orden de inserción y el número de elementos CSS según su tipo, enlazándolos para formar las clases e IDs de las filas, columnas y elementos textarea de cada uno de los elementos siguientes: reglas-AT, descriptores, propiedades de descriptores, valores de descriptores, selectores, declaraciones, propiedades de declaraciones, valores de declaraciones.

Entonces la jerarquía entre elementos CSS se gestionará mediante la formación y asignación de clases e IDs a las filas, columnas y elementos textarea de la tabla html las cuales representan a cualquier elemento dentro de una hoja de estilo. La teoría de nomenclaturas se clasifica, según el nivel de anidamiento de las reglas-AT a gestionar en: teoría de nomenclaturas para la administración de reglas-AT lineales y teoría de nomenclaturas para la administración de reglas-AT anidadas.

4.1.2.1. Nomenclatura regla-AT lineal (j = 1)

La teoría de nomenclaturas desarrollada para la regla-AT lineal, mostrada en la Figura 13, establece la formación y asignación de clases e IDs CSS para cada fila, columna y elemento textarea que representan a un determinado elemento dentro de la hoja de estilo, teniendo a la unidad como único nivel de anidamiento (j = 1); es decir, las reglas-AT contenidas no tienen reglas-AT descendientes, pero sí descriptores, selectores o declaraciones y sus respectivas propiedades y valores.

AT:

– tr	tr_cls_at_1	tr_id_at_n_1
– td	td_cls_at_1	td_id_at_n_tr_id_at_n_1
– txta	cls_at_1	txta_id_at_n_td_id_at_n_tr_id_at_n_1

AT DESCRIPTOR:

– tr	tr_cls_ds_tr_id_at_n_1	tr_id_ds_h_tr_id_at_n_1

PROPIEDAD:

– td	td_cls_ps_tr_id_at_n_1	td_id_ps_h_tr_id_ds_h_tr_id_at_n_1
– txta	cls_ps_tr_id_at_n_1	txta_id_ps_h_td_id_ps_h_tr_id_ds_h_tr_id_at_n_1

VALOR:

– td	td_cls_vs_tr_id_at_n_1	td_id_vs_h_tr_id_ds_h_tr_id_at_n_1
– txta	cls_vs_tr_id_at_n_1	txta_id_vs_h_td_id_vs_h_tr_id_ds_h_tr_id_at_n_1

AT SELECTOR:

– tr	tr_cls_s_tr_id_at_n_1	tr_id_s_m_tr_id_at_n_1
– td	td_cls_s_tr_id_at_n_1	td_id_s_m_tr_id_s_m_tr_id_at_n_1
– txta	cls_s_tr_id_at_n_1	txta_id_s_m_td_id_s_m_tr_id_s_m_tr_id_at_n_1

AT SELECTOR DECLARACION:

– tr	tr_cls_d_tr_id_s_m_tr_id_at_n_1	tr_id_d_k_tr_id_s_m_tr_id_at_n_1

PROPIEDAD:

– td	td_cls_p_tr_id_s_m_tr_id_at_n_1	td_id_p_k_tr_id_d_k_tr_id_s_m_tr_id_at_n_1
– txta	cls_p_tr_id_s_m_tr_id_at_n_1	txta_id_p_k_td_id_p_k_tr_id_d_k_tr_id_s_m_tr_id_at_n_1

VALOR:

– td	td_cls_v_tr_id_s_m_tr_id_at_n_1	td_id_v_k_tr_id_d_k_tr_id_s_m_tr_id_at_n_1
– txta	cls_v_tr_id_s_m_tr_id_at_n_1	txta_id_v_k_td_id_v_k_tr_id_d_k_tr_id_s_m_tr_id_at_n_1

Figura 13 **Nomenclatura Regla-AT Lineal (j = 1).**

Fuente: Elaboración propia

Donde:

$\boldsymbol{n}$ **:** número correlativo de la regla-AT lineal

$\boldsymbol{h}$ **:** número correlativo del descriptor perteneciente a la regla-AT lineal

$\boldsymbol{m}$ **:** número correlativo del selector perteneciente a la regla-AT lineal

$\boldsymbol{k}$ **:** número correlativo de la declaración perteneciente al selector perteneciente a la regla-AT lineal

Es sumamente importante tener presente el orden entre filas, columnas y elementos textarea, las que son contenidas unas dentro de otras pues será utilizado para lograr la formación de las clases e IDs mencionados. Así, el ID de cualquier fila será unida al ID de su columna y el ID de esta columna será unida al ID de su elemento textarea. Además, el ID de la fila de cualquier regla-AT: “tr_id_at_n_1”, será utilizada para formar todas las clases e IDs de las filas, columnas y elementos textarea de los

descriptores, selectores, declaraciones, propiedades de descriptores, valores de descriptores, propiedades de declaraciones y valores de declaraciones contenidos por dicha regla-AT.

Partiremos entonces de la formación de la clase CSS para la fila del elemento regla-AT lineal la cual es: “tr_cls_at_1”. Como se verifica esta clase ha sido compuesta usando el mecanismo de identificación abreviado el cual es “tr” para hacer referencia que se trata del elemento fila de la tabla html, la abreviación “cls” que denota que la expresión representa una clase, la abreviación “at” indica que se trata de una clase perteneciente a una regla-AT y el número uno (01) para hacer referencia a la gestión de reglas-AT lineales. Luego de formar la clase para la fila de la regla-AT lineal se procede a formar el ID el cual es: “tr_id_at_n_1”. Este ID claramente está compuesto por las abreviaciones: “tr” que indica que se trata de una fila, “id” que hace referencia que la expresión representa un identificador único (ID), “at” para referenciar que es el ID perteneciente a una regla-AT, la variable global “n” indicando el número correlativo de dicha regla-AT y el número uno (01) para hacer referencia a la gestión de reglas-AT lineales.

En el párrafo anterior donde se explica la formación de la clase e ID de la fila del elemento regla-AT lineal, se distingue claramente la variable global “n”. El valor que tenga esta variable está en función a la clase formada. Esto significa que al momento de insertar una nueva regla-AT lineal, se verificará la cantidad n-ésima de reglas-AT lineales que tengan asignada la clase: “tr_cls_at_1”, y se incrementará en uno (01) para asignarla como parte del ID de la nueva regla-AT a insertar. Posterior a la formación y asignación de la clase e ID de la fila de la regla-AT lineal, se procede a generar y asignar la clase e ID de la columna del elemento regla-AT lineal y para ello se parte del hecho de que la clase será la misma que la clase de la fila: “tr_cls_at_1”. Sin embargo, el ID será formado de la siguiente manera: “td_id_at_n_tr_id_at_n_1”, donde se evidencia claramente que su estructura consta de nueve (09) abreviaciones o es de longitud nueve, tiene la abreviación “td” haciendo referencia que se trata del ID de la columna de una regla-AT y lo principal, contiene como fragmento final el mismo ID de la fila que la contiene.

Por otro lado, la clase del elemento textarea de la regla-AT en cuestión es: “cls_at_1” que a diferencia de las clases de la fila y columna esta inicia directamente con la abreviación “cls”. El ID está formado por: “txta_id_at_n_td_id_at_n_tr_id_at_n_1” donde se evidencia que las nueve (09) últimas abreviaciones corresponden al ID de la columna que la contiene: “td_id_at_n_tr_id_at_n_1” y éste, a su vez, contiene el ID de su respectiva fila: “tr_id_at_n_1”. Con este método de formación de IDs resulta fácil ubicar el elemento columna que la contiene y además nos permite saber cuál es el elemento fila al cual pertenece. La variable “n” es calculada incrementando uno (01) a la cantidad de reglas-AT lineales en el caso de invocar a la operación de inserción, pero en el caso de realizar una actualización, el valor de dicha variable se calculará restando uno (01) al valor que tenga dicha variable en el momento de realizar la actualización, esto evidentemente porque la actualización es invocada inmediatamente después de eliminar el elemento en cuestión, reduciendo en uno (01) el número de reglas-AT lineales.

Continuando con el análisis de la formación de las clases e IDs gestionadas por la teoría de nomenclaturas de reglas-AT lineales se evidencia que en el caso de los descriptores el manejo de las nomenclaturas de sus filas y columnas varían un poco, pues ahora la fila no sólo contendrá una sola columna, sino que albergará a dos columnas que definen su propiedad y su valor, respectivamente. Bajo esta lógica la clase de la fila del descriptor es: “tr_cls_ds_tr_id_at_n_1” lo que quiere decir que la abreviación hace referencia a la clase “cls” de la fila “tr” del descriptor “ds” que pertenece a la regla-AT cuyo ID de fila es: “tr_id_at_n_1” permitiendo el manejo de jerarquías. El ID de la fila del descriptor es: “tr_id_ds_h_tr_id_at_n_1” y como se puede apreciar hace referencia a que se trata del identificador único “id” de la fila “tr” del descriptor “ds” que pertenece a la regla-AT lineal cuyo ID de fila es: “tr_id_at_n_1”. Es significativo mencionar que la variable “h” representa el número de descriptores pertenecientes a la regla-AT en cuestión y su valor se calcula como la cantidad total de elementos descriptores que tengan asignada la clase: “tr_cls_ds_tr_id_at_n_1”.

Es así que la clase asignada a la columna del elemento llamado propiedad del descriptor, mencionado en el párrafo anterior, es: “td_cls_ps_tr_id_at_n_1” que hace referencia a la clase “cls” de la columna “td” del elemento propiedad “ps” de un

descriptor que pertenece a la regla-AT lineal cuyo ID de fila es: “tr_id_at_n_1”; es decir, pertenece a la n-ésima regla-AT de nivel (j = 1). El ID de la columna del elemento propiedad del descriptor es: “td_id_ps_h_tr_id_ds_h_tr_id_at_n_1”, lo cual demuestra claramente que se trata del identificador único “id” de la columna “td” de la h-ésima propiedad “ps” del descriptor “ds” cuyo ID de fila es: “tr_id_ds_h_tr_id_at_n_1”, el cual forma parte de la n-ésima regla-AT lineal (j = 1) con ID de fila: “tr_id_at_n_1”. De igual forma la clase del elemento textarea de la propiedad es: “cls_ps_tr_id_at_n_1” que hace referencia a la clase “cls” del elemento textarea de la propiedad “ps” del descriptor que pertenece a la n-ésima regla-AT cuyo ID de fila es: “tr_id_at_n_1” y el ID de dicho elemento textarea es: “txta_id_ps_h_td_id_ps_h_tr_id_ds_h_tr_id_at_n_1” que referencia al identificador único “id” del elemento textarea “txta” de la h-ésima propiedad “ps” del descriptor “ds” que pertenece a la columna “td” cuyo ID es: “td_id_ps_h_tr_id_ds_h_tr_id_at_n_1” la cual está contenida dentro de la fila “tr” cuyo ID es: “tr_id_ds_h_tr_id_at_n_1” quien pertenece finalmente a la n-ésima regla-AT lineal cuyo ID de fila es: “tr_id_at_n_1”. El mismo procedimiento aplica para el elemento valor del descriptor mencionado, reemplazando sólo la abreviación “ps” por “vs”.

En el caso de la clase asignada a la fila del elemento selector cuya representación es: “tr_cls_s_tr_id_at_n_1” que se refiere a la clase “cls” de la fila “tr” del selector “s” que pertenece a la n-ésima regla-AT cuyo ID de fila es: “tr_id_at_n_1”. El ID de la fila del elemento selector es: “tr_id_s_m_tr_id_at_n_1” que indica que se trata del identificador único “id” de la fila “tr” del m-ésimo selector “s” perteneciente a la regla-AT lineal cuyo ID de fila es: “tr_id_at_n_1”. La clase de la columna del elemento selector es: “td_cls_s_tr_id_at_n_1” que hace referencia a la clase “cls” de la columna “td” del selector “s” perteneciente a la regla-AT lineal cuyo ID de fila es: “tr_id_at_n_1”. El ID de la columna del elemento selector es: “td_id_s_m_tr_id_s_m_tr_id_at_n_1” que representa el identificador único “id” de la columna “td” del m-ésimo selector “s” contenida dentro de la fila “tr” cuyo ID es: “tr_id_s_m_tr_id_at_n_1” que pertenece a la regla-AT lineal cuyo ID de fila es: “tr_id_at_n_1”.

La clase del elemento textarea del selector es: “cls_s_tr_id_at_n_1” que hace referencia a la clase “cls” del elemento textarea del selector “s” que pertenece a la n-ésima regla-AT lineal cuyo ID de fila es: “tr_id_at_n_1”. El ID del elemento textarea del selector es: “txta_id_s_m_td_id_s_m_tr_id_s_m_tr_id_at_n_1” que se interpreta como el identificador único “id” del elemento textarea “txta” del m-ésimo selector “s” ubicado dentro de la columna cuyo ID es: “td_id_s_m_tr_id_s_m_tr_id_at_n_1” la cual está contenida dentro de la fila cuyo ID es: “tr_id_s_m_tr_id_at_n_1” que pertenece a la n-ésima regla-AT lineal con ID de fila: “tr_id_at_n_1”. En este caso la variable “m” representa la cantidad total de selectores ubicados dentro de la n-ésima regla-AT lineal (j = 1), se incrementa en uno (01) al realizar la operación inserción y disminuye también en uno (01) al realizar la operación actualización luego de eliminar un selector perteneciente a dicha n-ésima regla-AT.

En el caso de la clase de la fila de una declaración su formación es la siguiente: “tr_cls_d_tr_id_s_m_tr_id_at_n_1”, lo que quiere decir que la abreviación hace referencia a la clase “cls” de la fila “tr” de la declaración “d” que pertenece al m-ésimo selector “s” perteneciente a la n-ésima regla-AT “at” cuyo ID de fila “tr” es: “tr_id_at_n_1” haciendo posible totalmente el manejo de jerarquías. El ID de la fila de la declaración es: “tr_id_d_k_tr_id_s_m_tr_id_at_n_1” y como se puede apreciar hace referencia a que se trata del identificador único “id” de la fila “tr” de la k-ésima declaración “d” que pertenece al m-ésimo selector “s” el cual forma parte de la regla-AT “at” lineal cuyo ID de fila es: “tr_id_at_n_1”. Es significativo mencionar que la variable “k” representa el número de declaraciones que pertenecen al m-ésimo selector y su valor se calcula como la cantidad total de elementos declaraciones que tengan asignada la clase: “tr_cls_d_tr_id_s_m_tr_id_at_n_1”.

Y de la misma forma la clase de la columna del elemento propiedad de la declaración mencionada en el párrafo anterior es: “td_cls_p_tr_id_s_m_tr_id_at_n_1” que hace referencia a la clase “cls” de la columna “td” del elemento propiedad “p” de una declaración que pertenece al m-ésimo selector perteneciente a la n-ésima regla-AT lineal cuyo ID de fila es: “tr_id_at_n_1”. El ID de la columna del elemento propiedad de la declaración es: “td_id_p_k_tr_id_d_k_tr_id_s_m_tr_id_at_n_1” lo cual demuestra claramente que se trata del identificador único “id” de la columna “td” de la k-ésima propiedad “p” de la declaración “d” cuyo ID de fila es:

“tr_id_d_k_tr_id_s_m_tr_id_at_n_1”, el cual forma parte del m-ésimo selector con ID de fila: “tr_id_s_m_tr_id_at_n_1” perteneciente a la n-ésima regla-AT lineal cuyo ID de fila es: “tr_id_at_n_1”.

De igual forma la clase del elemento textarea de la propiedad es: “cls_p_tr_id_s_m_tr_id_at_n_1” que hace referencia a la clase “cls” del elemento textarea de la propiedad “p” de la declaración que pertenece al m-ésimo selector cuyo ID de fila es: “tr_id_s_m_tr_id_at_n_1” y el ID de dicho elemento textarea es: “txta_id_p_k_td_id_p_k_tr_id_d_k_tr_id_s_m_tr_id_at_n_1”. Como se puede verificar la longitud del ID es de veintiuno (21) abreviaciones las que se han formado con: el identificador único “id” del elemento textarea “txta” de la k-ésima propiedad “p” de la declaración, el identificador único “id” de la columna “td” cuyo ID es: “td_id_p_k_tr_id_d_k_tr_id_s_m_tr_id_at_n_1”, el identificador único “id” de la fila “tr” cuyo ID es: “tr_id_d_k_tr_id_s_m_tr_id_at_n_1” el cual pertenece al m-ésimo selector cuyo ID de fila es: “tr_id_s_m_tr_id_at_n_1” que forma parte de la n-ésima regla-AT lineal cuyo ID de fila es: “tr_id_at_n_1”.

El mismo procedimiento aplica para la clase e ID de los elementos columna y textarea, del elemento valor de la declaración mencionada en los dos párrafos anteriores, reemplazando sólo la abreviación “p” por “v”. Finalmente cabe mencionar que las variables globales: “n”, “h”, “m” y “k” son referentes a la cantidad total de reglas-AT lineales, descriptores, selectores y declaraciones respectivamente.

4.1.2.2. Nomenclatura regla-AT anidada (j > 1)

La teoría de nomenclaturas para la administración de reglas-AT anidadas, mostrada en la Figura 14, es una generalización de la teoría de nomenclaturas para la administración de reglas-AT lineales expuesta en la sección anterior. Esta teoría pone especial énfasis en la gestión del nivel de anidamiento y la longitud de los IDs formados en función de las clases CSS asignadas a cada fila, columna y elemento textarea que contienen a un determinado elemento dentro de una hoja de estilo. Es así que se distinguen a las reglas-AT anidadas padre y reglas-AT anidadas hijo o descendientes, donde una regla-AT anidada puede cumplir el doble rol de ser padre e hija a la vez.

AT:

– tr	tr_cls_at_j	tr_id_at_n_j
– td	td_cls_at_j	td_id_at_n_tr_id_at_n_j
– txta	cls_at_j	txta_id_at_n_td_id_at_n_tr_id_at_n_j

AT DESCRIPTOR:

– tr	tr_cls_ds_tr_id_at_n_j	tr_id_ds_h_tr_id_at_n_j
PROPIEDAD:		
– td	td_cls_ps_tr_id_at_n_j	td_id_ps_h_tr_id_ds_h_tr_id_at_n_j
– txta	cls_ps_tr_id_at_n_j	txta_id_ps_h_td_id_ps_h_tr_id_ds_h_tr_id_at_n_j
VALOR:		
– td	td_cls_vs_tr_id_at_n_j	td_id_vs_h_tr_id_ds_h_tr_id_at_n_j
– txta	cls_vs_tr_id_at_n_j	txta_id_vs_h_td_id_vs_h_tr_id_ds_h_tr_id_at_n_j

AT SELECTOR:

– tr	tr_cls_s_tr_id_at_n_j	tr_id_s_m_tr_id_at_n_j
– td	td_cls_s_tr_id_at_n_j	td_id_s_m_tr_id_s_m_tr_id_at_n_j
– txta	cls_s_tr_id_at_n_j	txta_id_s_m_td_id_s_m_tr_id_s_m_tr_id_at_n_j

AT SELECTOR DECLARACION:

– tr	tr_cls_d_tr_id_s_m_tr_id_at_n_j	tr_id_d_k_tr_id_s_m_tr_id_at_n_j
PROPIEDAD:		
– td	td_cls_p_tr_id_s_m_tr_id_at_n_j	td_id_p_k_tr_id_d_k_tr_id_s_m_tr_id_at_n_j
– txta	cls_p_tr_id_s_m_tr_id_at_n_j	txta_id_p_k_td_id_p_k_tr_id_d_k_tr_id_s_m_tr_id_at_n_j
VALOR:		
– td	td_cls_v_tr_id_s_m_tr_id_at_n_j	td_id_v_k_tr_id_d_k_tr_id_s_m_tr_id_at_n_j
– txta	cls_v_tr_id_s_m_tr_id_at_n_j	txta_id_v_k_td_id_v_k_tr_id_d_k_tr_id_s_m_tr_id_at_n_j

Figura 14 **Nomenclatura Regla-AT Anidada (j > 1).**

Fuente: Elaboración propia

Donde:

n **:** número correlativo de la regla-AT anidada padre

h **:** número correlativo del descriptor perteneciente a la regla-AT anidada padre

m **:** número correlativo del selector perteneciente a la regla-AT anidada padre

k **:** número correlativo de la declaración perteneciente al selector perteneciente a la regla-AT anidada padre

j **:** número correlativo del nivel de anidamiento de la regla-AT anidada padre y sus elementos descendientes

A simple vista pareciera ser que la única diferencia que presenta la teoría de nomenclaturas para reglas-AT anidadas es el cambio, al final de la formación de sus clases e IDs, del número uno (01) presentado en la teoría anterior por la variable global de anidamiento “j”. Esto no debería de ser interpretado tan ligeramente, pues no es del todo cierta. La variable global de anidamiento “j” cumple el rol principal de representar

al número correlativo del nivel de anidamiento en el que se encuentra un determinado elemento: regla-AT, descriptor, selector, declaración, propiedad de descriptor, valor de descriptor, propiedad de declaración, valor de declaración. No obstante, el procedimiento de asignar una clase y un ID a cada fila, columna y elemento contenedor de texto llamado textarea sigue siendo el mismo. Por esta razón, se pondrá mayor atención en la formación de los IDs desde la perspectiva de los niveles de profundidad o anidamiento.

La determinación del nivel j-ésimo de cualquier regla-AT hija dependerá del nivel de la regla-AT padre desde donde se invoca la operación de inserción, lo que significa que el nivel j-ésimo de la nueva regla-AT hija se calculará incrementando una unidad (01) al nivel que tiene su padre: (j + 1). Esta lógica de gestión del nivel de anidamiento permite conocer con precisión las características siguientes de cualquier elemento dentro de una regla-AT anidada: nos permite saber el número correlativo del elemento según su tipo y nivel, nos permite conocer el nivel propiamente en el cual se encuentra y nos permite identificar la regla-AT anidada padre del cual desciende, no importando si es un descendiente directo o si es un j-ésimo descendiente.

Por otro lado, la longitud de los identificadores únicos (IDs), a diferencia de la teoría de nomenclaturas para reglas-AT lineales, difiere en su procedimiento de formación. Esto significa que la longitud de éstos IDs es variable y está en función del nivel en el cual se encontrará la regla-AT hija a insertar. Esto se traduce en la función siguiente: f (longitud ID) = [(nuevo nivel j-ésimo) x (2)], que quiere decir que se añadirán al final de las abreviaciones generales: “tr_at_id”, que conforman el identificador único “id” de la fila “tr” de una regla-AT “at”, una cantidad de dígitos igual al resultado de multiplicar el nivel de la nueva regla-AT por dos (02). Así, por ejemplo, si el nuevo nivel fuese 2 (segundo nivel de descendencia), entonces la longitud de números a insertar al final de las abreviaciones generales de la fila “tr” del elemento regla-AT “at” sería cuatro (04) pues el resultado de multiplicar 2 por 2 es igual a 4, o por ejemplo, si el nuevo nivel fuese 3 (tercer nivel de descendencia), entonces la longitud de números a insertar al final de las abreviaciones generales de la fila “tr” del elemento regla-AT “at” sería seis (06) pues el resultado de multiplicar 3 por 2 es igual a 6.

La interpretación que se da a la longitud o cantidad de números a insertar al final de las abreviaciones generales de la fila “tr” del elemento regla-AT “at”: “tr_at_id”, es: el primer dígito representa la cantidad correlativa de elementos AT en el mismo nivel j-ésimo, el segundo dígito representa el nivel j-ésimo propiamente en el cual se encuentra la regla-AT y, la cantidad de dígitos restantes, representan la numeración insertada al final de las abreviaciones generales del ID de la fila “tr” de la regla-AT “at” padre del cual desciende. Entonces resulta evidente que enlazar, como forma de herencia, la numeración de la regla-AT padre luego de los dos primeros dígitos del ID de la regla-AT hija (número correlativo y nivel propiamente), proporciona un mecanismo óptimo de rastreo para conocer en cualquier instante de tiempo las características de cualquier elemento anidado dentro de una hoja de estilo CSS.

La teoría de nomenclaturas para reglas-AT anidadas no sólo permite la realización de la operación de inserción, también permite las operaciones de eliminación y actualización. Sucede entonces que al eliminar una regla-AT anidada padre se eliminará dicha regla-AT propiamente, así como todos sus elementos hijos, siendo éstos: reglas-AT hijas, descriptores, selectores, declaraciones, propiedades de descriptores, valores de descriptores, propiedades de declaraciones, valores de declaraciones en todos los niveles a partir de la regla-AT anidada padre desde donde se invocó la operación.

Luego de eliminar la regla-AT anidada padre, se actualizará el dígito que representa el número correlativo de reglas-AT en el mismo nivel j-ésimo de la regla-AT eliminada, el cual forma parte de la numeración insertada al final de cada clase e ID de las reglas-AT hermanas y sus elementos descendientes en todos sus niveles de descendencia o anidamiento partiendo de la regla-AT eliminada. Es decir, se disminuirá en uno (01) el valor del dígito que representa el número correlativo de reglas-AT de todas las clases e IDs de las filas, columnas y elementos textarea de aquellas reglas-AT hermanas y de sus elementos descendientes, en todos los niveles de anidamiento, a partir de la regla-AT eliminada. Esto evidentemente debido a que al eliminar una regla-AT se afecta también el valor de los números correlativos que representan la cantidad de reglas-AT en un nivel j-ésimo dado, garantizando así que las características de: número correlativo de elementos según su tipo y nivel, nivel propiamente y la identificación

de la regla-AT anidada padre del cual desciende una regla-AT hija, se mantendrán consistentes en todo momento.

Si bien es cierto que la actualización del dígito que indica el número correlativo de reglas-AT anidadas en un determinado nivel j-ésimo debe llevarse a cabo luego de eliminar una determinada regla-AT anidada padre, es también cierto que no es posible identificar fácilmente dicho dígito, cuyo valor tendrá que ser disminuido en uno (01). Esto es debido a que la ubicación del dígito mencionado está en función a la longitud del ID y éste, a su vez, depende del nivel de anidamiento en el que se encuentra el elemento a actualizar. Para salvaguardar esta situación se ha aplicado el concepto de "tupla" entendiéndose a ésta como la formación de pares de números hasta cubrir el total de dígitos de la numeración del ID de cada elemento en cuestión. Luego de formar las tuplas se procede a identificar el nivel de la regla-AT padre eliminada y el nivel del elemento cuya numeración se desea actualizar para que finalmente se aplique la fórmula siguiente, para determinar el número de tupla cuyo primer dígito será actualizado, con total certeza, pues representa el número correlativo de reglas-AT en un nivel: f (tupla actualizar) = [(nivel elemento hijo + 1) – (nivel padre eliminado)].

Claramente se identifica que la tupla, cuyo valor de su primer dígito se disminuirá en uno (01), es determinado sumando uno (01) al nivel en el que se encuentre el elemento cuya numeración se desea actualizar y restándole el nivel de la regla-AT padre eliminada. Una vez se tenga la nueva numeración, es decir, se haya disminuido uno al primer dígito de la tupla identificada, se procede a recorrer cada uno de los elementos descendientes y los hijos de éste en todos los niveles de anidamiento actualizando sus respectivas clases e IDs. Otro concepto igual de importante viene a ser el hecho que no es necesario actualizar los índices de las numeraciones de los elementos padre de la regla-AT anidada eliminada pues estos no fueron eliminados y por ende no necesitan ser actualizados.

La teoría de nomenclaturas para reglas-AT lineales y anidadas es óptima para administrar el orden de inserción, el número y la jerarquía de cualquier elemento mediante la formación y asignación de clases e IDs a las filas, columnas y elementos textarea de la tabla html las cuales representan a los elementos que pertenezcan o descienda única y exclusivamente de reglas-AT. Sin embargo, dentro de una hoja de

estilo se encuentran elementos CSS que no dependen necesariamente de reglas-AT para su existencia. En estos casos las teorías de nomenclaturas expuestas en las secciones anteriores resultan poco apropiadas. Es así que a continuación se detallan nomenclaturas derivadas, propicias para la gestión de: selectores lineales (que no se encuentran dentro de ningún otro elemento pero que contienen declaraciones) y para la gestión de comentarios que forman parte de una hoja de estilo CSS.

4.1.2.3. Nomenclatura del selector lineal

La teoría de nomenclaturas para la administración de selectores lineales, mostrada en la Figura 15, rige cuando un elemento selector representado por la abreviación que lleva su propio nombre, no desciende de ninguna regla-AT.

SELECTOR:

– tr	tr_cls_selector	tr_id_selector_n
– td	td_cls_selector	td_id_selector_n_tr_id_selector_n
– txta	cls_selector	txta_id_selector_n_td_id_selector_n_tr_id_selector_n

SELECTOR DECLARACION:

– tr	tr_cls_declaracion_tr_id_selector_n	tr_id_declaracion_n_tr_id_selector_n

PROPIEDAD:

– td	td_cls_propiedad_tr_id_selector_n	td_id_propiedad_n_tr_id_declaracion_n_tr_id_selector_n
– txta	cls_propiedad_tr_id_selector_n	txta_id_propiedad_n_td_id_propiedad_n_tr_id_declaracion_n_tr_id_selector_n

VALOR:

– td	td_cls_valor_tr_id_selector_n	td_id_valor_n_tr_id_declaracion_n_tr_id_selector_n
– txta	cls_valor_tr_id_selector_n	txta_id_valor_n_td_id_valor_n_tr_id_declaracion_n_tr_id_selector_n

Figura 15 **Nomenclatura Selector Lineal.** Fuente: Elaboración propia

Donde:

n : número correlativo de los selectores de la hoja de estilos CSS 3 / tr : fila de la tabla / td : columna perteneciente a la fila tr de la tabla txta : textarea perteneciente a la columna td de la fila tr de la tabla / izq |: nomenclatura para representar las Clases CSS de los elementos de la tabla / | der : nomenclatura para representar los IDs CSS de los elementos de la tabla

Sigue los criterios de formación de clases e IDs que el de la teoría de nomenclaturas para la administración de reglas-AT. Asimismo, las columnas “td” heredan al final el ID de la fila “tr” que las contiene y, los elementos textarea heredan a su vez, el ID de la columna “td” que las contiene. Así, por ejemplo, el ID de la fila “tr” de la declaración se constituye de su ID propiamente: “tr_id_declaracion_n”, más el ID del selector al que define: “tr_id_selector_n”, de tal manera que el ID final de la fila “tr” de la declaración es: “tr_id_declaracion_n_tr_id_selector_n”. Y el ID de la columna “td” de la propiedad perteneciente a la declaración es formado uniendo el ID de la fila “tr” de dicha declaración más el ID de la fila “tr” del selector al que define: “td_id_propiedad_n_tr_id_declaracion_n_tr_id_selector_n”.

El ID del elemento textarea “txta” contenido por la columna “td” de la propiedad es: “txta_id_propiedad_n_td_id_propiedad_n_tr_id_declaracion_n_tr_id_selector_n” el cual se forma uniendo el ID de la columna “td”: “td_id_propiedad_n_tr_id_declaracion_n_tr_id_selector_n” la cual está dentro de la fila “tr” de la declaración cuyo ID es: “tr_id_declaracion_n_tr_id_selector_n” perteneciente al selector cuyo ID de fila es: “tr_id_selector_n” el cual cumple la función de abreviación general a partir del cual se forman todas las clases e IDs de sus declaraciones, propiedades y valores de declaraciones.

En el caso de las declaraciones el ID de la fila “tr” del selector del cual desciende es unida al final de su ID: “td_id_declaracion_n_tr_id_selector_n”, con el propósito de indicar explícitamente su pertenencia y número. El número de cada declaración está en función a la clase que se le asigna, la cual contiene el ID de la fila “tr” del selector: “tr_cls_declaracion_tr_id_selector_n”.

La variable “n” indica el número correlativo de selectores lineales (que no pertenece a ninguna regla-AT) y es calculado sumando uno (01) a la cantidad de selectores que tengan asignada la clase: “tr_cls_selector”. De esta forma resulta sumamente fácil administrar las operaciones de inserción, actualización y eliminación de éstos elementos selectores, sus declaraciones y sus respectivas propiedades y valores, así como sus elementos textarea “txta” que contienen los valores de cada elemento, de las columnas “td” que contienen a los textarea y de las filas “tr” que finalmente contienen a las columnas.

De igual forma, se puede inferir una versión más reducida de la teoría de nomenclaturas de reglas-AT, para hacerla encajar con la nomenclatura para la administración de comentarios.

4.1.2.4. Nomenclatura del comentario

En la teoría de nomenclaturas para la administración de comentarios, mostrada en la Figura 16, notamos que el ID de la fila "tr" del elemento comentario contiene la variable "n", la cual representa la cantidad de comentarios existentes en toda la hoja de estilos CSS.

COMENTARIO:

– tr	tr_cls_comentario	tr_id_comentario_n
– td	td_cls_comentario	td_id_comentario_n_tr_id_comentario_n
– txta	cls_comentario	txta_id_comentario_n_td_id_comentario_n_tr_id_comentario_n

Figura 16 **Nomenclatura Comentario.**

Fuente: Elaboración propia

Donde:

$\boldsymbol{n}$ **:** número correlativo de los comentarios de la hoja de estilos CSS 3

$\boldsymbol{tr}$ **:** fila de la tabla

$\boldsymbol{td}$ **:** columna perteneciente a la fila tr de la tabla

$\boldsymbol{txta}$ **:** textarea perteneciente a la columna td de la fila tr de la tabla

izq | **:** nomenclatura para representar las Clases CSS de los elementos de la tabla

| $\boldsymbol{der}$ **:** nomenclatura para representar los IDs CSS de los elementos de la tabla

Además, se observa que el ID de la fila "tr" forma parte del ID de la columna "td" y que, finalmente el ID del elemento textarea "txta" está conformado por el ID de la columna "td" que lo contiene y el ID de la fila "tr" que contiene a la columna: "txta_id_comentario_n_td_id_comentario_n_tr_id_comentario_n".

Las operaciones de inserción, actualización y eliminación también son permitidas. Una característica clave de la nomenclatura para comentarios radica en el hecho de que éstas no forman parte de ningún otro elemento y tampoco contienen ningún otro

elemento. Es posible entonces, catalogarlos como elementos flotantes que pueden ser insertados dentro de una hoja de estilos en cualquier parte y en cualquier momento sin restricciones de ningún tipo.

Hasta el momento se ha detallado el modelo matemático que permite realizar las operaciones de inserción, actualización y eliminación de cualquier elemento CSS y se ha desarrollado también, como complemento al modelo matemático, la teoría de nomenclaturas para describir un mecanismo que nos permita identificar cualquier elemento, su orden de inserción, su número correlativo según su tipo y nivel y su jerarquía mediante la formación y asignación de clases e IDs a las filas, columnas y elementos textarea de una tabla html tradicional para representar a cualquier elemento dentro de una hoja de estilo. Estas teorías permiten presentar finalmente a la Interface MFML: Modelo físico – Modelo lógico, que hace posible la generación automatizada de hojas de estilo CSS 3 mediante el uso de comandos de voz.

4.1.3. Interface MFML: Modelo Físico – Modelo Lógico

La Interface MFML, mostrada en la Figura 17, permite la integración de las tecnologías Web Speech API y CSS; es decir, nos permite usar la voz en el desarrollo de código para la generación de hojas de estilo CSS 3. Entendiéndose como tal al hecho de vincular y ejecutar comandos de voz reconocidos por el framework como un mecanismo de programación.

La interface se concibe entonces con el propósito de permitir el desarrollo de código CSS usando comandos de voz, los cuales son gestionados mediante la tecnología de reconocimiento de voz a través de la API Web Speech, para convertirlos y devolverlos en formato texto y, una vez devueltos estos comandos, se proceda con su identificación y ejecución desencadenando acciones programadas como respuesta, que permitan la gestión de una hoja de estilo dentro del framework. Por otro lado, la interface se concibe también con el propósito de permitir la transformación de una tabla html, la cual gestiona a los elementos CSS de una hoja de estilo dentro del framework, en un archivo .css propiamente, el cual sea apto para ser incluido directamente en la carpeta de estilos del proyecto de maquetado de cualquier sitio web.

Mejor aún, la interface se concibe con el propósito de permitir, una vez exportado el archivo .css, la reconstrucción de la tabla html dentro del framework. Para ello propone la exportación, junto al archivo .css, del archivo .xml el cual cumplirá la función de metadata de la hoja de estilo generada y contendrá toda la información sobre el: número de fila, nombre del elemento, clase de la fila, ID de la fila, clase de la columna, ID de la columna, clase del elemento textarea, ID del elemento textarea, contenido o valor del elemento y el nivel j-ésimo de anidamiento entre reglas-AT.

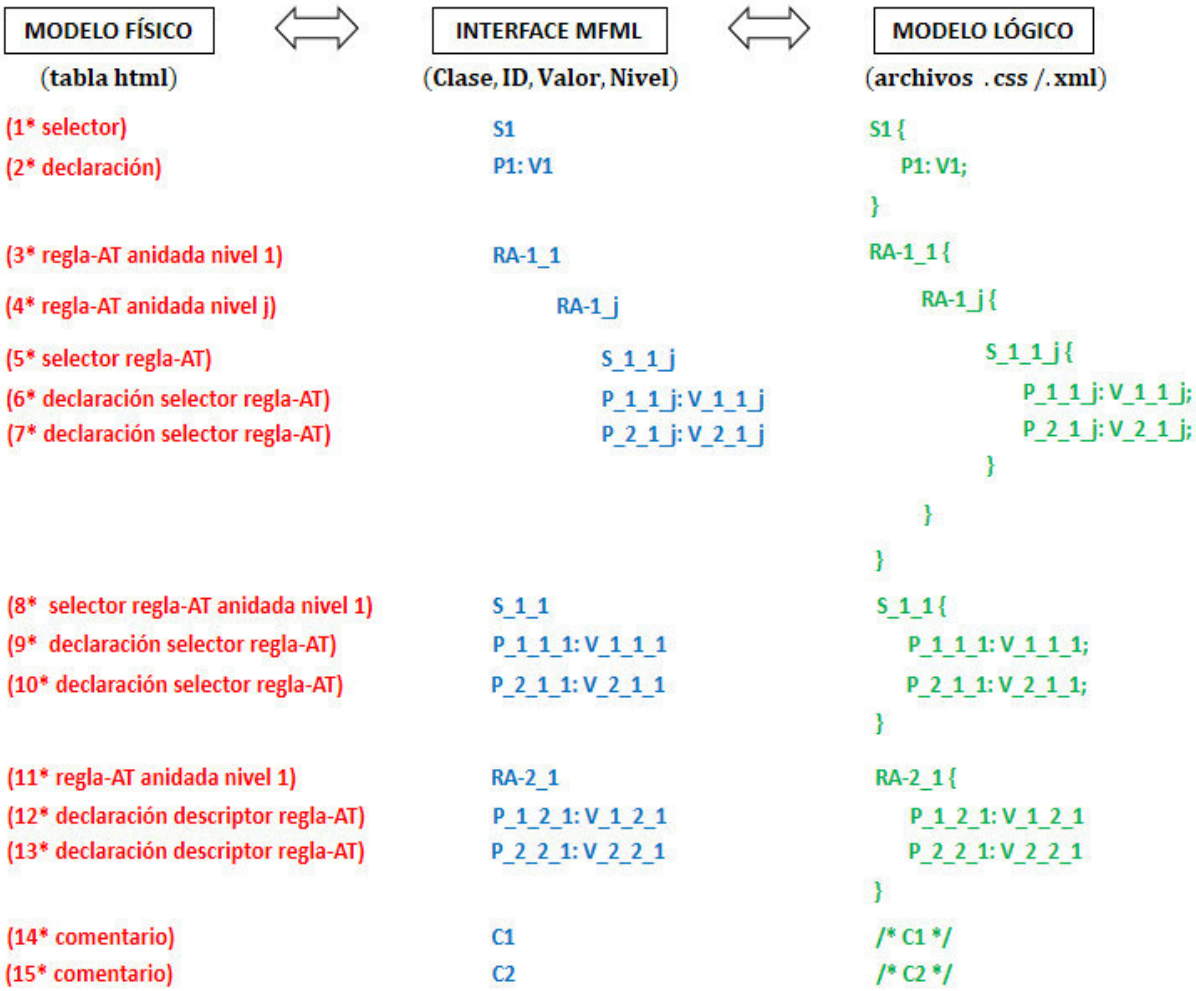


Figura 17 **Interface MFML: Modelo Físico - Modelo Lógico.**

Fuente: Elaboración propia

Donde:

∗ **:** número de fila de la tabla html

Clase **:** clase CSS asignada a cada elemento html de la tabla (fila, columna y textarea)

ID **:** identificador asignado a cada elemento html de la tabla (fila, columna y textarea)

Valor **:** contenido del elemento html textarea representando el valor de cada elemento CSS

Nivel **:** nivel de cada elemento CSS dentro de la tabla html

Resulta obvio entonces mencionar que la Interface MFML permite la interconexión de la tabla html y sus respectivos elementos fila, columna y textarea, así como de sus respectivas clases e IDs, con los archivos físicos con extensión: .css y .xml. Es decir, permite generar y exportar hojas de estilo CSS con todos los elementos contenidos en la tabla html incluyendo los niveles de anidamiento y, además, hace posible la reconstrucción de dicha tabla html junto con todos sus elementos exportados mediante el uso del archivo .xml, el cual contiene la metadata de la hoja de estilo .css. Es fundamental mencionar que la tabla html y todos sus elementos con sus respectivas clases e IDs forman el componente llamado modelo físico y los archivos .css y .xml exportados forman el componente llamado modelo lógico.

4.1.3.1. Modelo físico

El modelo físico permite introducir la voz en el desarrollo de código CSS lo cual es posible gracias a los comandos de voz. Los comandos de voz son expresiones habladas a través del micrófono del dispositivo conectado a Internet mediante el cual se interactúa con los elementos de la tabla html. A través del uso de la Web Speech API, estas expresiones son enviadas a un servicio web de reconocimiento de voz y son devueltas en formato texto. Una vez recibido el texto es evaluado internamente mediante el uso de expresiones regulares para determinar si es un comando reconocido y proceder a ejecutar la instrucción hablada.

De esta forma se permite la ejecución de los comandos de voz, los cuales hacen posible la inserción de: reglas-AT (at), descriptores (ds), selectores (s), declaraciones (d) y comentarios (c). Permiten también actualizar: reglas-AT (at), descriptores (ds), selectores (s), declaraciones (d) y comentarios (c). Además, hacen posible la operación de eliminación de: reglas-AT (at), descriptores (ds), selectores (s), declaraciones (d) y comentarios (c). Finalmente, las funciones de edición de contenido son posibles,

entendiéndose como tales al: desplazamiento entre filas y columnas de la tabla html, copiar, cortar, pegar, borrar y seleccionar texto, así como avanzar y retroceder entre el contenido de los elementos del textarea.

A manera de ejemplo, con la intención de simular el funcionamiento de la interface desarrollada, asumiremos que tenemos una tabla html tradicional la cual consta de siete (07) filas. En la primera fila se ubica un selector, en la segunda fila se ubica una declaración, la tercera fila está constituida por una regla-AT anidada padre de nivel uno (01), la cuarta fila contiene una regla-AT anidada de nivel dos (02) hija de la regla-AT anterior, la quinta fila está compuesta de un selector perteneciente a la regla-AT anidada de nivel dos (02), la sexta fila contiene una declaración la cual pertenece al selector de la quinta fila, por último, en la séptima fila se encuentra un comentario. Cabe mencionar que cada uno de los siete elementos está contenido dentro de una sola fila la cual contiene una sola columna en el caso de selectores, reglas-AT y comentarios, pero en el caso de las declaraciones la fila contiene dos columnas: una de la propiedad y la otra del valor. Además, cada columna contiene un solo elemento textarea que alberga el valor de cada elemento. Es así que cada fila, columna y elemento textarea tienen asignados una clase, un ID, un valor y un nivel lineal ($j = 1$) o anidado ($j > 1$).

Decimos entonces que el modelo físico está compuesto por siete (07) filas donde la primera contiene un selector y este hecho es representado por “1° selector”, la segunda contiene una declaración cuya lógica de representación, que será la misma de ahora en adelante, es “2° declaración”, la tercera contiene una regla-AT anidada padre de nivel dos (02) representada con “3° regla-AT anidada nivel 1”, la cuarta contiene una regla-AT anidada de nivel dos (02) representada con “4° regla-AT anidada nivel 2” hija de la regla-AT anterior, la quinta contiene un selector que se representa como “5° selector” perteneciente a la regla-AT anidada de nivel dos (02), la sexta contiene una declaración perteneciente al selector anterior representada por “6° declaración” y la séptima fila contiene un comentario representado por “8° comentario”. Evidentemente, para la inserción de estos elementos se usó la operación de inserción a través de comandos de voz.

La interface considera entonces la clase, ID, valor y nivel de cada uno de los siete (07) elementos anteriores. Estos elementos serán representados de la siguiente manera: el selector se abreviará con “S1”, la declaración será representada por “P1: V1” que se refiere a la propiedad y valor que la definen, la regla-AT anidada padre de nivel uno (01) se representará por la abreviación “RA-1_1”, la regla-AT anidada de nivel dos (02) hija de la regla-AT anterior se representará por “RA-1_2” (haciendo hincapié que el primer dígito es referente al número correlativo de elementos en el mismo nivel y el segundo dígito indica el nivel propiamente de dicho elemento), el selector perteneciente a la regla-AT anidada de nivel dos (02) será representado por “S_1_1_2”, la declaración perteneciente al selector anterior se representará con la abreviación “P_1_1_2: V_1_1_2” y finalmente el comentario será representado por “C1”.

Para continuar con la simulación del funcionamiento de la interface desarrollada, es necesario explicar el componente modelo lógico el cual se detalla en la sección siguiente.

4.1.3.2. Modelo lógico

El modelo lógico está compuesto por la hoja de estilo CSS 3, la cual una vez generada y exportada, está lista para ser incrustada en los archivos CSS de la carpeta de estilos del proyecto de maquetado de páginas o sistemas web. El archivo xml contiene los elementos siguientes: número de fila, nombre, clase de la fila, ID de la fila, clase de la columna, ID de la columna, clase del elemento textarea, ID del elemento textarea, contenido o valor del elemento y el nivel j-ésimo de anidamiento entre reglas-AT. El archivo xml permite que la Interface MFML puede reconstruir la tabla html para continuar desarrollando código CSS 3.

El archivo .css generado cumple con todas las normas sintácticas, semánticas y estructurales establecidas por la tecnología CSS 3. Precisamente esa es la función del modelo lógico: estructurar la hoja de estilo a generar y exportar. Esto incluye, añadir el caracter dos puntos (:) entre las propiedades y los valores de los descriptores, así como de las declaraciones; añadir el carácter punto y coma (;) después del valor que define a una declaración o descriptor; añadir las llaves de inicio y fin ({ }) que delimitan

los bloques de código CSS definiendo a las reglas-AT o selectores; inicio y cierre de comentarios anteponiendo los caracteres barra diagonal y asterisco (/*) al inicio del comentario y finalizándolo con los caracteres asterisco y barra diagonal (*/); añadir la sangría entre los elementos padre e hijos según el nivel de anidamiento.

El orden inverso que determina el procedimiento de importación es llevado a cabo en función al contenido del archivo xml, que es la metadata del archivo CSS. Esto es, cada vez que se genera y exporta una hoja de estilo se genera y exporta también un archivo xml cuyo nombre es el mismo que se dio a la hoja de estilo. La metadata es sumamente útil pues contiene toda la información necesaria que determinan las clases, IDs, valores, niveles, número de fila del elemento CSS dentro de la tabla html. Con esta información, es posible reconstruir el modelo físico que es representada por la tabla html, con todas las filas, columnas, elementos textarea, valores contenidos dentro de cada textarea y nivel de jerarquía. Esto permite seguir desarrollando código CSS 3.

Continuando con la simulación del funcionamiento de la interface desarrollada diremos que una vez identificada la clase, ID, valor y nivel de anidamiento de cada uno de los siete (07) elementos se procederá a generar y exportar los archivos .css y .xml. Para ello el componente modelo lógico recorrerá en orden ascendente cada fila del modelo físico; es decir, recorrerá cada una de las filas, columnas y elementos textarea de la tabla html y recogerá su clase, ID, valor y nivel. Cuando se identifique que un determinado elemento corresponde a un selector o a una regla-AT verificará su nivel para comenzar a identar (asignar espacios en blanco de izquierda a derecha llamados sangría) a partir del segundo (02) nivel en adelante según la fórmula de identado: f (identado) = [nivel x 4], así se comenzará a dar forma y estructura al archivo .css a generar. Otro aspecto fundamental es la inserción de los signos de puntuación. El carácter llave de apertura y cierre ({ }) que delimita y forma un bloque de código, el signo denominado dos puntos (:) es usado para dividir las propiedades de los valores pertenecientes a declaraciones o descriptores, el carácter punto y coma (;) sirve para delimitar una línea de código dentro de cualquier bloque. De esta forma el modelo lógico luego de identificar si el elemento es una regla-AT, un selector o un comentario asignará el valor de dicho elemento seguido del carácter llave de apertura ({), una vez construida esta línea generará un salto de línea para empezar a asignar el valor de las propiedades, insertar el carácter dos puntos (:) y continuar con la inserción del valor

finalizando con el signo de punto y coma (;) hasta finalizar todos los descriptores o declaraciones pertenecientes a dicho elemento, insertando luego una nueva línea con el carácter llave de cierre (}) que define el fin del bloque de código. Entonces, el procedimiento se repetirá hasta asignar: sangrías que visualmente denotarán niveles de jerarquía, saltos de línea y signos de puntuación que definirán la sintaxis de la hoja de estilo.

El procedimiento descrito en el párrafo anterior es válido para la generación del archivo .css. No obstante, la generación del archivo .xml sigue un procedimiento similar con la ligera diferencia de que ahora se creará un documento basado en el uso de etiquetas. Para ello, el modelo lógico define la etiqueta global: “<modeloLogicoCSSXML>” que encapsulará todo el contenido del documento. Dentro de la etiqueta anterior se encontrará la etiqueta: “<arrayMFML>” haciendo referencia a que dentro de ésta se encontrarán cada uno de los siete (07) elementos descritos anteriormente. Estos elementos serán contenidos por la etiqueta: “<elementoMFML>” y habrá tantas de estas etiquetas como cantidades de elementos tenga la tabla html del modelo físico, incluso se asignará una etiqueta propia a las propiedades y valores independientemente, las cuales definen a una declaración o descriptor; es decir, tanto la propiedad como el valor estarán contenidos dentro de sus propias etiquetas “<elementoMFML>”.

Las etiquetas que definirán a cada elemento CSS son: “<fila>” que determinará el número de fila de la tabla html, “<elemento>” que definirá el tipo de elemento, “<tr_clase>” que contendrá el valor de la clase de la fila donde se encuentra el elemento, “<tr_id>” que contendrá el valor del ID de la fila donde se encuentra el elemento, “<td_clase>” que contendrá el valor de la clase de la columna donde se encuentra el elemento, “<td_id>” que contendrá el valor del ID de la columna donde se encuentra el elemento, “<txta_clase>” que contendrá el valor de la clase del elemento textarea que contiene el valor del elemento, “<txta_id>” que contendrá el valor del ID del elemento textarea que contiene el valor del elemento, “<contenido>” que es la etiqueta que contendrá el valor propiamente del elemento y la etiqueta “<nivel>” que contendrá el nivel en el que se encuentra dicho elemento.

Si bien es cierto que el modelo matemático y la teoría de nomenclaturas sustentan la Interface MFML con sus respectivos componentes llamados modelo físico que permite gestionar la hoja de estilo dentro de un entorno web usando la voz y el modelo lógico que hace posible la generación y exportación automatizada de archivos .css y .xml, es necesario mencionar que nada de esto no sería posible sin la intervención de los comandos de voz. Explicar y estudiar en detalle los comandos de voz resulta imprescindible en esta etapa pues son el medio por el que resulta posible aplicar la tecnología de reconocimiento de voz para la generación automatizada de hojas de estilo CSS. Podemos decir entonces que los comandos de voz no pueden ser tomados a la ligera ya que estos engloban complejos procesos de análisis, categorización, pruebas y programación de las acciones a realizar en respuesta a su invocación y uso.

4.2. Comandos de voz

La Interface MFML, haciendo uso de la Web Speech API, hace posible el uso de la voz para generar código CSS. Para que esto sea posible se emplean los comandos de voz, los cuales son frases pronunciadas para provocar una respuesta. Cada comando de voz tiene dos (02) formas de ser pronunciado. La primera es llamada la forma completa y consiste en pronunciar la palabra “comando” seguido de la “acción a realizar” y la segunda es llamada forma abreviada que consiste en pronunciar la palabra “comando” seguido de su abreviación representada por el “número de orden” en que se encuentra el comando ordenado alfabéticamente de forma ascendente.

Así, por ejemplo, para insertar un nuevo selector, se tiene que pronunciar la frase: “comando insertar selector” o “comando uno”, las cuales representan la forma completa y abreviada respectivamente. En el caso del comando usado para insertar una regla-AT, la pronunciación de su acción contiene el caracter ampersand “&” para formar la frase “insertar at&t” para insertar una regla-AT de nivel (j = 1) e “insertar at&t hermano” en el caso de insertar una regla-AT en el nivel (j + 1) que el de la regla-AT desde donde se invoca la operación.

Al momento de diseñar cada comando de voz se puso especial énfasis en la facilidad de su pronunciación e identificación por parte del servicio de reconocimiento de voz

usando la tecnología Web Speech API. La gestión de la tecnología de reconocimiento de voz estará a cargo de los comandos de voz los cuales se agrupan en siete (07) categorías: comandos propiamente que modifican la estructura del modelo físico y gestionan el contenido de sus elementos; comandos auxiliares que permiten insertar letras del abecedario, vocales y caracteres; comandos de texto que permiten insertar cualquier número, palabra o frase literalmente; comandos de selector que permiten insertar selectores de tipo, de pseudoclase, de pseudoelemento, de clase, de ID, de reglas-AT page, de reglas-AT keyframes y de reglas-AT font-feature-values; comandos de declaración que permiten insertar propiedades y sus valores respectivos; comandos de reglas-AT que permiten insertar reglas-AT lineales y anidadas; comandos de descriptores que permiten insertar propiedades y sus valores respectivos pertenecientes a las reglas-AT anidadas.

4.2.1. Comandos propiamente

Los comandos propiamente son aquellos que modifican la estructura del modelo físico y gestionan el contenido de sus elementos. La modificación de la estructura se refiere a añadir o eliminar elementos CSS: reglas-AT (at), descriptores (ds), selectores (s), declaraciones (d) y comentarios. En tanto que la gestión se refiere a las operaciones de desplazamiento y edición de contenido dentro del modelo físico como son: desplazamiento vertical y horizontal, seleccionar, copiar, cortar, pegar y borrar texto dentro de los elementos textarea (txta).

Tabla 7 **Reconocimiento de voz - Comandos propiamente**

Pronunciación			Abr.	Descripción
comando	(+)	insertar selector	1	Inserta un nuevo selector de nivel (j = 1) con fondo de color rojo
comando	(+)	insertar declaracion	2	Si es invocado desde un selector, inserta una nueva declaración de nivel (j = 1) con fondo de color azul. Si es invocado desde una regla-AT anidada, inserta un nuevo descriptor de nivel (j > 1) con fondo de color naranja

comando	(+)	insertar at&t	3	Inserta un nueva regla-AT de nivel (j = 1) con fondo de color verde
comando	(+)	insertar at&t hermano	4	Inserta una nueva regla-AT con fondo de color verde y de nivel (j + 1), respecto del nivel de la regla-AT desde donde se invoca
comando	(+)	insertar comentario	5	Inserta un nuevo comentario de nivel (j = 1) con fondo de color violeta
comando	(+)	eliminar elemento	6	Elimina el elemento desde donde es invocado. Si es un selector o una regla-AT padre, entonces elimina también toda su descendencia y actualiza la nomenclatura de sus elementos hermanos (mismo nivel j) y de la de sus descendientes, restando uno a la numeración del elemento padre
comando	(+)	abajo	7	Establece el cursor en el elemento inmediatamente posterior al elemento desde el cual fue invocado
comando	(+)	arriba	8	Establece el cursor en el elemento inmediatamente anterior al elemento desde el cual fue invocado
comando	(+)	atras	9	Retrocede el cursor una posición, de derecha a izquierda, en el elemento desde el cual fue invocado
comando	(+)	avanzar	10	Avanza el cursor una posición, de izquierda a derecha, en el elemento desde el cual fue invocado
comando	(+)	borrar	11	Elimina el contenido seleccionado en el elemento desde el cual fue invocado
comando	(+)	comentario	12	Asigna el cursor en el n-ésimo comentario
comando	(+)	copiar	13	Guarda el contenido seleccionado en el elemento desde el cual fue invocado
comando	(+)	cortar	14	Corta y guarda el contenido seleccionado en el elemento desde el cual fue invocado
comando	(+)	espacio	15	Asigna un espacio en blanco en la posición actual del cursor en el elemento desde el cual fue invocado
comando	(+)	final	16	Establece el cursor en la posición de fin en el elemento desde el cual fue invocado

comando	(+)	inicio	17	Establece el cursor en la posición de inicio en el elemento desde el cual fue invocado
comando	(+)	pegar	18	Inserta el contenido guardado previamente por las operaciones de copiar o cortar en la posición actual del cursor en el elemento desde el cual fue invocado
comando	(+)	regla at&t	19	Asigna el cursor en la n-ésima regla-AT de nivel (j=1)
comando	(+)	seleccionar	20	Selecciona la fracción de texto desde la posición actual del cursor y se detiene al encontrar un espacio en blanco o un caracter: dos puntos, punto, punto y coma, coma, paréntesis inicio, paréntesis fin, llave inicio, llave fin, corchete inicio, corchete fin, barra diagonal, barra diagonal invertida, doble barra diagonal invertida, igual, signo de admiración apertura, signo de admiración cierre, signo de pregunta apertura, signo de pregunta cierre, porcentaje, arroba, numeral, dólar, ampersand, asterisco, comilla simple, doble comilla o, el final del contenido en el elemento desde el cual fue invocado
comando	(+)	seleccionar todo	21	Selecciona todo el contenido en el elemento desde el cual fue invocado
comando	(+)	selector	22	Asigna el cursor en el m-ésimo selector

Fuente. Elaboración propia

4.2.2. Comandos auxiliares

Los comandos auxiliares son aquellos que permiten insertar letras del abecedario, vocales y caracteres. Las letras del abecedario posibles de ser insertadas con este comando son: s, d, m, n, t. Esto debido a que por facilidad de pronunciación e interpretación del servicio de reconocimiento de voz que usa la tecnología Web Speech API no es necesario generar comandos para el resto de las letras del abecedario, tan sólo se usará el comando “texto”.

Tabla 8 **Reconocimiento de voz - Comandos auxiliares**

Pronunciación			Abr.	Descripción
abecedario	(+)	De	1	Inserta la letra "d" en la posición actual del cursor en el elemento desde el cual fue invocado
abecedario	(+)	M	2	Inserta la letra "m" en la posición actual del cursor en el elemento desde el cual fue invocado
abecedario	(+)	n	3	Inserta la letra "n" en la posición actual del cursor en el elemento desde el cual fue invocado
abecedario	(+)	si	4	Inserta la letra "c" en la posición actual del cursor en el elemento desde el cual fue invocado
abecedario	(+)	ti	5	Inserta la letra "t" en la posición actual del cursor en el elemento desde el cual fue invocado
vocal	(+)	a	1	Inserta la vocal "a" en la posición actual del cursor en el elemento desde el cual fue invocado
vocal	(+)	e	2	Inserta la vocal "e" en la posición actual del cursor en el elemento desde el cual fue invocado
vocal	(+)	i	3	Inserta la vocal "i" en la posición actual del cursor en el elemento desde el cual fue invocado
vocal	(+)	o	4	Inserta la vocal "o" en la posición actual del cursor en el elemento desde el cual fue invocado
vocal	(+)	u	5	Inserta la vocal "u" en la posición actual del cursor en el elemento desde el cual fue invocado
caracter	(+)	acento circunflejo	1	Inserta el caracter "^" en la posición actual del cursor en el elemento desde el cual fue invocado
caracter	(+)	ampersand	2	Inserta el caracter "&" en la posición actual del cursor en el elemento desde el cual fue invocado
caracter	(+)	asterisco	3	Inserta el caracter "*" en la posición actual del cursor en el elemento desde el cual fue invocado
caracter	(+)	coma	4	Inserta el caracter "," en la posición actual del cursor en el elemento desde el cual fue invocado
caracter	(+)	comilla simple	5	Inserta el caracter " ' " en la posición actual del cursor en el elemento desde el cual fue invocado

caracter	(+)	comillas	6	Inserta el caracter ' " ' en la posición actual del cursor en el elemento desde el cual fue invocado
caracter	(+)	corchete fin	7	Inserta el caracter "]" en la posición actual del cursor en el elemento desde el cual fue invocado
caracter	(+)	corchete inicio	8	Inserta el caracter "[" en la posición actual del cursor en el elemento desde el cual fue invocado
caracter	(+)	diagonal	9	Inserta el caracter "/" en la posición actual del cursor en el elemento desde el cual fue invocado
caracter	(+)	diagonal invertida	10	Inserta el caracter "\" en la posición actual del cursor en el elemento desde el cual fue invocado
caracter	(+)	dolar	11	Inserta el caracter "$" en la posición actual del cursor en el elemento desde el cual fue invocado
caracter	(+)	dos puntos	12	Inserta el caracter ":" en la posición actual del cursor en el elemento desde el cual fue invocado
caracter	(+)	guion	13	Inserta el caracter "-" en la posición actual del cursor en el elemento desde el cual fue invocado
caracter	(+)	guion bajo	14	Inserta el caracter "_" en la posición actual del cursor en el elemento desde el cual fue invocado
caracter	(+)	igual	15	Inserta el caracter "=" en la posición actual del cursor en el elemento desde el cual fue invocado
caracter	(+)	llave fin	16	Inserta el caracter "}" en la posición actual del cursor en el elemento desde el cual fue invocado
caracter	(+)	llave inicio	17	Inserta el caracter "{" en la posición actual del cursor en el elemento desde el cual fue invocado
caracter	(+)	mas	18	Inserta el caracter "+" en la posición actual del cursor en el elemento desde el cual fue invocado
caracter	(+)	mayor que	19	Inserta el caracter ">" en la posición actual del cursor en el elemento desde el cual fue invocado
caracter	(+)	menor que	20	Inserta el caracter "<" en la posición actual del cursor en el elemento desde el cual fue invocado
caracter	(+)	numeral	21	Inserta el caracter "#" en la posición actual del cursor en el elemento desde el cual fue invocado

caracter	(+)	palote	22	Inserta el caracter "|" en la posición actual del cursor en el elemento desde el cual fue invocado
caracter	(+)	parentesis fin	23	Inserta el caracter ")" en la posición actual del cursor en el elemento desde el cual fue invocado
caracter	(+)	parentesis inicio	24	Inserta el caracter "(" en la posición actual del cursor en el elemento desde el cual fue invocado
caracter	(+)	porcentaje	25	Inserta el caracter "%" en la posición actual del cursor en el elemento desde el cual fue invocado
caracter	(+)	punto	26	Inserta el caracter "." en la posición actual del cursor en el elemento desde el cual fue invocado
caracter	(+)	punto y coma	27	Inserta el caracter ";" en la posición actual del cursor en el elemento desde el cual fue invocado
caracter	(+)	virgulilla	28	Inserta el caracter "~" en la posición actual del cursor en el elemento desde el cual fue invocado

Fuente. Elaboración propia

4.2.3. Comandos de texto

Los comandos de texto son aquellos que permiten insertar cualquier número, palabra o frase libre y literalmente. Es decir, este comando, es usado en casos donde se requiera insertar contenido alternativo, útil en el caso de comentarios, u otros motivos que se consideren apropiados por el usuario.

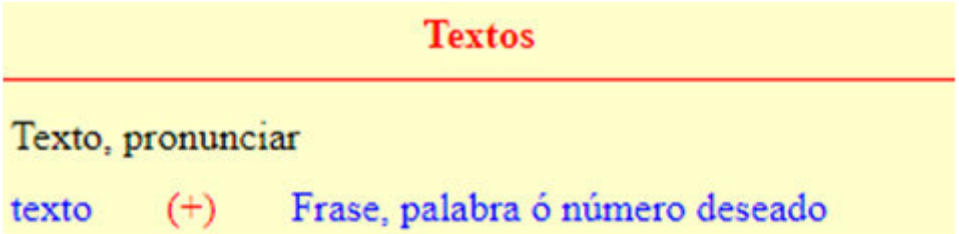


Figura 18 **Reconocimiento de voz - Comandos de texto.**

Fuente: Elaboración propia

Como se puede notar la pronunciación de este comando está determinado por la palabra “texto” seguido de cualquier otra palabra o frase que se desee insertar como

contenido de cualquier elemento de la hoja de estilo, principalmente usado para insertar contenido en el elemento textarea comentario.

4.2.4. Comandos de selector

Los comandos de selector son aquellos que permiten insertar selectores de tipo, de pseudoclase, de clase, de ID, de reglas-AT page, de reglas-AT keyframes y de reglas-AT font-feature-values. En el caso de los selectores de pseudoclase y pseudoelemento, por motivos de facilidad de pronunciación e interpretación por parte del servicio de reconocimiento de voz usado por la tecnología Web Speech API, son pronunciados como “clase” y “elemento” seguido del selector, respectivamente. En el caso de selectores de Clase e ID, son pronunciados como “class” e “id” seguido del nombre del selector, respectivamente.

En el caso de los selectores de reglas-AT page, de reglas-AT keyframes y de reglas-AT font-feature-values, por motivos de facilidad de pronunciación e interpretación por parte del servicio de reconocimiento de voz usado por la tecnología Web Speech API, son pronunciados como “pagina”, “marco principal” y “valor letra” seguido del nombre del selector, respectivamente.

Tabla 9 **Reconocimiento de voz - Comandos de selector**

Pronunciación			Abr.	Descripción
selector	(+)	elemento html	número	Inserta el selector de tipo, siendo un total de 111 elementos html soportados por el framework, en el elemento selector desde el cual fue invocado
clase	(+)	pseudoclase	número	Inserta el selector de pseudoclase, siendo estos: active, any-link, any, blank, checked, current, defined, default, dir, disabled, empty, enabled, first-child, first-of-type, first, fullscreen, focus-visible, focus-within, focus, future, hover, host-context, host, indeterminate, in-range, invalid, lang, last-

				child, last-of-type, left, local-link, link, is, not, nth-child, nth-of-type, nth-last-child, nth-last-of-type, only-child, only-of-type, optional, out-of-range, past, placeholder-shown, playing, paused, read-only, read-write, required, right, root, scope, valid, target y visited, anteponiendo el caracter ":", en el elemento selector desde el cual fue invocado
elemento	(+)	pseudoelemento	número	Inserta el selector de pseudoelemento, siendo estos: after, backdrop, before, cue, first-letter, first-line, grammar-error, marker, placeholder, selection y spelling-error, anteponiendo el caracter "::", en el elemento selector desde el cual fue invocado
class	(+)	nombre selector	-	Inserta el selector de clase, anteponiendo el caracter ".", en el elemento selector desde el cual fue invocado
id	(+)	nombre selector	-	Inserta el selector de ID, anteponiendo el caracter "#", en el elemento selector desde el cual fue invocado
pagina	(+)	nombre selector	número	Inserta el selector de regla-AT page, siendo estos: @top-left-corner, @top-left, @top-center, @top-right, @top-right-corner, @bottom-left-corner, @bottom-left, @bottom-center, @bottom-right, @bottom-right-corner, @left-top, @left-middle, @left-bottom, @right-top, @right-middle, @right-bottom y @footnote, anteponiendo el caracter "@", en el elemento selector desde el cual fue invocado
marco principal	(+)	nombre selector	número	Inserta el selector de regla-AT keyframes, siendo estos: from, to o valores porcentuales,

				en el elemento selector desde el cual fue invocado
valor letra	(+)	nombre selector	número	Inserta el selector de regla-AT font-feature-values, siendo estos: @swash, @annotation, @ornaments, @stylistic, @styleset y @character-variant, anteponiendo el caracter "@", en el elemento selector desde el cual fue invocado

Fuente. Elaboración propia

4.2.5. Comandos de declaración

Los comandos de declaración son aquellos que permiten insertar propiedades y sus valores respectivos. Esto quiere decir que los valores a insertar son mostrados según la propiedad elegida.

Tabla 10 **Reconocimiento de voz - Comandos de declaración**

Pronunciación			Abr.	Descripción
propiedad	(+)	nombre propiedad	número	Inserta la propiedad en el elemento propiedad de la declaración desde el cual fue invocado
valor	(+)	nombre valor	número	Inserta el valor, según la propiedad elegida, en el elemento valor de la declaración desde el cual fue invocado

Fuente. Elaboración propia

4.2.6. Comandos de reglas-AT

Los comandos de reglas-AT son aquellos que permiten insertar reglas-AT lineales y anidadas. Algunas reglas-AT requieren de condiciones lógicas, tipos de media queries, características de media queries, pseudoclases o identificadores asignados por el usuario seguidos de los nombres de dichas reglas-AT.

De esta forma la regla-AT lineal import puede contener en su definición una url y, opcionalmente una lista de media queries; la regla-AT lineal namespace contiene, a su

vez, en su definición una url y, opcionalmente un prefijo; la regla-AT anidada media contiene en su definición una lista de media queries compuesta de tipos y características de media unidas por condiciones lógicas; la regla-AT anidada document contiene en su definición una url, un dominio o una expresión regular. La regla-AT anidada page puede contener en su definición una pseudoclase; la regla-AT anidada supports contiene en su definición una serie de declaraciones: propiedad – valor, unidas opcionalmente por condiciones lógicas; finalmente, las reglas-AT anidadas counter-style, keyframes y font-feature-values contienen en su definición un identificador (nombre) asignado por el usuario.

Tabla 11 **Reconocimiento de voz - Comandos de reglas-AT**

Pronunciación			Abr.	Descripción
at&t lineal	(+)	nombre regla-AT	número	Inserta las reglas-AT lineales: charset, import y namespace, anteponiendo el caracter "@", en el elemento regla-AT desde el cual fue invocado
at&t anidada	(+)	nombre regla-AT	número	Inserta las reglas-AT anidadas: media, document, page, font-face, viewport, counter-style, keyframes, supports y font-feature-values, anteponiendo el caracter "@", en el elemento regla-AT desde el cual fue invocado
importar	(+)	url	número	Inserta la definición: url, de la regla-AT lineal import, en el elemento regla-AT desde el cual fue invocado
tipo	(+)	nombre tipo	número	Inserta la definición: all, aural, braille, handheld, print, projection, screen, tty, tv, embossed y speech, de la regla-AT lineal import ó la regla-AT anidada media, en el elemento regla-AT desde el cual fue invocado
caracteristica	(+)	nombre caracteristica	número	Inserta la definición: width, min-width, max-width, height, min-height, max-

				height, device-width, min-device-width, max-device-width, device-height, min-device-height, max-device-height, aspect-ratio, min-aspect-ratio, max-aspect-ratio, device-aspect-ratio, min-device-aspect-ratio, max-device-aspect-ratio, color, min-color, max-color, color-index, min-color-index, max-color-index, monochrome, min-monochrome, max-monochrome, resolution, min-resolution, max-resolution, update-frequency, overflow-block, overflow-inline, display-mode, inverted-colors, pointer, hover, any-pointer, any-hover, light-level, scripting, scan y grid, de la regla-AT lineal import o la regla-AT anidada media, en el elemento regla-AT desde el cual fue invocado
condicion	(+)	nombre condicion	número	Inserta la definición: and, or, not y only, de la regla-AT lineal import, la regla-AT anidada media o la regla-AT anidada supports, en el elemento regla-AT desde el cual fue invocado
espacio nombre	(+)	url	número	Inserta la definición: url, de la regla-AT lineal namespace, en el elemento regla-AT desde el cual fue invocado
documento	(+)	nombre documento	número	Inserta la definición: url, url-prefix, domain y regex, de la regla-AT anidada document, en el elemento regla-AT desde el cual fue invocado
pagina	(+)	nombre pseudoclase	número	Inserta la definición: :blank, :first, :left y :right, de la regla-AT anidada page, en el elemento regla-AT desde el cual fue invocado

Fuente. Elaboración propia

4.2.7. Comandos de descriptores

Los comandos de descriptores son aquellos que permiten insertar propiedades y sus valores respectivos, correspondientes a reglas-AT anidadas. Las reglas-AT anidadas page, font-face, viewport y counter-style tiene descriptores: propiedades y valores, claramente definidos.

Tabla 12 **Reconocimiento de voz - Comandos de descriptores**

Pronunciación			Abr.	Descripción
pagina	(+)	nombre pagina	número	Inserta el descriptor propiedad, correspondiente a las propiedades margin, padding, border y background de la regla-AT anidada page, en el elemento regla-AT desde el cual fue invocado
letra	(+)	nombre letra	número	Inserta el descriptor propiedad de la regla-AT anidada font-face en el elemento regla-AT desde el cual fue invocado
ventana	(+)	nombre ventana	número	Inserta el descriptor propiedad de la regla-AT anidada viewport en el elemento regla-AT desde el cual fue invocado
contador estilo	(+)	nombre contador	número	Inserta el descriptor propiedad de la regla-AT anidada counter-style en el elemento regla-AT desde el cual fue invocado

Fuente. Elaboración propia

La traducción al idioma español de palabras como: document a documento, viewport a ventana, counter-style a contador estilo, page a pagina, font-face a letra, font-feature-values a valor letra y supports a soporte es por motivos de facilidad de pronunciación e interpretación por parte del servicio de reconocimiento de voz usado por la tecnología Web Speech API.

Habiendo expuesto las bases teóricas representadas por la Interface MFML y los comandos de voz se procederá entonces a describir el proceso seguido para el análisis, diseño y construcción del framework CSS 3 con reconocimiento de voz. Para ello se

ha elegido la metodología de desarrollo ágil Scrum la cual nos ha guiado en todo el proceso de desarrollo del software. Además de Scrum, se ha considerado conveniente modelar la arquitectura del framework usando el Lenguaje de Modelado Unificado (UML) para obtener una mejor comprensión del sistema a construir.

4.3. CONSTRUCCIÓN DEL FRAMEWORK WEB CSS 3 CON SCRUM Y UML

Detallaremos el procedimiento seguido en el análisis, diseño, construcción y prueba del framework CSS 3 con reconocimiento de voz, aplicando el modelo de desarrollo ágil Scrum y los modelos del UML 2.0: modelo de casos de uso, diagramas de clases, diagramas de actividad, diagramas de secuencia, diagramas de componentes y diagramas de despliegue, que nos han permitido definir la arquitectura del framework.

Así que en primera instancia se definirán los roles establecidas por Scrum, posteriormente se describirán los eventos que marcan el ritmo de desarrollo y las actividades realizadas en cada uno de ellos. Finalmente se detallarán los artefactos pila del producto, pila del sprint y el incremento propiamente.

4.3.1. Roles de scrum

Los roles establecidos son: propietario del producto (product owner) quien determina cómo será el producto terminado y el orden de construcción de los incrementos entregados al cliente; desarrollador (developer) que es el profesional que realiza el incremento que permitirá obtener el producto final; scrum master quien se responsabiliza del cumplimiento de las reglas de la metodología Scrum. En esta tesis estos roles han sido desempeñados por el autor.

Tabla 13 **Scrum - Roles de scrum desempeñados por el autor**

Rol	Investigador
Propietario del producto	Julio Prudencio Nieves
Desarrollador	Julio Prudencio Nieves

Scrum master	Julio Prudencio Nieves
Interesados	Diseñadores web

Fuente. Elaboración propia

4.3.2. Historias de usuario

Los artefactos de Scrum son sus herramientas y ayudan a los roles a desarrollar los incrementos del producto que dan valor a los clientes. Scrum gestiona los artefactos siguientes: pila del producto (product backlog) que registra y prioriza las historias de usuario que son los requisitos de los clientes y que suelen llamarse "historias de usuario"; pila del sprint (sprint backlog) que representa la lista de actividades a realizar durante un sprint; incremento que viene a ser el resultado de cada sprint. Las historias de usuario son descritas a continuación.

4.3.2.1. Cargar archivo html

El inicio del maquetado web es precisar el archivo .html sobre el cual se desean ejecutar reglas de diseño CSS 3, así que para cumplir tal propósito el framework CSS 3 tiene que permitir subir o cargar dicho archivo, garantizando que su estructura sea la indicada por la tecnología HTML.

Tabla 14 **Scrum - Historia de usuario Cargar archivo html**

HU-01: Cargar archivo html	
Como	Usuario
Quiero	Que el framework CSS 3 me permita subir cualquier archivo html que cumpla la estructura establecida por la tecnología HTML 5
Para	Poder maquetar el archivo html usando la tecnología de diseño CSS 3

Fuente. Elaboración propia

4.3.2.2. Importar archivo xml

Luego de subir el archivo html a maquetar, se inicia el proceso de diseño. Para el proceso de diseño, el framework nos permite importar un archivo con extensión .xml que forma parte del modelo lógico de la Interface MFML y contiene toda la

información necesaria para generar el modelo físico que es la tabla html con sus filas, columnas, elementos textarea, valores, clases e IDs que representan a la hoja de estilos CSS 3 generada y exportada con anterioridad a través del framework.

Tabla 15 **Scrum - Historia de usuario Importar archivo xml**

HU-02: Importar archivo xml	
Como	Usuario
Quiero	Que el framework CSS 3 me permita importar el archivo .xml que contiene toda la información necesaria para generar la tabla html
Para	Continuar el maquetado del archivo html a partir del código CSS 3 generado hasta el momento de exportar el archivo .xml

Fuente. Elaboración propia

4.3.2.3. Crear hoja de estilo CSS 3

El proceso de diseño no sólo podrá ser realizado importando el archivo xml del modelo lógico, también se tiene que poder crear una nueva hoja de estilo CSS 3. Para ello, el framework tiene que permitir ingresar el nombre de la nueva hoja de estilo acorde a la función que desempeñará el archivo html a maquetar para iniciar luego con la pronunciación de los comandos de voz.

Tabla 16 **Scrum - Historia de usuario Crear archivo css**

HU-03: Crear archivo css	
Como	Usuario
Quiero	Que el framework CSS 3 me permita asignar el nombre de la nueva hoja de estilo a crear acorde a la función que desempeñará el archivo html a maquetar
Para	Poder iniciar la pronunciación de los comandos de voz que me permitirán crear bloques de código CSS 3

Fuente. Elaboración propia

4.3.2.4. Pronunciación de comandos de voz

El proceso de maquetación propiamente se lleva a cabo con el uso de los comandos de voz que son las instrucciones habladas por parte del usuario las cuales han sido enviadas al servicio de reconocimiento de voz mediante el uso de la Web Speech API y se han devuelto en formato texto. Una vez se tenga el comando de voz en formato texto, el framework verifica que dicho comando sea válido y ejecuta los procedimientos establecidos.

Así, por ejemplo, si el usuario pronunciara el comando para insertar un nuevo selector: "comando insertar selector", se consultaría al servicio de reconocimiento de voz esperando la devolución de dicho comando en formato texto y luego de obtener la respuesta validaría si es reconocida por el framework para que, luego de validado y aceptado, realice los procedimientos de: crear un nuevo modelo físico (tabla html), si no existiera aún, que representará a la hoja de estilo CSS 3 a generar, crear también una nueva fila, generar y asignarle una clase e ID, además de crear una nueva columna y elemento textarea, los cuales serán contenidos por la fila anteriormente creada, con sus propias clases e IDs respectivamente.

Tabla 17 **Scrum - Historia de usuario Pronunciación de comandos de voz**

HU-04: Pronunciar comandos de voz	
Como	Usuario
Quiero	Que el framework CSS 3 me permita pronunciar los comandos de voz que permitirán gestionar el modelo físico (tabla html) que representa a la hoja de estilo a generar posteriormente
Para	Poder construir los bloques de código CSS 3 de la hoja de estilo que maquetará el archivo html importado anteriormente

Fuente. Elaboración propia

4.3.2.5. Ocultar framework

Durante el procedimiento de maquetado del archivo html, no importando que el modelo físico (tabla html) se haya generado mediante la importación del archivo xml del modelo lógico o se haya creado una nueva hoja de estilo CSS 3 desde cero, el

framework tiene que permitir previsualizar el avance del maquetado en cualquier momento para obtener una retroalimentación y así cambiar algunos bloques de código o continuar con el diseño web. Entonces, el procedimiento ocultar framework conlleva a generar de manera interna el modelo lógico compuesto por los archivos .xml y .css y luego mostrar, en la misma ventana del navegador web donde se ejecuta el framework, el archivo html aplicando la hoja de estilo CSS 3 desarrollada hasta el momento.

Tabla 18 **Scrum - Historia de usuario Ocultar framework**

HU-05: Ocultar framework	
Como	Usuario
Quiero	Que el framework CSS 3 me permita ocultar el framework y generar, hasta el momento, el modelo lógico: archivos .css y .xml
Para	Poder aplicar la hoja de estilo CSS 3 al archivo html que me permita previsualizar el avance del maquetado y obtener una retroalimentación del proceso de diseño web

Fuente. Elaboración propia

4.3.2.6. Mostrar framework

Luego de generar internamente el modelo lógico (archivos xml y css) que permiten aplicar la hoja de estilo sobre el archivo html, generada hasta el momento de ocultar el framework, se requiere volver al framework para poder continuar con el proceso de maquetado web. Para ello el framework tiene que permitir cambiar del modo de previsualización al modo de edición de la hoja de estilo CSS 3 para continuar el proceso de diseño.

Tabla 19 **Scrum - Historia de usuario Mostrar framework**

HU-06: Mostrar framework	
Como	Usuario
Quiero	Que el framework CSS 3 me permita volver al modo de edición
Para	Poder continuar el proceso de maquetado web

Fuente. Elaboración propia

4.3.2.7. Exportar archivos .xml y .css

Finalizada la maquetación del archivo html a través del desarrollo de los bloques de código de la hoja de estilo CSS 3 haciendo uso de los comandos de voz y luego de previsualizar que el diseño realizado es correcto y acorde al requerimiento del usuario, es momento de generar el modelo lógico de la Interface MFML. Esto quiere decir que el framework tiene que permitir exportar la hoja de estilo en formato .css para poder ser incluido en el directorio css de nuestra página o sistema web, y que genere también el archivo .xml que contiene toda la información necesaria para generar nuevamente el modelo físico (tabla html) que nos permita continuar con el proceso de maquetado web del archivo html correspondiente.

Tabla 20 **Scrum - Historia de usuario Exportar archivos .xml y .css**

HU-07: Exportar archivos .xml y .css	
Como	Usuario
Quiero	Que el framework CSS 3 me permita generar el modelo lógico que son los archivos .xml y .css
Para	Poder incluir la hoja de estilo en la carpeta css de nuestra página o sistema web y generar el modelo físico (tabla html) posteriormente

Fuente. Elaboración propia

4.3.2.8. Descargar código fuente

El framework debe permitir realizar la descarga de sus propios archivos de código fuente. Esto es, se debe poder descargar el proyecto: framework web CSS 3, a través de un enlace de descarga incrustado en la propia página web del manual del framework. Así, la investigación de la presente tesis y el producto desarrollado podrá ser analizado completamente por otros profesionales, investigadores, aficionados y público en general motivados por el maquetado o diseño web, enriqueciendo la investigación en sí misma y generando nuevos aportes en la materia.

Tabla 21 **Scrum - Historia de usuario Descargar código fuente**

HU-08: Descargar código fuente	
Como	Usuario
Quiero	Que el framework CSS 3 me permita descargar los archivos de su código fuente
Para	Poder analizar completamente el proyecto y contribuir con otros aportes en la materia

Fuente. Elaboración propia

4.3.3. Pila del producto (Product backlog)

La pila del producto representa el conjunto de funcionalidades a incorporar en el producto a lo largo de los sprints. Es decir, está conformada por todas las historias de usuario desarrolladas anteriormente.

Tabla 22 **Scrum - Pila del producto (product backlog)**

Historia de usuario	Descripción	Prioridad
HU-01	Cargar archivo html	1
HU-02	Importar archivo xml	2
HU-03	Crear archivo css	2
HU-04	Pronunciar comandos de voz	3
HU-05	Ocultar framework	3
HU-06	Mostrar framework	3
HU-07	Exportar archivos .xml y .css	3
HU-08	Descargar código fuente	3

Fuente. Elaboración propia

4.3.4. Estimación de la pila del producto

Scrum emplea puntos como unidad de trabajo, usando denominaciones como puntos de historia, la cual ayuda a dimensionar la estimación de una tarea comparándola con una ya conocida. De esta forma estimar el tiempo y el esfuerzo de cada historia de usuario suele ser una unidad relativa o abstracta.

Por otro lado, se usará el método planing pocker para asignar valores a cada historia de usuario clasificándolos como: XtraSmall (XS) con un valor de 01 como punto de historia y 1/2 día como tiempo de trabajo; Small (S) con un valor de 02 como punto de historia y 01 día como tiempo de trabajo; Medium (M) con un valor de 03 como punto de historia y 02 días como tiempo de trabajo; Large (L) con un valor de 05 como punto de historia y 03 días como tiempo de trabajo; XtraLarge (XL) con un valor de 08 como punto de historia y 05 días como tiempo de trabajo (Mamani, 2019). La estimación de la pila del producto se ha realizado siguiendo un orden de prioridad de implementación.

Tabla 23 **Scrum - Estimación de la pila del producto**

Sprint	HU	Descripción	Tamaño	Puntos de historia	Tiempo (días)
Primer Sprint	HU-01	Cargar archivo html	M	3	2
	HU-02	Importar archivo xml	M	3	2
	HU-03	Crear archivo css	S	2	1
	HU-04	Pronunciar comandos de voz	XL	8	5
Segundo Sprint	HU-05	Ocultar framework	L	5	3
	HU-06	Mostrar framework	S	2	1
	HU-07	Exportar archivos .xml y .css	L	5	3
	HU-08	Descargar código fuente	L	5	3
Puntos de historia / Tiempo estimado (Time Boxing)				**33**	**20**

Fuente. Elaboración propia

De acuerdo al tamaño y a la cantidad de historias de usuario, el tiempo estimado para el desarrollo del sistema es de 20 días, los cuales lo dividimos en 02 iteraciones o sprints de 10 días cada uno.

4.3.5. Primer sprint

El sprint es considerado el evento principal en Scrum pues es aquí donde todos los roles desempeñados por el equipo de trabajo realizan las actividades de construcción

de un entregable, por ello se dice que el sprint tiene un objetivo bien definido dentro del proyecto y una duración fija y constante (Palacio, 2020).

4.3.5.1. Planificación del sprint

Determina el punto de inicio del sprint definiendo su objetivo y la lista de tareas a realizar para su cumplimiento (Palacio, 2020).

Tabla 24 **Scrum - Planificación del primer sprint**

HU	Descripción	Puntos de historia	Tiempo (días)	Inicio	Final
HU-01	Cargar archivo html	3	2	01/04/2021	02/04/2021
HU-02	Importar archivo xml	3	2	03/04/2021	04/04/2021
HU-03	Crear archivo css	2	1	05/04/2021	05/04/2021
HU-04	Pronunciar comandos de voz	8	5	06/04/2021	10/04/2021
Puntos de historia / Tiempo estimado (Time Boxing)		16	10		

Fuente. Elaboración propia

Se muestra que el esfuerzo estimado para completar el primer sprint es de 16 puntos de historia con un tiempo de desarrollo de 10 días. Se considera preciso modelar el diagrama de casos de uso con el fin de comprender la funcionalidad global del sprint.

4.3.5.2. Diagrama de casos de uso

Un diagrama de casos de uso provee información adicional acerca de las relaciones entre los casos de uso y el usuario externo. El diagrama de casos de uso asociado al primer sprint ha sido elaborado considerando las historias de usuario mencionadas anteriormente.

Tabla 25 **UML - Casos de uso del primer sprint**

Casos de uso	Historia de usuario	Descripción
Cargar html	HU-01	Cargar archivo html
Importar xml	HU-02	Importar archivo xml
Crear css	HU-03	Crear hoja de estilo CSS 3
Pronunciar comandos de voz	HU-04	Pronunciación de comandos de voz
- Gestionar edición texto		
- Gestionar selector CSS		
- Gestionar declaracion CSS		
- Gestionar reglaAT CSS		
- Gestionar descriptor CSS		
- Gestionar comentario CSS		

Fuente. Elaboración propia

Claramente se verifica que la historia de usuario, HU-04 llamada Pronunciación de comandos de voz, requiere de un caso de uso propiamente y de seis casos de uso incluidos, esto debido a la complejidad de la historia de usuario en cuestión, pues resulta conveniente separar los comandos de voz que gestionan cada uno de los elementos CSS.

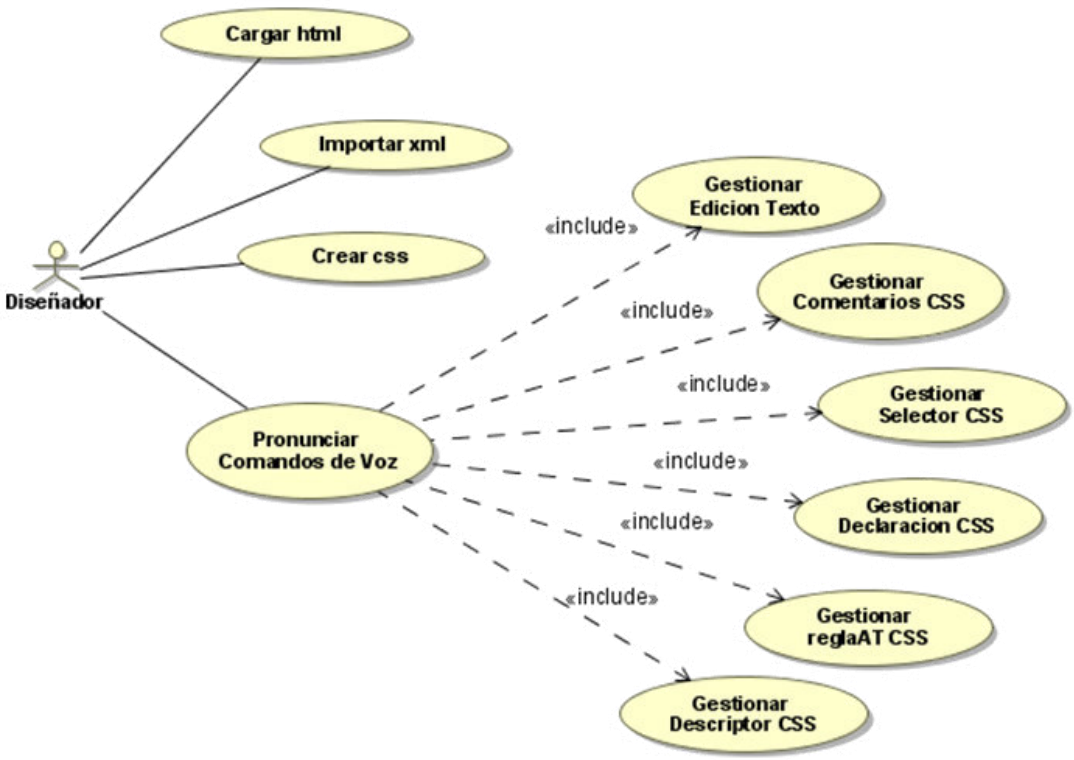


Figura 19 **UML - Diagrama de Casos de uso del primer sprint.**

Fuente: Elaboración propia

4.3.5.3. Diagrama de clases

El diagrama de clases pertenece a la vista de estructura definida por UML donde se muestran los atributos y operaciones de las clases, así como su interrelación. En el diagrama de clases asociado al primer sprint se aprecian las historias de usuario mencionadas.

Tabla 26 **UML - Clases del primer sprint**

Clases	Historia de usuario	Descripción
Inicialización	HU-01	Cargar archivo html
	HU-02	Importar archivo xml
	HU-03	Crear hoja de estilo CSS 3
Pronunciación	HU-04	Pronunciación de comandos de voz
- edicionTexto		
- selector		

- declaracion		
- reglaAT		
- descriptor		
- comentario		

Fuente. Elaboración propia

Claramente se aprecia que para cumplir las historias de usuario de este primer sprint se han establecido dos clases principales llamadas Inicialización y Pronunciación. La clase Inicialización consta de diez (10) operaciones y cinco (05) atributos principales, en tanto que la clase Pronunciación consta de trece (13) operaciones y veintiséis (26) atributos principales. Además, la clase Pronunciación generaliza a seis (06) clases: la clase edicionTexto que consta de doce (12) operaciones y cinco (05) atributos principales, la clase selector que consta de cinco (05) operaciones y siete (07) atributos principales, la clase declaración que consta de cinco (05) operaciones y cuatro (04) atributos principales, la clase reglaAT que consta de dieciocho (18) operaciones y trece (13) atributos principales, la clase descriptor que consta de cinco (05) operaciones y siete (07) atributos principales, la clase comentario que consta de dos (02) operaciones y cuatro (04) atributos principales.

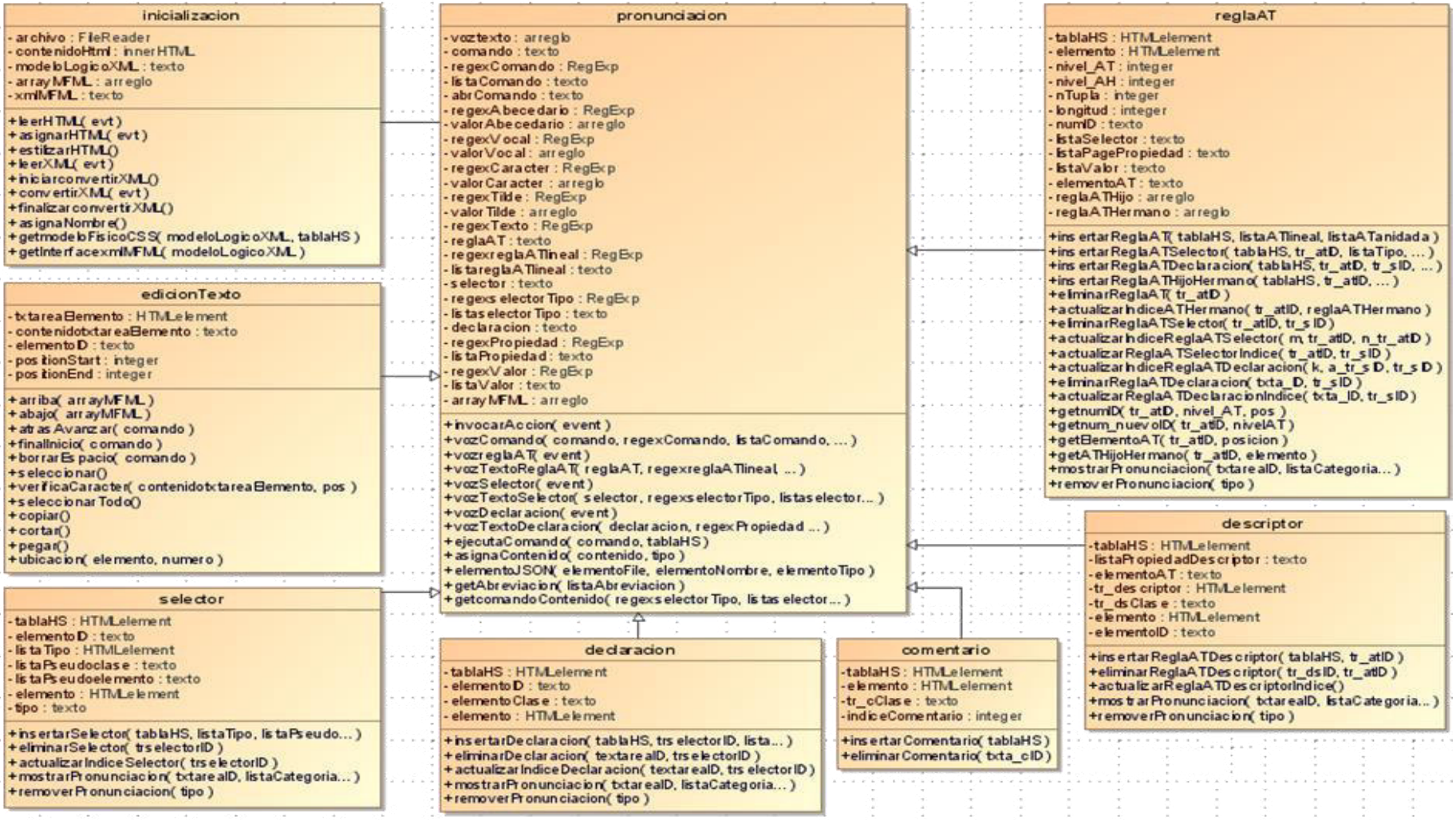


Figura 20 **UML - Diagrama de Clases del primer sprint.** Fuente: Elaboración propia

4.3.5.4. Especificación de Casos de uso

La especificación de casos de uso es utilizada para detallar un caso de uso. Las tablas siguientes muestran la especificación de casos de uso para cada caso de uso correspondiente al primer sprint.

A. Especificación Caso de uso – Cargar html

Tabla 27 **UML - Especificación Caso de uso Cargar html**

Caso de uso:	Cargar html
Fuentes	Julio Prudencio
Actor	Diseñador
Descripción	El caso de uso permite al diseñador subir el archivo html 5 sobre el cual se aplicará la hoja de estilo CSS 3 a desarrollar a través de comandos de voz
Flujo básico	1. **Seleccionar archivo html** El diseñador elige un archivo html. Una vez elegido el archivo el sistema procede a leer, asignar y estilizar el contenido del archivo para mostrarlo al usuario. El caso de uso termina.
Flujos alternos	1. **El archivo html es incorrecto** En el paso 1 del flujo básico el archivo seleccionado presenta inconsistencia en su estructura o no es el que se desea maquetar. El sistema permite al usuario elegir otro archivo, retornando el caso de uso al paso 1 del flujo básico.
Pre-condiciones	No existen para este caso de uso
Post-condiciones	1. **Archivo html cargado** El sistema muestra el contenido estilizado con colores distintivos pero no permite su edición.
Puntos de inclusión	No existen para este caso de uso
Puntos de extensión	No existen para este caso de uso

Fuente. Elaboración propia

Para una mejor compresión del caso de uso “Cargar html” se muestran a continuación los diagramas de actividad y secuencia. El propósito del diagrama de actividad es mostrar el comportamiento interno de un caso de uso representado a través de acciones. En este caso se verifica que luego de que el usuario ha seleccionado un archivo html el framework procede a extraer el contenido, verifica y valida su estructura para que finalmente pinte con colores distintivos las etiquetas html y presente el contenido al usuario.

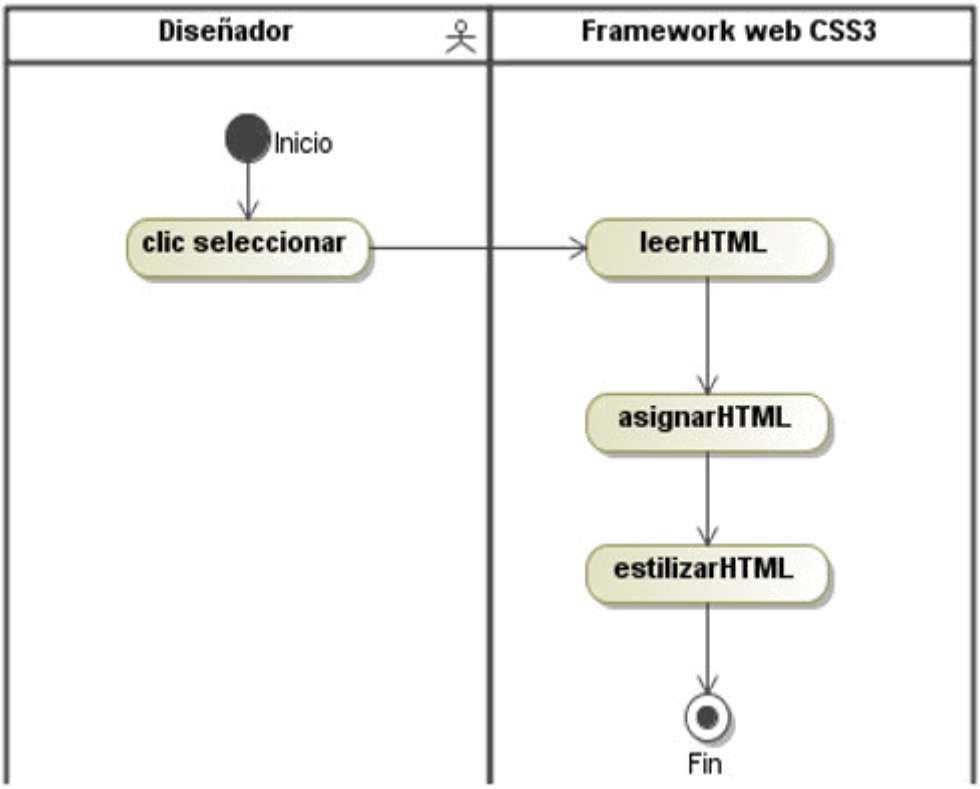


Figura 21 **UML - Diagrama de Actividad del Caso de uso Cargar html.**

Fuente: Elaboración propia

El propósito del diagrama de secuencia es mostrar la información de la interacción enfatizando la secuencia de tiempo. El eje Y representa la secuencia de tiempo y el eje X muestra los objetos participantes. En este caso de uso se verifica que el diseñador es quien inicia la secuencia de mensajes indicando que el sistema extraiga el contenido del archivo html que ha elegido, verifique y valide su estructura y finalmente estilice el contenido a ser mostrado al usuario.

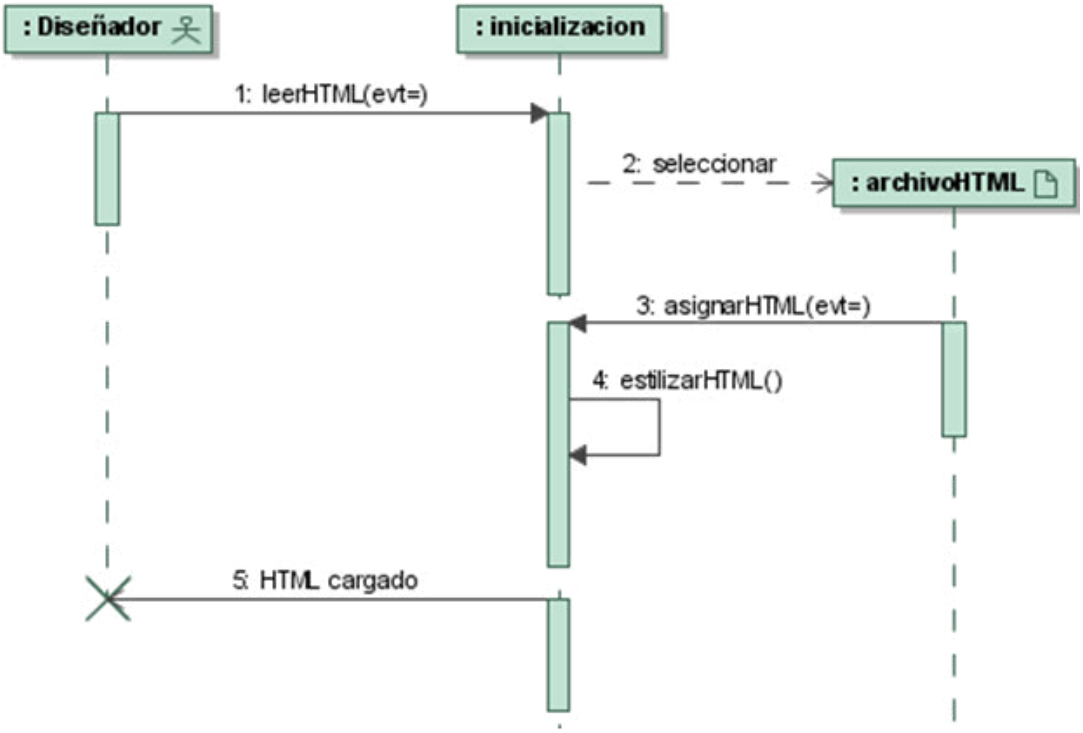


Figura 22 **UML - Diagrama de Secuencia del Caso de uso Cargar html.**

Fuente: Elaboración propia

B. Especificación Caso de uso – Importar xml

Tabla 28 **UML - Especificación Caso de uso Importar xml**

Caso de uso	Importar xml
Fuentes	Julio Prudencio
Actor	Diseñador
Descripción	El caso de uso permite al diseñador subir el archivo xml perteneciente al modelo lógico de la Interface MFML para generar el modelo físico (tabla html) que le permita seguir desarrollando la hoja de estilo a través de comandos de voz
Flujo básico	1. **Seleccionar archivo xml** El diseñador elige un archivo xml. Una vez elegido el archivo el sistema muestra un mensaje de procesamiento y procede a leer el contenido del archivo.

	2. **Generar modelo físico (tabla html)** El sistema procede a construir los elementos fila, columna y textarea, asignándoles su clase, ID, valor y nivel extraídos del archivo xml seleccionado. 3. **Presentar modelo físico al usuario** El sistema destruye el mensaje de procesamiento y muestra al usuario la tabla html generada. El caso de uso termina.
Flujos alternos	1. **El archivo xml es incorrecto** En el paso 1 del flujo básico el archivo seleccionado presenta inconsistencia en su estructura. El sistema permite al usuario elegir otro archivo, retornando el caso de uso al paso 1 del flujo básico.
Pre-condiciones	1. **Archivo html cargado** El sistema ha extraído, analizado, estilizado y mostrado al usuario el contenido del archivo html. 2. **Existencia del archivo xml** El sistema ha generado y exportado previamente el archivo xml.
Post-condiciones	1. **Archivo xml cargado** El sistema muestra la tabla html generada con colores distintivos y permite su edición.
Puntos de inclusión	No existen para este caso de uso
Puntos de extensión	No existen para este caso de uso

Fuente. Elaboración propia

Para una mejor compresión del caso de uso “Importar xml” se muestran a continuación los diagramas de actividad y secuencia. El propósito del diagrama de actividad es mostrar el comportamiento interno de un caso de uso representado a través de acciones. En este caso se aprecia que el usuario selecciona el archivo .xml y el framework

procede a realizar dos acciones en paralelo: la primera es mostrar un mensaje de procesamiento y la segunda acción es leer, verificar y validad la estructura del archivo xml y proceder a generar los elementos fila, columna y textarea de la tabla html. Finalmente, destruye el mensaje de procesamiento y muestra el resultado al diseñador.

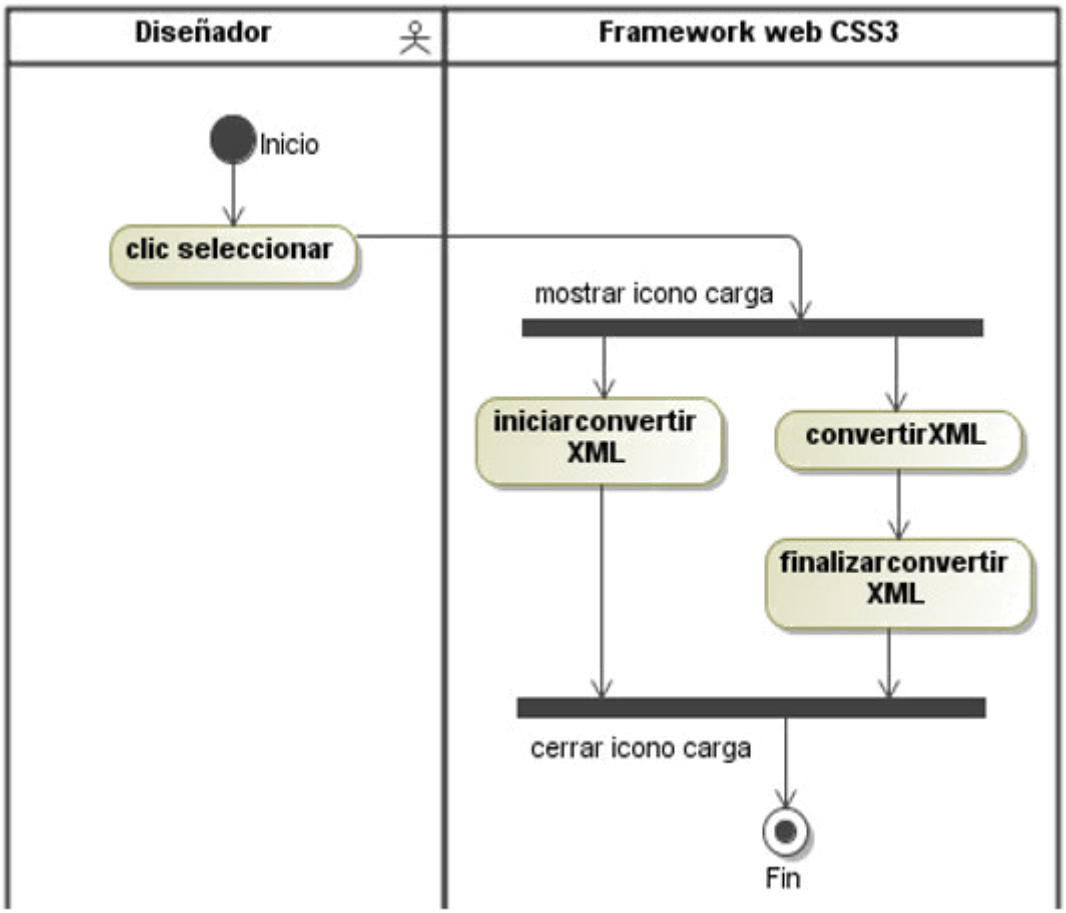


Figura 23 **UML - Diagrama de Actividad del Caso de uso Importar xml.**

Fuente: Elaboración propia

El propósito del diagrama de secuencia es mostrar la información de la interacción enfatizando la secuencia de tiempo. El eje Y representa la secuencia de tiempo y el eje X muestra los objetos participantes. En este caso el diseñador inicia la secuencia de mensajes indicando al framework leer el archivo xml que ha seleccionado. El framework entonces procede a mostrar un ícono de carga y en paralelo, luego de leer y validar el contenido del archivo xml, genera la tabla html la cual es mostrada al diseñador.

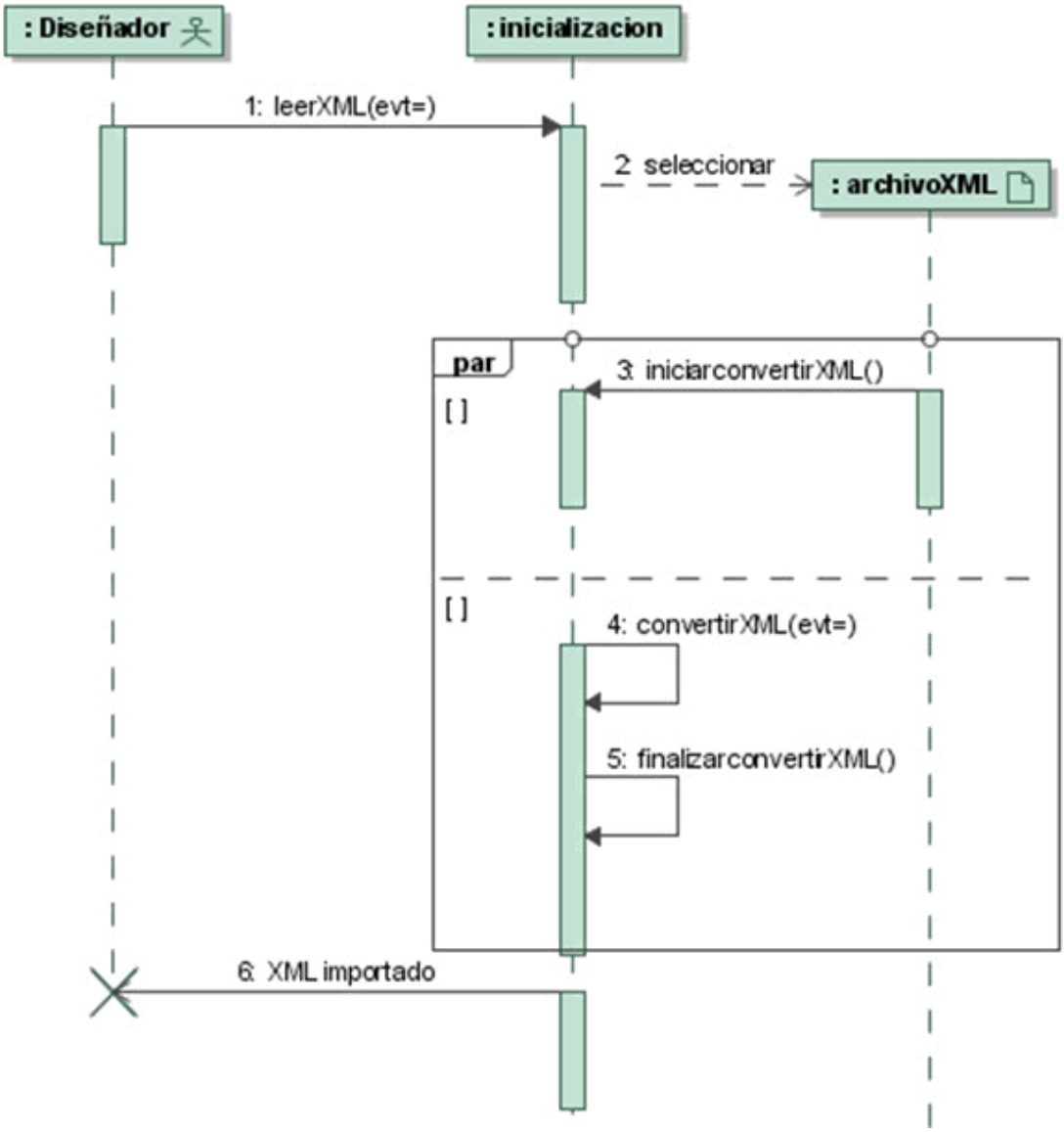


Figura 24 **UML - Diagrama de Secuencia del Caso de uso Importar xml.**

Fuente: Elaboración propia

C. Especificación Caso de uso – Crear css

Tabla 29 **UML - Especificación Caso de uso Crear css**

Caso de uso	Crear css
Fuentes	Julio Prudencio
Actor	Diseñador
Descripción	El caso de uso permite al diseñador especificar el nombre de la hoja de estilo CSS 3 a desarrollar a través de comandos de voz

Flujo básico	**1. Escribir nombre de la hoja de estilo** El diseñador ingresa el nombre a asignar a la hoja de estilo. El sistema valida que el nombre ingresado no contenga la extensión .css. El caso de uso termina.
Flujos alternos	**1. El nombre escrito es incorrecto** En el paso 1 del flujo básico el nombre de la hoja de estilo contiene la extensión .css. El sistema permite al usuario ingresar otro nombre, retornando el caso de uso al paso 1 del flujo básico.
Pre-condiciones	**1. Archivo html cargado** El sistema ha extraído, analizado, estilizado y mostrado al usuario el contenido del archivo html.
Post-condiciones	**1. Hoja de estilo con nombre válido** El nombre ha sido verificado y se ha validado la no existencia de la extensión .css
Puntos de inclusión	No existen para este caso de uso
Puntos de extensión	No existen para este caso de uso

Fuente. Elaboración propia

Para una mejor compresión del caso de uso “Crear css” se muestran a continuación los diagramas de actividad y secuencia. El propósito del diagrama de actividad es mostrar el comportamiento interno de un caso de uso representado a través de acciones. En este caso el diseñador es quien ingresa el nombre de la hoja de estilo a desarrollar y el framework es quien valida que dicho nombre no contenga la extensión .css. Si el framework detecta que el nombre ingresado contiene la extensión .css entonces rechaza el nombre, muestra un mensaje al diseñador y vuelve a solicitar su ingreso, en caso contrario acepta el nombre y finaliza la actividad.

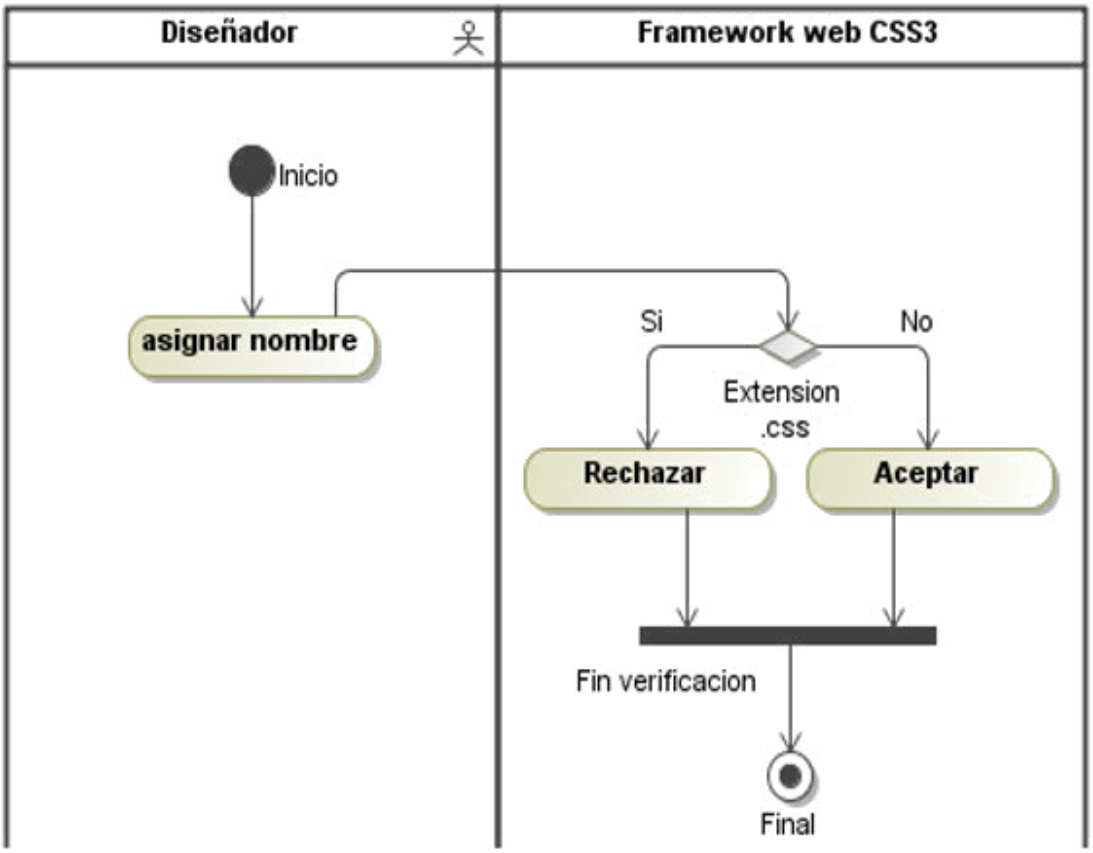


Figura 25 **UML - Diagrama de Actividad del Caso de uso Crear css.**

Fuente: Elaboración propia

El propósito del diagrama de secuencia es mostrar la información de la interacción enfatizando la secuencia de tiempo. El eje Y representa la secuencia de tiempo y el eje X muestra los objetos participantes. En este caso se visualiza claramente que el diseñador ingresa el nombre a asigna a la hoja de estilo, pero el framework realiza la comparación de dicho nombre verificando que no contenga la extensión .css. Si no contiene dicha extensión entonces se acepta, en caso contrario se rechaza.

Esto es realizado por la clase inicialización, la cual contiene, entre otras operaciones, a la operación llamada “asignaNombre”. Precisamente es esta operación la encargada de validar el ingreso del nombre de la hoja de estilo, y en última instancia es quien acepta o muestra un mensaje de rechazo indicando al diseñador ingresar un nombre válido para dicha hoja de estilo.

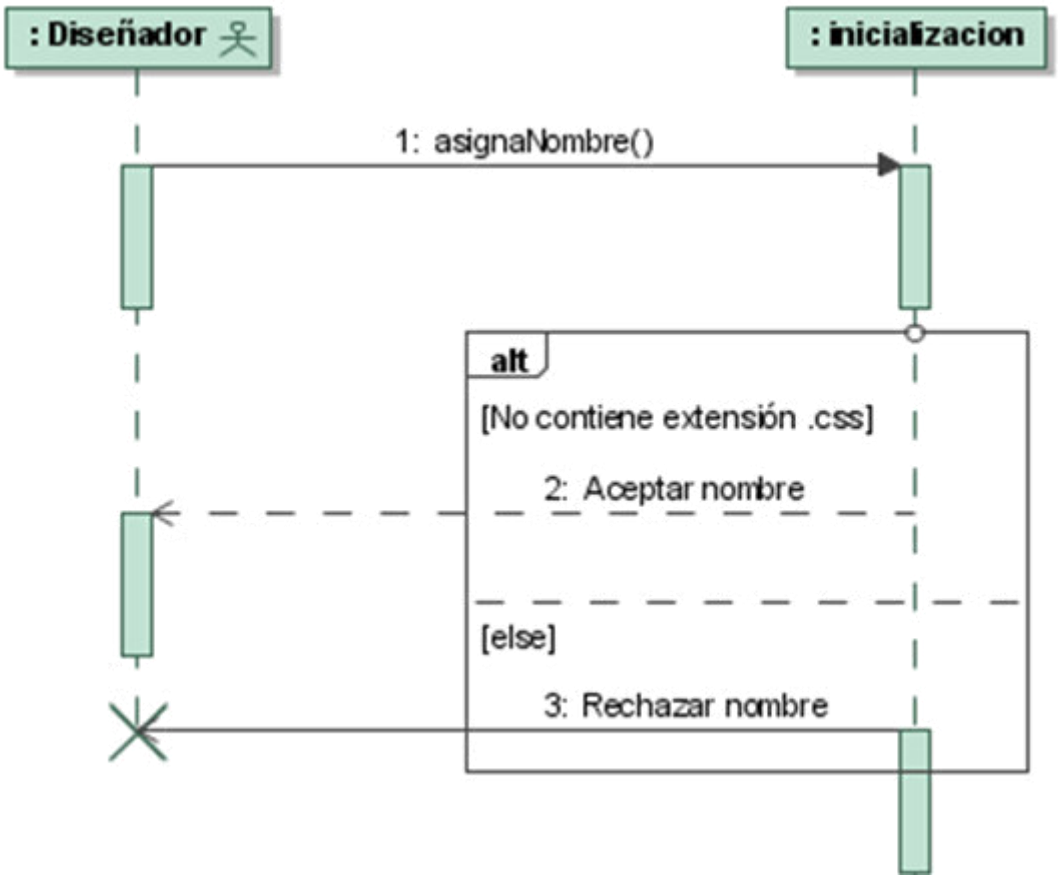


Figura 26 **UML - Diagrama de Secuencia del Caso de uso Crear css.**

Fuente: Elaboración propia

D. Especificación Caso de uso - Pronunciar comandos de voz

Tabla 30 **UML - Especificación Caso de uso Pronunciar comandos de voz**

Caso de uso	Pronunciar comandos de voz
Fuentes	Julio Prudencio
Actor	Diseñador
Descripción	El caso de uso permite al diseñador pronunciar instrucciones que el framework reconoce, llamados comandos de voz, para permitirle insertar, eliminar, asignar contenido o realizar funciones de edición de texto sobre el modelo físico de la Interface MFML (tabla html)
Flujo básico	**1. Pronunciar comando de voz**

	El diseñador pronuncia el comando de voz para realizar la acción de inserción o eliminación, asignar contenido o realizar funciones de edición de texto. **2. Enviar comando al servicio de reconocimiento de voz** El sistema envía el comando pronunciado al servicio de reconocimiento de voz para que sea transformado de audio a texto. **3. Validar comando pronunciado** El sistema recibe el comando en formato texto y valida que sea una instrucción válida y reconocida. **4. Ejecutar el comando pronunciado** Una vez identificado el comando el sistema procede a realizar la acción deseada: insertar o eliminar, asignar contenido o realizar una función de edición de texto. **5. Presentar el resultado al diseñador** El sistema muestra la acción realizada por el comando de voz pronunciado por el usuario. El caso de uso termina.
Flujos alternos	**1. El comando pronunciado es erróneo** En el paso 3 del flujo básico el comando pronunciado devuelto en formato texto no es válido. El sistema muestra un mensaje al usuario informando que la instrucción no es válida, regresando el caso de uso al paso 1 del flujo básico.
Pre-condiciones	**1. Archivo html cargado** El sistema ha extraído, analizado, estilizado y mostrado al usuario el contenido del archivo html

	2. **Archivo xml cargado** El sistema muestra la tabla html generada con colores distintivos y permite su edición 3. **Hoja de estilo con nombre válido** El nombre ha sido verificado y se ha validado la no existencia de la extensión .css
Post-condiciones	1. **Comando de voz realizado** El sistema ha realizado la acción correspondiente al comando de voz pronunciado
Requerimientos trazados	1. **Comando de voz fácil de pronunciar** Resulta indispensable que el comando a pronunciar sea lo más breve y fácil de pronunciar
Puntos de inclusión	1. Gestionar Edición Texto 2. Gestionar Comentario CSS 3. Gestionar Selector CSS 4. Gestionar Declaracion CSS 5. Gestionar reglaAT CSS 6. Gestionar Descriptor CSS
Puntos de extensión	No existen para este caso de uso

Fuente. Elaboración propia

Para una mejor compresión del caso de uso “Pronunciar comandos de voz” se muestran a continuación los diagramas de actividad y secuencia. El propósito del diagrama de actividad es mostrar el comportamiento interno de un caso de uso representado a través de acciones. En este caso se muestra que el diseñador es quien pronuncia el comando de voz y el sistema valida que el comando pronunciado sea correcto y reconocido. Si no fuera reconocido entonces el sistema envía un mensaje. En caso contrario se procede a verificar si se desea realizar alguna acción o se desea asignar algún contenido. Finalmente, la acción realizada o el contenido asignado es mostrado al diseñador.

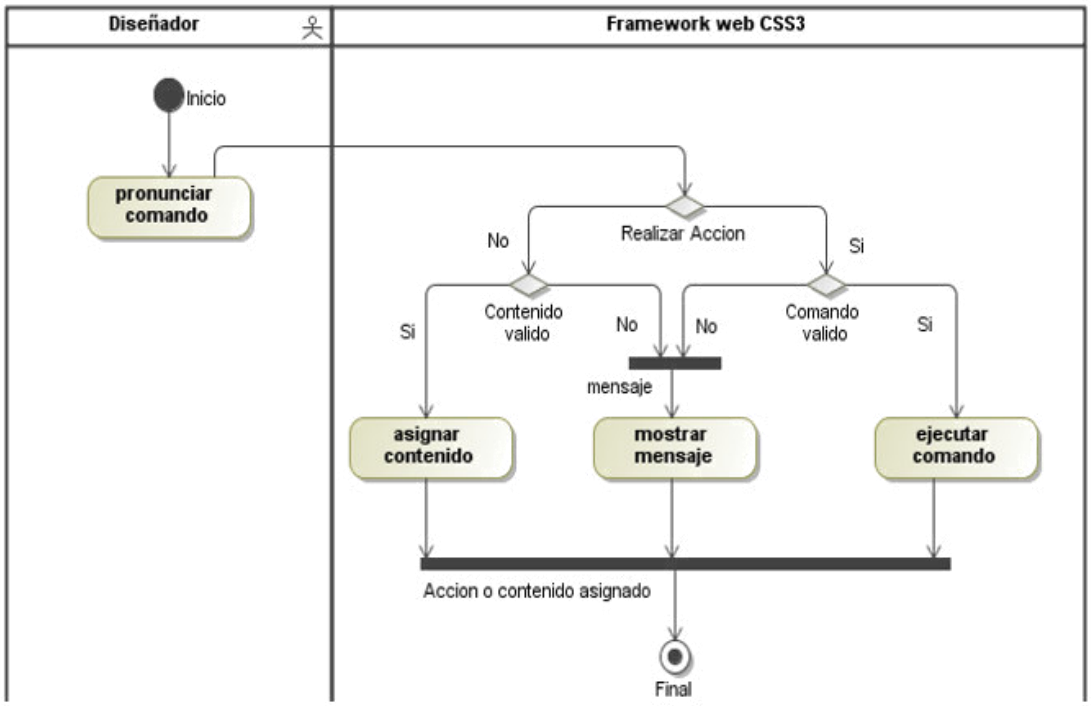


Figura 27 **UML - Diagrama de Actividad del Caso de uso Pronunciar comandos de voz.**

Fuente: Elaboración propia

El propósito del diagrama de secuencia es mostrar la información de la interacción enfatizando la secuencia de tiempo. El eje Y representa la secuencia de tiempo y el eje X muestra los objetos participantes. En este caso se muestra que el diseñador puede enviar tanto el mensaje de asignar contenido, así como de realizar alguna acción. Es así que la clase llamada "edicionTexto", a través de su operación "vozComando", verifica si se desea asignar algún contenido el cual es asignado mediante su operación "asignaContenido". Por otro lado, cuando el diseñador desea realizar la operación de inserción o eliminación de alguno de los elementos CSS: selector, declaración, regla-AT, descriptor o comentario, entonces se invoca a la operación "ejecutaComando". Así, la clase selector implementa el método "insertarSelector", la clase declaración implementa el método "insertarDeclaracion", la clase reglaAT implementa el método "insertarReglaAT", la clase descriptor implementa el método "insertarDescriptor" y la clase comentario implementa el método "insertarComentario".

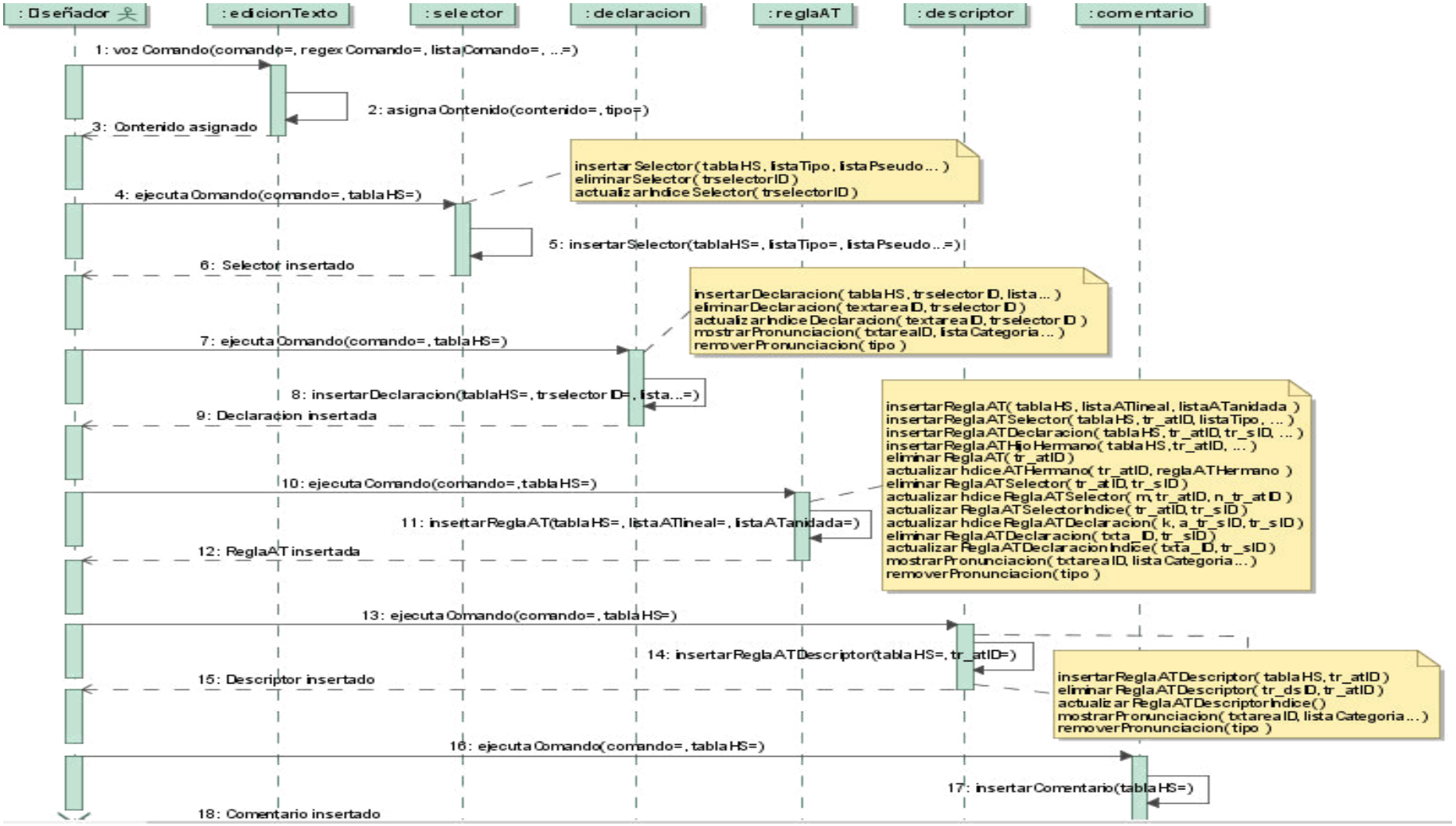


Figura 28 **UML - Diagrama de Secuencia del Caso de uso Pronunciar comandos de voz.** Fuente: Elaboración propia

E. Especificación Caso de uso - Gestionar edición texto

Tabla 31 **UML - Especificación Caso de uso Gestionar edicion texto**

Caso de uso	Gestionar edición texto
Fuentes	Julio Prudencio
Actor	Diseñador
Descripción	El caso de uso permite al diseñador pronunciar instrucciones que el framework reconoce para asignar contenido: selectores, declaraciones, reglas-AT, descriptores, comentarios. O para realizar funciones de edición de texto: subir, bajar, avanzar, retroceder, asignar el cursor al inicio del texto, asignar el cursor al final del texto, borrar contenido, asignar espacios entre textos, seleccionar todo o parte de un texto, copiar, cortar, pegar y desplazarse por los elementos del modelo físico de la Interface MFML (filas y columnas de la tabla html)
Flujo básico	1. **Pronunciar comando de voz** El diseñador pronuncia el comando de voz para realizar la acción de asignar contenido o realizar funciones de edición de texto. 2. **Enviar comando al servicio de reconocimiento de voz** El sistema envía el comando pronunciado al servicio de reconocimiento de voz para que sea transformado de audio a texto. 3. **Validar comando pronunciado** El sistema recibe el comando en formato texto y valida que sea una instrucción válida y reconocida. 4. **Ejecutar el comando pronunciado** Una vez identificado el comando el sistema procede a asignar contenido o realizar una

	función de edición de texto: subir, bajar, avanzar, retroceder, asignar el cursor al inicio del texto, asignar el cursor al final del texto, borrar contenido, asignar espacios entre textos, seleccionar todo o parte de un texto, copiar, cortar, pegar y desplazarse por los elementos del modelo físico de la Interface MFML **5. Presentar el resultado al diseñador** El sistema muestra el contenido asignado o la acción realizada por el comando de voz pronunciado por el usuario. El caso de uso termina.
Flujos alternos	**1. El comando pronunciado es erróneo** En el paso 3 del flujo básico el comando pronunciado devuelto en formato texto no es válido. El sistema muestra un mensaje al usuario informando que la instrucción no es válida, retornando el caso de uso al paso 1 del flujo básico.
Pre-condiciones	**1. Archivo html cargado** El sistema ha extraído, analizado, estilizado y mostrado al usuario el contenido del archivo html **2. Archivo xml cargado** El sistema muestra la tabla html generada con colores distintivos y permite su edición **3. Hoja de estilo con nombre válido** El nombre ha sido verificado y se ha validado la no existencia de la extensión .css
Post-condiciones	**1. Comando de voz realizado**

	El sistema ha asignado el contenido o realizado la acción correspondiente al comando de voz pronunciado
Puntos de inclusión	No existen para este caso de uso
Puntos de extensión	No existen para este caso de uso

Fuente. Elaboración propia

Para una mejor compresión del caso de uso "Gestionar Edición Texto" se muestran a continuación los diagramas de actividad y secuencia. El propósito del diagrama de actividad es mostrar el comportamiento interno de un caso de uso representado a través de acciones. En este caso el diseñador pronuncia el comando a realizar o el contenido a asignar y el framework evalúa el comando si el comando pronunciado es válido o no. Si no es válido muestra un mensaje al diseñador indicando que la instrucción no es válida. En caso contrario, se verifica si el comando es para asignar contenido de texto o es para realizar alguna acción de edición de texto. Finalmente, el framework muestra el contenido asignado o la acción de edición de texto realizada: subir, bajar, avanzar, retroceder, asignar el cursor al inicio del texto, asignar el cursor al final del texto, borrar contenido, asignar espacios entre textos, seleccionar todo o parte de un texto, copiar, cortar, pegar y desplazarse por los elementos de la tabla html.

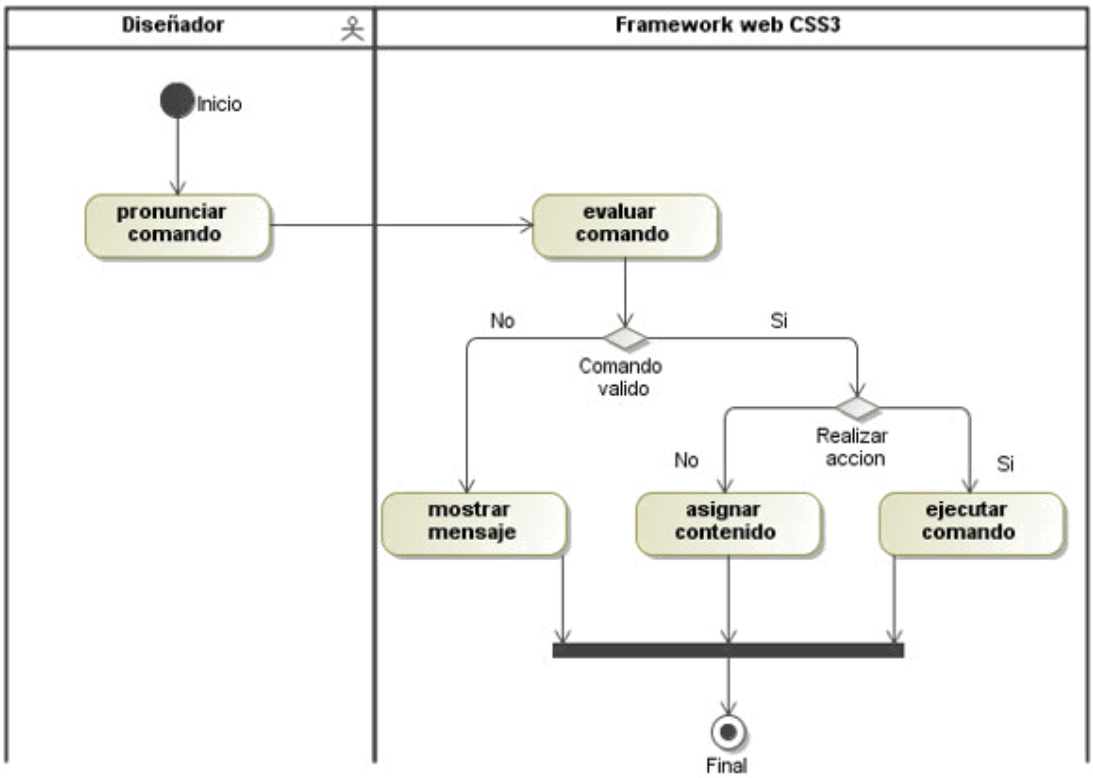


Figura 29 **UML - Diagrama de Actividad del Caso de uso Gestionar edicion texto.**

Fuente: Elaboración propia

El propósito del diagrama de secuencia es mostrar la información de la interacción enfatizando la secuencia de tiempo. El eje Y representa la secuencia de tiempo y el eje X muestra los objetos participantes. En este caso el diseñador inicia el mensaje de asignar algún contenido: selectores, declaraciones, reglas-AT, descriptores o comentarios. O realizar alguna función de edición de texto.

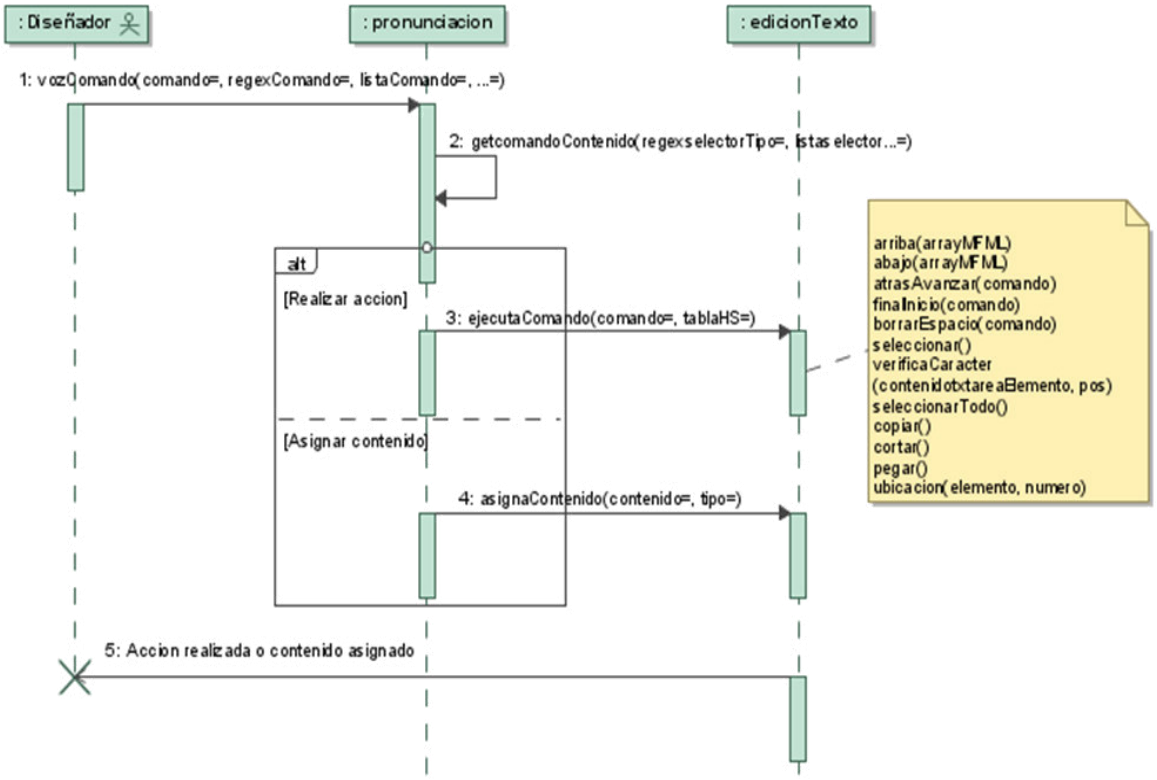


Figura 30 **UML - Diagrama de Secuencia del Caso de uso Gestionar edicion texto.**

Fuente: Elaboración propia

F. Especificación Caso de uso – Gestionar selector CSS

Tabla 32 **UML - Especificación Caso de uso Gestionar selector css**

Caso de uso	Gestionar selector CSS
Fuentes	Julio Prudencio
Actor	Diseñador
Descripción	El caso de uso permite al diseñador insertar o eliminar selectores
Flujo básico	1. **Pronunciar comando de voz** El diseñador pronuncia el comando de voz para insertar o eliminar un selector. 2. **Enviar comando al servicio de reconocimiento de voz** El sistema envía el comando pronunciado al servicio de reconocimiento de voz para que sea transformado de audio a texto.

	3. **Validar comando pronunciado** El sistema recibe el comando en formato texto y valida que sea una instrucción válida y reconocida de inserción o eliminación de un selector. 4. **Ejecutar el comando pronunciado** Una vez identificado el comando el sistema procede a insertar o eliminar el selector dentro del modelo físico de la Interface MFML 5. **Presentar el resultado al diseñador** El sistema muestra el selector insertado o eliminado por el comando de voz pronunciado por el usuario. El caso de uso termina.
Flujos alternos	1. **El comando pronunciado es erróneo** En el paso 3 del flujo básico el comando pronunciado devuelto en formato texto no es válido. El sistema muestra un mensaje al usuario informando que la instrucción no es válida, retornando el caso de uso al paso 1 del flujo básico. 2. **Mostrar lista completa de selectores** En el paso 5 del flujo básico luego de mostrar el selector insertado el sistema procede a mostrar la lista completa de selectores y su forma de pronunciación. 3. **Remover lista completa de selectores** En el paso 5 del flujo básico luego de eliminar el selector el sistema procede a remover la lista completa de selectores y su forma de pronunciación.
Pre-condiciones	1. **Archivo html cargado**

	El sistema ha extraído, analizado, estilizado y mostrado al usuario el contenido del archivo html 2. **Archivo xml cargado** El sistema muestra la tabla html generada con colores distintivos y permite su edición 3. **Hoja de estilo con nombre válido** El nombre ha sido verificado y se ha validado la no existencia de la extensión .css
Post-condiciones	1. **Selector insertado o eliminado** El sistema ha insertado o eliminado el selector, correspondiente al comando de voz pronunciado
Puntos de inclusión	No existen para este caso de uso
Puntos de extensión	No existen para este caso de uso

Fuente. Elaboración propia

Para una mejor compresión del caso de uso “Gestionar selector CSS” se muestran a continuación el diagrama de secuencia. El propósito del diagrama de secuencia es mostrar la información de la interacción enfatizando la secuencia de tiempo. El eje Y representa la secuencia de tiempo y el eje X muestra los objetos participantes. En este caso el diseñador envía el mensaje a la clase pronunciación que le indica al sistema insertar o eliminar un selector a través de la operación "vozComando". Esta clase verifica que el comando pronunciado, el cual ha sido devuelto en formato texto por el servicio de reconocimiento de voz, sea un comando válido y reconocido lo cual es realizado a través de la operación "getcomandoContenido".

Luego de verificar la validez del comando la clase pronunciación procede a enviar el mensaje a la clase selector que le indica que se desea insertar o eliminar un selector a través de la operación "ejecutaComando". Luego de realizar la operación de inserción, si fuese el caso, la clase selector mostrará la lista completa de selectores: de tipo, de pseudoclase y de pseudoelemento, así como sus formas de pronunciación a través de la operación: "mostrarPronunciacion". En caso se haya eliminado el selector, entonces

la clase selector invocará a la operación: "removerPronunciacion" la cual eliminará el listado completo de selectores.

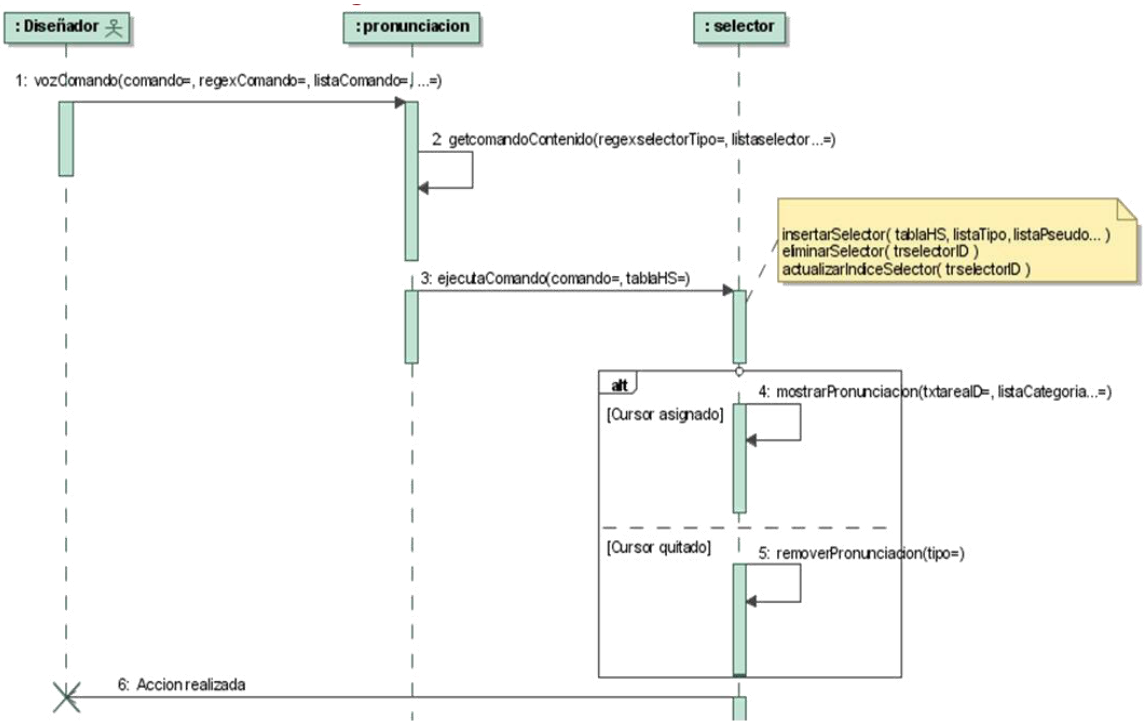


Figura 31 **UML - Diagrama de Secuencia del Caso de uso Gestionar selector css.**

Fuente: Elaboración propia

G. Especificación Caso de uso – Gestionar declaración CSS

Tabla 33 **UML - Especificación Caso de uso Gestionar declaracion css**

Caso de uso	Gestionar declaración CSS
Fuentes	Julio Prudencio
Actor	Diseñador
Descripción	El caso de uso permite al diseñador insertar o eliminar declaraciones
Flujo básico	1. **Pronunciar comando de voz** El diseñador pronuncia el comando de voz para insertar o eliminar una declaración. 2. **Enviar comando al servicio de reconocimiento de voz**

	El sistema envía el comando pronunciado al servicio de reconocimiento de voz para que sea transformado de audio a texto. **3. Validar comando pronunciado** El sistema recibe el comando en formato texto y valida que sea una instrucción válida y reconocida de inserción o eliminación de una declaración. **4. Ejecutar el comando pronunciado** Una vez identificado el comando el sistema procede a insertar o eliminar la declaración dentro del modelo físico de la Interface MFML **5. Presentar el resultado al diseñador** El sistema muestra la declaración insertada o eliminada por el comando de voz pronunciado por el usuario. El caso de uso termina.
Flujos alternos	**1. El comando pronunciado es erróneo** En el paso 3 del flujo básico el comando pronunciado devuelto en formato texto no es válido. El sistema muestra un mensaje al usuario informando que la instrucción no es válida, retornando el caso de uso al paso 1 del flujo básico. **2. Mostrar lista completa de propiedades** En el paso 5 del flujo básico luego de mostrar la declaración insertada el sistema procede a mostrar la lista completa de propiedades y su forma de pronunciación. **3. Remover lista completa de propiedades** En el paso 5 del flujo básico luego de eliminar la declaración el sistema procede a

	remover la lista completa de propiedades y su forma de pronunciación. 4. **Mostrar lista completa de valores** En el paso 5 del flujo básico luego mostrar la declaración insertada el sistema procede a mostrar la lista completa de valores y su forma de pronunciación, en función a una determinada propiedad elegida. 5. **Remover lista completa de valores** En el paso 5 del flujo básico luego de eliminar la declaración el sistema procede a remover la lista completa de valores y su forma de pronunciación, en función a una determinada propiedad elegida.
Pre-condiciones	1. **Archivo html cargado** El sistema ha extraído, analizado, estilizado y mostrado al usuario el contenido del archivo html 2. **Archivo xml cargado** El sistema muestra la tabla html generada con colores distintivos y permite su edición 3. **Hoja de estilo con nombre válido** El nombre ha sido verificado y se ha validado la no existencia de la extensión .css
Post-condiciones	1. **Declaración insertada o eliminada** El sistema ha insertado o eliminado la declaración, correspondiente al comando de voz pronunciado
Puntos de inclusión	No existen para este caso de uso
Puntos de extensión	No existen para este caso de uso

Fuente. Elaboración propia

Para una mejor compresión del caso de uso “Gestionar declaración CSS” se muestran a continuación el diagrama de secuencia. El propósito del diagrama de secuencia es

mostrar la información de la interacción enfatizando la secuencia de tiempo. El eje Y representa la secuencia de tiempo y el eje X muestra los objetos participantes. En este caso el diseñador envía el mensaje a la clase pronunciación que le indica al sistema insertar o eliminar una declaración a través de la operación "vozComando". Esta clase verifica que el comando pronunciado, el cual ha sido devuelto en formato texto por el servicio de reconocimiento de voz, sea un comando válido y reconocido lo cual es realizado a través de la operación "getcomandoContenido".

Luego de verificar la validez del comando la clase pronunciación procede a enviar el mensaje a la clase declaración que le indica que se desea insertar o eliminar una declaración a través de la operación "ejecutaComando". Luego de realizar la operación de inserción, si fuese el caso, la clase declaración mostrará la lista completa de propiedades, así como sus formas de pronunciación a través de la operación: "mostrarPronunciacion". En caso se haya eliminado la declaración, entonces la clase declaración invocará a la operación: "removerPronunciacion" la cual eliminará el listado completo de propiedades.

El mismo caso sucede para los valores de cada una de las propiedades elegidas. Es decir, según la propiedad pronunciada por el diseñador y asignada en el framework por la clase edicionTexto, expuesta en el caso de uso Gestionar edición texto, se creará una lista completa de todos los valores pertenecientes a dicha propiedad, en el caso de que la acción a realizar haya sido insertar declaración. En el caso de que la acción a realizar haya sido eliminar declaración, entonces se removerá también la lista completa de valores, según la propiedad asignada.

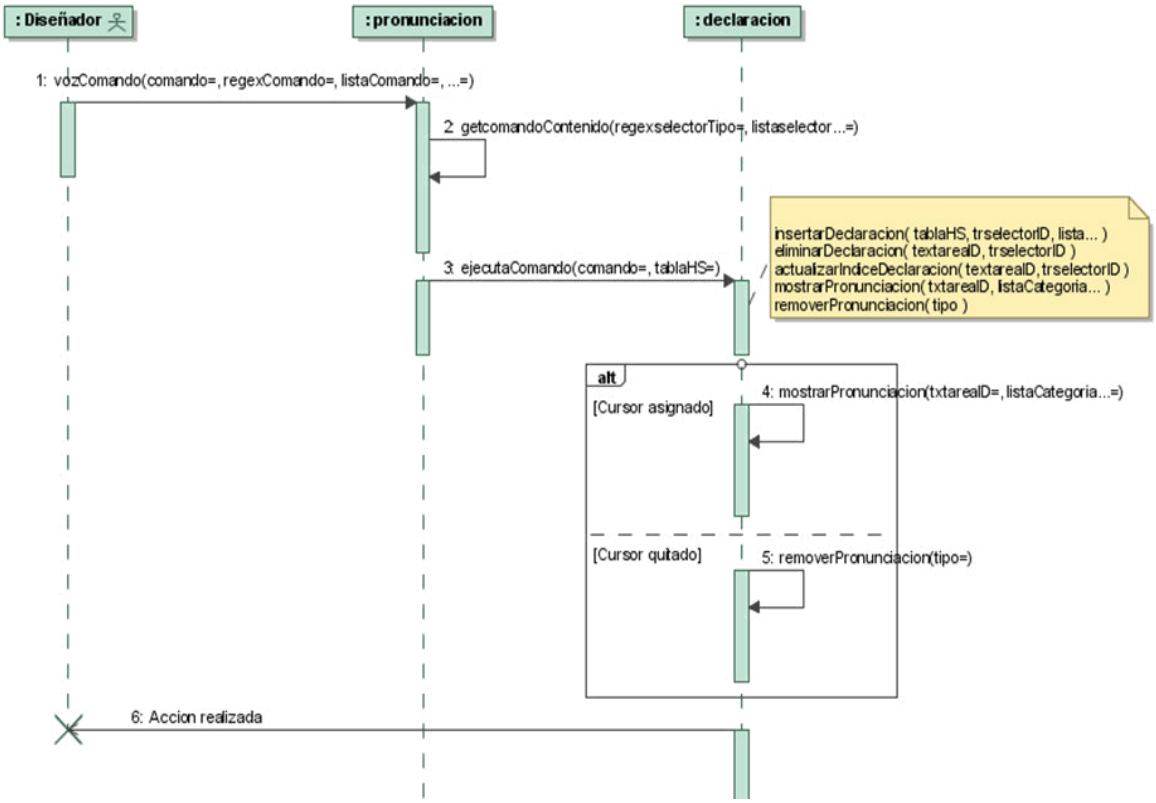


Figura 32 **UML - Diagrama de Secuencia del Caso de uso Gestionar declaracion css.**

Fuente: Elaboración propia

H. Especificación Caso de uso – Gestionar reglaAT CSS

Tabla 34 **UML - Especificación Caso de uso Gestionar reglaAT css**

Caso de uso	Gestionar reglaAT CSS
Fuentes	Julio Prudencio
Actor	Diseñador
Descripción	El caso de uso permite al diseñador insertar o eliminar reglas-AT lineales o anidadas
Flujo básico	1. **Pronunciar comando de voz** El diseñador pronuncia el comando de voz para insertar o eliminar una regla-AT. 2. **Enviar comando al servicio de reconocimiento de voz**

	El sistema envía el comando pronunciado al servicio de reconocimiento de voz para que sea transformado de audio a texto. **3. Validar comando pronunciado** El sistema recibe el comando en formato texto y valida que sea una instrucción válida y reconocida de inserción o eliminación de una regla-AT. **4. Ejecutar el comando pronunciado** Una vez identificado el comando el sistema procede a insertar o eliminar la regla-AT dentro del modelo físico de la Interface MFML **5. Presentar el resultado al diseñador** El sistema muestra la regla-AT insertada o eliminada por el comando de voz pronunciado por el usuario. El caso de uso termina.
Flujos alternos	**1. El comando pronunciado es erróneo** En el paso 3 del flujo básico el comando pronunciado devuelto en formato texto no es válido. El sistema muestra un mensaje al usuario informando que la instrucción no es válida, retornando el caso de uso al paso 1 del flujo básico. **2. Mostrar lista completa de reglas-AT** En el paso 5 del flujo básico luego de mostrar la regla-AT insertada el sistema procede a mostrar la lista completa de reglas-AT. **3. Remover lista completa de reglas-AT** En el paso 5 del flujo básico luego de eliminar la regla-AT el sistema procede a remover la lista completa de reglas-AT.

Pre-condiciones	1. **Archivo html cargado** El sistema ha extraído, analizado, estilizado y mostrado al usuario el contenido del archivo html 2. **Archivo xml cargado** El sistema muestra la tabla html generada con colores distintivos y permite su edición 3. **Hoja de estilo con nombre válido** El nombre ha sido verificado y se ha validado la no existencia de la extensión .css
Post-condiciones	1. **Regla-AT insertada o eliminada** El sistema ha insertado o eliminado la regla-AT, correspondiente al comando de voz pronunciado
Puntos de inclusión	No existen para este caso de uso
Puntos de extensión	No existen para este caso de uso

Fuente. Elaboración propia

Para una mejor compresión del caso de uso "Gestionar reglaAT CSS" se muestran a continuación los diagramas de actividad y secuencia. El propósito del diagrama de actividad es mostrar el comportamiento interno de un caso de uso representado a través de acciones. En este caso el diseñador puede pronunciar el comando de voz de inserción o eliminación de una regla-AT la cual es evaluada por el framework para determinar si es un comando válido o no. En caso de ser un comando no válido se procede a mostrar un mensaje al diseñador indicando que la instrucción es no válida. En el caso de que el comando sea identificado como válido y sea para insertar una regla-AT, entonces luego de insertado el framework muestra la lista completa de reglas-AT lineales: import y namespace, así como la lista completa de reglas-AT anidadas: media, document, page, supports, counter-style, keyframes y font-feature-values, ordenadas ascendentemente en orden alfabético.

En el caso de que el comando de voz pronunciado y validado sea referente a la eliminación de una regla-AT, entonces luego de su eliminación se procede también a remover la lista completa de reglas-AT lineales y anidadas.

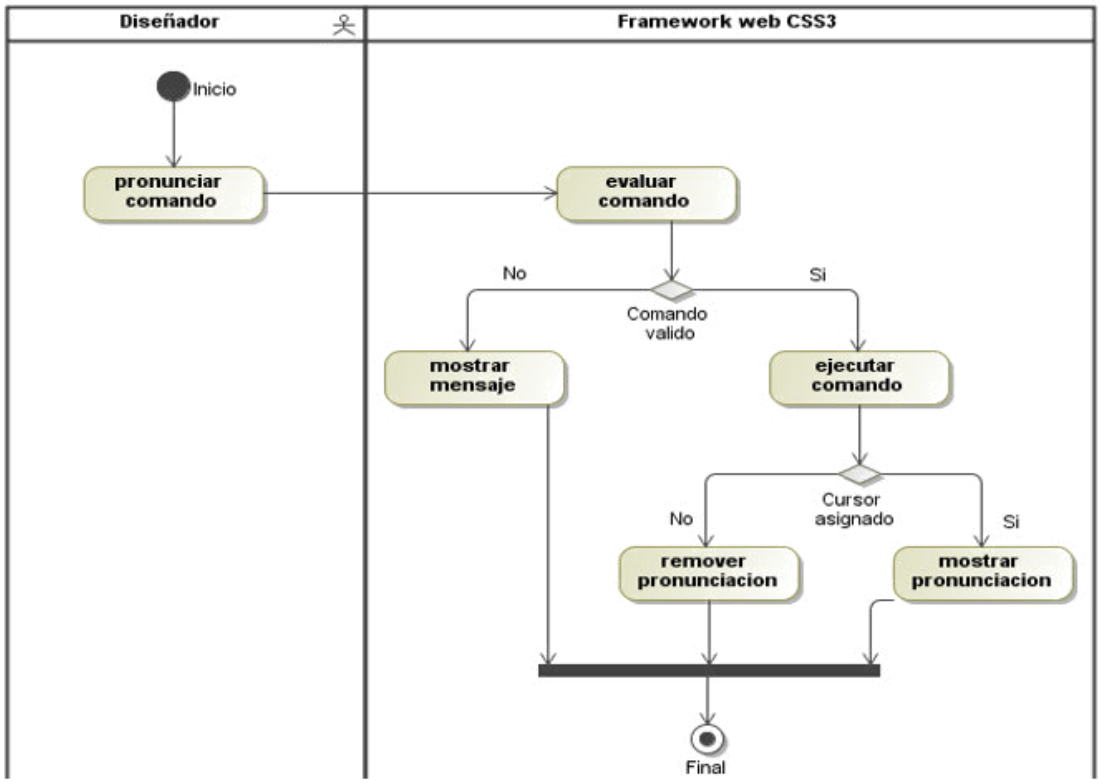


Figura 33 **UML - Diagrama de Actividad del Caso de uso Gestionar reglaAT css.**

Fuente: Elaboración propia

El propósito del diagrama de secuencia es mostrar la información de la interacción enfatizando la secuencia de tiempo. El eje Y representa la secuencia de tiempo y el eje X muestra los objetos participantes. En este caso el diseñador es quien envía el mensaje de inserción o eliminación de una regla-AT a través de la operación "vozComando" de la clase pronunciación. Esta, luego de obtener el comando pronunciado y transformado en formato texto por el servicio de reconocimiento de voz, evalúa si dicho comando es válido y reconocido a través de la operación "getcomandoContenido" para que posteriormente invoque a la clase reglaAT con la finalidad de realizar la inserción o eliminación a través de la operación "ejecutaComando". Luego de insertada la regla-AT en caso de que la instrucción haya sido de inserción, entonces el framework mostrará la lista completa de todas las reglas-AT junto con sus respectivas pronunciaciones. En caso de que se haya eliminado la regla-AT, entonces se procederá a remover la lista completa de reglas-AT.

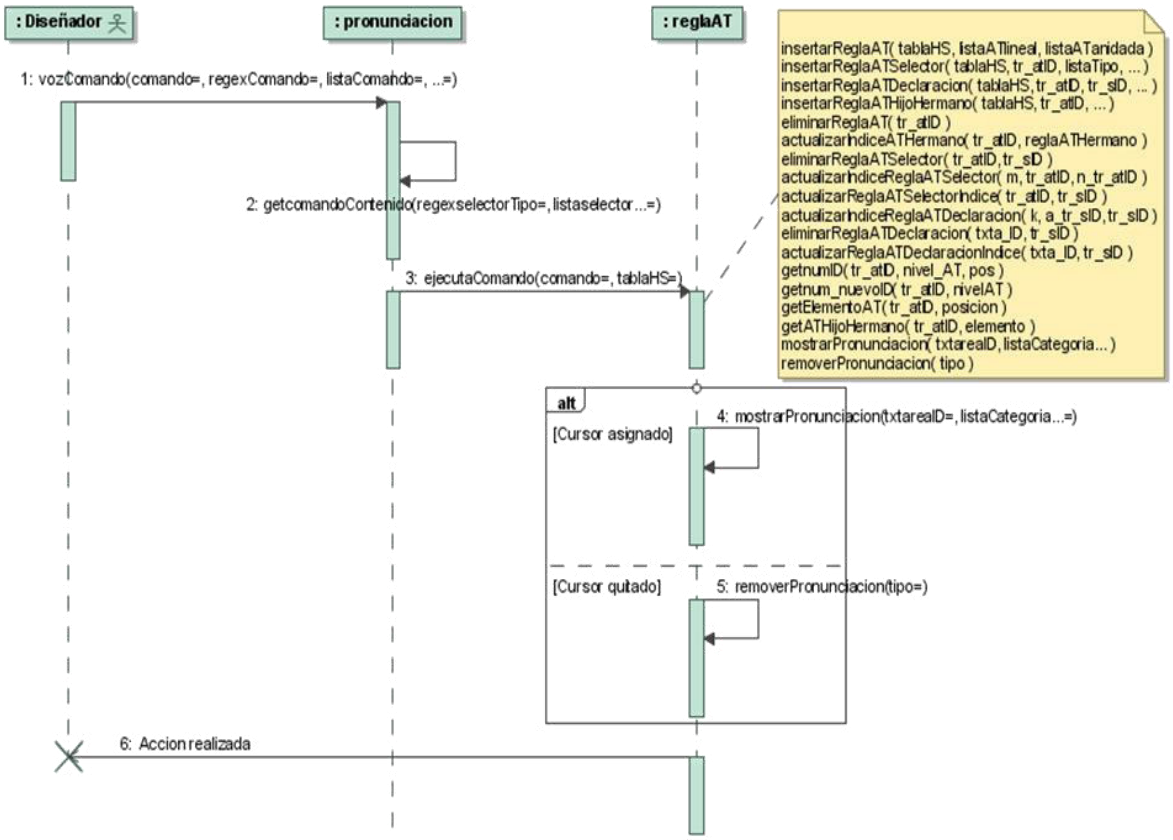


Figura 34 **UML - Diagrama de Secuencia del Caso de uso Gestionar reglaAT css.**

Fuente: Elaboración propia

I. Especificación Caso de uso – Gestionar descriptor CSS

Tabla 35 **UML - Especificación Caso de uso Gestionar descriptor CSS**

Caso de uso	Gestionar descriptor CSS
Fuentes	Julio Prudencio
Actor	Diseñador
Descripción	El caso de uso permite al diseñador insertar o eliminar descriptores pertenecientes a reglas-AT
Flujo básico	1. **Pronunciar comando de voz** El diseñador pronuncia el comando de voz para insertar o eliminar un descriptor. 2. **Enviar comando al servicio de reconocimiento de voz**

	El sistema envía el comando pronunciado al servicio de reconocimiento de voz para que sea transformado de audio a texto. **3. Validar comando pronunciado** El sistema recibe el comando en formato texto y valida que sea una instrucción válida y reconocida de inserción o eliminación de un descriptor. **4. Ejecutar el comando pronunciado** Una vez identificado el comando el sistema procede a insertar o eliminar el descriptor dentro del modelo físico de la Interface MFML **5. Presentar el resultado al diseñador** El sistema muestra el descriptor insertado o eliminado por el comando de voz pronunciado por el usuario. El caso de uso termina.
Flujos alternos	**1. El comando pronunciado es erróneo** En el paso 3 del flujo básico el comando pronunciado devuelto en formato texto no es válido. El sistema muestra un mensaje al usuario informando que la instrucción no es válida, retornando el caso de uso al paso 1 del flujo básico. **2. Mostrar lista completa de propiedades** En el paso 5 del flujo básico luego de mostrar el descriptor insertado el sistema procede a mostrar la lista completa de propiedades y su forma de pronunciación según la regla-AT a la que pertenezca. **3. Remover lista completa de propiedades**

	En el paso 5 del flujo básico luego de eliminar el descriptor el sistema procede a remover la lista completa de propiedades y su forma de pronunciación según la regla-AT a la que pertenezca. **4. Mostrar lista completa de valores** En el paso 5 del flujo básico luego mostrar el descriptor insertado el sistema procede a mostrar la lista completa de valores y su forma de pronunciación, en función a una determinada propiedad elegida. **5. Remover lista completa de valores** En el paso 5 del flujo básico luego de eliminar el descriptor el sistema procede a remover la lista completa de valores y su forma de pronunciación, en función a una determinada propiedad elegida.
Pre-condiciones	**1. Archivo html cargado** El sistema ha extraído, analizado, estilizado y mostrado al usuario el contenido del archivo html **2. Archivo xml cargado** El sistema muestra la tabla html generada con colores distintivos y permite su edición **3. Hoja de estilo con nombre válido** El nombre ha sido verificado y se ha validado la no existencia de la extensión .css
Post-condiciones	**1. Descriptor insertado o eliminado** El sistema ha insertado o eliminado el descriptor, correspondiente al comando de voz pronunciado
Puntos de inclusión	No existen para este caso de uso
Puntos de extensión	No existen para este caso de uso

Fuente. Elaboración propia

Para una mejor compresión del caso de uso “Gestionar descriptor CSS” se muestran a continuación el diagrama de secuencia. El propósito del diagrama de secuencia es mostrar la información de la interacción enfatizando la secuencia de tiempo. El eje Y representa la secuencia de tiempo y el eje X muestra los objetos participantes. En este caso el diseñador envía el mensaje a la clase pronunciación que le indica al sistema insertar o eliminar un descriptor a través de la operación "vozComando". Esta clase verifica que el comando pronunciado, el cual ha sido devuelto en formato texto por el servicio de reconocimiento de voz, sea un comando válido y reconocido lo cual es realizado a través de la operación "getcomandoContenido".

Luego de verificar la validez del comando la clase pronunciación procede a enviar el mensaje a la clase descriptor que le indica que se desea insertar o eliminar un descriptor a través de la operación "ejecutaComando". Luego de realizar la operación de inserción, si fuese el caso, la clase descriptor mostrará la lista completa de propiedades según la regla-AT a la cual pertenezca, así como sus formas de pronunciación a través de la operación: "mostrarPronunciacion". En caso se haya eliminado el descriptor, entonces la clase descriptor invocará a la operación: "removerPronunciacion" la cual eliminará el listado completo de propiedades según la regla-AT a la cual pertenezca.

El mismo caso sucede para los valores de cada una de las propiedades elegidas. Es decir, según la propiedad pronunciada por el diseñador y asignada en el framework por la clase edicionTexto, expuesta en el caso de uso Gestionar edición texto, se creará una lista completa de todos los valores pertenecientes a dicha propiedad, en el caso de que la acción a realizar haya sido insertar descriptor. En el caso de que la acción a realizar haya sido eliminar descriptor, entonces se removerá también la lista completa de valores, según la propiedad asignada según la regla-AT a la cual pertenezca.

Esto significa que las propiedades que definen en este caso a los descriptores, son mostradas en función a una regla-AT a la cual pertenecen. Así, por ejemplo, la regla-AT anidada page, por definición sólo contiene descriptores cuyas propiedades pertenecen y pueden modificar sólo a las propiedades CSS: margin, padding, border, background.

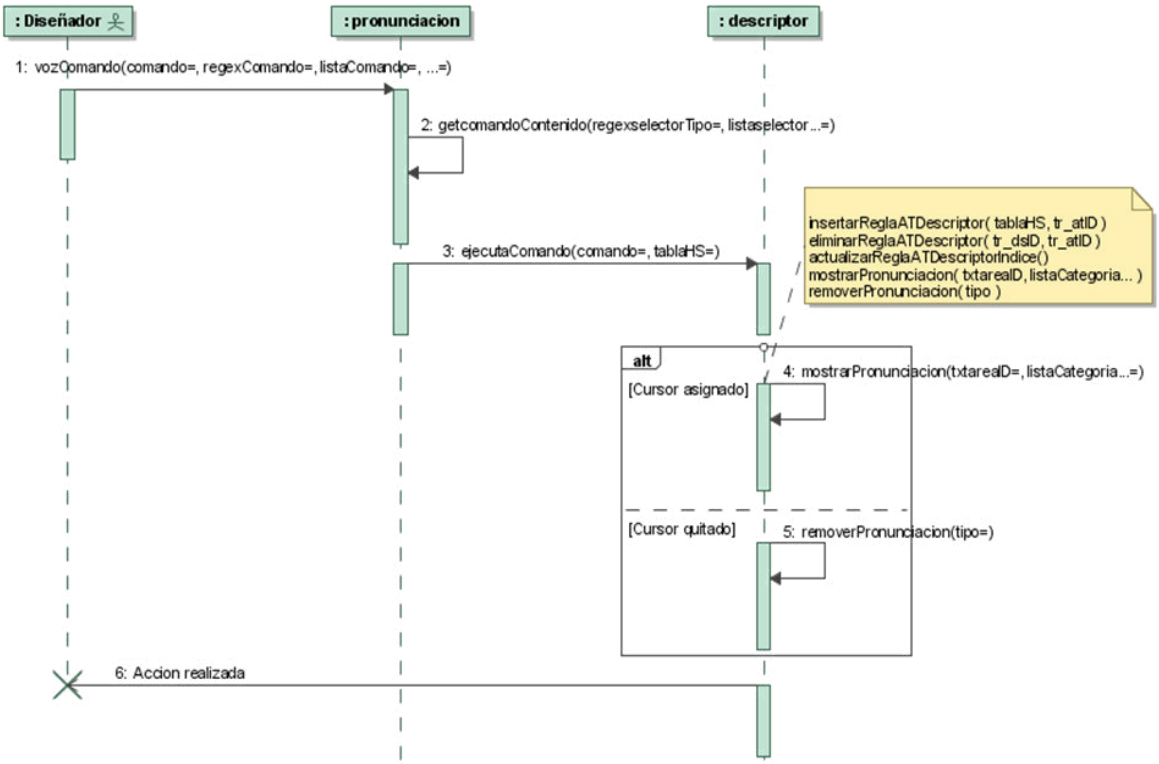


Figura 35 **UML - Diagrama de Secuencia del Caso de uso Gestionar descriptor css.**

Fuente: Elaboración propia

J. Especificación Caso de uso – Gestionar comentario CSS

Tabla 36 **UML - Especificación Caso de uso Gestionar comentario css**

Caso de uso	Gestionar comentario CSS
Fuentes	Julio Prudencio
Actor	Diseñador
Descripción	El caso de uso permite al diseñador insertar o eliminar comentarios
Flujo básico	1. **Pronunciar comando de voz** El diseñador pronuncia el comando de voz para insertar o eliminar un comentario. 2. **Enviar comando al servicio de reconocimiento de voz**

	El sistema envía el comando pronunciado al servicio de reconocimiento de voz para que sea transformado de audio a texto. **3. Validar comando pronunciado** El sistema recibe el comando en formato texto y valida que sea una instrucción válida y reconocida de inserción o eliminación de un comentario. **4. Ejecutar el comando pronunciado** Una vez identificado el comando el sistema procede a insertar o eliminar el comentario dentro del modelo físico de la Interface MFML **5. Presentar el resultado al diseñador** El sistema muestra el comentario insertado o eliminado por el comando de voz pronunciado por el usuario. El caso de uso termina.
Flujos alternos	**1. El comando pronunciado es erróneo** En el paso 3 del flujo básico el comando pronunciado devuelto en formato texto no es válido. El sistema muestra un mensaje al usuario informando que la instrucción no es válida, retornando el caso de uso al paso 1 del flujo básico.
Pre-condiciones	**1. Archivo html cargado** El sistema ha extraído, analizado, estilizado y mostrado al usuario el contenido del archivo html **2. Archivo xml cargado** El sistema muestra la tabla html generada con colores distintivos y permite su edición **3. Hoja de estilo con nombre válido**

	El nombre ha sido verificado y se ha validado la no existencia de la extensión .css
Post-condiciones	**1. Comentario insertado o eliminado** El sistema ha insertado o eliminado el comentario, correspondiente al comando de voz pronunciado
Puntos de inclusión	No existen para este caso de uso
Puntos de extensión	No existen para este caso de uso

Fuente. Elaboración propia

Para una mejor compresión del caso de uso "Gestionar comentario CSS" se muestran a continuación los diagramas de actividad y secuencia. El propósito del diagrama de actividad es mostrar el comportamiento interno de un caso de uso representado a través de acciones.

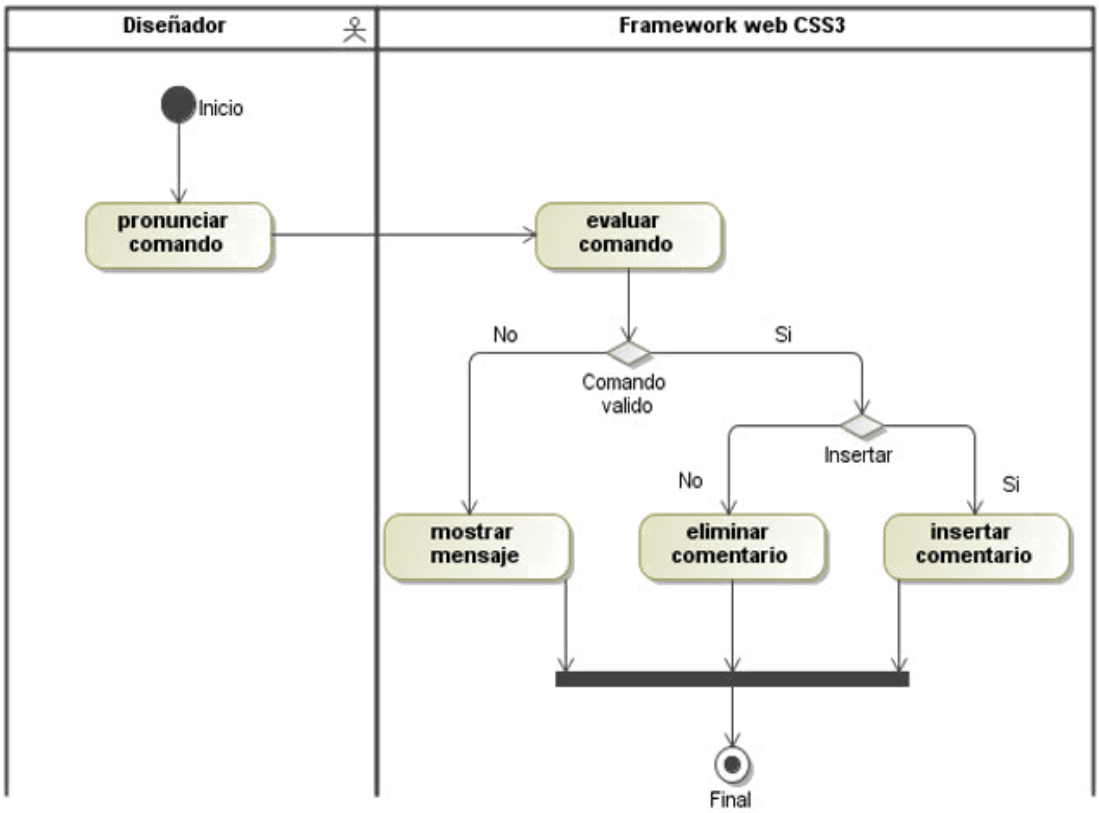


Figura 36 **UML - Diagrama de Actividad del Caso de uso Gestionar comentario css.**

Fuente: Elaboración propia

El propósito del diagrama de secuencia es mostrar la información de la interacción enfatizando la secuencia de tiempo. El eje Y representa la secuencia de tiempo y el eje X muestra los objetos participantes.

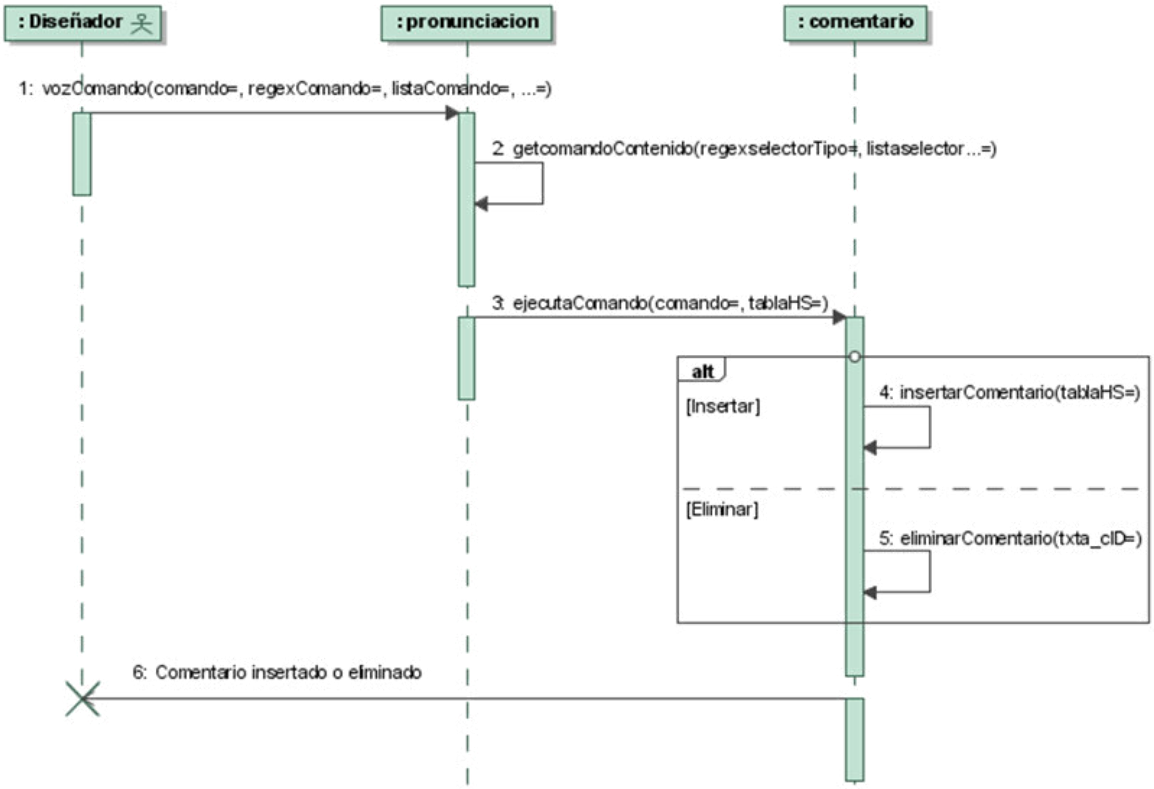


Figura 37 **UML - Diagrama de Secuencia del Caso de uso Gestionar comentario css.**

Fuente: Elaboración propia

4.3.5.5. Pila del sprint (sprint backlog)

Para el desarrollo de las historias de usuario pertenecientes al primer sprint hemos identificado las tareas siguientes.

Tabla 37 **Scrum - Pila del primer sprint**

HU	Descripción	Tareas
HU-01	Cargar archivo html	T1. Diseñar la UI del sistema
		T2. Desarrollar la carga y estilización del contenido del archivo html

HU-02	Importar archivo xml	T3. Desarrollar la carga y estilización del modelo físico de la Interface MFML (tabla html)
HU-03	Crear hoja de estilo CSS 3	T4. Desarrollar y validar el ingreso del nombre de la hoja de estilo CSS 3 a generar
HU-04	Pronunciación de comandos de voz	T5. Recopilar y organizar todos los selectores CSS 3
		T6. Recopilar y organizar todas las propiedades y sus respectivos valores, pertenecientes a las declaraciones CSS 3
		T7. Recopilar y organizar todas las reglasAT CSS3
		T8. Identificar y organizar todos los comandos de edición de texto que permitan realizar la acción deseada como respuesta a su pronunciación
		T9. Identificar y organizar todos los comandos de inserción y eliminación de selectores, declaraciones, reglas-AT, descriptores y comentarios que permitan realizar la acción deseada como respuesta a su pronunciación
		T10. Identificar y organizar todos los caracteres, vocales y letras del abecedario usados en el desarrollo de la hoja de estilo CSS 3 para la creación de selectores, declaraciones, reglas-AT, descriptores y comentarios
		T11. Desarrollar las acciones a realizar sobre el modelo físico de la Interface MFML (tabla html) como respuesta a la pronunciación, por parte del usuario del sistema, de cada uno de los comandos identificados en las tareas anteriores

Fuente. Elaboración propia

4.3.5.6. Ejecución del sprint

Las tareas del primer sprint han sido realizadas dentro del tiempo establecido.

4.3.5.7. Revisión del sprint

Este evento se realiza al final del sprint y tiene como objetivo probar completamente el incremento a entregar al cliente lo cual permite, al propietario del producto y al equipo en general, obtener una retroalimentación sobre el incremento. Posteriormente se procede a actualizar la pila del producto para continuar con la retrospectiva del sprint (Palacio, 2020).

4.3.5.8. Retrospectiva del sprint

Esta reunión es llevada a cabo luego de la revisión del sprint y abre paso a la planificación del siguiente sprint, en caso lo hubiera. El propósito de esta reunión es el análisis del trabajo desarrollado durante el sprint, planificando la mejora de las deficiencias y resaltando las fortalezas (Palacio, 2020). Se aprecian las historias de usuario: Cargar archivo html (3), Importar archivo xml (3), Crear archivo css (2) y Pronunciar comandos de voz (8), junto a sus esfuerzos estimados, así como el avance diario de puntos de historias completadas. En total son 16 puntos de historia que se realizaron en 10 días.

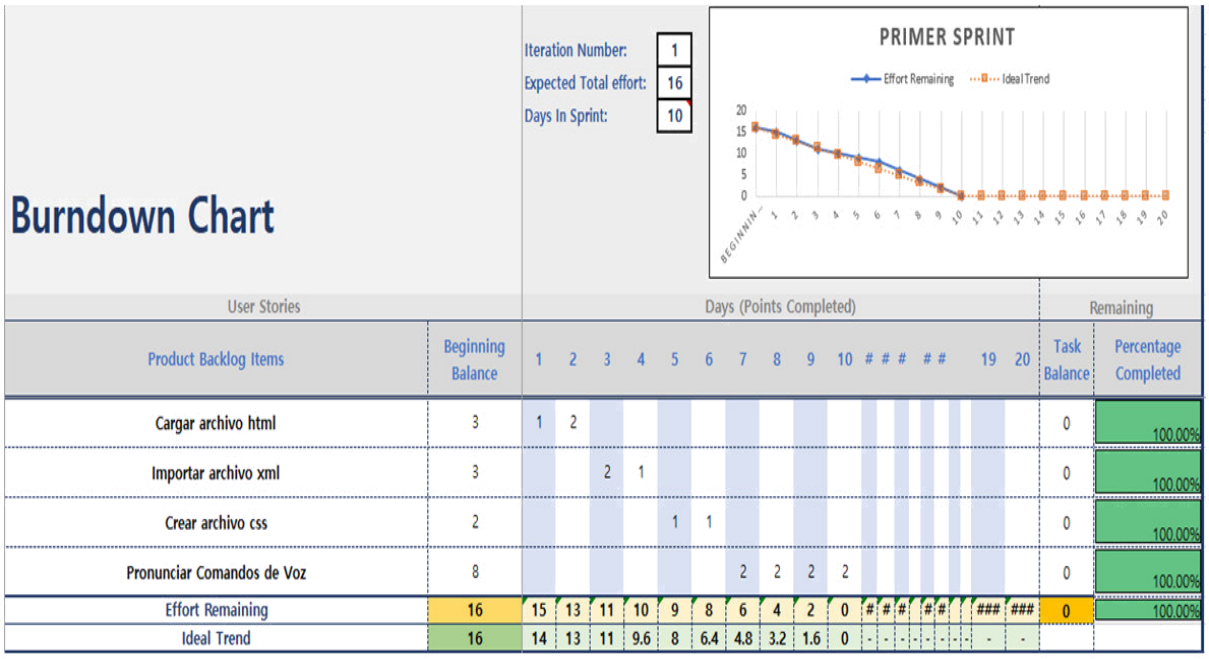


Figura 38 **Scrum - Seguimiento de Historias de usuario del primer sprint.**

Fuente: Elaboración propia

El gráfico de avance, también llamado gráfico burndown, es mostrado en la figura siguiente donde se aprecia que la línea de color naranja indica el esfuerzo ideal y la línea azul indica el esfuerzo real. Resulta evidente que el avance del trabajo diario se realizó acorde a lo planificado según los puntos de historia. De todos modos, diariamente se controló el avance evitando cualquier tipo de contratiempos.

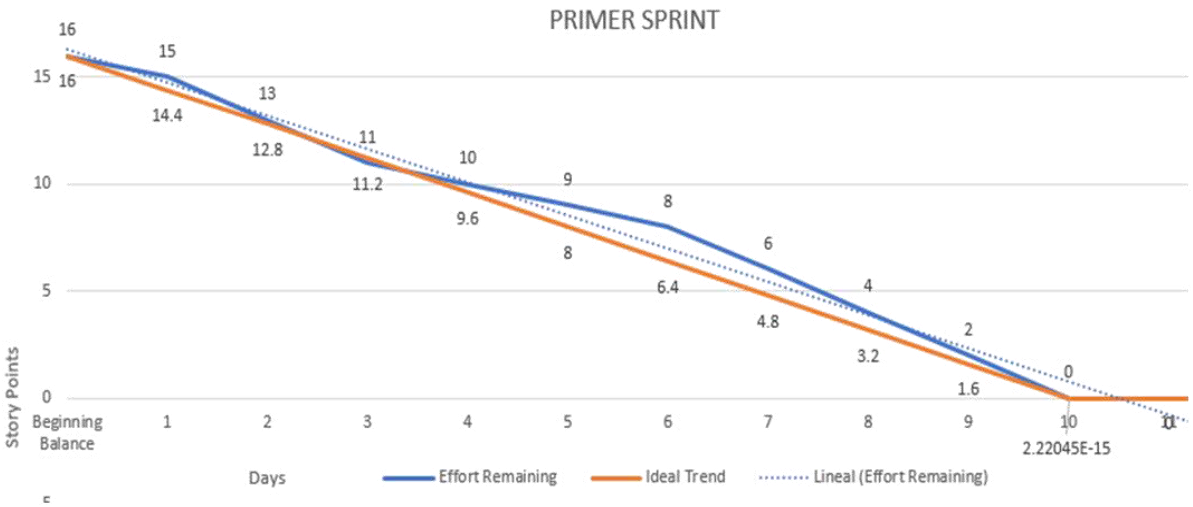


Figura 39 **Scrum - Gráfico burndown del primer sprint.**

Fuente: Elaboración propia

4.3.6. Segundo sprint

El sprint es considerado el evento principal en Scrum pues es aquí donde todos los roles desempeñados por el equipo de trabajo realizan las actividades de construcción de un entregable, por ello se dice que el sprint tiene un objetivo bien definido dentro del proyecto y una duración fija y constante (Palacio, 2020).

4.3.6.1. Planificación del sprint

Determina el punto de inicio del sprint definiendo su objetivo y la lista de tareas a realizar para su cumplimiento (Palacio, 2020). Las historias de usuario a desarrollar y los tiempos requeridos para llevarlos a cabo son descritos a continuación.

Tabla 38 **Scrum - Planificación del segundo sprint**

HU	Descripción	Puntos de historia	Tiempo (días)	Inicio	Final
HU-05	Ocultar framework	5	3	07/04/2021	09/04/2021
HU-06	Mostrar framework	2	1	10/04/2021	10/04/2021
HU-07	Exportar archivos .xml y .css	5	3	11/04/2021	13/04/2021
HU-08	Descargar código fuente	5	3	14/04/2021	16/04/2021
Puntos de historia / Tiempo estimado (Time Boxing)		17	10		

Fuente. Elaboración propia

En la tabla se muestra que el esfuerzo estimado para completar el segundo sprint es de 17 puntos de historia con un tiempo de desarrollo de 10 días. Se considera preciso modelar el diagrama de casos de uso con el fin de comprender la funcionalidad global del sprint.

4.3.6.2. Diagrama de Casos de uso

Un diagrama de casos de uso provee información adicional acerca de las relaciones entre los casos de uso y el usuario externo. El diagrama de casos de uso asociado al segundo sprint ha sido elaborado considerando las historias de usuario mencionadas anteriormente.

Tabla 39 **UML - Casos de uso del segundo sprint**

Casos de uso	Historia de usuario	Descripción
Ocultar framework	HU-05	Ocultar framework
Mostrar framework	HU-06	Mostrar framework
Exportar archivos xml – css	HU-07	Exportar archivos .xml y .css
Descargar código fuente	HU-08	Descargar código fuente

Fuente. Elaboración propia

En este caso los casos de uso han sido nombrados de igual forma que las historias de usuario pues representan perfectamente la interacción del diseñador con el framework para cumplir con este segundo sprint.

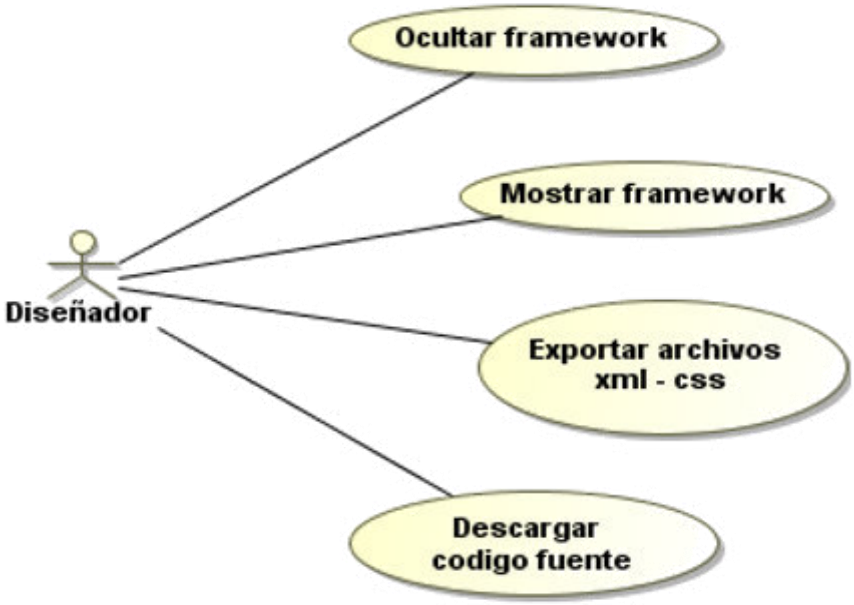


Figura 40 **UML - Diagrama de Casos de uso del segundo sprint.**

Fuente: Elaboración propia

4.3.6.3. Diagrama de clases

El diagrama de clases pertenece a la vista de estructura definida por UML donde se muestran los atributos y operaciones de las clases, así como su interrelación. En el diagrama de clases asociado al segundo sprint se aprecian las historias de usuario mencionadas.

Tabla 40 **UML - Clases del segundo sprint**

Clases	Historia de usuario	Descripción
Operación - Previsualización	HU-05	Ocultar framework
	HU-06	Mostrar framework
Operación - Exportación	HU-07	Exportar archivos .xml y .css
Descarga	HU-08	Descargar código fuente

Fuente. Elaboración propia

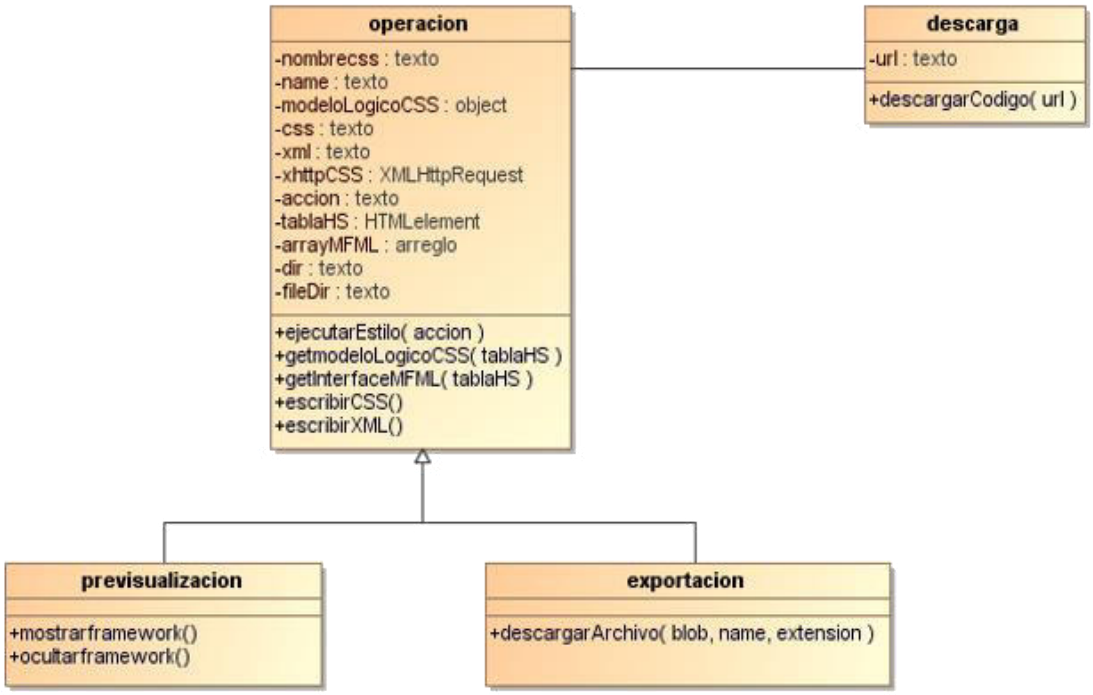


Figura 41 **UML - Diagrama de Clases del segundo sprint.**

Fuente: Elaboración propia

Claramente se aprecia que para cumplir las historias de usuario de este segundo sprint se han establecido dos clases principales llamadas Operación y Descarga. La clase Operación que consta de cinco (05) operaciones y de once (11) atributos generaliza a dos (02) clases: la clase previsualización que consta de dos (02) operaciones y la clase exportación que consta de una (01) operación. La clase Descarga consta de una (01) operación y de una (01) atributo principal.

4.3.6.4. Especificación de Casos de uso

La especificación de casos de uso es utilizada para detallar un caso de uso. Las tablas siguientes muestran la especificación de casos de uso para cada caso de uso correspondiente al segundo sprint.

A. Especificación Caso de uso – Ocultar framework

Tabla 41 **UML - Especificación Caso de uso Ocultar framework**

Caso de uso	Ocultar framework
Fuentes	Julio Prudencio
Actor	Diseñador
Descripción	El caso de uso permite al diseñador ocultar el framework y mostrar el archivo html cargado inicialmente sobre el cual se ha aplicado la hoja de estilo CSS 3 desarrollada hasta el momento; es decir, permite visualizar el avance del maquetado
Flujo básico	1. **Asignar un nombre a la hoja de estilo** El diseñador asigna un nombre válido a la hoja de estilo a generar. 2. **Recorrer la tabla html** El sistema recorre las filas, columnas y elementos textarea de la tabla html para obtener la clase, ID, valor y nivel de cada elemento contenido. 3. **Generar el archivo .css** El sistema procede a generar el archivo .css estableciendo las sangrías, los signos de puntuación: llave de inicio, llave de cierre, dos puntos, punto y coma para formar los bloques de código del archivo. 4. **Generar el archivo .xml** El sistema procede a generar el archivo .xml creando las etiquetas globales: “<modeloLogicoCSSXML>” y “<arrayMFML>”, así como la etiqueta: “<elementoMFML>” para cada elemento. 5. **Mostrar resultados al diseñador**

	El sistema muestra el archivo html al cual se le ha aplicado la hoja de estilo desarrollada.
Flujos alternos	1. **Hoja de estilo con nombre no válido** En el paso 1 del flujo básico el nombre de la hoja de estilo que define a la tabla html contiene la extensión .css. El sistema muestra un mensaje al usuario informando que el nombre no es válido, retornando el caso de uso al paso 1 del flujo básico. 2. **La hoja de estilo no tiene asignado un nombre** En el paso 1 del flujo básico la hoja de estilo que define a la tabla html no tiene asignado un nombre. El sistema muestra un mensaje al usuario indicando que asigne un nombre, retornando el caso de uso al paso 1 del flujo básico.
Pre-condiciones	1. **Archivo html cargado** El sistema ha extraído, analizado, estilizado y mostrado al usuario el contenido del archivo html
Post-condiciones	1. **Framework en modo previsualización** El framework se ha ocultado y se muestra el archivo html al cual se le ha aplicado la hoja de estilo desarrollada
Puntos de inclusión	No existen para este caso de uso
Puntos de extensión	No existen para este caso de uso

Fuente. Elaboración propia

Para una mejor compresión del caso de uso “Ocultar framework” se muestran a continuación los diagramas de actividad y secuencia. El propósito del diagrama de actividad es mostrar el comportamiento interno de un caso de uso representado a través de acciones. En este caso el diseñador invoca a la operación ocultar framework. Es en

este momento que verifica que la hoja de estilo a generar tenga asignado un nombre y, además, que dicho nombre no tenga asignado la extensión .css. En caso tenga asignada la extensión, el framework muestra un mensaje de error al diseñador. Si el diseñador ha asignado un nombre válido a la hoja de estilo, el sistema procede a recorrer la tabla html para obtener la clase, ID, valor y nivel de cada elemento de la tabla y genera la estructura y contenido del archivo .css, genera, además las etiquetas del archivo .xml, oculta el framework y muestra el archivo html al cual se ha aplicado la hoja de estilo.

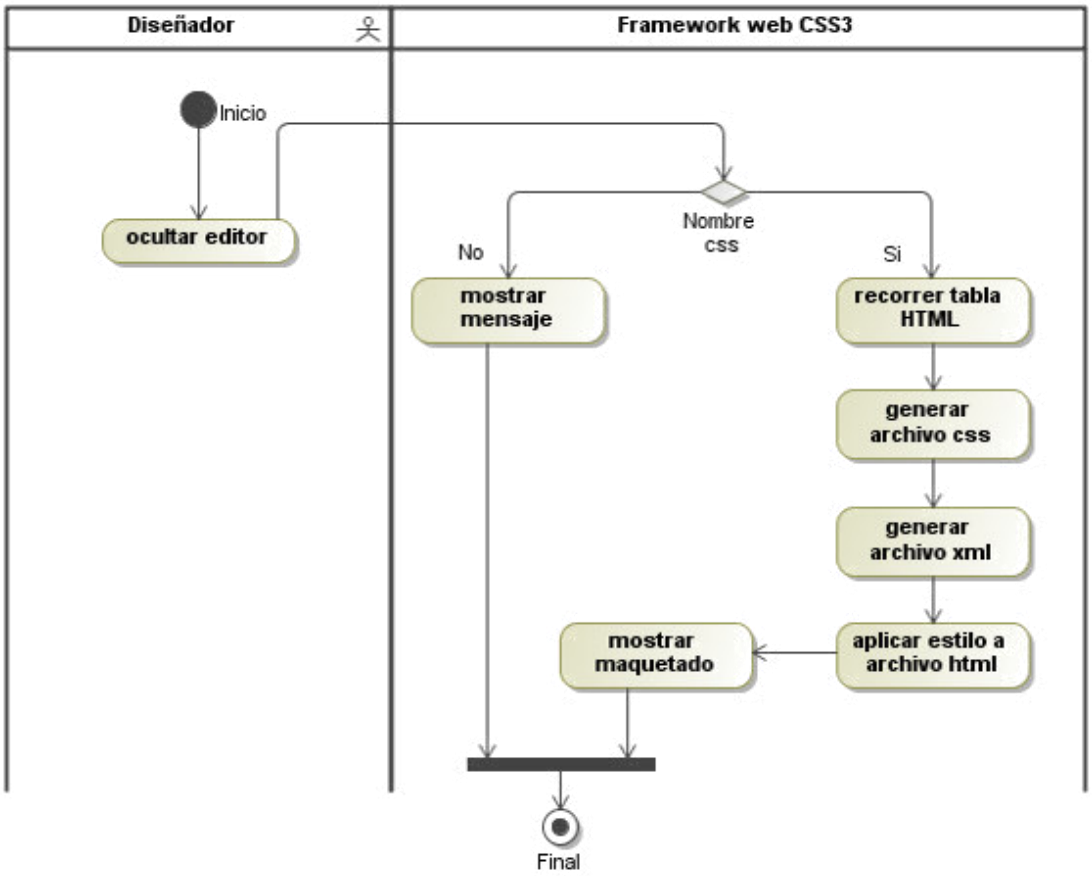


Tabla 42 **UML - Diagrama de Actividad del Caso de uso Ocultar framework.**

Fuente: Elaboración propia

El propósito del diagrama de secuencia es mostrar la información de la interacción enfatizando la secuencia de tiempo. El eje Y representa la secuencia de tiempo y el eje X muestra los objetos participantes. En este caso el diseñador envía un mensaje a la clase previsualización indicándole que se desea previsualizar el avance del maquetado hasta el momento. Esto es realizado invocando la operación "ejecutarEstilo". La clase verifica que la hoja de estilo a generar tenga asignado un nombre y que éste no

contenga la extensión .css. Una vez verificada la condición anterior procede a obtener la clase, ID, valor y nivel de la tabla html a través de la operación "getmodeloLogicoCSS" y "getInterfaceMFML". Posteriormente procede a generar el archivo .css propiamente invocando a la operación "escribirCSS". Genera además el archivo .xml invocando a la operación "escribirXML". Finalmente oculta el framework y muestra el archivo html sobre el cual se ha aplicado la hoja de estilo generada.

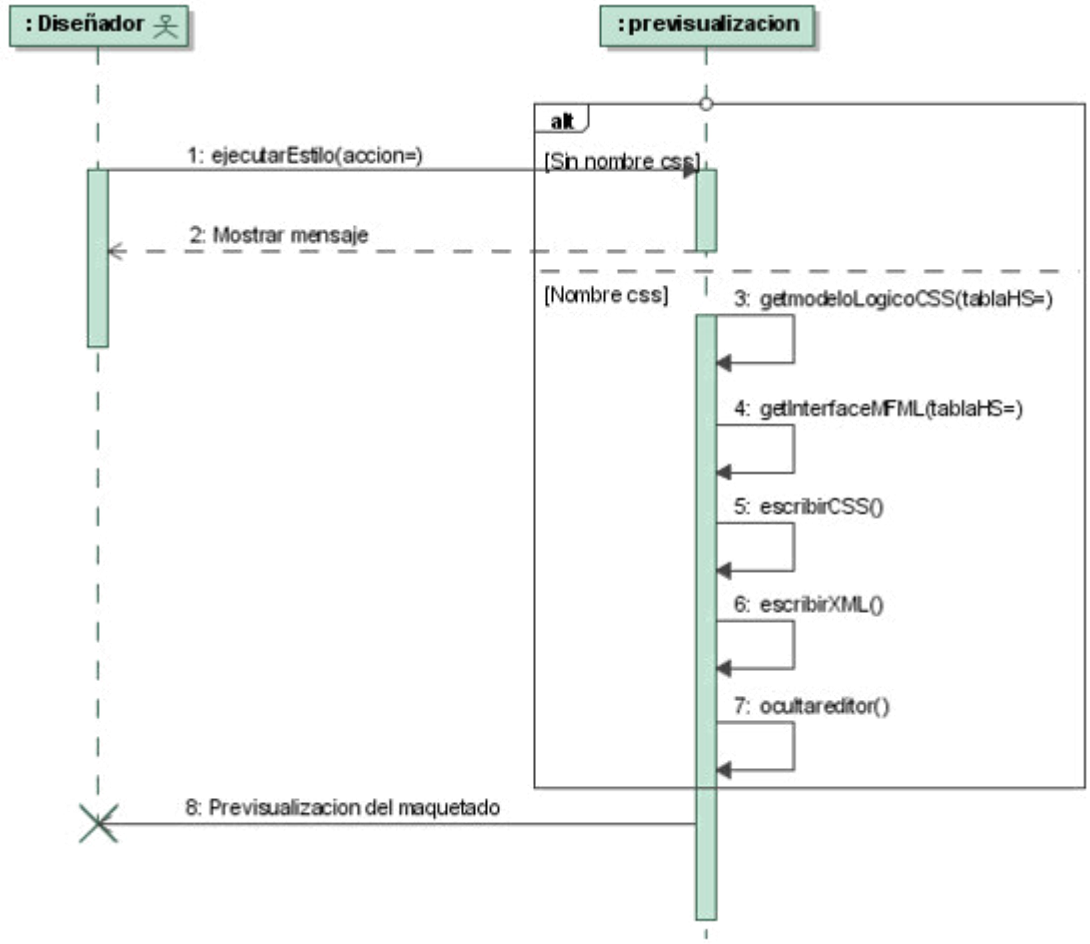


Tabla 43 **UML - Diagrama de Secuencia del Caso de uso Ocultar framework.**

Fuente: Elaboración propia

B. Especificación Caso de uso – Mostrar framework

Tabla 44 **UML - Especificación Caso de uso Mostrar framework**

Caso de uso	Mostrar framework
Fuentes	Julio Prudencio

Actor	Diseñador
Descripción	El caso de uso permite al diseñador mostrar el framework para continuar desarrollando la hoja de estilo CSS 3; es decir, sale del modo previsualización y cambia al modo framework
Flujo básico	1. **Salir del modo previsualización** El sistema oculta el archivo html 2. **Mostrar resultados al diseñador** El sistema muestra el framework para continuar el desarrollo de la hoja de estilo.
Flujos alternos	No existen para este caso de uso
Pre-condiciones	1. **Framework en modo previsualización** El framework se ha ocultado y se muestra el archivo html al cual se le ha aplicado la hoja de estilo desarrollada
Post-condiciones	1. **Framework en modo edición** El archivo html se ha ocultado y se muestra el framework para continuar el desarrollo de la hoja de estilo
Puntos de inclusión	No existen para este caso de uso
Puntos de extensión	No existen para este caso de uso

Fuente. Elaboración propia

Para una mejor compresión del caso de uso “Mostrar framework” se muestran a continuación el diagrama de secuencia. El propósito del diagrama de secuencia es mostrar la información de la interacción enfatizando la secuencia de tiempo. El eje Y representa la secuencia de tiempo y el eje X muestra los objetos participantes. En este caso se puede apreciar que el diseñador envía el mensaje de mostrar el framework a la clase previsualización, a través de la operación “mostrar framework”, en respuesta a esta solicitud la clase oculta el archivo html sobre el cual se ha aplicado la hoja de estilo desarrollada hasta el momento y vuelve a mostrar el framework; es decir, cambia del modo previsualización al modo framework para continuar con el desarrollo de la hoja de estilo.

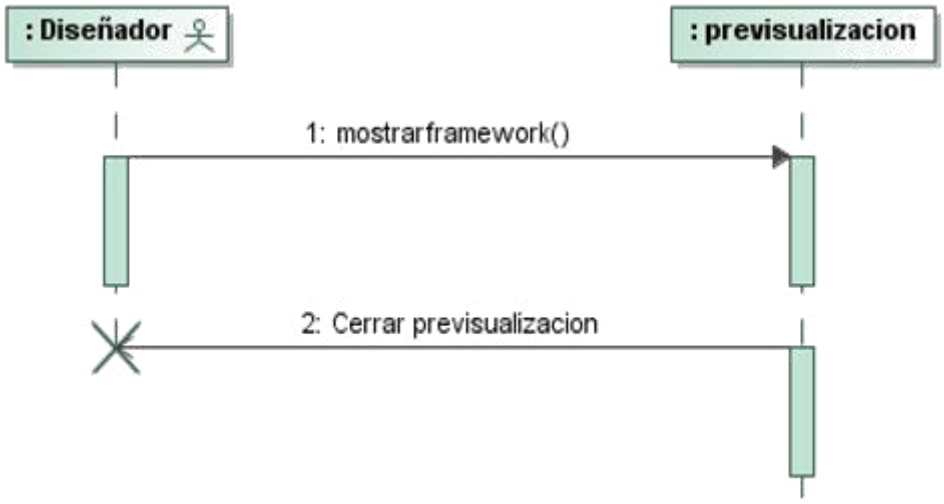


Tabla 45 **UML - Diagrama de Secuencia del Caso de uso Mostrar framework.**

Fuente: Elaboración propia

C. Especificación Caso de uso - Exportar archivos .xml y .css

Tabla 46 **UML - Especificación Caso de uso Exportar archivos .xml y .css**

Caso de uso	Exportar archivos .xml y .css
Fuentes	Julio Prudencio
Actor	Diseñador
Descripción	El caso de uso permite al diseñador exportar el modelo lógico de la Interface MFML (filas, columnas y elementos textarea de la tabla html) en los archivos con extensión .xml que permite importarlo posteriormente y, en formato .css, que es la hoja de estilo desarrollada hasta el momento de la exportación, el cual está lista para ser incluido en la carpeta css del proyecto de maquetado
Flujo básico	1. **Asignar un nombre a la hoja de estilo** El diseñador asigna un nombre válido a la hoja de estilo a generar. 2. **Recorrer la tabla html** El sistema recorre las filas, columnas y elementos textarea de la tabla html para

	obtener la clase, ID, valor y nivel de cada elemento contenido. **3. Generar el archivo .css** El sistema procede a generar el archivo .css estableciendo las sangrías, los signos de puntuación: llave de inicio, llave de cierre, dos puntos, punto y coma para formar los bloques de código del archivo. **4. Generar el archivo .xml** El sistema procede a generar el archivo .xml creando las etiquetas globales: “<modeloLogicoCSSXML>” y “<arrayMFML>”, así como la etiqueta: “<elementoMFML>” para cada elemento. **5. Exportar archivos generados** El sistema exporta los archivos .xml el cual puede ser importado posteriormente para continuar el desarrollo y, .css, el cual está apto y listo para ser incluido en el proyecto de maquetado
Flujos alternos	**1. Hoja de estilo con nombre no válido** En el paso 1 del flujo básico el nombre de la hoja de estilo que define a la tabla html contiene la extensión .css. El sistema muestra un mensaje al diseñador informando que el nombre no es válido, retornando el caso de uso al paso 1 del flujo básico. **2. La hoja de estilo no tiene asignado un nombre** En el paso 1 del flujo básico la hoja de estilo que define a la tabla html no tiene asignado un nombre. El sistema muestra un mensaje al diseñador indicando que asigne un nombre,

	retornando el caso de uso al paso 1 del flujo básico.
Pre-condiciones	1. **Archivo html cargado** El sistema ha extraído, analizado, estilizado y mostrado al usuario el contenido del archivo html
Post-condiciones	2. **Archivos .xml y .css exportados** El framework ha descargado los archivos .xml y .css en la carpeta de descargas del dispositivo del diseñador a través del cual se conecta al sistema
Puntos de inclusión	No existen para este caso de uso
Puntos de extensión	No existen para este caso de uso

Fuente. Elaboración propia

Para una mejor compresión del caso de uso “Exportar archivos .xml y .css” se muestran a continuación el diagrama de secuencia. El propósito del diagrama de secuencia es mostrar la información de la interacción enfatizando la secuencia de tiempo. El eje Y representa la secuencia de tiempo y el eje X muestra los objetos participantes. En este caso los mensajes y operaciones realizadas son similares que las del caso de uso Ocultar framework; es decir, el diseñador envía un mensaje a la clase previsualización indicándole que se desea descargar los archivos .xml y .css. Esto es realizado invocando la operación "ejecutarEstilo". La clase verifica que la hoja de estilo a generar tenga asignado un nombre y que éste no contenga la extensión .css. Una vez verificada la condición anterior procede a obtener la clase, ID, valor y nivel de la tabla html a través de la operación "getmodeloLogicoCSS" y "getInterfaceMFML". Posteriormente procede a generar el archivo .css propiamente invocando a la operación "escribirCSS". Genera además el archivo .xml invocando a la operación "escribirXML".

No obstante, la diferencia respecto de este caso de uso es que el framework no se oculta ni muestra tampoco el archivo html sobre el cual se ha aplicado la hoja de estilo generada, simplemente procede a invocar a la operación “descargarArchivo”, la cual

se encarga de descargar los archivos en la carpeta descargas de la máquina del diseñador desde donde se conecta al framework.

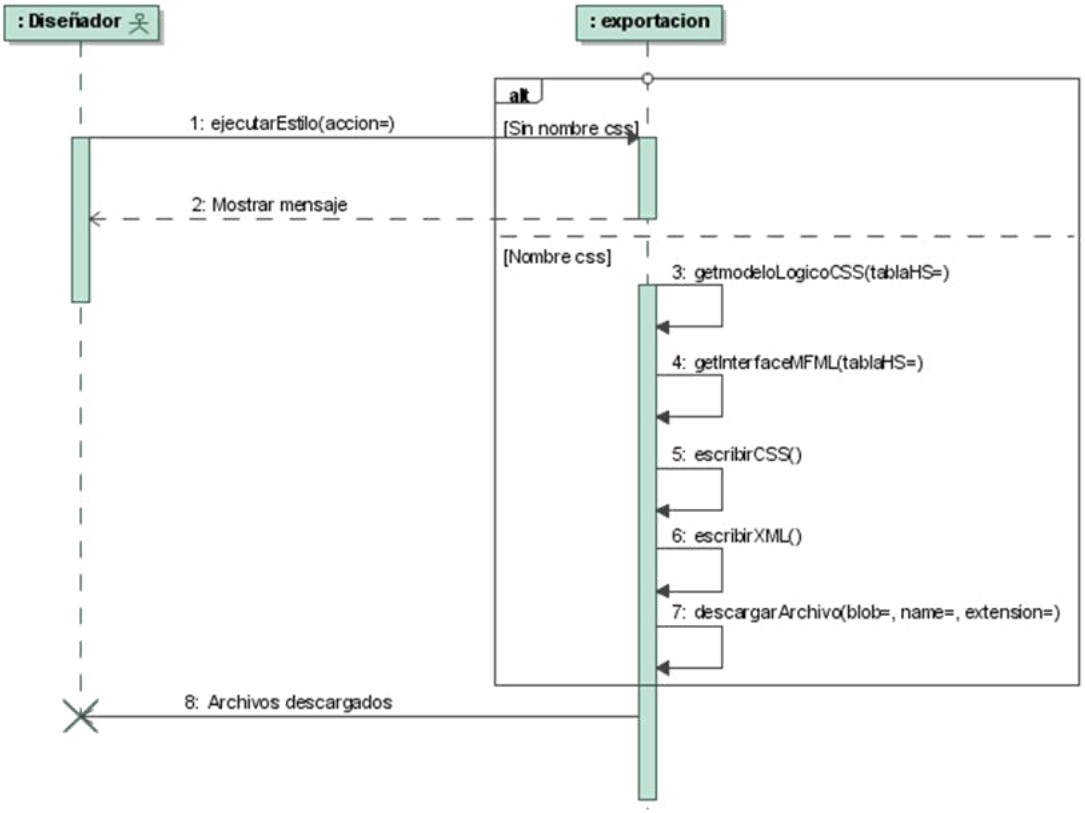


Figura 42 **UML - Diagrama de Secuencia del Caso de uso Exportar archivos .xml y .css.**

Fuente: Elaboración propia

D. Especificación Caso de uso – Descargar código fuente

Tabla 47 **UML - Especificación Caso de uso Descargar código fuente**

Caso de uso	Descargar código fuente
Fuentes	Julio Prudencio
Actor	Diseñador
Descripción	El caso de uso permite al diseñador descargar el proyecto completo del framework CSS 3 con reconocimiento de voz, con fines académicos, para que pueda analizarlo, modificarlo o mejorarlo según estime conveniente

Flujo básico	1. **Descargar proyecto** El diseñador indica al sistema que desea realizar la descarga del proyecto
Flujos alternos	No existen para este caso de uso
Pre-condiciones	No existen para este caso de uso
Post-condiciones	1. **Proyecto descargado** El framework descarga el proyecto en la carpeta de descargas del dispositivo del diseñador a través del cual se conecta al sistema
Puntos de inclusión	No existen para este caso de uso
Puntos de extensión	No existen para este caso de uso

Fuente. Elaboración propia

Para una mejor compresión del caso de uso "Descargar código fuente" se muestran a continuación los diagramas de actividad y secuencia. El propósito del diagrama de actividad es mostrar el comportamiento interno de un caso de uso representado a través de acciones. En este caso el diseñador inicia el proceso de descarga del proyecto completo del framework CSS 3 con reconocimiento de voz. Para ello, hace clic en el link de descarga lo que activa la descarga propiamente, por parte del sistema, en la carpeta de descarga del dispositivo con el cual el diseñador se conecta al sistema a través de internet.

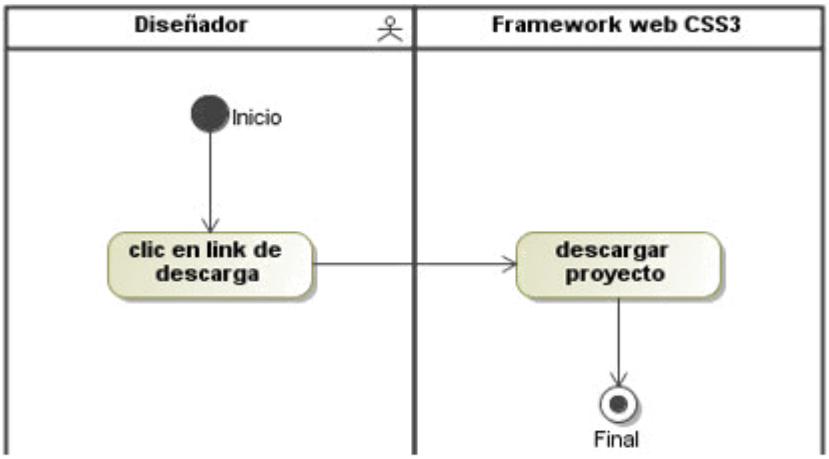


Figura 43 **UML - Diagrama de Actividad del Caso de uso Descargar codigo fuente.**

Fuente: Elaboración propia

El propósito del diagrama de secuencia es mostrar la información de la interacción enfatizando la secuencia de tiempo. El eje Y representa la secuencia de tiempo y el eje X muestra los objetos participantes.

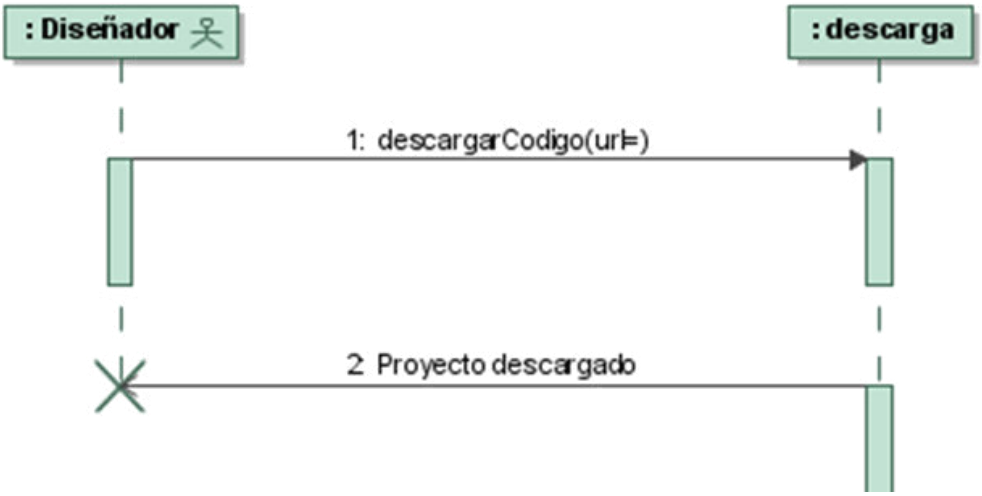


Figura 44 **UML - Diagrama de Secuencia del Caso de uso Descargar código fuente.**

Fuente: Elaboración propia

4.3.6.5. Pila del sprint (sprint backlog)

Para el desarrollo de las historias de usuario pertenecientes al segundo sprint hemos identificado las tareas siguientes.

Tabla 48 **Scrum - Pila del segundo sprint**

HU	Descripción	Tareas
HU-05	Ocultar framework	T1. Generar los archivos .xml y .css
		T2. Aplicar la hoja de estilo CSS 3 desarrollada hasta el momento sobre el archivo html
HU-06	Mostrar framework	T3. Cambiar del modo previsualización al modo framework
HU-07	Exportar archivos .xml y .css	T4. Generar los archivos .xml y .css
		T5. Generar la descarga del modelo lógico de la Interface MFML (archivos .xml y .css)

HU-08	Descargar código fuente	T6. Generar la descarga del proyecto framework CSS 3 con reconocimiento de voz

Fuente. Elaboración propia

4.3.6.6. Ejecución del sprint

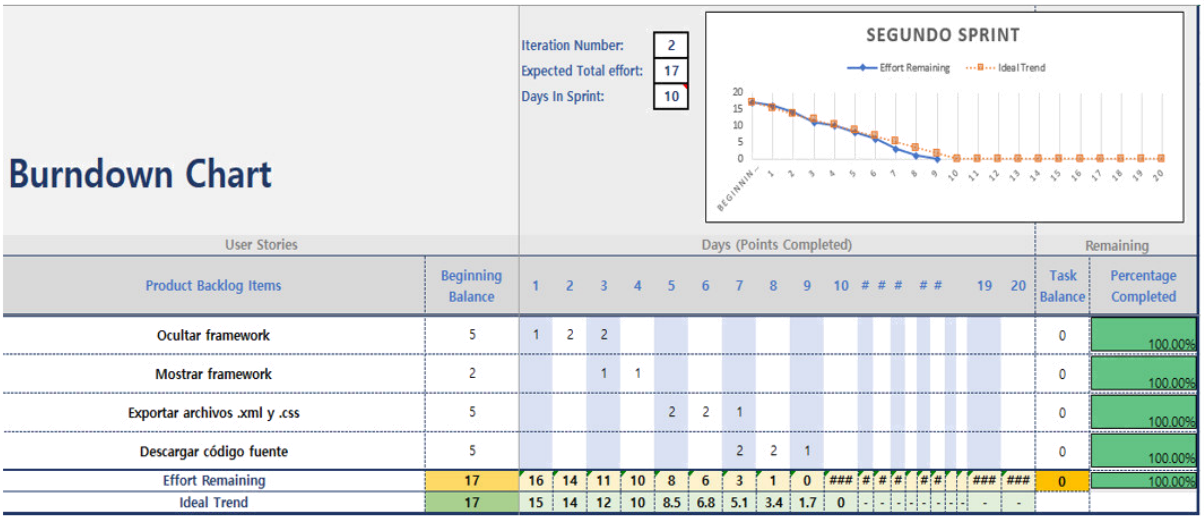
Las tareas del segundo sprint han sido realizadas dentro del tiempo establecido.

4.3.6.7. Revisión del sprint

Este evento se realiza al final del sprint y tiene como objetivo probar completamente el incremento a entregar al cliente lo cual permite, al propietario del producto y al equipo en general, obtener una retroalimentación sobre el incremento. Posteriormente se procede a actualizar la pila del producto para continuar con la planeación del siguiente sprint (Palacio, 2020).

4.3.6.8. Retrospectiva del sprint

Esta reunión es llevada a cabo luego de la revisión del sprint y abre paso a la planificación del siguiente sprint, en caso lo hubiera. El propósito de esta reunión es el análisis del trabajo desarrollado durante el sprint, planificando la mejora de las deficiencias y resaltando las fortalezas (Palacio, 2020). Se aprecian las historias de usuario: Ocultar framework (5), Mostrar framework (2), Exportar archivos .xml y .css (5) y Descargar código fuente (5), junto a sus esfuerzos estimados, así como el avance diario de puntos de historias completadas. En total son 17 puntos de historia que se realizaron en 10 días.



Burndown Chart

Iteration Number: 2
Expected Total effort: 17
Days In Sprint: 10

User Stories		Days (Points Completed)																	Remaining	
Product Backlog Items	Beginning Balance	1	2	3	4	5	6	7	8	9	10	#	#	#	#	#	19	20	Task Balance	Percentage Completed
Ocultar framework	5	1	2	2															0	100.00%
Mostrar framework	2			1	1														0	100.00%
Exportar archivos .xml y .css	5					2	2	1											0	100.00%
Descargar código fuente	5							2	2	1									0	100.00%
Effort Remaining	17	16	14	11	10	8	6	3	1	0	###	#	#	#	#	#	###	###	0	100.00%
Ideal Trend	17	15	14	12	10	8.5	6.8	5.1	3.4	1.7	0	-	-	-	-	-	-	-		

Figura 45 **Scrum - Seguimiento de Historias de usuario del segundo sprint.**

Fuente: Elaboración propia

El gráfico de avance, también llamado gráfico burndown, es mostrado en la figura siguiente donde se aprecia claramente que la línea de color naranja indica el esfuerzo ideal y la línea azul indica el esfuerzo real. Resulta evidente que el avance del trabajo diario se realizó acorde a lo planificado según los puntos de historia. De todos modos, diariamente se controló el avance evitando cualquier tipo de contratiempos.

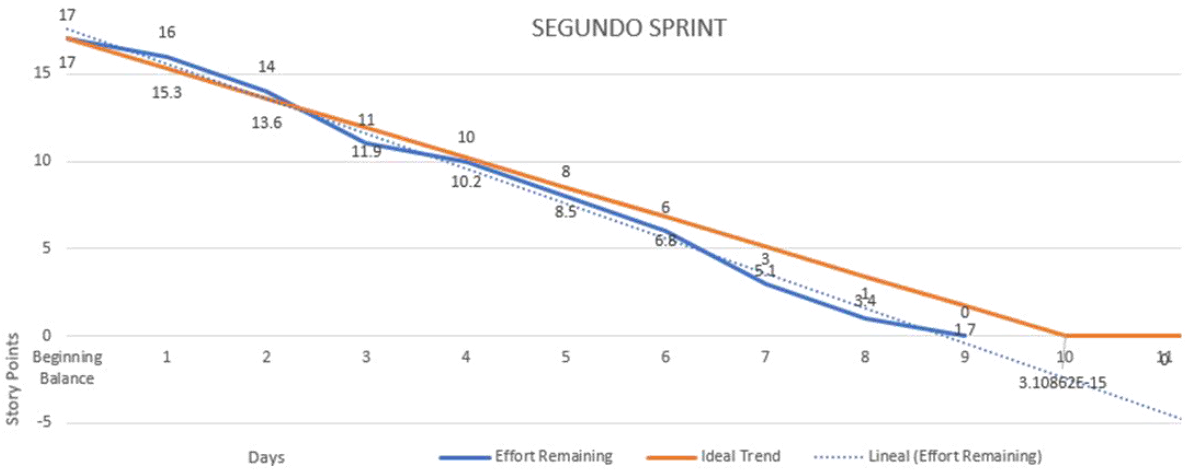


Figura 46 **Scrum - Gráfico burndown del segundo sprint.**

Fuente: Elaboración propia

4.3.7. Arquitectura general del framework CSS 3

La arquitectura general del sistema está compuesta por el diagrama de paquetes, el diagrama de casos de uso, el diagrama de clases, el diagrama de componentes y el diagrama de despliegue.

4.3.7.1. Diagrama de paquetes

El diagrama de paquetes pertenece a la vista de estructura definida por UML donde se muestran los componentes del sistema y su interrelación. En este caso se muestra claramente que el framework ha sido desarrollado siguiendo el patrón de arquitectura de software MVC: modelo – vista – controlador. De esta forma, el controlador es quien responde a las acciones del usuario y se comunica con el modelo, el controlador está conformado por los archivos js. El modelo representa la información con la cual el sistema opera y está conformado por los archivos json que contienen los comandos de voz, los archivos .css y .xml generados, y la clase php que es la que se encarga de exportar y descargar los archivos .css y .xml. La vista representa la interfaz gráfica del usuario y está conformada por los archivos html y css.

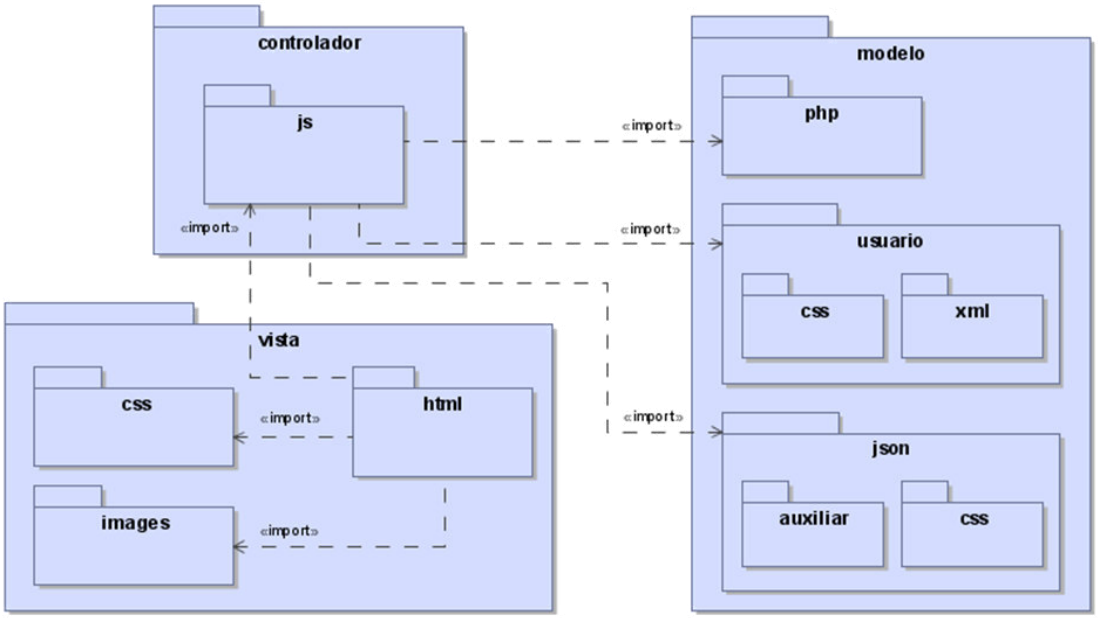


Figura 47 **UML - Diagrama de paquetes.**

Fuente: Elaboración propia

4.3.7.2. Diagrama de casos de uso

La funcionalidad global del framework y su interacción con el diseñador es mostrada en la figura siguiente. Claramente se constata que el framework cuenta con 14 casos de uso, de los cuales 06 de ellos son casos de uso incluidos. Sus nombres son claramente representativos para todas las funciones realizadas por el framework.

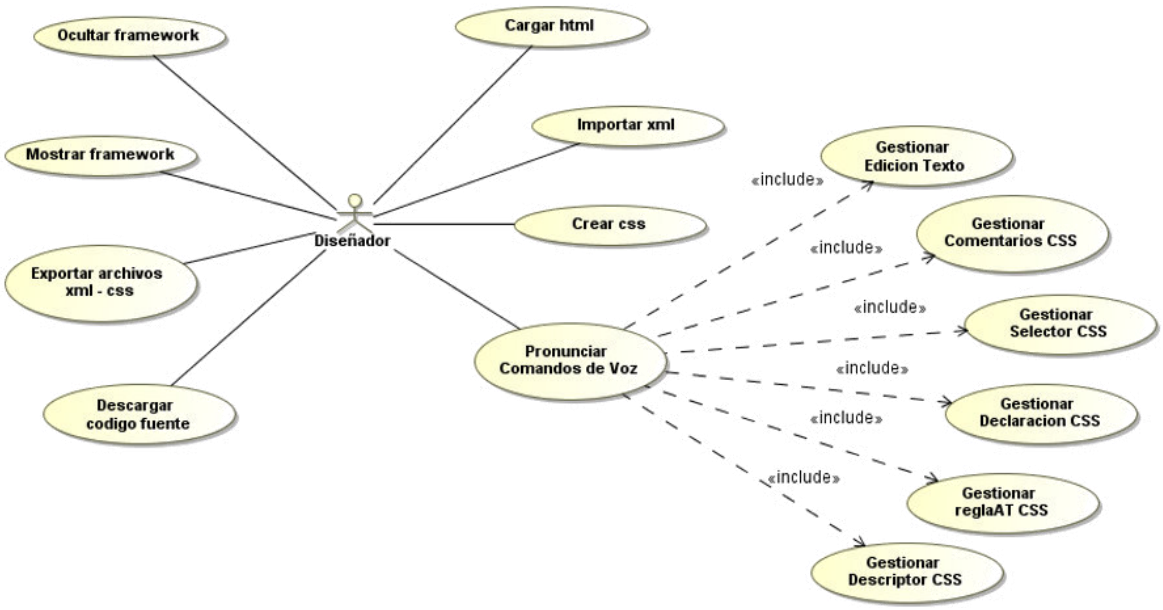


Figura 48 **UML - Diagrama de casos de uso.**

Fuente: Elaboración propia

4.3.7.3. Diagrama de clases

El diagrama de clases muestra la estructura lógica global del framework la cual consta de 12 clases, de las cuales 06: edicionTexto, selector, declaracion, reglaAT, descriptor y comentario son clases que heredan las operaciones y atributos de la clase pronunciación y 02: previsualización y exportación son las clases que heredan, a su vez, las operaciones y atributos de la clase operación. Las clases inicialización y descarga en cambio no heredan ni son heredadas de ninguna otra clase.

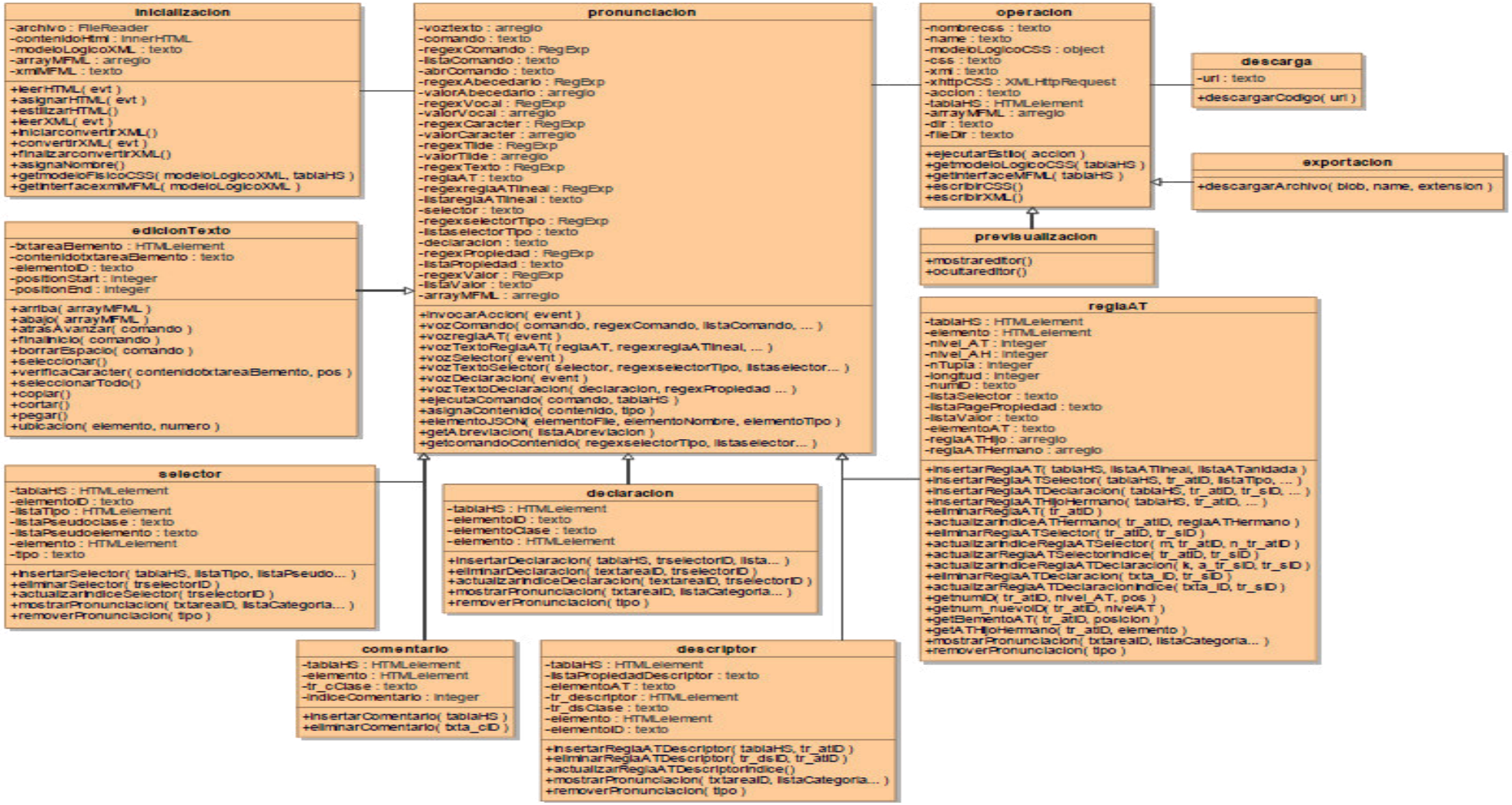


Figura 49 **UML - Diagrama de clases.** Fuente: Elaboración propia

4.3.7.4. Diagrama de componentes

El diagrama de componentes describe componentes software y sus dependencias donde un componente es una parte del software encapsulada, reutilizable, reemplazable, tiene mayor responsabilidad que una clase e interactúa con otros a través de interfaces requeridas y proveídas. El diagrama de componentes del framework consta de 05 componentes: inicialización, pronunciación, operación, descarga e index.

El componente inicialización consta de 10 operaciones proveídas a través de la interface "cargaArchivo" la cual es requerida por el componente index para cargar el archivo html a maquetar y/o cargar el archivo xml que contiene la metadata de la hoja de estilo generada para reconstruir la tabla html del modelo físico; el componente pronunciación está compuesto de 15 operaciones proveídas a través de la interface "comandoVoz" la cual es requerida por el componente index para gestionar las operaciones de inserción, eliminación, actualización de: reglas-AT, descriptores, selectores, declaraciones y comentarios así como para realizar las funciones de edición de texto; el componente descarga consta de 01 operación proveída a través de la interface "descargaArchivo" la cual provee la funcionalidad de descargar el código fuente del framework propiamente; el componente operación consta de 08 operaciones proveídas a través de las interfaces "previsualización" y "exportaMFML" requeridas por el componente index para gestionar: la previsualización del avance del maquetado y la generación y descarga de los archivos .css y .xml del modelo lógico. Finalmente, el componente index que consta de 01 operación proveída por la interface "verificaDispositivo" para gestionar la carga del framework según el dispositivo desde el cual el usuario accede mostrando así la versión para smartphone o computador, además, este componente requiere de todas las interfaces proveídas por los componentes antes mencionados.

El diagrama de componentes es mostrado en la figura siguiente.

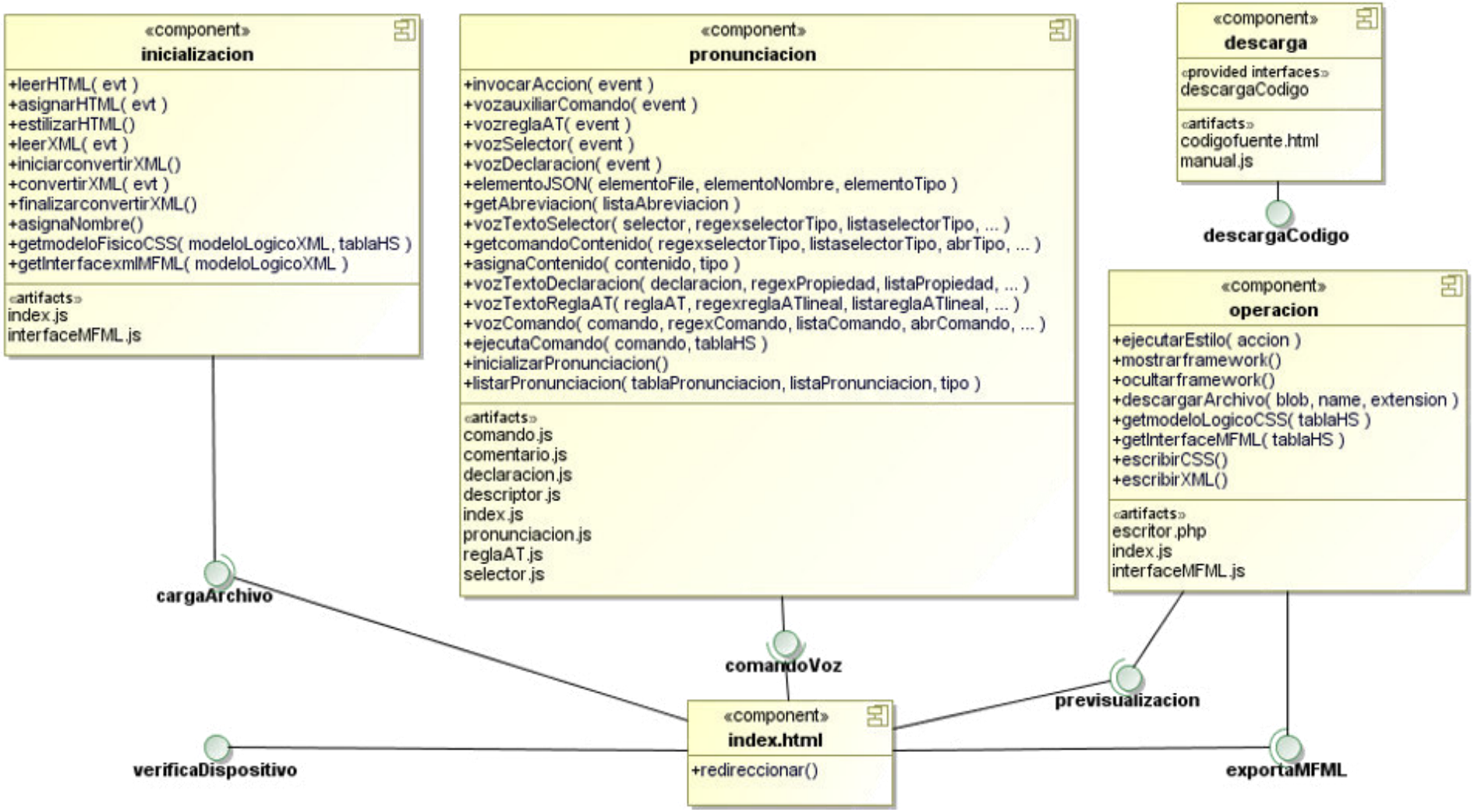


Figura 50 **UML - Diagrama de componentes.** Fuente: Elaboración propia

4.3.7.5. Diagrama de despliegue

El diagrama de despliegue representa la arquitectura en tiempo de ejecución de procesadores, dispositivos (nodos) y de los artefactos software que se ejecutan en esa arquitectura. El diagrama de despliegue del framework consta de 01 dispositivo llamado cliente y de 02 procesadores llamados servidor web y servidor de aplicaciones donde están instalados: Apache Web Server v.2.4.48 y PHP Application Server v.7.2.34, respectivamente. Además, se muestra que en el servidor web se encuentran desplegados: 25 artefactos de los cuales 10 son archivos js, 09 son archivos html, 05 son archivos json y 01 archivo comprimido. Los artefactos que representan a los 10 archivos js son: index.js, pronunciacion.js, comando.js, selector.js, declaracion.js, declaracion.js, reglaAT.js, descriptor.js, interfaceMFML.js, manual.js, comentario.js; los artefactos que representan a los 09 archivos html son: cargar.html, codigofuente.html, adiseniar.html, ejemplodisenio.html, mostrarframework.html, webspeechapi.html, pronunciacion.html, importarxml.html, manual.html; los artefactos que representan a los 05 archivos json son: complemento.json, comando.json, selector.json, declaracion.json, reglaAT.json; el artefacto que representa al archivo comprimido es: Frameworkcss3.rar.

En el servidor de aplicaciones se encuentra desplegado 01 archivo php. El artefacto que representa al archivo php es: escritor.php. Estos procesadores y dispositivos se conectan y comunican entre sí a través del protocolo TCP/IP mediante mecanismos de seguridad para evitar el acceso indebido a archivos protegidos del sistema.

El diagrama de despliegue es mostrado en la figura siguiente.

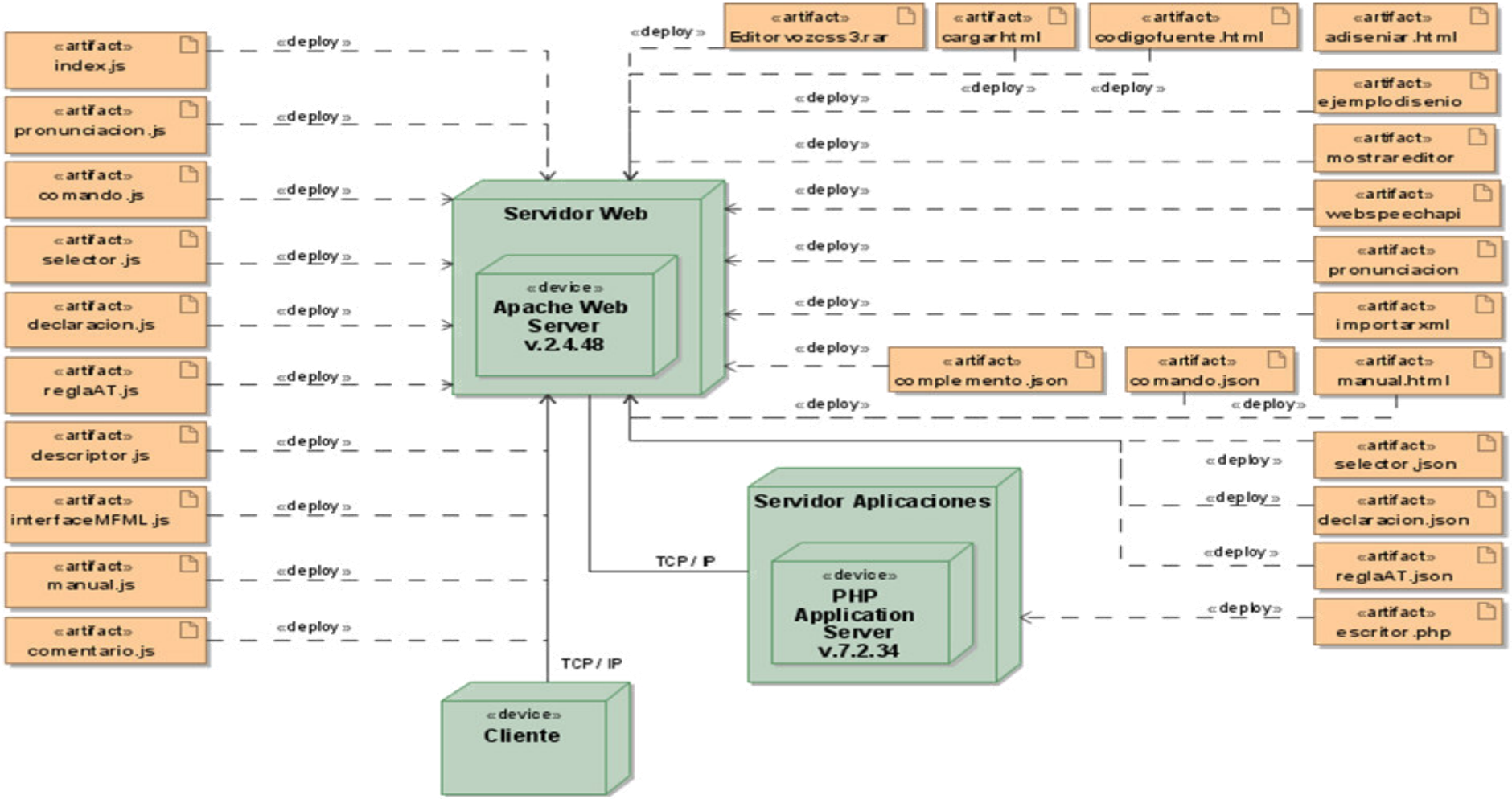


Figura 51 **UML - Diagrama de despliegue.** Fuente: Elaboración propia

4.4. FRAMEWORK WEB CSS 3 CON RECONOCIMIENTO DE VOZ

A continuación, se explicará el framework CSS 3 con reconocimiento de voz propuesto con el fin de integrar la tecnología de reconocimiento de voz mediante Web Speech API y la tecnología de maquetado web CSS 3, a través de la Interface MFML: Modelo físico – Modelo lógico.

De aquí en adelante, explicaremos el uso del framework CSS 3 con reconocimiento de voz. El estudio del framework de voz se dividirá en seis (06) partes pasos: Paso 1 donde explicaremos la carga del archivo .html al framework sobre el cual deseamos ejecutar el código CSS 3 a generar; Paso 2 donde detallamos el procedimiento a seguir para importar el archivo .xml o crear la hoja de estilo .css desde cero; Ejecutar código donde se explica cómo generar una prevista del maquetado que se realiza para modificar o continuar el maquetado web; Generar MFML donde se explica el procedimiento a seguir para exportar los archivos .xml y .css; Ejemplo aplicativo donde se documenta el procedimiento completo de maquetado de un archivo .html desde el desarrollo de código CSS mediante comandos de voz hasta la previsualización del avance del maquetado y la generación y exportación de los archivos .xml y .css; Código fuente que es la última parte del estudio donde se pone a disposición la descarga del código fuente del framework de voz CSS 3.

4.4.1. Paso 1 – Cargar .html

El primer paso a seguir para hacer uso del framework es cargar el archivo con extensión .html sobre el cual deseamos ejecutar el código CSS 3 a generar, es decir, cargar uno de los archivos de la página web a estilizar. Evidentemente, el archivo .html tiene que seguir la estructura descrita por la tecnología HTML5, una estructura definida por elementos: header, nav, aside, section, article y footer, entre otras etiquetas html que cumplirán el propósito del sitio web a construir.

Entonces, el archivo .html a maquetar tiene que poseer las características y elementos de cualquier archivo web guiado por la tecnología HTML5. Entre estas características se encuentra la adición de la etiqueta “link”, dentro de la sección “head”, importando

la hoja de estilos CSS 3 a generar desde el framework, aún sin haber sido creado con anterioridad.

```
<link rel="stylesheet" href="./project/usuario/css/nombreArchivo.css">
```

Figura 52 **Framework web CSS 3 - Importar archivo .css a generar.**

Fuente: Elaboración propia

Claramente el valor (ruta) de la propiedad href es: "./project/usuario/css/", la cual es estática y siempre debe ser la misma para cualquier archivo .html. Sin embargo, la última parte de la ruta está compuesta por el "nombre" que se dará a la hoja de estilos CSS 3 a generar, evidentemente con extensión .css. Así, por ejemplo, si se deseara maquetar la ventana de autenticación, el nombre más sugerido sería: "login", el framework asignará la extensión .css, resultando: "login.css" como nombre de la hoja de estilo.

Por lo tanto, la sección "head" del archivo .html a maquetar contendrá, entre otros elementos, la etiqueta "link" con el "valor href" explicado.

```
<head>
        <title>Login</title>
        <meta charset="UTF-8">
         ...
        <link rel="stylesheet" href="./project/usuario/css/login.css">
<head>
```

Figura 53 **Framework web CSS 3 - Sección head archivo .html.**

Fuente: Elaboración propia

El nombre "Login" y "login.css" dependerá del criterio del diseñador web. Para la carga del archivo .html hacemos clic en el botón "Seleccionar archivo" para abrir la ventana de búsqueda y selección de archivos. Esta ventana sólo mostrará archivos con extensión .html facilitando su búsqueda y selección.

Figura 54 **Framework web CSS 3 - Botón seleccionar archivo .html.**

Fuente: Elaboración propia

Una vez se muestre la ventana de búsqueda y selección de archivos, procedemos a seleccionar el archivo .html deseado, el cual cumple con las normas de la tecnología HTML5 y contiene la etiqueta "link" dentro de la sección "head".

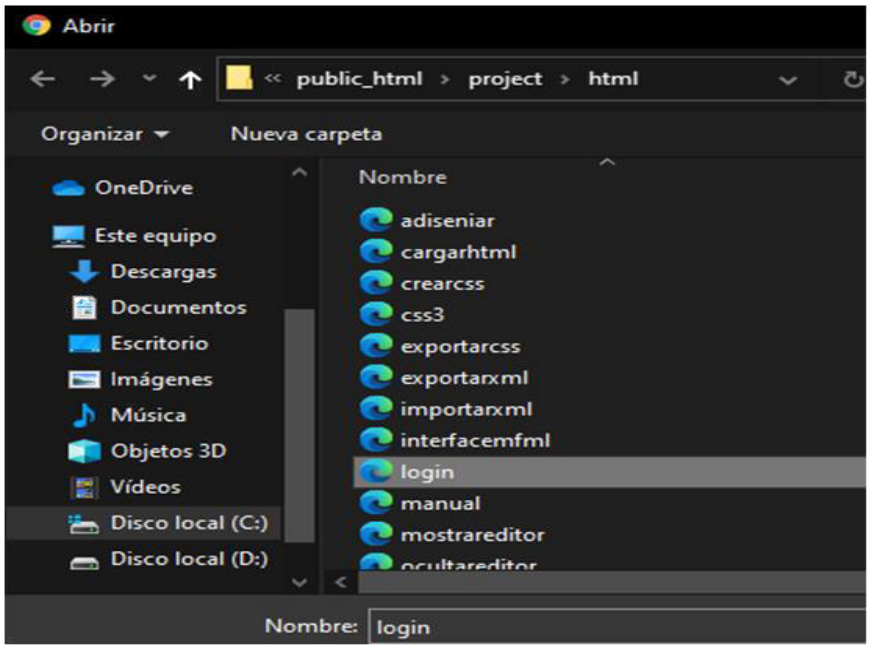


Figura 55 **Framework web CSS 3 - Seleccionar archivo .html.**

Fuente: Elaboración propia

Finalmente, notamos que el texto "Paso 1 – cargar .html", es reemplazado por el nombre y extensión del archivo importado "login.html". Además, se muestra el código html del archivo.

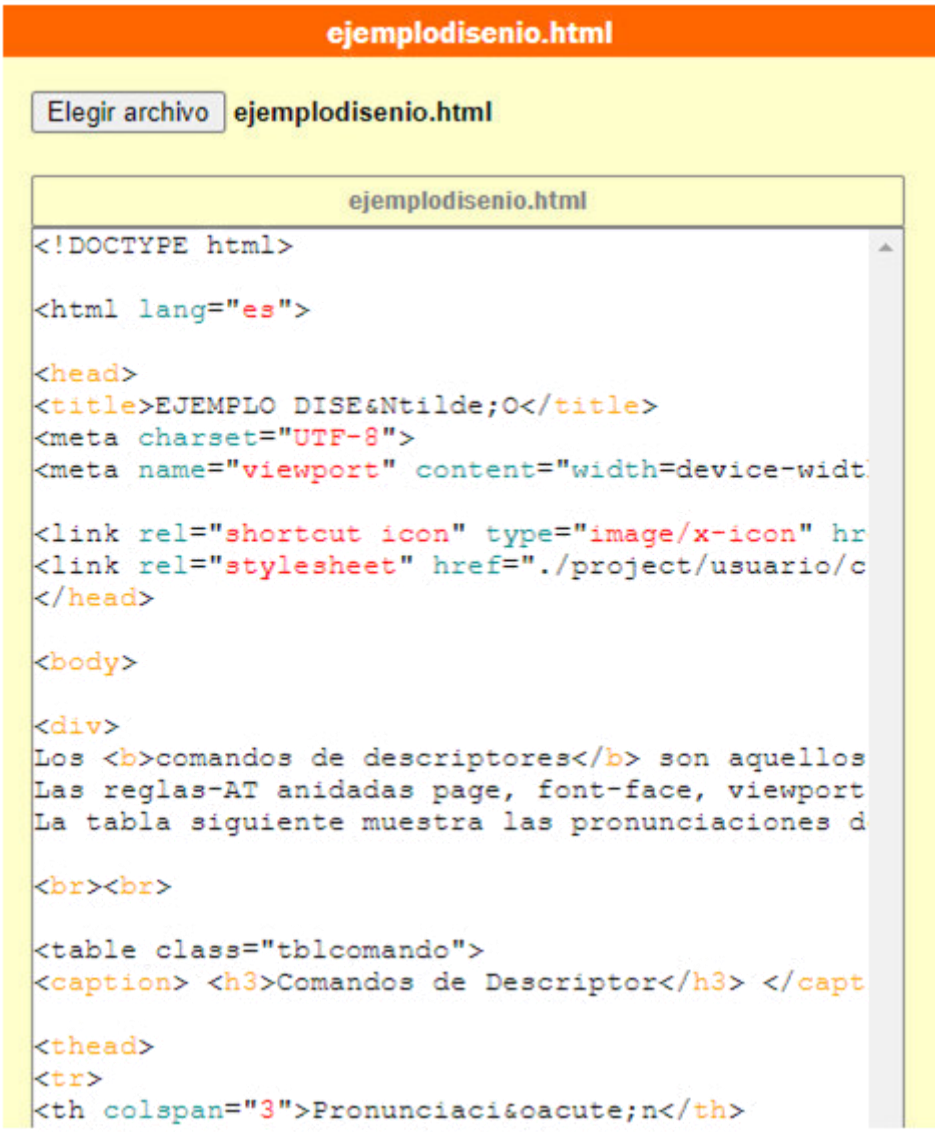


Figura 56 **Framework web CSS 3 - Archivo .html cargado.**

Fuente: Elaboración propia

Una vez se hayan completado con éxito los pasos anteriores, se concluye el paso de carga del archivo .html a maquetar.

4.4.2. Paso 2 – Importar .XML o Crear .CSS

4.4.2.1. Importar .xml

Los archivos .xml son utilizados como repositorios de datos, y en este caso, los datos almacenados son: número de fila, tipo de elemento, clases e IDs de las filas, columnas y elementos textarea, contenido del textarea y el nivel de anidamiento de cada

elemento CSS 3, conteniendo a los elementos: reglas-AT lineales, reglas-AT anidadas, descriptores, selectores, declaraciones, comentarios. El archivo .xml inicia su existencia luego de que el framework de voz CSS 3 haya terminado de generar el modelo lógico y descargado en el dispositivo del usuario los archivos .css y .xml con el nombre que el usuario haya especificado para la hoja de estilos CSS 3 generada. Para importar el archivo .xml mencionado hacemos clic en el botón "Seleccionar archivo" para abrir la ventana de búsqueda y selección de archivos. Esta ventana sólo mostrará archivos con extensión .xml facilitando su búsqueda y selección.

Figura 57 **Framework web CSS 3 - Botón seleccionar archivo .xml.**

Fuente: Elaboración propia

Una vez se muestre la ventana de búsqueda y selección de archivos, procedemos a seleccionar el archivo .xml deseado, el cual como se explicó presenta las características siguientes: contiene a los elementos reglas-AT lineales, reglas-AT anidadas, descriptores, selectores, declaraciones, comentarios y el comportamiento de la hoja de estilo: nivel de anidamiento (descendencia).

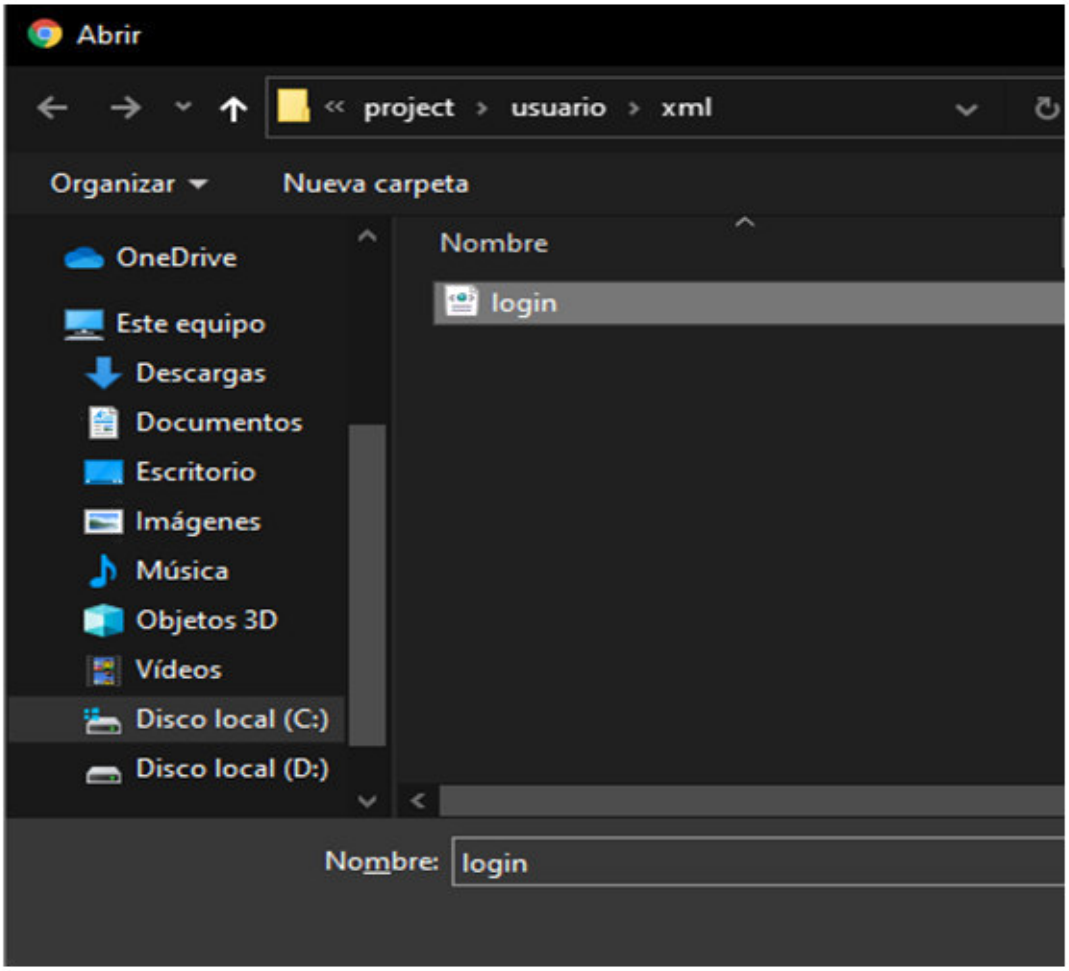


Figura 58 **Franework web CSS 3 - Seleccionar archivo .xml.**

Fuente: Elaboración propia

Finalmente, notamos que el texto "Paso 2 – importar .xml o crear .css", es reemplazado por el nombre y extensión del archivo importado "login.css". Además, se muestra el modelo físico el cual es la tabla html, con sus respectivas filas, columnas, elementos textarea, contenido, clases e IDs. Esta tabla html es generada por la Interface MFML: Modelo Físico - Modelo Lógico y es posible continuar desarrollando código CSS 3.

Cabe mencionar que el color verde representa a las reglas-AT anidadas y lineales, el color naranja simboliza a los descriptores pertenecientes a la regla-AT anidada, el color rojo representa a los selectores y finalmente el color azul simboliza a los descriptores.

Figura 59 **Framework web CSS 3 - Modelo Físico generado - Archivo .xml cargado.**

Fuente: Elaboración propia

Una vez se hayan completado con éxito los pasos anteriores, se concluye el paso de carga del archivo .xml (modelo lógico) a partir del cual, la Interface MFML: Modelo Físico – Modelo Lógico, genera la tabla html (modelo físico).

4.4.2.2. Crear .css

Una vez hayamos terminado de cargar el archivo .html, es posible continuar administrando la hoja de estilo CSS 3, a través de la importación del archivo .xml o podemos crear una nueva hoja de estilo .css, haciendo uso de los comandos de voz que nos permitirán interactuar con el framework de voz CSS 3 a través de la tecnología Web Speech API.

Entonces, los comandos de voz, son instrucciones que el usuario expresa verbalmente al framework de voz CSS 3 para que este, a su vez, haciendo uso de la tecnología Web Speech API, obtenga en formato texto la instrucción verbal, procese internamente y devuelva como respuesta el texto en sí mismo o la realización de alguna acción como: inserción o eliminación de elementos, realizar desplazamientos entre componentes o, funciones de edición de texto.

Como primer paso, se tiene que asignar el nombre del archivo .css a generar. El nombre a asignar tiene que ser acorde al propósito que tendrá dicho archivo dentro de nuestra página web, es decir, se sugiere asignar un nombre acorde con la función que se espera que realice. Por ejemplo, si deseamos maquetar el archivo web de la ventana de autenticación, pues sería lógico asignarle de nombre "login.css". La figura siguiente muestra el nombre asignado a la hoja de estilo CSS 3 a generar.

Figura 60 **Framework web CSS 3 - Asignar nombre al archivo .css.**

Fuente: Elaboración propia

Una vez se haya escrito el nombre del archivo .css notamos que el texto "Paso 2 – importar .xml o crear .css" es reemplazado por el nombre asignado más la extensión .css, la cual es asignada por el mismo framework, y no es necesario escribir dicha extensión.

Luego de asignado el nombre del archivo .css a generar, procedemos a insertar los elementos que contendrá la hoja de estilo. Para ello, el framework CSS 3 tiene incorporado comandos de voz. Cada comando tiene dos (02) formas de ser pronunciado: la primera es pronunciar la palabra "comando" seguido de la "acción a realizar". Así, por ejemplo, para insertar un nuevo selector, se tiene que pronunciar la frase: "comando insertar selector". La segunda forma es usar la abreviación de dicho

comando, la cual es la “palabra comando” seguido del “número de comando”, en el caso anterior será: “comando uno”.

La figura siguiente muestra los elementos: regla-AT, descriptor, selector y declaración insertados en la tabla html (modelo físico) a través del uso de estos comandos.

Figura 61 **Framework web CSS 3 - Inserción de elementos por comandos de voz.**

Fuente: Elaboración propia

Como en el caso anterior, para la inserción de declaraciones pronunciamos la frase: “comando insertar declaración” o “comando dos”; para insertar reglas-AT pronunciamos: “comando insertar at&t” o “comando tres”. La inserción de descriptores es igual que la inserción de declaraciones, sólo que tiene que ser pronunciado desde una regla-AT anidada.

4.4.3. Ejecutar Código – Ocultar o Mostrar el framework

4.4.3.1. Ocultar framework

A medida que vayamos avanzando el desarrollo del modelo físico, conformado por la tabla html y sus respectivos elementos, clases e IDs, los cuales representan la estructura de la hoja de estilo CSS 3 a generar a través de los comandos de voz y luego de asignar un nombre al archivo .css acorde a la función que realizará dentro del proyecto de nuestra página web, podemos ir visualizando progresivamente el resultado del maquetado del archivo .html, en cualquier momento.

Para ello el framework presenta el botón llamado "Ocultar framework", el cual al hacer clic sobre él hará tres cosas:

1. Ocultará el framework CSS 3 pero seguirá mostrando los botones "Ocultar framework" y "Mostrar framework" en la parte superior del navegador web,
2. Aplicará el estilo CSS 3 desarrollado hasta el momento de hacer clic sobre él y,
3. Mostrará el archivo .html con el estilo aplicado

De tal manera que se pueda obtener una retroalimentación sobre el desarrollo del maquetado del archivo .html y la posterior generación del modelo lógico (archivos .css y .xml). La figura siguiente muestra el botón "Ocultar framework", el cual se encuentra ubicado en la parte superior derecha de la barra de comandos de voz.

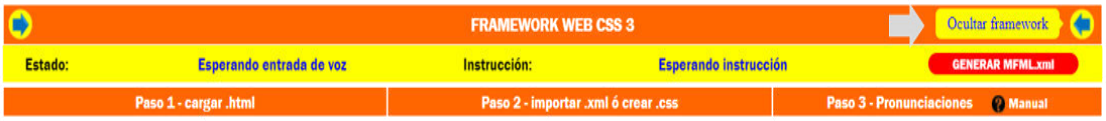


Figura 62 **Framework web CSS 3 - Botón ocultar framework.**

Fuente: Elaboración propia

Cabe mencionar, que el archivo .html es encapsulado dentro de un "elemento div padre", el cual desciende directamente de la etiqueta body; es decir, el archivo .html

es insertado dentro de una etiqueta div con ID “maincontent”: <body> archivo.html </body>.

Finalmente, la barra que contiene los botones “Ocultar framework” y “Mostrar framework” siempre se encuentran visibles, en el estado de edición de contenido y en el estado de ejecución del maquetado, de tal manera que el usuario siempre pueda editar o ejecutar contenido CSS 3.

4.4.3.2. Mostrar framework

A medida que vayamos avanzando el desarrollo del modelo físico, conformado por la tabla html y sus respectivos elementos, clases e IDs, los cuales representan la estructura de la hoja de estilo CSS 3 a generar a través de los comandos de voz y luego de asignar un nombre al archivo .css acorde a la función que realizará dentro del proyecto de nuestra página web, podemos ir visualizando progresivamente el resultado del maquetado del archivo .html, en cualquier momento.

Para ello el framework presenta el botón llamado “Mostrar framework”, el cual al hacer clic sobre él hará dos cosas:

1. Ocultará el archivo .html maquetado, pero seguirá mostrando los botones “Ocultar framework” y “Mostrar framework” en la parte superior y,
2. Mostrará el framework de voz CSS 3 para continuar desarrollando el modelo físico

De tal manera que se pueda obtener una retroalimentación sobre el desarrollo del maquetado del archivo .html y la posterior generación del modelo lógico (archivos .css y .xml). La figura siguiente muestra el botón “Ocultar framework”, el cual se encuentra ubicado en la parte superior derecha de la barra de comandos de voz.

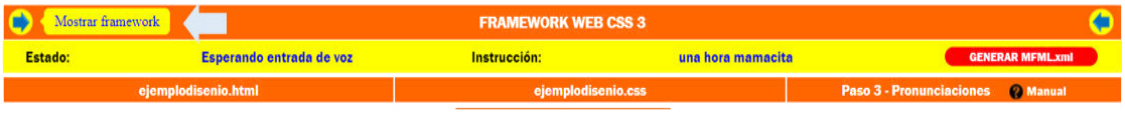


Figura 63 **Framework web CSS 3 - Botón mostrar framework.**

Fuente: Elaboración propia

Cabe mencionar, que el archivo .html es encapsulado dentro de un “elemento div padre”, el cual desciende directamente de la etiqueta body; es decir, el archivo .html es insertado dentro de una etiqueta div con ID “maincontent”: <body> archivo.html </body>.

Finalmente, la barra que contiene los botones “Ocultar framework” y “Mostrar framework” siempre se encuentran visibles, en el estado de edición de contenido y en el estado de ejecución del maquetado, de tal manera que el usuario siempre pueda editar o ejecutar contenido CSS 3.

4.4.4. Generar MFML – Exportar archivos .XML y .CSS

Luego de desarrollado completamente el modelo físico que viene a ser la tabla html y sus componentes: filas, columnas, elementos textarea, sus respectivas clases, IDs y contenido, los cuales representan a los elementos de la hoja de estilo CSS 3 y el nivel de anidamiento, que es el nivel (j = 1 o j > 1) de descendencia de reglas-AT, gestionado a través de la teoría de nomenclaturas.

Y luego de haber comprobado el maquetado del archivo .html mediante las funciones ocultar y mostrar framework, es momento de finalizar la generación de código CSS 3 a través de comandos de voz. Se entiende por finalizar la generación de código CSS 3 al procedimiento de creación y descarga del modelo lógico de la Interface MFML, compuesta por los archivos .xml y .css. Para ello, hacemos clic sobre el botón “GENERAR MFML.xml”, ubicado en la barra de comandos de voz del framework.

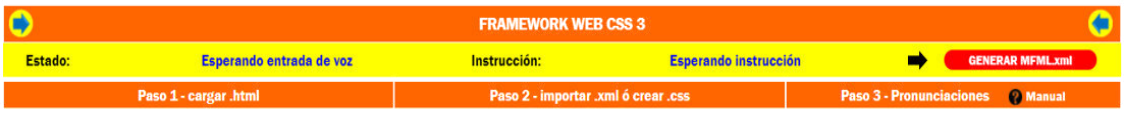


Figura 64 **Framework web CSS 3 - Barra de comandos de voz.**

Fuente: Elaboración propia

Este botón realiza dos acciones:

1. Verifica que el archivo .css a generar tenga asignado un nombre apropiado a la función que realizará y,

2. Genera y descarga los archivos .xml y .css en el dispositivo del usuario a través del cual se conecta al framework de voz CSS 3

Luego de hacer clic en el botón GENERAR MFML.xml, se puede visualizar que en la parte inferior del navegador web del usuario aparece una barra que contiene el enlace a los dos archivos descargados.

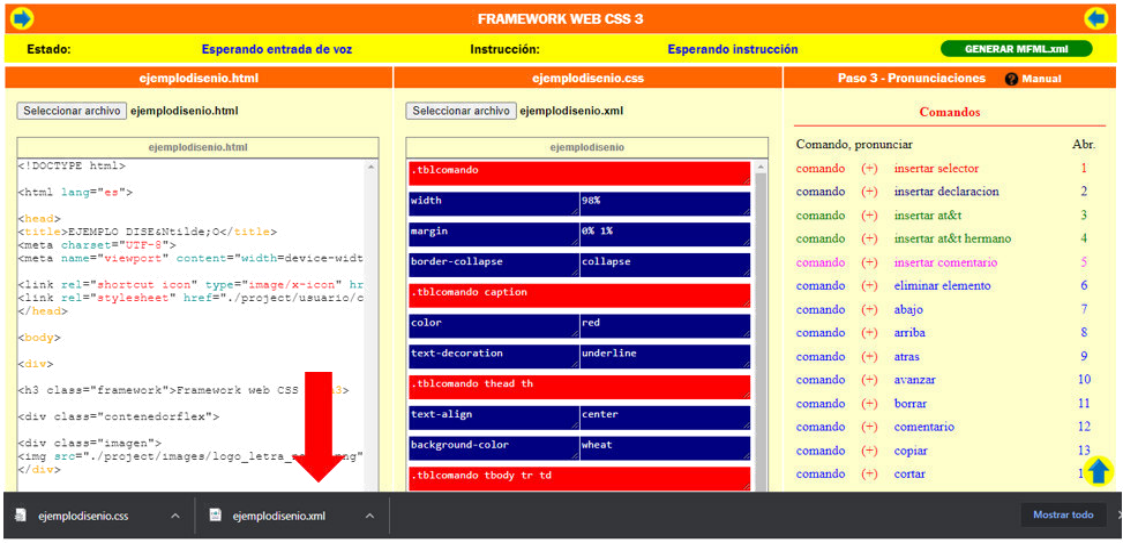


Figura 65 **Framework web CSS 3 - Exportación Modelo Lógico (archivos .xml y .css).**

Fuente: Elaboración propia

Estos archivos se pueden verificar en la carpeta de descargas del dispositivo desde donde el usuario se conecta al framework CSS 3.

Figura 66 **Framework web CSS 3 - Archivos descargados .xml y .css.**

Fuente: Elaboración propia

Al abrir el archivo “login.css” se muestra el contenido de dicho archivo, el cual ahora puede ser integrado en el directorio css del proyecto de maquetado de nuestra página web.

```
html {
    color: red;
}

@media all {
    link {
        animation: ease;
    }

    @viewport {
        max-width: max-content;
    }

    @supports (max-height: 19px) {
        border: 1px solid red;
    }

}
```

Figura 67 **Framework web CSS 3 - Archivo descargado .css.**

Fuente: Elaboración propia

Al abrir el archivo “login.xml” se muestra el contenido de dicho archivo, el cual de ahora en adelante servirá para que la Interface MFML pueda volver a generar el modelo físico (tabla html) para continuar maquetando. La figura siguiente muestra el contenido .XML generado.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<modeloLogicoCSSXML>
    <arrayMFML>
        <elementoMFML>
            <fila>1</fila>
            <elemento>selector</elemento>
            <tr_clase>tr_cls_selector</tr_clase>
            <tr_id>tr_id_selector_1</tr_id>
            <td_clase>td_cls_selector</td_clase>
            <td_id>td_id_selector_1_tr_id_selector_1</td_id>
            <txta_clase>cls_selector</txta_clase>
            <txta_id>txta_id_selector_1_td_id_selector_1_tr_id_selector_1</txta_id>
            <contenido>html</contenido>
            <nivel>1</nivel>
        </elementoMFML>
        <elementoMFML>
            <fila>2</fila>
            <elemento>propiedad</elemento>
            <tr_clase>tr_cls_declaracion_tr_id_selector_1</tr_clase>
            <tr_id>tr_id_declaracion_1_tr_id_selector_1</tr_id>
            <td_clase>td_cls_propiedad_tr_id_selector_1</td_clase>
            <td_id>td_id_propiedad_1_tr_id_declaracion_1_tr_id_selector_1</td_id>
            <txta_clase>cls_propiedad_tr_id_selector_1</txta_clase>
            <txta_id>txta_id_propiedad_1_td_id_propiedad_1_tr_id_declaracion_1_tr_id_selector_1</txta_id>
            <contenido>color</contenido>
            <nivel>2</nivel>
        </elementoMFML>
        <elementoMFML>
            <fila>2</fila>
            <elemento>valor</elemento>
            <tr_clase>tr_cls_declaracion_tr_id_selector_1</tr_clase>
```


Figura 68 **Framework web CSS 3 - Archivo descargado .xml.**

Fuente: Elaboración propia

Claramente el archivo .xml almacena los datos siguientes: número de fila, tipo de elemento, clases e IDs de las filas, columnas y elementos textarea, contenido del textarea y el nivel de anidamiento de cada elemento CSS 3.

Finalmente, es más que obligatorio recordar no dañar, modificar ni extraviar el archivo .xml pues si no, evidentemente, la Interface MFML no podrá volver a generar el modelo físico (tabla html), el cual nos permite continuar agregando más contenido o modificar el archivo .css que ahora forma parte de nuestro proyecto de maquetado web.

4.4.5. Ejemplo Aplicativo

Para demostrar el funcionamiento de nuestro framework CSS 3, vamos a seguir todos los pasos a continuación descritos con los alumnos del décimo ciclo de la Escuela Profesional de Ingeniería de Sistemas e Informática de la UNASAM, los cuales nos permitirán maquetar el archivo ejemplodisenio.html.

L**comandos de descriptores** son aquellos que **permiten insertar propiedades y sus valores respectivos, correspondientes a reglas-AT anidadas**.Las reglas-AT anidadas page, font-face, viewport y co r-st tienen descriptores: propiedades - valores, claramente definidos.La tabla siguiente muestra las pronunciaciones de esta categoría.

Comandos de Descriptor

	Pronunciación	Abr.	Descripción
pagina	(+) nombre pagina	número	Inserta el descriptor propiedad, correspondiente a las propiedades margin, padding, border y background de la regla-AT anidada page, en el elemento regla-AT desde el cual fue invocado
letra	(+) nombre letra	número	Inserta el descriptor propiedad de la regla-AT anidada font-face en el elemento regla-AT desde el cual fue invocado
ventana	(+) nombre ventana	número	Inserta el descriptor propiedad de la regla-AT anidada viewport en el elemento regla-AT desde el cual fue invocado
contador estilo	(+) nombre contador	número	Inserta el descriptor propiedad de la regla-AT anidada counter-style en el elemento regla-AT desde el cual fue invocado

Fuente: Elaboración propia

Figura 69 **Framework web CSS 3 - Archivo ejemplodisenio.html.**

Fuente: Elaboración propia

Como se muestra en la figura anterior, el archivo ejemplodisenio.html sólo contiene código html sin formato ni colores, lo cual no es agradable para los usuarios que visitan nuestra página web.

Así que, el primer paso es importar dicho archivo al framework CSS 3. Este archivo debe tener una etiqueta “link” dentro de la sección “head”, la cual tiene como valor (ruta) de la propiedad href a: “./project/usuario/css/”, la cual es estática y siempre debe ser la misma para cualquier otro archivo .html que se desee cargar, seguida por el “nombre” que se dará a la hoja de estilos CSS 3 a generar, en nuestro caso “ejemplodisenio.css”.

La figura siguiente muestra la etiqueta “link” dentro de la sección “head” del archivo ejemplodisenio.html.

```
1   <!DOCTYPE html>
2
3   <html lang="es">
4
5       <head>
6           <title>EJEMPLO DISE&Ntilde;O</title>
7           <meta charset="UTF-8">
8           <meta name="viewport" content="width=device-width, initial-scale=1.0">
9
10          <link rel="shortcut icon" type="image/x-icon" href="./project/images/favicon.ico">
11          <link rel="stylesheet" href="./project/usuario/css/ejemplodisenio.css">
12      </head>
```

Figura 70 **Framework web CSS 3 - Sección head del archivo ejemplodisenio.html.**

Fuente: Elaboración propia

Una vez garantizada esta ruta procedemos a hacer clic en el botón “Elegir archivo” del panel lateral izquierdo llamado, “Paso 1 – cargar .html”, para seleccionar finalmente el archivo ejemplodisenio.html.

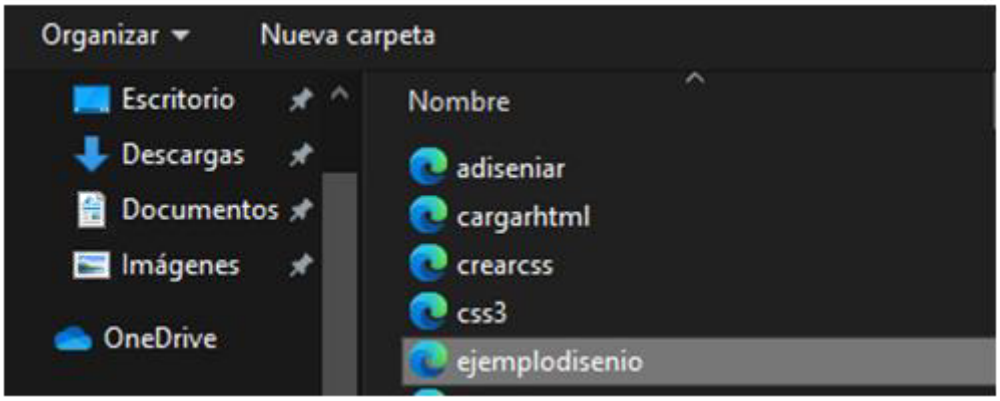


Figura 71 **Framework web CSS 3 - Seleccionar archivo ejemplodisenio.html.**

Fuente: Elaboración propia

Luego de seleccionar y abrir el archivo ejemplodisenio.html, este es cargado en el panel lateral izquierdo, reemplazando el nombre “Paso 1 – cargar .html” por ejemplodisenio.html.

ejemplodisenio.html

Elegir archivo ejemplodisenio.html

ejemplodisenio.html

```
<!DOCTYPE html>

<html lang="es">

<head>
<title>EJEMPLO DISE&Ntilde;O</title>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-widt

<link rel="shortcut icon" type="image/x-icon" hr
<link rel="stylesheet" href="./project/usuario/c
</head>

<body>

<div>
Los <b>comandos de descriptores</b> son aquellos
Las reglas-AT anidadas page, font-face, viewport
La tabla siguiente muestra las pronunciaciones d

<br><br>

<table class="tblcomando">
<caption> <h3>Comandos de Descriptor</h3> </capt

<thead>
<tr>
<th colspan="3">Pronunciaci&oacute;n</th>
```

Figura 72 **Framework web CSS 3 - Archivo ejemplodisenio.html cargado.**

Fuente: Elaboración propia

Proseguimos con el paso 2, en nuestro caso, la creación del archivo ejemplodisenio.css. Para ello, reemplazamos el texto “Ingrese nombre hoja CSS a crear” por “ejemplodisenio” de la caja de texto que se encuentra en el panel central, con nombre “Paso 2 – importar .xml o crear .css”.

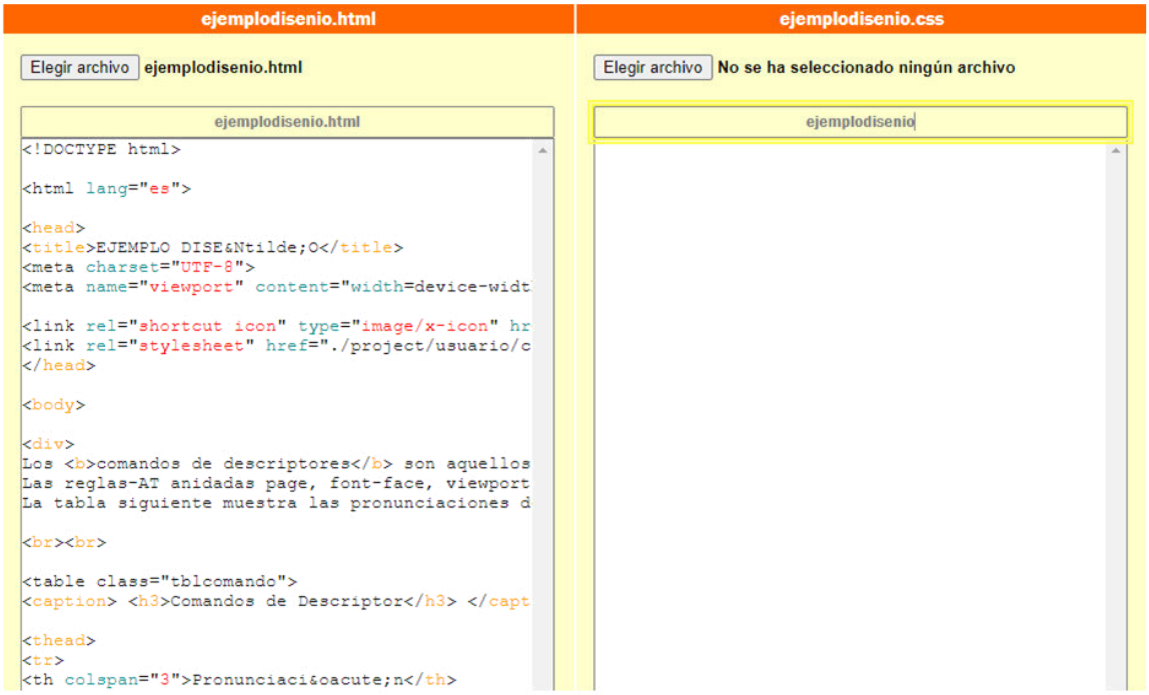


Figura 73 **Framework web CSS 3 - Escribir nombre ejemplodisenio de archivo .css.**

Fuente: Elaboración propia

Ahora estamos listos para empezar a crear y añadir contenido al modelo físico. Para ello haremos uso de los comandos de voz. Los comandos a pronunciar serán:

1. comando uno: para insertar un nuevo selector, con fondo de color rojo, iniciando la creación del modelo físico
2. class t: para insertar el texto ".t" en el selector desde el cual fue invocado
3. texto b: para insertar el texto "b" en el selector desde el cual fue invocado
4. texto l: para insertar el texto "l" en el selector desde el cual fue invocado
5. texto comando: para insertar el texto "comando" en el selector desde el cual fue invocado

Con la pronunciación de los comandos anteriores se formó el contenido: ".tblcomando", del nuevo selector de clase. Bajo esta lógica procedemos a insertar los demás selectores, sus respectivas declaraciones y contenidos. La tabla mostrada en el ANEXO 12 muestra los comandos a pronunciar, en el orden consecutivo, para maquetar completamente el archivo ejemplodisenio.html. La figura siguiente muestra el modelo físico desarrollado.

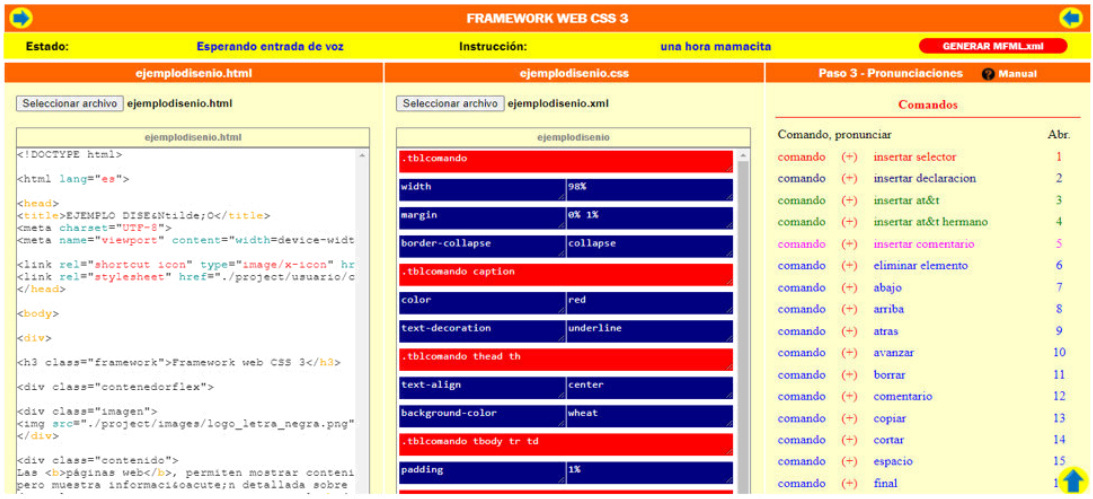


Figura 74 **Framework web CSS 3 - Modelo Físico final archivo ejemplodisenio.html.**

Fuente: Elaboración propia

En cualquier momento de desarrollo del modelo físico se puede ir comprobando el maquetado del archivo ejemplodisenio.html, para ello hacemos clic en el botón "Ocultar framework", ubicado en la parte superior del framework CSS 3, entonces el framework se ocultará y en cambio se muestra el archivo ejemplodisenio.html maquetado con la hoja de estilo CSS 3.

Figura 75 **Framework web CSS 3 - Archivo ejemplodisenio.html Maquetado.**

Fuente: Elaboración propia

Para volver a visualizar el framework CSS 3, hacemos clic en el botón "Mostrar framework", ubicado en la parte superior del framework. Una vez mostrado el framework, procedemos a generar el modelo lógico, haciendo clic en el botón "GENERAR MFML.xml" y entonces, en la parte baja del navegador del dispositivo

del usuario con el cual se conecta al framework de voz CSS 3, se muestran los archivos “ejemplodisenio.css” y “ejemplodisenio.xml” generados y descargados por el framework.

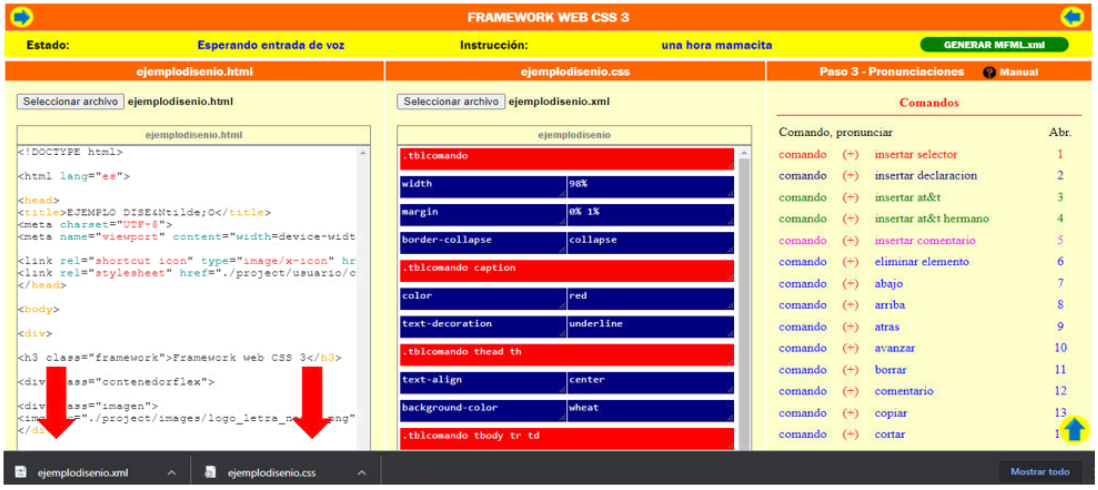


Figura 76 **Framework web CSS 3 - Modelo Lógico ejemplodisenio descargado.**

Fuente: Elaboración propia

En la carpeta de descargas del dispositivo del usuario con el cual se conecta al framework de voz CSS 3, se muestran los archivos “ejemplodisenio.css” y “ejemplodisenio.xml” descargados por el framework.

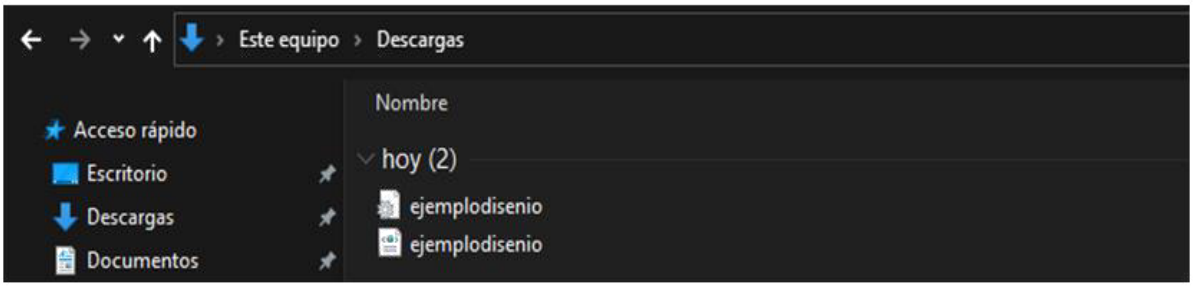


Figura 77 **Framework web CSS 3 - Archivos ejemplodisenio .css / .xml descargados.**

Fuente: Elaboración propia

Entonces ahora es posible integrar el archivo ejemplodisenio.css en la carpeta css de nuestro proyecto de maquetado web, tal como se muestra en la figura siguiente.

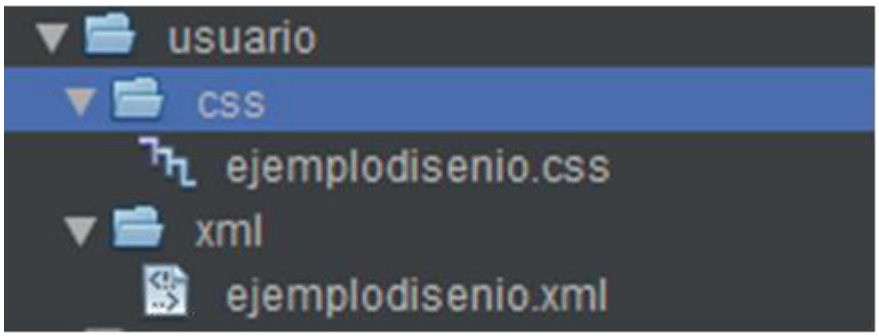


Figura 78 **Framework web CSS 3 - Archivo ejemplodisenio.css integrado en el proyecto.**

Fuente: Elaboración propia

Al abrir el archivo ejemplodisenio.xml se muestra el contenido de dicho archivo. Claramente el archivo .xml almacena los datos siguientes: número de fila, tipo de elemento, clases e IDs de las filas, columnas y elementos textarea, contenido del textarea y el nivel de anidamiento de cada elemento CSS 3.

Es más que obligatorio recordar no dañar, modificar ni extraviar el archivo .xml pues si no, evidentemente, la Interface MFML no podría volver a generar el modelo físico (tabla html), el cual nos permite continuar agregando más contenido o modificar el archivo .css que ahora forma parte de nuestro proyecto de maquetado web.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<modeloLogicoCSSXML>
    <arrayMFML>
        <elementoMFML>
            <fila>1</fila>
            <elemento>selector</elemento>
            <tr_clase>tr_cls_selector</tr_clase>
            <tr_id>tr_id_selector_1</tr_id>
            <td_clase>td_cls_selector</td_clase>
            <td_id>td_id_selector_1_tr_id_selector_1</td_id>
            <txta_clase>cls_selector</txta_clase>
            <txta_id>txta_id_selector_1_td_id_selector_1_tr_id_selector_1</txta_id>
            <contenido>.tblcomando</contenido>
            <nivel>1</nivel>
        </elementoMFML>
        <elementoMFML>
            <fila>2</fila>
            <elemento>propiedad</elemento>
            <tr_clase>tr_cls_declaracion_tr_id_selector_1</tr_clase>
            <tr_id>tr_id_declaracion_1_tr_id_selector_1</tr_id>
            <td_clase>td_cls_propiedad_tr_id_selector_1</td_clase>
            <td_id>td_id_propiedad_1_tr_id_declaracion_1_tr_id_selector_1</td_id>
            <txta_clase>cls_propiedad_tr_id_selector_1</txta_clase>
            <txta_id>txta_id_propiedad_1_td_id_propiedad_1_tr_id_declaracion_1_tr_id_selector_1</txta_id>
            <contenido>width</contenido>
            <nivel>2</nivel>
        </elementoMFML>
        <elementoMFML>
            <fila>2</fila>
            <elemento>valor</elemento>
            <tr_clase>tr_cls_declaracion_tr_id_selector_1</tr_clase>
```


Figura 79 **Framework web CSS 3 - Contenido archivo ejemplodisenio.xml.**

Fuente: Elaboración propia

En la figura siguiente se aprecia al selector de clase llamado ".tblcomando", el cual representa a una tabla html, que según las declaraciones que contiene: "width: 95%; margin: 0% 2%; border-collapse: collapse", el ancho que se le ha asignado es del 95% del ancho total del dispositivo, tiene un margen derecho e izquierdo de 2% y los bordes entre sus filas y columnas no contienen ningún espacio entre ellos. Además, se aprecia también que la etiqueta html "caption", la cual se encuentra dentro de la tabla llamada "tblcomando" representado mediante el selector compuesto: ".tblcomando caption", según las declaraciones que contienen: "color: red; text-decoration: underline", indica que el texto contenido será de color rojo y que se dibujará una línea horizontal, también de color rojo, en la parte inferior. Otro de los aspectos importantes a recalcar es la generación de sangrías para cada declaración debajo de cada selector y la inserción de los signos de puntuación, como son: las llaves de apertura ({) y cierre (}) entre los bloques de código, los dos puntos (:) entre cada propiedad y valor, así como los signos de punto y coma (;) al final de cada declaración.

```
1  .tblcomando {
2      width: 95%;
3      margin: 0% 2%;
4      border-collapse: collapse;
5  }
6
7  .tblcomando caption {
8      color: red;
9      text-decoration: underline;
10 }
11
12 .tblcomando thead th {
13     text-align: center;
14     background-color: wheat;
15 }
16
17 .tblcomando tbody tr td {
18     padding: 1%;
19 }
20
21 .tblcomando tbody tr:nth-child(2n) {
22     background-color: wheat;
23 }
24
25 .tblcomando tbody tr:hover {
26     background-color: yellow;
27 }
28
29 .tblcomando tbody td:nth-child(1) {
30     width: 8%;
31 }
```

Figura 80 **Framework web CSS 3 - Contenido archivo ejemplodisenio.css.**

Fuente: Elaboración propia

4.4.6. Código Fuente

El código fuente del framework CSS 3, es puesto a disposición del usuario en general, bajo la licencia GNU, y se autoriza la libre distribución, modificación o reproducción, por parte del público en general con fines de:

1. Investigación científica

2. Mejora del diseño o de las funciones del framework
3. Implementación de nuevas funcionalidades
4. Otros fines que el profesional, investigador, estudiante, aficionado o usuario en general considere pertinente

Para descargar el código fuente del proyecto, visitamos la url siguiente: https://www.editorvozcss3.com/Frameworkcss3/public_html/project/html/manual.html. Una vez accedido al manual del framework CSS 3, hacemos clic en el link "Descargar" del ítem "Código fuente" perteneciente a la sección llamada "Tabla de contenidos", y esperamos a que finalice la descarga.

Una vez descargado el archivo "Frameworkcss3.rar", nos dirigimos a la carpeta descargas del dispositivo a través del cual se accede al framework y procedemos a extraerlo. Finalizada la extracción del archivo se tendrá la carpeta "Frameworkcss3", dentro del cual, se encontrará la carpeta "public_html".

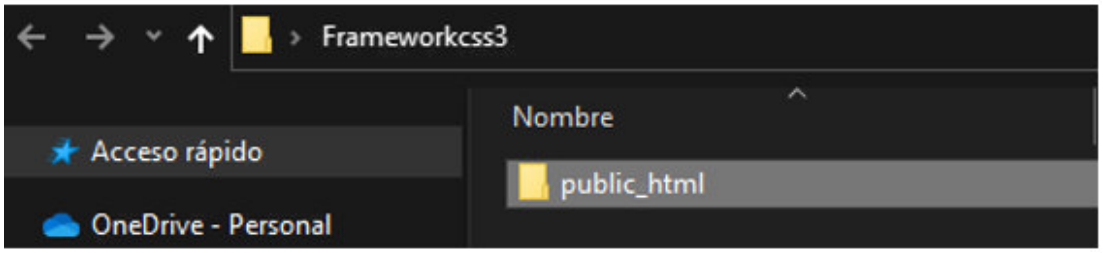


Figura 81 **Framework web CSS 3 - Extracción del archivo Frameworkcss3.rar.**

Fuente: Elaboración propia

Dentro de la carpeta "public_html" se encontrará la carpeta "project" y los archivos "index.html", "indexescritorio.html" e "indexmobile.html".

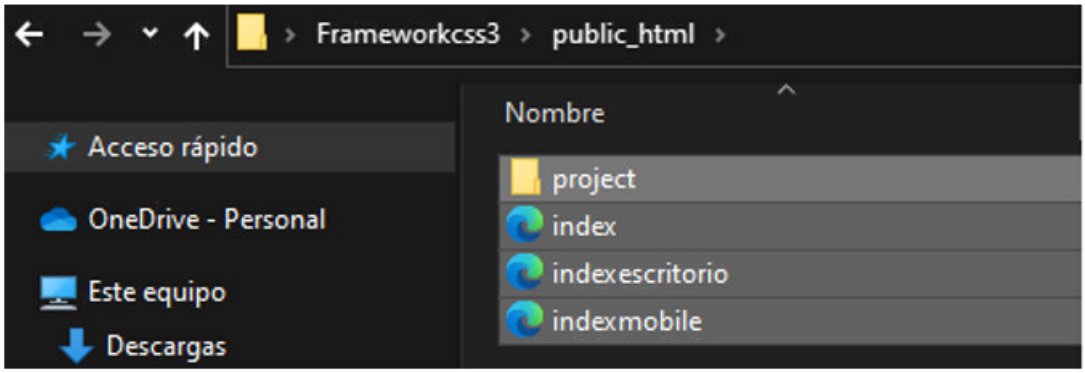


Figura 82 **Framework web CSS 3 - Archivos dentro de la carpeta public_html de Frameworkcss3.**

Fuente: Elaboración propia

Para ejecutar el framework de forma local subir la carpeta "Frameworkcss3" al directorio del servidor web que se elija. En este caso, se ha utilizado el servidor web Apache, y se ha pegado la carpeta "Frameworkcss3" dentro del directorio "htdocs", propio de este servidor web.

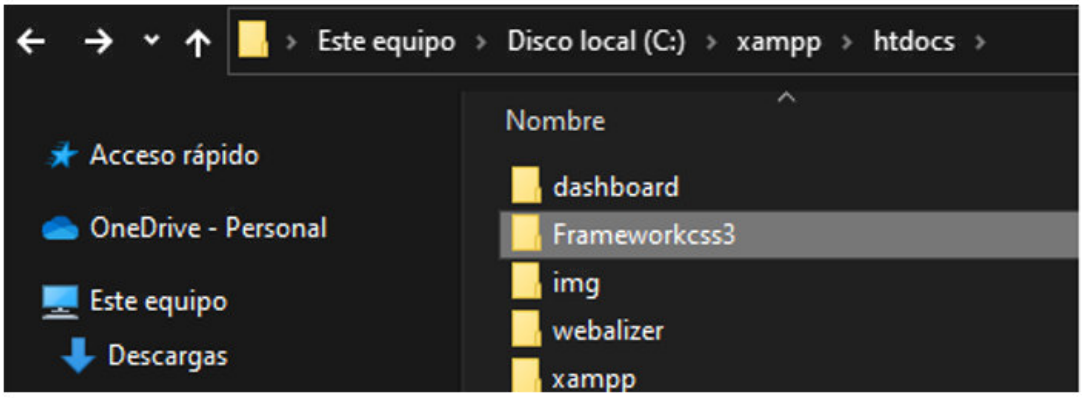


Figura 83 **Framework web CSS 3 - Subir carpeta Frameworkcss3 al servidor web.**

Fuente: Elaboración propia

Finalmente, abrir el navegador web Chrome v.90 o superior, e ingresar la ruta http del proyecto para ejecutar el framework de voz CSS 3.

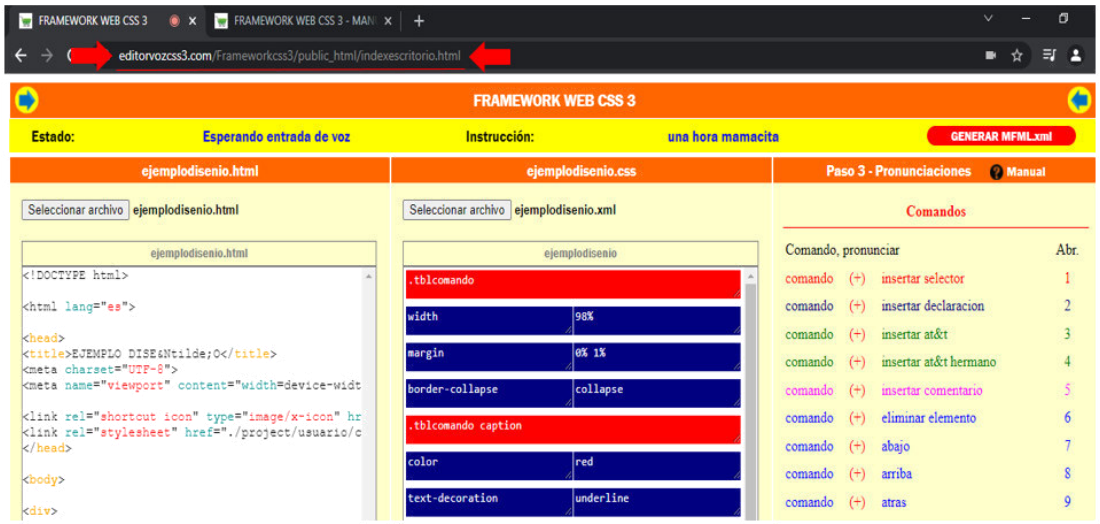


Figura 84 **Framework web CSS 3 - Ruta http local para ejecución del framework.**

Fuente: Elaboración propia

De igual forma, se puede subir el framework de voz CSS 3 a un Hosting y Dominio en la nube. Para ello usaremos cualquier programa que gestione el protocolo FTP y, luego de subir la carpeta “Frameworkcss3” del proyecto al servidor en la nube, ingresar la ruta http o https, desde el navegador web Chrome v.90 o superior.

Evidentemente la carpeta “project” localizada, dentro de la carpeta “Frameworkcss3/public_html”, contiene todos los archivos fuente del e framework CSS 3, y es posible realizar la modificación de dichos archivos a través del uso de cualquier framework de desarrollo que soporte los lenguajes: JavaScript, HTML 5, CSS 3 y PHP.

CAPÍTULO V: VALIDACIÓN Y RESULTADOS

En este capítulo se detallará el procedimiento de validación y los resultados obtenidos luego de implementar el framework web CSS 3 con reconocimiento de voz en la Escuela de Ingeniería de Sistemas e Informática de la UNASAM, 2021. Se ha considerado conveniente dividir este capítulo en: validación donde explicaremos todas las actividades realizadas durante el proceso de implementación de la tesis; resultados descriptivos donde se muestran los gráficos de barra de los porcentajes de mejora de las dimensiones evaluadas comparando el pre-test frente al post-test; resultados según objetivos donde se muestran las tablas descriptivas con las medias de los resultados y las tablas con los resultados de las pruebas T de Student de igualdad de medias por cada dimensión evaluada.

5.1. Validación

La etapa de validación consistió en la realización de dos reuniones virtuales mediante el programa Zoom. En la primera reunión virtual se evaluó las capacidades de los participantes sobre maquetado web, mediante la aplicación del cuestionario por parte del investigador, con la finalidad de garantizar que los resultados obtenidos para el pre-test reflejen el grado de conocimiento de los participantes con respecto a las dimensiones e indicadores de diseño, mediante el maquetado de una página web de prueba; posteriormente, se capacitó también a los participantes sobre el uso del framework propuesto. En la segunda reunión virtual, se procedió a aplicar el framework mediante el maquetado de la página web de prueba y, luego de finalizada la demostración de diseño de dicha página, se entregó a los participantes el cuestionario del post-test el cual fue llenado según su percepción sobre el grado de cumplimiento de cada indicador. Con los cuestionarios del pre-test y post-test llenados, procedimos a aplicar la técnica estadística medidas de correlación (Vinuesa, 2016) y T de Student para la obtención de resultados de cada dimensión.

La Figura 85 muestra todas las actividades del proceso de validación para probar la utilidad del framework propuesto y comprobar la hipótesis de esta investigación que indica que el framework ayuda en la automatización del maquetado web.

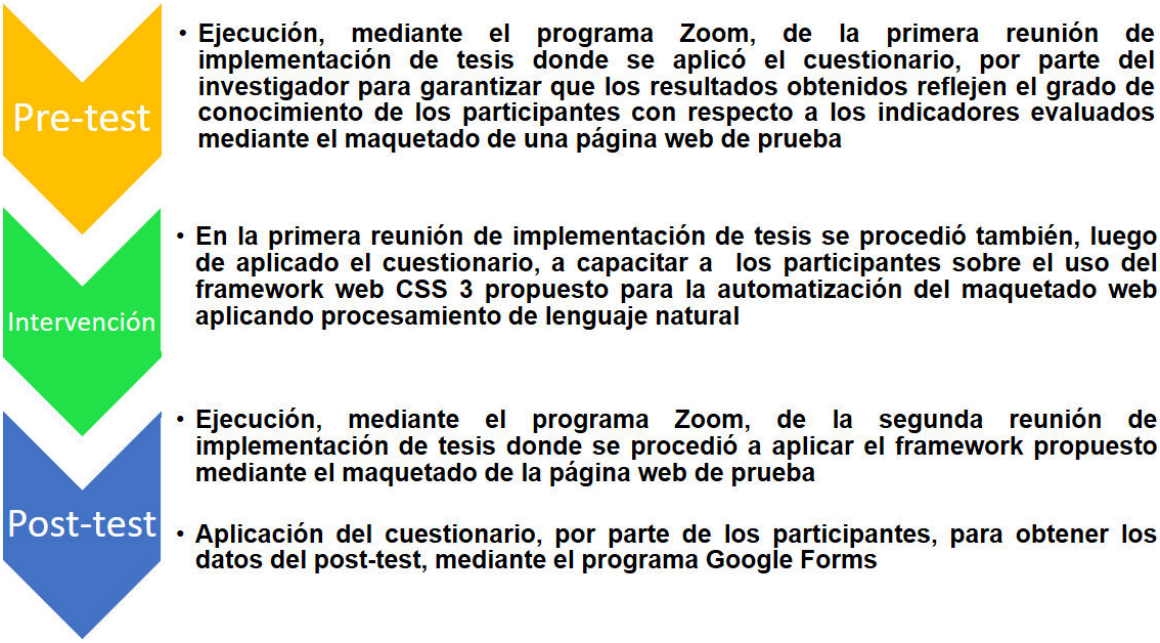


Figura 85 **Procedimiento de validación del framework propuesto.**

Fuente: Elaboración propia

Claramente el proceso de validación presenta tres escenarios: el primer escenario es el que engloba las actividades para obtener los datos del pre-test, el segundo escenario engloba el proceso de capacitación a los participantes sobre el uso del framework y, finalmente, el tercer escenario es el que engloba las actividades para obtener los datos del post-test.

5.1.1. Escenario 1: obtención de datos del pre-test

El primer escenario inició con la solicitud de apoyo para la implementación de tesis, mediante un email dirigido a la Decanatura de la Facultad de Ciencias de la UNASAM, el cual es mostrado en el Anexo 3. El email en cuestión contiene como datos adjuntos el formato único de trámite de la facultad, mostrado en el Anexo 2, donde se expuso claramente la solicitud de apoyo para la implementación del framework propuesto con un grupo de alumnos de la Escuela de Ing. de Sistemas e Informática matriculados en el noveno ciclo.

5.1.1.1. Aplicación del cuestionario para el pre-test

Una vez concedido el permiso para la implementación de la tesis procedimos a programar, mediante el programa Zoom, la primera reunión de implementación del framework como se muestra en el Anexo 4. Como primera actividad de la primera reunión virtual iniciamos la evaluación, mediante las escalas cuantitativas de Likert, del grado de conocimiento de las dimensiones e indicadores de maquetado web de los participantes, los cuales sirven para determinar si el maquetado será o no ampliamente aceptado por los usuarios finales (Szigeti, 2012). Este proceso consistió en la evaluación, por parte del investigador, del grado de conocimiento y habilidad de cada participante en cuanto a maquetado web, con lo cual se obtuvieron los datos del pre-test. La participación del investigador en la ejecución del pre-test fue con la finalidad de garantizar que los resultados obtenidos sean los más cercanos a la realidad.

5.1.2. Escenario 2: capacitación a los participantes

5.1.2.1. Capacitación sobre el uso del framework propuesto

En esta primera reunión de implementación del framework propuesto se capacitó también a los participantes sobre su uso indicando los conceptos básicos sobre los que se fundamenta su construcción, las actividades necesarias para su uso como son: el paso uno para cargar el archivo html a maquetar, el paso dos para importar una hoja de estilo anteriormente generada o para crear una nueva hoja desde cero, el paso tres donde se explicaron los comandos de voz reconocidos por el framework, como se muestra en el Anexo 6. Finalmente, se les mostró la página web piloto a ser maquetada en la segunda reunión virtual como medio de aplicación del framework, la cual se muestra en el Anexo 5.

5.1.3. Escenario 3: obtención de datos del post-test

En este tercer escenario finalizamos la capacitación sobre el funcionamiento del framework y se procedió a aplicarlo mediante la maquetación de la página web de prueba mostrada al final de la capacitación en el segundo escenario. Finalmente, solicitamos a cada participante llenar el cuestionario para obtener los datos del post-test.

5.1.3.1. Maquetado de una página web mediante el framework

Mediante el programa Zoom, procedimos a programar la segunda reunión virtual de implementación del framework propuesto, como se muestra en el Anexo 7. Es así que luego de finalizar la capacitación sobre el framework procedimos a maquetar la página web de prueba mediante el uso del framework como medio para corroborar su correcto funcionamiento y cumplimiento con las dimensiones e indicadores del cuestionario, como se muestra en el Anexo 8.

5.1.3.2. Aplicación del cuestionario para el post-test

Finalizado el maquetado de la página web piloto solicitamos a los participantes el llenado del cuestionario mediante el programa Google Forms, mostrado en los Anexos 10 y 11. Este cuestionario nos sirvió para obtener los datos del post-test y fue llenado por los participantes propiamente.

5.2. Resultados descriptivos

5.2.1. Maquetado de páginas web

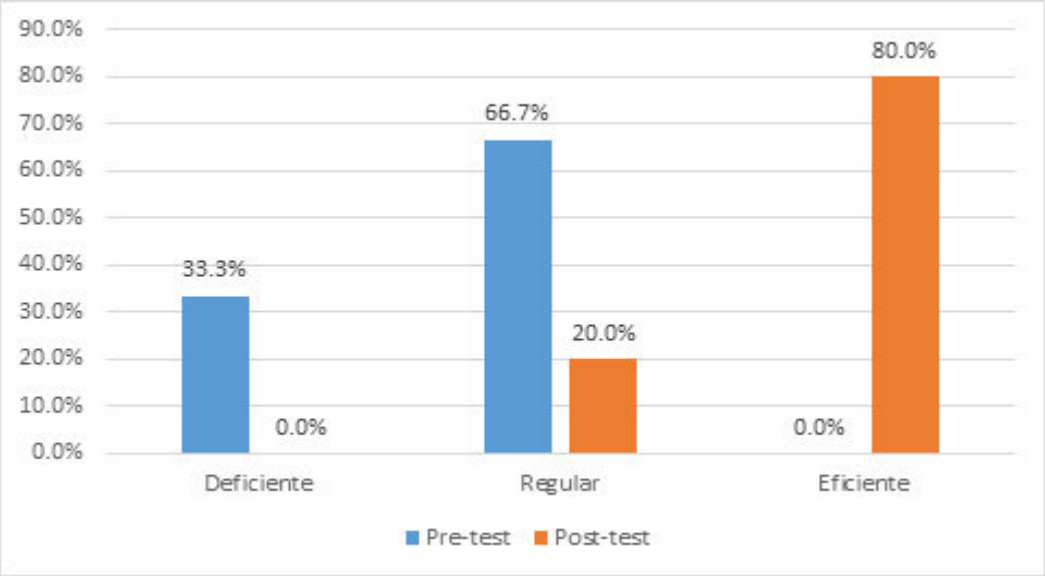


Figura 86 **Niveles del maquetado web en el pre-test y post-test.**

Fuente: Elaboración propia

Se observa que el 33.3% de los estudiantes de la escuela de Ingeniería de Sistemas e Informática de la UNASAM 2021, tuvieron niveles deficientes en la maquetación web antes del diseño e implementación del framework web CSS 3 con reconocimiento de voz, mientras que luego de la implementación los niveles disminuyeron a 0%. Por otro lado, el 66.7% tuvieron niveles regulares en el pre-test, mientras que en el post-test los niveles disminuyeron al 20%, representando una mejora de 46.7%.

Finalmente, no hubo estudiantes con niveles eficientes en el pre-test, mientras que, en el pos-test, los niveles de eficiencia alcanzaron un 80%.

5.2.2. Dimensión animación del maquetado de páginas web

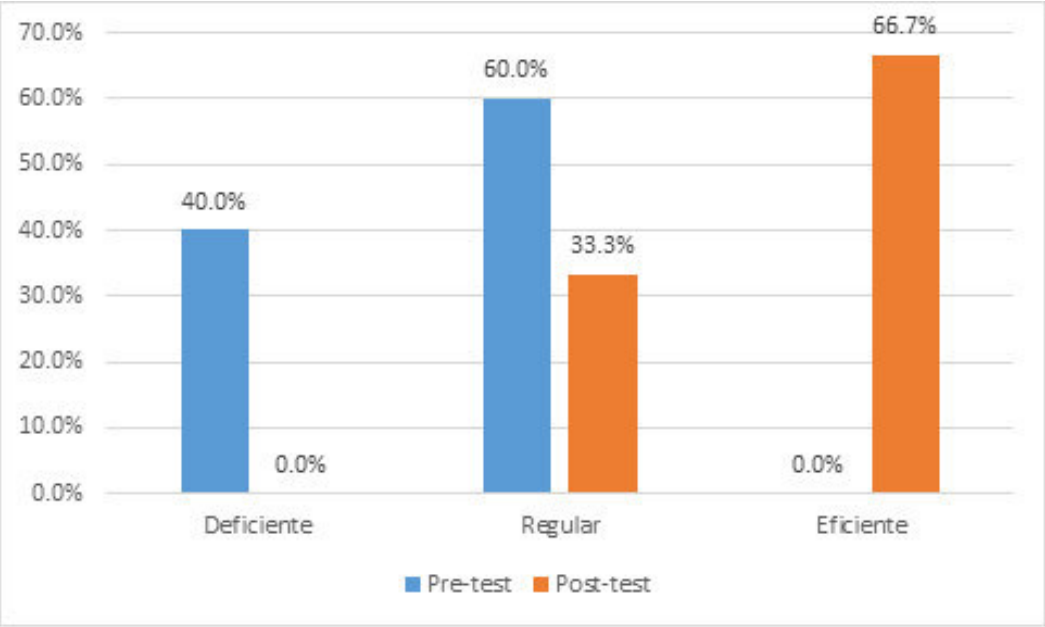


Figura 87 **Niveles de la dimensión animación del maquetado web en el pre-test y post-test.**

Fuente: Elaboración propia

Se observa que el 40% de los estudiantes de la escuela de Ingeniería de Sistemas e Informática de la UNASAM 2021, tuvieron niveles deficientes en la dimensión animación de la maquetación web antes del diseño e implementación del framework web CSS 3 con reconocimiento de voz, mientras que luego de la implementación los niveles disminuyeron a 0%, por otro lado, el 60% de los estudiantes tuvieron niveles regulares en el pre-test, mientras que en el post-test los niveles disminuyeron a 33.3%.

Finalmente, no hubo estudiantes con niveles eficientes en el pre-test, mientras que, en el pos-test, los niveles de los estudiantes de Ingeniería de Sistemas e Informática de la UNASAM, en el periodo 2021 aumentaron a un 66.7%.

Se puede afirmar que la dimensión animación de páginas web mejoró con el diseño e implementación del framework web CSS 3 con reconocimiento de voz.

5.2.3. Dimensión botones del maquetado de páginas web

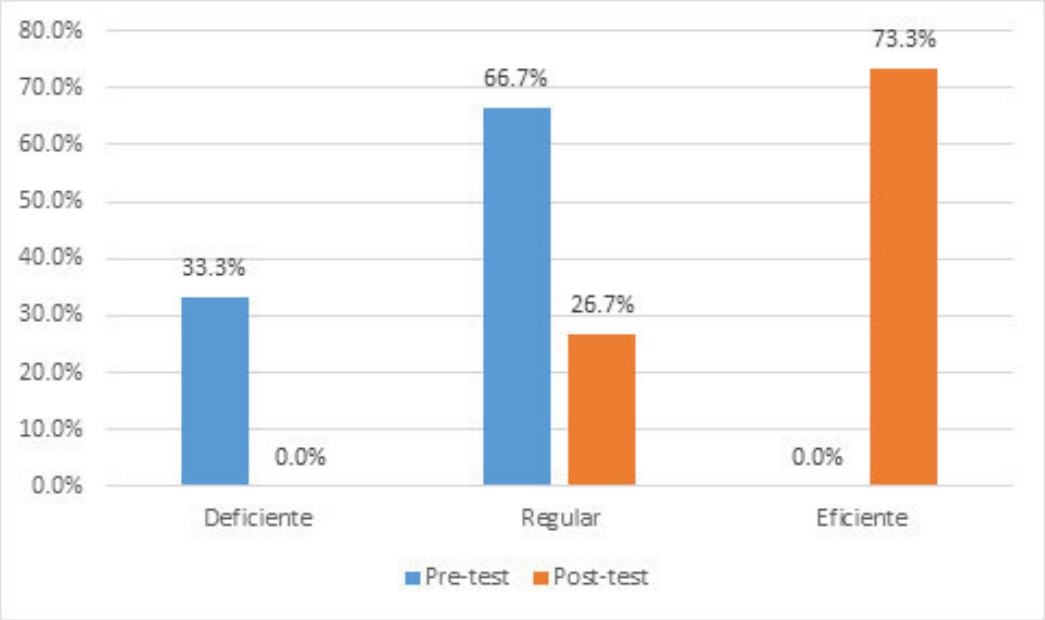


Figura 88 **Niveles de la dimensión botones del maquetado web en el pre-test y post-test.**

Fuente: Elaboración propia

Se observa que el 33.3% de los estudiantes de la escuela de Ingeniería de Sistemas e Informática de la UNASAM 2021, tuvieron niveles deficientes en la dimensión botones de la maquetación web antes del diseño e implementación del framework web CSS 3 con reconocimiento de voz, mientras que luego de la implementación los niveles disminuyeron a 0%, por otro lado, el 66.7% de los estudiantes tuvieron niveles regulares en el pre-test, mientras que en el post-test los niveles disminuyeron a 26.7%.

Finalmente, no hubo estudiantes con niveles eficientes en el pre-test, mientras que, en el pos-test, los niveles de los estudiantes de Ingeniería de Sistemas e Informática de la UNASAM, en el periodo 2021 aumentaron a un 73.3%.

Se puede afirmar que la dimensión botones de páginas web mejoró con el diseño e implementación del framework web CSS 3 con reconocimiento de voz.

5.2.4. Dimensión contenido del maquetado de páginas web

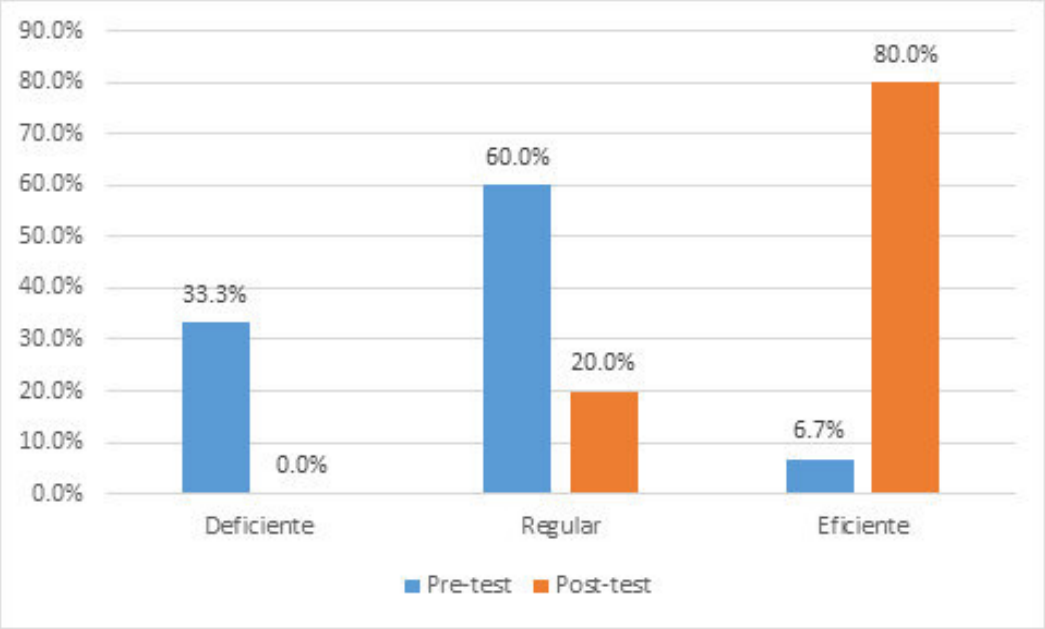


Figura 89 **Niveles de la dimensión contenido del maquetado web en el pre-test y post-test.**

Fuente: Elaboración propia

Se observa que el 33.3% de los estudiantes de la escuela de Ingeniería de Sistemas e Informática de la UNASAM 2021, tuvieron niveles deficientes en la dimensión contenido de la maquetación web antes del diseño e implementación del framework web CSS 3 con reconocimiento de voz, mientras que en con la implementación los niveles disminuyeron a 0%, por otro lado, el 60.0% de los estudiantes tuvieron niveles regulares en el pre-test, mientras que en el post-test los niveles disminuyeron a 20.0%.

Finalmente, el 6.7% de los estudiantes tuvieron niveles eficientes en el pre-test, mientras que, en el pos-test, los niveles de los estudiantes de Ingeniería de Sistemas de la UNASAM, en el periodo 2021 aumentaron a un 80.0%.

Se puede afirmar que la dimensión contenido de páginas web mejoró con el diseño e implementación del framework web CSS 3 con reconocimiento de voz.

5.2.5. Dimensión fondos del maquetado de páginas web

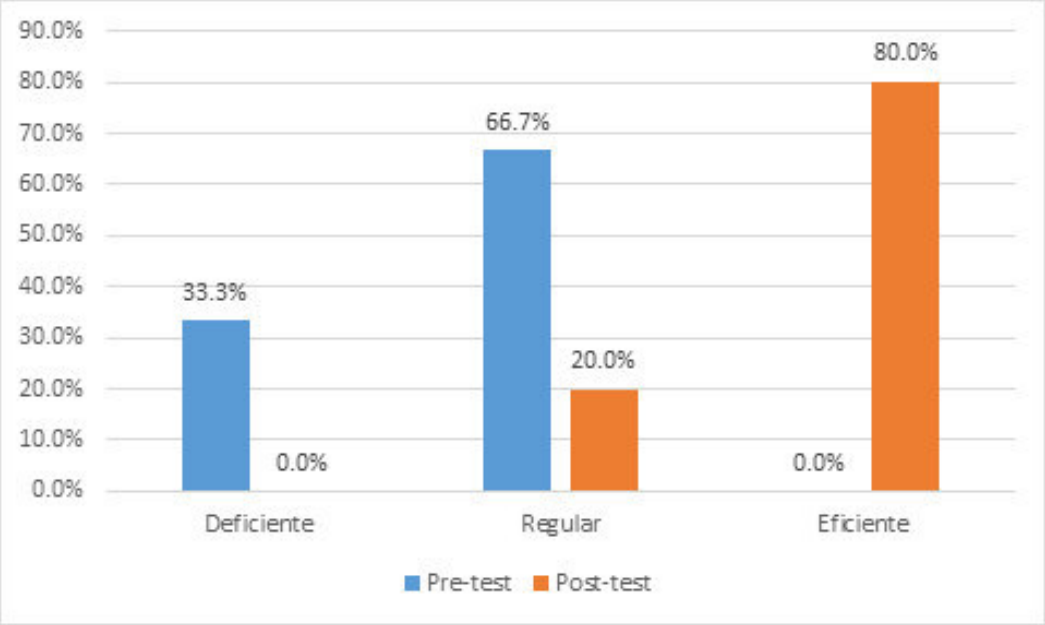


Figura 90 **Niveles de la dimensión fondo del maquetado web en el pre-test y post-test.**

Fuente: Elaboración propia

Se observa que el 33.3% de los estudiantes de la escuela de Ingeniería de Sistemas e Informática de la UNASAM 2021, tuvieron niveles deficientes en la dimensión fondo de la maquetación web antes del diseño e implementación del framework web CSS 3 con reconocimiento de voz, mientras que luego de la implementación los niveles disminuyeron a 0%, por otro lado, el 66.7% de los estudiantes tuvieron niveles regulares en el pre-test, mientras que en el post-test los niveles disminuyeron a 20.0%.

Finalmente, no hubo estudiantes con niveles eficientes en el pre-test, mientras que, en el pos-test, los niveles de los estudiantes de Ingeniería de Sistemas e Informática de la UNASAM, en el periodo 2021 aumentaron a un 80.0%.

Se puede afirmar que la dimensión fondos de páginas web mejoró con el diseño e implementación del framework web CSS 3 con reconocimiento de voz.

5.2.6. Dimensión imágenes del maquetado de páginas web

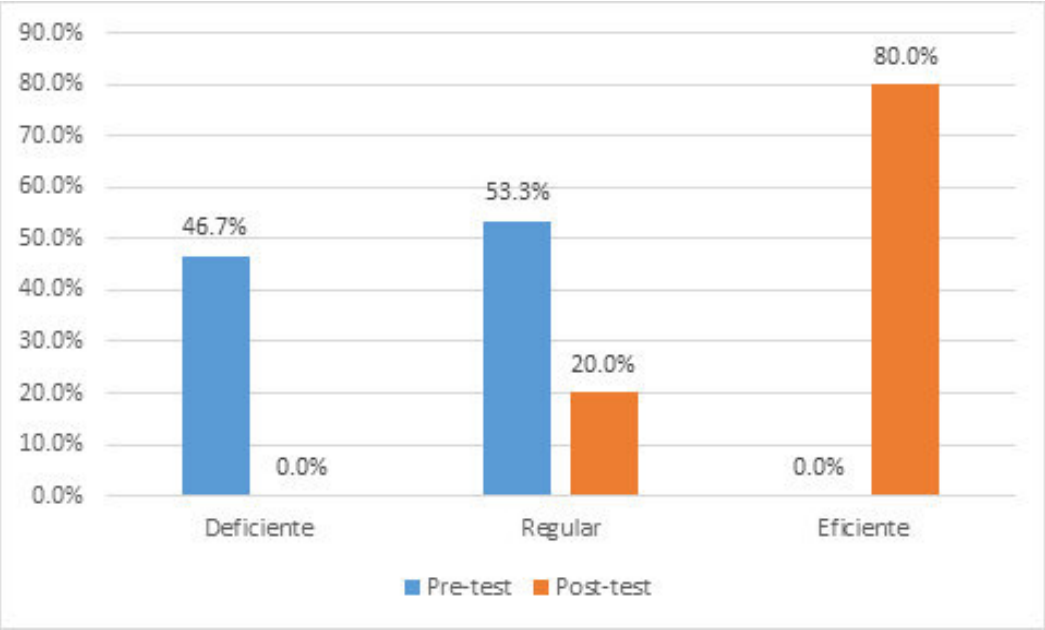


Figura 91 **Niveles de la dimensión imágenes del maquetado web en el pre-test y post-test.**

Fuente: Elaboración propia

Se observa que el 46.7% de los estudiantes de la escuela de Ingeniería de Sistemas e Informática de la UNASAM 2021, tuvieron niveles deficientes en la dimensión imágenes de la maquetación web antes del diseño e implementación del framework web CSS 3 con reconocimiento de voz, mientras que con la implementación de la propuesta los niveles disminuyeron a 0%, por otro lado, el 53.3% de los estudiantes tuvieron niveles regulares en el pre-test, mientras que en el post-test los niveles disminuyeron a 20.0%.

Finalmente, no hubo estudiantes con niveles eficientes en el pre-test, mientras que, en el pos-test, los niveles de los estudiantes de Ingeniería de Sistemas e Informática de la UNASAM, en el periodo 2021 aumentaron a un 80.0%.

Se puede afirmar que la dimensión imágenes de páginas web mejoró con el diseño e implementación del framework web CSS 3 con reconocimiento de voz.

5.2.7. Dimensión tipografías del maquetado de páginas web

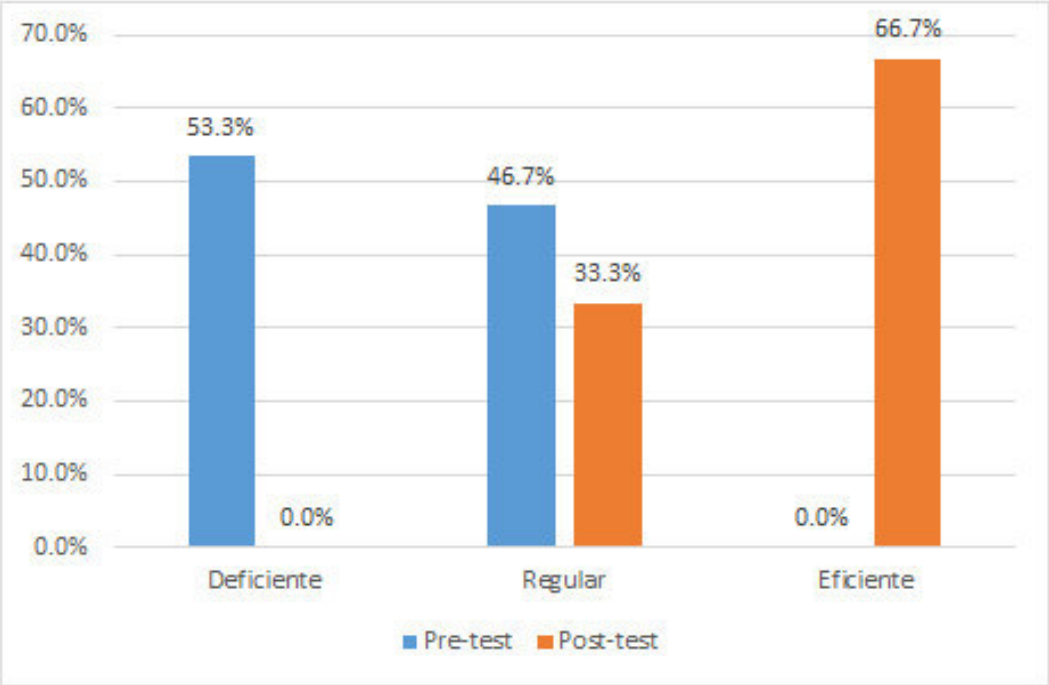


Figura 92 **Niveles de la dimensión tipografías del maquetado web en el pre-test y post-test.**

Fuente: Elaboración propia

Se observa que el 53.3% de los estudiantes de la escuela de Ingeniería de Sistemas e Informática de la UNASAM 2021, tuvieron niveles deficientes en la dimensión tipografías de la maquetación web antes del diseño e implementación del framework web CSS 3 con reconocimiento de voz, mientras que con la implementación de la propuesta de generación del framework, los niveles disminuyeron a 0%, por otro lado, el 46.7% de los estudiantes tuvieron niveles regulares en el pre-test, mientras que en el post-test los niveles disminuyeron a 33.3%.

Finalmente, no hubo estudiantes con niveles eficientes en el pre-test, mientras que, en el pos-test, los niveles de los estudiantes de Ingeniería de Sistemas e Informática de la UNASAM, en el periodo 2021 aumentaron a un 66.7%.

Se puede afirmar que la dimensión tipografías de páginas web mejoró con el diseño e implementación del framework web CSS 3 con reconocimiento de voz.

5.3. Resultados según dimensiones de maquetado web

El proceso de generación de los valores de las medias en el pre-test y post-test forma parte del proceso de análisis de datos que se dio luego de la aplicación del post-test, en donde se analizaron las 6 dimensiones de la variable dependiente (maquetado de páginas web) distribuidos en 31 indicadores con escala tipo Likert. Es así que la generación de las medias en pre-test y post-test es el resultado de la suma de los valores asignados a cada indicador correspondiente a cada dimensión, para luego ser dividido entre 15, por ser el número de participantes elegidos para esta investigación.

Así, por ejemplo, la dimensión animación cuenta con 4 indicadores, los cuales han sido valorizados por cada participante. La suma de los valores asignados a cada indicador de esta dimensión, por cada uno de los 15 participantes en el post-test, resulta ser 246 (16, 18, 15, 20, 12, 12, 19, 20, 17, 20, 12, 16, 12, 20, 17) y, dividido entre 15, finalmente da como resultado 16.4. Por otro lado, la dimensión botones cuenta con 5 indicadores, los cuales sumando sus valores en el post-test resulta 306 (20, 21, 20, 25, 15, 15, 24, 25, 22, 25, 15, 20, 15, 25, 19) y, dividido entre 15, resulta 20.4. De la misma forma se obtienen las medias de cada una de las dimensiones restantes y, finalmente, de la variable dependiente maquetado de páginas web en el pre-test y post-test.

Por otro lado, para la correcta interpretación de las tablas que contienen las medias y las pruebas T de Student, se describirá el contenido de cada columna. Entonces, para el caso de las tablas que contienen el valor de las medias, la columna **"N°"** corresponde a la cantidad de participantes; la columna **"media"** corresponde a la media cuyo cálculo se ha explicado en el párrafo anterior; la columna **"desviación estándar"** resulta del promedio de las diferencias del valor medio entre cada valor, lo cual indica

que en el intervalo entre la media +/- la desviación estándar se halla la mayor cantidad de datos; la columna **"media de error estándar"** corresponde a la media de los errores estándar, que corresponde al promedio de las diferencias entre la media con cada valor.

Para el caso de las tablas que contienen los resultados de las pruebas T de Student, la columna **"F"** corresponde a la prueba F de Snedecor y determina si la variación de los puntajes en el pre-test y post-test son iguales; la columna **"Sig."** indica que al ser un valor mayor al nivel 0.05, se acepta la prueba de hipótesis de igualdad de varianzas, por lo que se puede afirmar que las variaciones de los puntajes de una determinada dimensión en el pre-test y post-test son similares a un 95% de confianza; la columna **"gl"** muestra los grados de libertad de la prueba; la columna **"Sig. (bilateral)"** se observa la significancia de la prueba T para determinar si existe diferencias entre los puntajes promedios de cada dimensión en el pre-test y post-test, en este caso se rechaza la hipótesis de igualdad de medias (entre el pre-test y post-test), por lo que se puede afirmar que el puntaje promedio del pre-test es diferente al puntaje promedio del post-test; la columna **"Dif. de medias"** muestra la diferencia entre los promedios de los puntajes de una determinada dimensión en el pre-test y post-test; la columna **"Dif. de error estándar"** muestra la diferencia del error estándar, que es la diferencia de la desviación estándar de los puntajes de una determinada dimensión en el pre-test con el post-test.

5.3.1. Resultados para el objetivo general

Determinar de qué manera desarrollo de un framework web CSS 3 basado en reconocimiento de voz influye en el maquetado de páginas web, en la Escuela de Ingeniería de Sistemas e Informática de la UNASAM, 2021.

Tabla 49 **Estadísticas descriptivas de los puntajes del maquetado web en el pre-test y post-test**

Test		N°	Media	Desviación estándar	Media de error estándar
Maquetado web	Pre-test	15	77.53	8.262	2.133
	Post-test	15	126.27	23.426	6.049

Fuente. Elaboración propia

Se observa que la media de los puntajes de los estudiantes de la Escuela de Ingeniería de Sistemas e Informática de la UNASAM 2021, con respecto al maquetado web aplicando reconocimiento de voz, en el pre-test fue de 77.53, mientras que en el post-test aumentó a 126.27.

Tabla 50 **Prueba t de igualdad de medias de los puntajes del maquetado web en el pre-test y post-test**

	Prueba de Levene de igualdad de varianzas		**Prueba T para igualdad de medias**				
	F	**Sig.**	**t**	**gl**	**Sig. (bilateral)**	**Dif. de medias**	**Dif. de error estándar**
Se asumen varianzas iguales	9.356	0.005	-7.598	28	0.000	-48.733	6.414
No se asumen varianzas iguales			-7.598	17.430	0.000	-48.733	6.414

Fuente. Elaboración propia

Se observa que, según la prueba T de Student para comparar los promedios de los puntajes del maquetado web en el pre-test y el post-test, se evidenció que existe diferencia significativa entre ambas pruebas, (sig, bilateral de 0.00 es menor al 0.05), por lo que se puede afirmar que los puntajes del post-test del maquetado web son superiores, a un 95% de confianza.

5.3.2. Resultados para las dimensiones de maquetado web

5.3.2.1. Maquetado de animaciones

Determinar de qué manera el diseño e implementación de un framework web CSS 3 con reconocimiento de voz influye en el maquetado de animaciones de páginas web, en la Escuela de Ingeniería de Sistemas e Informática de la UNASAM, 2021.

Tabla 51 **Estadísticas descriptivas de los puntajes de la dimensión animación del maquetado web en el pre-test y post-test**

Test		N°	Media	Desviación estándar	Media de error estándar
Dimensión Animación	Pre-test	15	9.60	1.454	0.375
	Post-test	15	16.40	3.180	0.821

Fuente. Elaboración propia

Se observa que los puntajes de los estudiantes de la Escuela de Ingeniería de Sistemas e Informática de la UNASAM 2021, con respecto a la dimensión animación del maquetado web en el pre-test fue de 9.60, mientras que en el post-test aumentó a 16.40.

Tabla 52 **Prueba t de igualdad de medias de los puntajes de la dimensión animación del maquetado web en el pre-test y post-test**

	Prueba de Levene de igualdad de varianzas		Prueba T para igualdad de medias				
	F	Sig.	t	gl	Sig. (bilateral)	Dif. de medias	Dif. de error estándar
Se asumen varianzas iguales	9.978	0.004	-7.531	28	0.000	-6.800	0.903
No se asumen varianzas iguales			-7.531	19.608	0.000	-6.800	0.903

Fuente. Elaboración propia

Se observa que, según la prueba T de Student para comparar los promedios de los puntajes de la dimensión animación del maquetado web en el pre-test y el post-test, se evidenció que existe diferencia significativa entre ambas pruebas, (sig, bilateral de 0.00 es menor al 0.05), por lo que se puede afirmar que los puntajes del post-test de la dimensión animación del maquetado web son superiores, a un 95% de confianza.

5.3.2.2. Maquetado de botones

Determinar de qué manera el diseño e implementación de un framework web CSS 3 con reconocimiento de voz influye en el maquetado de botones de páginas web, en la Escuela de Ingeniería de Sistemas e Informática de la UNASAM, 2021.

Tabla 53 **Estadísticas descriptivas de los puntajes de la dimensión botones del maquetado web en el pre-test y post-test**

Test		N°	Media	Desviación estándar	Media de error estándar
Dimensión Botones	Pre-test	15	12.40	2.384	0.616
	Post-test	15	20.40	3.961	1.023

Fuente. Elaboración propia

Se observa que los puntajes de los estudiantes de la Escuela de Ingeniería de Sistemas e Informática de la UNASAM 2021, con respecto a la dimensión botones del maquetado web en el pre-test fue de 12.40, mientras que en el post-test aumentó a 20.40.

Tabla 54 **Prueba t de igualdad de medias de los puntajes de la dimensión botones del maquetado web en el pre-test y post-test**

	Prueba de Levene de igualdad de varianzas		Prueba T para igualdad de medias				
	F	Sig.	t	gl	Sig. (bilateral)	Dif. de medias	Dif. de error estándar
Se asumen varianzas iguales	3.898	0.58	-6.702	28	0.000	-8.000	1.194
No se asumen varianzas iguales			-6.702	22.971	0.000	-8.000	1.194

Fuente. Elaboración propia

Se observa que, según la prueba T de Student para comparar los promedios de los puntajes de la dimensión botones del maquetado web en el pre-test y el post-test, se evidenció que existe diferencia significativa entre ambas pruebas, (sig, bilateral de 0.00 es menor al 0.05), por lo que se puede afirmar que los puntajes del post-test de la dimensión botones del maquetado web son superiores, a un 95% de confianza.

5.3.2.3. Maquetado de contenido

Determinar de qué manera el diseño e implementación de un framework web CSS 3 con reconocimiento de voz influye en el maquetado de contenido de páginas web, en la Escuela de Ingeniería de Sistemas e Informática de la UNASAM, 2021.

Tabla 55 **Estadísticas descriptivas de los puntajes de la dimensión contenido del maquetado web en el pre-test y post-test**

Test		N°	Media	Desviación estándar	Media de error estándar
Dimensión Contenido	Pre-test	15	16.60	3.291	0.850
	Post-test	15	24.53	4.533	1.171

Fuente. Elaboración propia

Se observa que los puntajes de los estudiantes de la Escuela de Ingeniería de Sistemas e Informática de la UNASAM 2021, con respecto a la dimensión contenido del maquetado web en el pre-test fue de 16.60, mientras que en el post-test aumentó a 24.53.

Tabla 56 **Prueba t de igualdad de medias de los puntajes de la dimensión contenido del maquetado web en el pre-test y post-test**

	Prueba de Levene de igualdad de varianzas		Prueba T para igualdad de medias				
	F	Sig.	t	gl	Sig. (bilateral)	Dif. de medias	Dif. de error estándar

Se asumen varianzas iguales	0.965	0.334	-5.485	28	0.000	-7.933	1.446
No se asumen varianzas iguales			-5.485	25.547	0.000	-7.933	1.446

Fuente. Elaboración propia

Se observa que, según la prueba T de Student para comparar los promedios de los puntajes de la dimensión contenido del maquetado web en el pre-test y el post-test, se evidenció que existe diferencia significativa entre ambas pruebas, (sig, bilateral de 0.00 es menor al 0.05), por lo que se puede afirmar que los puntajes del post-test de la dimensión contenido del maquetado web son superiores, a un 95% de confianza.

5.3.2.4. Maquetado de fondos

Determinar de qué manera el diseño e implementación de un framework web CSS 3 con reconocimiento de voz influye en el maquetado de fondos de páginas web, en la Escuela de Ingeniería de Sistemas e Informática de la UNASAM, 2021.

Tabla 57 **Estadísticas descriptivas de los puntajes de la dimensión fondo del maquetado web en el pre-test y post-test**

Test		N°	Media	Desviación estándar	Media de error estándar
Dimensión Fondo	Pre-test	15	12.33	2.059	0.532
	Post-test	15	20.53	3.623	0.935

Fuente. Elaboración propia

Se observa que los puntajes los estudiantes de la Escuela de Ingeniería de Sistemas e Informática de la UNASAM 2021, con respecto a la dimensión fondo del maquetado web en el pre-test fue de 12.33, mientras que el post-test aumentó a 20.53.

Tabla 58 **Prueba t de igualdad de medias de los puntajes de la dimensión fondo del maquetado web en el pre-test y post-test**

	Prueba de Levene de igualdad de varianzas		Prueba T para igualdad de medias				
	F	**Sig.**	**t**	**gl**	**Sig. (bilateral)**	**Dif. de medias**	**Dif. de error estándar**
Se asumen varianzas iguales	3.466	0.073	-7.622	28	0.000	-8.200	1.076
No se asumen varianzas iguales			-7.622	22.188	0.000	-8.200	1.076

Fuente. Elaboración propia

Se observa que, según la prueba T de Student para comparar los promedios de los puntajes de la dimensión fondo del maquetado web en el pre-test y el post-test, se evidenció que existe diferencia significativa entre ambas pruebas, (sig, bilateral de 0.00 es menor al 0.05), por lo que se puede afirmar que los puntajes del post-test de la dimensión fondo del maquetado web son superiores, a un 95% de confianza.

5.3.2.5. Maquetado de imágenes

Determinar de qué manera el diseño e implementación de un framework web CSS 3 con reconocimiento de voz influye en el maquetado de imágenes de páginas web, en la Escuela de Ingeniería de Sistemas e Informática de la UNASAM, 2021.

Tabla 59 **Estadísticas descriptivas de los puntajes de la dimensión imágenes del maquetado web en el pre-test y post-test**

Test		N°	Media	Desviación estándar	Media de error estándar
	Pre-test	15	12.07	2.840	0.733

Dimensión Imagen	Post-test	15	20.47	4.121	1.064

Fuente. Elaboración propia

Se observa que de los puntajes los estudiantes de la Escuela de Ingeniería de Sistemas e Informática de la UNASAM 2021, con respecto a la dimensión imágenes del maquetado web en el pre-test fue de 12.07, mientras que en el post-test aumentó a 20.47.

Tabla 60 **Prueba t de igualdad de medias de los puntajes de la dimensión imágenes del maquetado web en el pre-test y post-test**

	Prueba de Levene de igualdad de varianzas		**Prueba T para igualdad de medias**				
	F	**Sig.**	**t**	**gl**	**Sig. (bilateral)**	**Dif. de medias**	**Dif. de error estándar**
Se asumen varianzas iguales	1.453	0.238	-6.500	28	0.000	-8.400	1.292
No se asumen varianzas iguales			-6.500	24.852	0.000	-8.400	1.292

Fuente. Elaboración propia

Se observa que, según la prueba T de Student para comparar los promedios de los puntajes de la dimensión imágenes del maquetado web en el pre-test y el post-test, se evidenció que existe diferencia significativa entre ambas pruebas, (sig, bilateral de 0.00 es menor al 0.05), por lo que se puede afirmar que los puntajes del post-test de la dimensión imágenes del maquetado web son superiores, a un 95% de confianza.

5.3.2.6. Maquetado de tipografías

Determinar de qué manera el diseño e implementación de un framework web CSS 3 con reconocimiento de voz influye en el maquetado de tipografías de páginas web, en la Escuela de Ingeniería de Sistemas e Informática de la UNASAM, 2021.

Tabla 61 **Estadísticas descriptivas de los puntajes de la dimensión tipografías del maquetado web en el pre-test y post-test**

Test		N°	Media	Desviación estándar	Media de error estándar
Dimensión Tipografía	Pre-test	15	14.53	2.532	0.654
	Post-test	15	23.93	4.698	1.213

Fuente. Elaboración propia

Se observa que los puntajes los estudiantes de la Escuela de Ingeniería de Sistemas e Informática de la UNASAM 2021, con respecto a la dimensión tipografías del maquetado web en el pre-test fue de 14.53, mientras que en el post-test aumentó a 23.93.

Tabla 62 **Prueba t de igualdad de medias de los puntajes de la dimensión tipografías del maquetado web en el pre-test y post-test**

	Prueba de Levene de igualdad de varianzas		Prueba T para igualdad de medias				
	F	Sig.	t	gl	Sig. (bilateral)	Dif. de medias	Dif. de error estándar
Se asumen varianzas iguales	2.859	0.102	-6.822	28	0.000	-9.400	1.378
No se asumen varianzas iguales			-6.822	21.500	0.000	-9.400	1.378

Fuente. Elaboración propia

Se observa que, según la prueba T de Student para comparar los promedios de los puntajes de la dimensión tipografía del maquetado web en el pre-test y el post-test, se evidenció que existe diferencia significativa entre ambas pruebas, (sig, bilateral de 0.00 es menor al 0.05), por lo que se puede afirmar que los puntajes del post-test de la dimensión tipografías del maquetado web son superiores, a un 95% de confianza.

CAPÍTULO VI: DISCUSIÓN

El framework web propuesto en esta tesis de maestría tiene como finalidad hacer posible la automatización del proceso de maquetado web aplicando reconocimiento de voz mediante las tecnologías: CSS 3 para el maquetado web y Web Speech API para el reconocimiento de voz. A continuación, realizaremos una discusión de cada resultado obtenido comparándolos con los del marco teórico.

6.1. Discusión para el objetivo general

El framework web desarrollado por (Jaimez & Vargas, 2017), tuvo como objetivo la generación automatizada de código HTML, CSS y JavaScript mediante la construcción de páginas y sistemas web dentro de un entorno gráfico, compuesto por los módulos, del lado del cliente, encargados de la interfaz gráfica de usuario y, del lado del servidor, del módulo generador de código para los lenguajes mencionados dentro de archivos .html, .css y .js, respectivamente. Este framework permitió al usuario seleccionar, colocar y configurar todos los atributos de un conjunto de elementos HTML sobre el espacio de trabajo, sobre los cuales fue posible aplicar estilos CSS y realizar validaciones JavaScript sobre aquellos elementos de entrada de datos, recibiendo por ello la denominación de framework WYSYWIG. A diferencia del framework mencionado, el framework web propuesto en la presente tesis de investigación incorpora reconocimiento de voz, aunque sólo se ocupa de gestionar todos los elementos de la tecnología CSS 3: selectores, declaraciones, reglas-AT, descriptores, propiedades y valores, así como de la generación automatizada de hojas de estilo permitiendo su descarga y posterior edición dentro del propio del framework o a través de cualquier sistema editor de preferencia del usuario. No obstante, se recomienda como trabajo futuro la implementación de comandos de voz para la gestión y generación automatizada de archivos HTML y JavaScript.

En la investigación realizada por (Hernández & Lemuz, 2016) se desarrolla un sistema web compuesto por cuatro módulos: Reading, Listening, Writing y Speaking, con la

finalidad de mejorar el nivel de capacidad de los participantes interesados en rendir los exámenes de certificación TOEFL correspondiente al dominio del idioma inglés, donde se utilizó la tecnología de síntesis de voz para convertir a audio las respuestas de las evaluaciones para cada nivel del examen y la tecnología de reconocimiento de voz como apoyo principal al módulo Speaking por el uso de la tecnología de reconocimiento de voz a través de comandos de voz que obedecieron a un vocabulario cerrado cuya gramática estuvo relacionada a cada una de las respuestas en una determinada evaluación. En comparación con el sistema de mejora del dominio del idioma inglés, el framework propuesto en esta tesis de maestría, se ha enfocado en fortalecer el conocimiento de los profesionales en maquetado de páginas web guiándolos durante todo el proceso de maquetado listando todos los elementos propios de una hoja estilo CSS 3: selectores, declaraciones, reglas-AT, descriptores, propiedades, valores y comentarios, permitiéndoles su manejo a disposición, mediante la pronunciación completa o abreviada de los comandos relacionados a cada uno de estos elementos. Por otro lado, el diseño estético del framework web propuesto en esta tesis de maestría, al igual que el sistema web de mejora del dominio inglés, ha tenido una consideración especial centrándose en un diseño sencillo, estético y adaptable para diferentes dispositivos: computador, table o smartphone que cuente con un navegador web y una conexión a internet, utilizando las tecnologías: HTML, CSS, JavaScript y Web Speech API para la gestión del reconocimiento de voz.

En comparación con el sistema de computadora desarrollada para el control de una silla de ruedas mediante instrucciones de voz propuesto por (Gil, Castillo, & Flórez, 2016), en la cual los comandos son un total de treinta y siete clasificados en las tres categorías siguientes: movimiento, domótica y salud, el framework propuesto en esta tesis consta de trescientos cuarenta y nueve comandos clasificados en seis categorías: comandos propiamente, comandos auxiliares, comandos de selector, comandos de declaración, comandos de reglas-AT, comandos de descriptores. Por otro lado, una de las características importantes de este sistema de computadora es que ha sido desarrollada utilizando: C# como lenguaje de programación, Microsoft SAPI (Interfaz de Programación de Aplicaciones de Voz) y Microsoft .NET Framework para el reconocimiento del habla en idioma español. No obstante, el framework web propuesto en esta tesis ha sido desarrollado con las tecnologías: HTML 5, CSS 3 y JavaScript para la interfaz gráfica de usuario y, Web Speech API para la gestión del

reconocimiento de voz, las cuales son multiplataforma, funcionando en entornos de escritorio o móviles, independientemente del sistema operativo del dispositivo con el cual el usuario acceda al framework.

En comparación con el Asistente de voz basado en web propuesto por (Collado & Aparicio, 2017), el cual permite al usuario realizar las funciones siguientes, clasificadas en cuatro categorías desarrolladas siguiendo la metodología ágil SCRUM, las cuales son: Reconocimiento de voz, Sintetizador de voz, Consulta de búsqueda y Secuencia de palabras clave, el framework web CSS 3 propuesto en esta tesis, ha sido desarrollado también siguiendo la metodología ágil SCRUM, pero implementa ocho historias de usuario: Cargar archivo html, Importar archivo xml, Crear hoja de estilo CSS 3, Pronunciación de comandos de voz, Ocultar framework, Mostrar framework, Exportar archivos .xml / .css y Descargar código fuente. Por otro lado, el Asistente de voz incorpora seis frases reservadas las cuales tienen la capacidad de realizar una acción en particular, pertenecientes al vocabulario cerrado reconocido por el asistente; no obstante, el framework propuesto incorpora veintidós frases reservadas que tienen la capacidad de realizar acciones sobre los elementos CSS 3 de la hoja de estilo además de modificar el aspecto de la interfaz gráfica de usuario.

La investigación realizada por (Pilamunga, 2012) tuvo como finalidad demostrar que la implementación de un sitio web mediante CSS y JavaScript tiene una incidencia positiva sobre la optimización de su tamaño y funcionalidad, determinándose que una correcta maquetación es aquella que usa las normas establecidas por la W3C junto con el uso de scripts resulta primordial para reducir el tiempo de carga, unificar el código de diseño, dividir elementos del contenido de los de la presentación y optimizar el ancho de banda de la conexión. El framework propuesto en la presente tesis de maestría considera todas las ventajas mencionadas y las pone en práctica añadiendo la posibilidad de gestionar la tecnología CSS 3 mediante el reconocimiento de voz a través de los comandos de voz.

6.2. Discusión para las dimensiones de maquetado web

6.2.1. Discusión del maquetado de animaciones

La incorporación de la regla-AT keyframes dentro del framework muestra al diseñador web todas las propiedades y valores posibles de ser utilizadas en la creación de animaciones sobre cualquier elemento dentro de una página web, permitiéndole conocer y disponer a voluntad de todas sus propiedades y valores mejorando de esa manera la capacidad creativa en el diseño de animaciones web. Por otro lado, la gestión de estas propiedades y valores a través de comandos de voz soportados por el framework, representa una innovación con respecto a la programación tradicional.

Los resultados obtenidos indican que el 40% de los participantes lograron un nivel deficiente en la dimensión animación del maquetado web en el pre-test pero luego de la capacitación e implementación del framework propuesto este porcentaje de participantes desapareció; el 60% lograron un nivel regular en el pre-test pero en el post-test este porcentaje disminuyó a 33.3% lo que representó un porcentaje de mejora de 26.7%; ningún participante alcanzó el nivel eficiente de maquetado web en el pre-test pero en el post-test este nivel alcanzó un porcentaje de 66.7%.

6.2.2. Discusión del maquetado de botones

La incorporación de todas las propiedades y valores posibles de ser utilizados para la maquetación de botones, como son principalmente las propiedades: background, color, margin, padding, border y font, permiten al diseñador web conocer y disponer a voluntad de todas las propiedades y valores mejorando así su capacidad creativa en el maquetado de botones en páginas web. Por otro lado, la gestión de estos elementos a través de comandos de voz soportados por el framework, representa una innovación con respecto a la programación tradicional.

Los resultados obtenidos indican que el 33.3% de los participantes lograron un nivel deficiente en la dimensión botones del maquetado web en el pre-test pero luego de la capacitación e implementación del framework propuesto este porcentaje de participantes desapareció; el 66.7% lograron un nivel regular en el pre-test pero en el

post-test este porcentaje disminuyó a 26.7% lo que representó un porcentaje de mejora de 40%; ningún participante alcanzó el nivel eficiente de maquetado web en el pre-test pero en el post-test este nivel alcanzó un porcentaje de 73.3%.

6.2.3. Discusión del maquetado de contenido

La incorporación de las reglas-AT: media, utilizada para asignar propiedades a los elementos de una página web según las características del dispositivo donde se desplegará; document, utilizada para restringir la asignación de propiedades a los elementos de la página web según la url o dominio donde se encuentren desplegados; page, utilizada para modificar las propiedades de los elementos de la página web sólo al momento de impresión; viewport, utilizada para configurar la pantalla del dispositivo donde se desplegará la página web; counter-style, utilizada para configurar estilos adicionales a los establecidos para los elementos de las listas de una página web; supports, utilizada para verificar si la asignación de las propiedades a los elementos de la página web son permitidos por el navegador o no; font-face, utilizada para seleccionar distintas fuentes usadas dentro de la página web las cuales pueden estar ubicadas en cualquier origen; font-feature-values, utilizada para renombrar glifos alternos utilizados para caracteres. Además, la incorporación de todas las propiedades y valores posibles de ser utilizados adicionalmente para la maquetación de contenido, el cual está constituido principalmente por textos, imágenes, videos, botones, menús, links, y otros, así como la organización de estos elementos, permiten al diseñador web conocer y disponer a voluntad de todas estas propiedades y valores mejorando así su capacidad creativa en el maquetado de contenido de páginas web. Además, la gestión de todos estos elementos a través de comandos de voz soportados por el framework, representa una innovación con respecto a la programación tradicional.

Los resultados obtenidos indican que el 33.3% de los participantes lograron un nivel deficiente en la dimensión contenido del maquetado web en el pre-test pero luego de la capacitación e implementación del framework propuesto este porcentaje de participantes desapareció; el 60% lograron un nivel regular en el pre-test pero en el post-test este porcentaje disminuyó a 20% lo que representó un porcentaje de mejora de 40%; el 6.7% de los participantes alcanzó el nivel eficiente de maquetado web en el pre-test pero en el post-test este nivel alcanzó un porcentaje de 80%.

6.2.4. Discusión del maquetado de fondos

La incorporación de la propiedad background y sus derivados posibles de ser utilizados para maquetar el fondo de los elementos de las páginas web, como son principalmente: el fondo de botones, imágenes, textos, el fondo de la página propiamente o el fondo de elementos contenedores, permite al diseñador web conocer y disponer a voluntad de todas estas propiedades y valores posibles de ser aplicados, mejorando así su creatividad creativa en el maquetado de fondos de páginas web. De la misma forma, la gestión de los elementos a través de comandos de voz soportados por el framework, representa una innovación con respecto a la programación tradicional.

Los resultados obtenidos indican que el 33.3% de los participantes lograron un nivel deficiente en la dimensión fondo del maquetado web en el pre-test pero luego de la capacitación e implementación del framework propuesto este porcentaje de participantes desapareció; el 66.7% lograron un nivel regular en el pre-test pero en el post-test este porcentaje disminuyó a 20% lo que representó un porcentaje de mejora de 46.7%; ningún participante alcanzó el nivel eficiente de maquetado web en el pre-test pero en el post-test este nivel alcanzó un porcentaje de 80%.

6.2.5. Discusión del maquetado de imágenes

La incorporación de todas las propiedades y valores posibles de ser utilizados para la maquetación de imágenes, como son principalmente: background, color, margin, padding, border y filter, permiten al diseñador web conocer y disponer a voluntad de todas estas propiedades y valores mejorando así su capacidad creativa en el maquetado de imágenes en páginas web. De la misma manera, la gestión de los elementos a través de comandos de voz soportados por el framework, representa una innovación con respecto a la programación tradicional.

Los resultados obtenidos indican que el 46.7% de los participantes lograron un nivel deficiente en la dimensión imágenes del maquetado web en el pre-test pero luego de la capacitación e implementación del framework propuesto este porcentaje de participantes desapareció; el 53.3% lograron un nivel regular en el pre-test pero en el post-test este porcentaje disminuyó a 20% lo que representó un porcentaje de mejora

de 33.3%; ningún participante alcanzó el nivel eficiente de maquetado web en el pre-test pero en el post-test este nivel alcanzó un porcentaje de 80%.

6.2.6. Discusión del maquetado de tipografías

La incorporación de las reglas-AT font-face y font-feature-values, además de todas las propiedades y valores posibles de ser aplicados en el maquetado de tipografías, como es principalmente la propiedad font y sus derivados, sobre cualquier texto dentro de una página web, permite al diseñador web conocer y disponer a voluntad de todas estas propiedades y valores mejorando así su capacidad creativa en el maquetado de tipografías web. Por otro lado, la gestión de las propiedades y valores a través de comandos de voz soportados por el framework, representa una innovación con respecto a la programación tradicional.

Los resultados obtenidos indican que el 46.7% de los participantes lograron un nivel deficiente en la dimensión tipografía del maquetado web en el pre-test pero luego de la capacitación e implementación del framework propuesto este porcentaje de participantes desapareció; el 53.3% lograron un nivel regular en el pre-test pero en el post-test este porcentaje disminuyó a 20% lo que representó un porcentaje de mejora de 33.3%; ningún participante alcanzó el nivel eficiente de maquetado web en el pre-test pero en el post-test este nivel alcanzó un porcentaje de 80%.

CAPÍTULO VII: CONCLUSIONES Y TRABAJOS FUTUROS

En este capítulo se describen las conclusiones obtenidas en esta tesis de maestría, así como los trabajos futuros posibles de ser realizados como continuación de la presente investigación. Las conclusiones han sido clasificadas en conclusiones generales y conclusiones específicas, según los objetivos generales y específicos planteados en esta tesis.

7.1. Conclusiones

7.1.1. Conclusión general

El desarrollo de un framework web CSS 3 basado en reconocimiento de voz influyó de manera significativa en la automatización del maquetado de páginas web y permitió el uso de comandos de voz CSS 3 dentro de un entorno web intuitivo, fácil, moderno e innovador mediante la realización de vistas previas del maquetado según el avance, generación y descarga automática de las hojas de estilo CSS gestionadas mediante el uso del mouse y el teclado, así como de la voz, lo que representó una innovación del método tradicional de diseño web.

Los resultados obtenidos indican que el 33.3% de los participantes lograron un nivel deficiente en el maquetado web en el pre-test pero luego de la capacitación e implementación del framework propuesto este porcentaje de participantes desapareció; el 66.7% lograron un nivel regular en el pre-test pero en el post-test este porcentaje disminuyó a 20% lo que representó un porcentaje de mejora de 46.7%; ningún participante alcanzó el nivel eficiente de maquetado web en el pre-test pero en el post-test este nivel alcanzó un porcentaje de 80%.

7.1.2. Conclusiones específicas

OE1. Recolectar, analizar y obtener trabajos de investigación sobre frameworks que aplican reconocimiento de voz.

Se logró obtener trabajos de investigación sobre frameworks que aplican reconocimiento de voz como son el trabajo de (Hernández & Lemuz, 2016) quienes implementaron un framework de síntesis y reconocimiento de voz con la finalidad de mejorar el nivel de los participantes interesados en rendir los exámenes de certificación TOEFL correspondiente al dominio del idioma inglés, en el cual se usa la síntesis de voz para convertir a audio las respuestas de las evaluaciones y el reconocimiento de voz, a través de comandos de voz, que obedecieron a un vocabulario cerrado cuya gramática estuvo relacionada a las respuestas en una determinada evaluación. Por otro lado, (Gil, Castillo, & Flórez, 2016), propusieron un framework que reconoce instrucciones de voz para el control de una silla de ruedas en la cual los comandos de voz, pertenecientes al módulo de reconocimiento de voz, fueron un total de treinta y siete y se clasificaron en tres categorías: movimiento, domótica y salud. Por su parte, (Collado & Aparicio, 2017), implementaron un asistente de voz basado en web utilizando las tecnologías de síntesis y reconocimiento de voz aplicados en la búsqueda de información para personas con discapacidad visual.

OE2. Recolectar, procesar y obtener indicadores de maquetado de páginas o sistemas web que aseguran su aceptación.

Se ha demostrado que los indicadores de maquetado web resultan adecuados como guía o directriz al profesional en maquetado web para ayudarle a incrementar las probabilidades de aceptación, por parte del usuario final, del sitio web sobre el cual se apliquen estos indicadores (Szigeti, 2012). Encontramos dos investigaciones resaltantes. La primera investigación se titula: Web Design Guidelines for WSDM (Jarrar, 2002), que propone la aplicación de un conjunto de indicadores como aporte a la fase de implementación del diseño, la cual es parte de la metodología de diseño web WSDM (De Troyer, 2001). La segunda investigación es titulada: Research-Based Web Design & Usability Guidelines (U.S. Department of Health and Human Services, 2006), que establece una metodología completa de diseño web abarcando las fases de

análisis, maquetado propiamente y pruebas de usabilidad de un sitio web. Una vez obtenidos los indicadores que forman parte del cuestionario, proseguimos con su validación mediante el análisis de validez por juicio de expertos en maquetado web obteniendo así la lista final de indicadores de maquetado.

OE3. Diseñar los comandos de voz reconocidos y gestionados por el framework para la realización de una acción en particular.

El framework web CSS 3 propuesto gestiona la tecnología de reconocimiento de voz mediante los comandos de voz los cuales fueron un total de trescientos cuarenta y nueve comandos clasificados en siete categorías: comandos propiamente que modifican la estructura del modelo físico y gestionan el contenido de sus elementos; comandos auxiliares que permiten insertar letras del abecedario, vocales y caracteres; comandos de texto que permiten insertar cualquier número, palabra o frase literalmente; comandos de selector que permiten insertar selectores de tipo, de pseudoclase, de pseudoelemento, de clase, de ID, de reglas-AT page, de reglas-AT keyframes y de reglas-AT font-feature-values; comandos de declaración que permiten insertar propiedades y sus valores respectivos; comandos de reglas-AT que permiten insertar reglas-AT lineales y anidadas; comandos de descriptores que permiten insertar propiedades y sus valores respectivos pertenecientes a las reglas-AT anidadas.

OE4. Diseñar un caso de estudio para evaluar el framework propuesto mediante la aplicación de los indicadores de maquetado web.

El proceso de validación constó de tres escenarios. El primer escenario englobó las actividades para la obtención de los datos del pre-test donde se evaluó las capacidades de los participantes sobre maquetado web mediante el maquetado de una página de prueba; el segundo escenario abarcó el proceso de capacitación a los participantes sobre el uso del framework; el tercer escenario constó de las actividades para la obtención de los datos del post-test donde se entregó a los participantes el cuestionario para que sea llenado según el grado de cumplimiento de cada indicador durante el maquetado de la página web de prueba. Con los cuestionarios del pre-test y post-test llenados, procedimos a aplicar la técnica estadística llamada medidas de correlación y T de Student para la obtención de resultados.

7.2. Trabajos futuros

1. El framework propuesto hace uso del reconocimiento de voz para automatizar el proceso de maquetado web mediante comandos de voz. Se recomienda incorporar comandos de voz que hagan posible la automatización del proceso de creación de páginas web mediante la tecnología HTML.

2. El framework propuesto hace uso del reconocimiento de voz, para automatizar el proceso de maquetado web mediante comandos de voz. Se recomienda incorporar comandos de voz que hagan posible la automatización del proceso de validación de páginas web mediante la tecnología JavaScript.

3. El framework propuesto gestiona gráficamente los elementos de cualquier hoja de estilos CSS 3 y una vez finalizada su construcción el framework permite generar y descargar automáticamente dicha hoja de estilo. Se recomienda extender esta funcionalidad para poder visualizar el desarrollo de la hoja de estilo dentro del framework.

REFERENCIAS BIBLIOGRÁFICAS

Aiken, L. (1985). Three coefficients for Analyzing the Reliability and Validity of Ratings. *Educational and Psychological Measurement*, 131-142.

Apaza, P. P. (2017). *Desarrollo de un sistema basado en la ingeniería web para la gestión académica del instituto de idiomas de la Universidad Nacional de Juliaca 2017.* Tesis para optar el título profesional de ingenieró de sistemas, Universidad Andina Néstor Cáceres Velasquez, Facultad de Ingeniería de Sistemas, Juliaca. Retrieved 08 10, 2020, from http://repositorio.uancv.edu.pe/bitstream/handle/UANCV/1695/T036-41656593.pdf?sequence=3&isAllowed=y

Arias-Gomez, J., Villasís-Keever, M., & Miranda-Novales, M. (2016). El protocolo de investigación III: la población de estudio. *Revista Alergia México*, 201-206.

Calle, J., & Trujillo, J. (2019). GTW (Google Web Toolkit). In S. Omatu, *Sensors Science (Congreso Iberoamericano de Filosofía de la Ciencia y la Tecnología)* (pp. 361-390). Salamanca: Ediciones Universidad de Salamanca.

Collado, K. Z., & Aparicio, E. A. (2017). *Diseño de asistente de voz pasado en web para personas con visión subnormal.* Tesis para optar el Título Profesional de ingeniero de sistemas en la especialidad de ingeniero de Software, Universidad Católica De Santa María, Facultad De Ciencias E Ingenierías Físicas Y Formales, Arequipa. Retrieved 08 15, 2020, from https://alicia.concytec.gob.pe/vufind/Record/UCSM_583690f83f97ba081bd38ed9f8ddac24/Details

De la calle, G. (2014). *Modelo basado en técnicas de procesamiento de lenguaje natural para extraer y anotar información de publicaciones científicas.* Madrid.

De Troyer, O. (2001). *Audience-driven web design.* Bruselas: IDEA GroupPublishing.

Delgado, C., Sandoval, J., & Arteaga, W. (2018). Blockly Voice: un entorno de programación guiado por voz. *ACTA NOVA; Vol. 9, N° 1, marzo 2019*, 115-129.

Dentzel, Z. (2013). El impacto de internet en la vida diaria. In BBVA, *C@mbio: 19 ensayos clave sobre cómo internet está cambiando nuestras vidas* (pp. 240-252). Madrid: Artes Gráficas Palermo.

Escurra, L. (1988). *Cuantificación de la validez de contenido por criterio de jueces*. Lima.

García, F. (2019, Octubre 20). *Introducción a la Ingeniería Web*. Retrieved from Procesos y Métodos de Modelado para: https://repositorio.grial.eu/bitstream/grial/1541/1/Tema1.pdf

Gil, L., Castillo, L., & Flórez, R. (2016). *Reconocimiento de comandos de voz en español orientado al control de una silla de ruedas.* Manizales, Colombia.

GNU. (2021, Agosto 25). *El Sistema Operativo GNU*. Retrieved from Licencias: https://www.gnu.org/licenses/licenses.html

Hernández , M., & Lemuz, R. (2016). *Aplicación multimedia para el entrenamiento en la certificación TOEFL mediante reconocimiento de voz* . Artículo, Universidad Autónoma de Puebla, Facultad de Ciencias de la Computación, Puebla-México. Retrieved 11 05, 2020, from https://doi.org/10.13053/rcs-128-1-6

Hernandez, C., & Carpio, N. (2019). Introducción a los tipos de muestreo. *ALERTA*, 76-79.

Jaimez, C., & Vargas, R. (2017). Editor web visual para HTML, CSS y JavaScript de apoyo a la docencia. *Virtualidad, Educación y Ciencia*, 136-152.

Jarrar, S. (2002). *Web Design Guidelines for WSDM.* Bruselas.

JSON. (2021, Noviembre 18). *Introducing JSON*. Retrieved from https://www.json.org/json-en.html

Malone, T. (2013). ¿Cómo está cambiando internet nuestra manera de trabajar? In BBVA, *C@mbio: 19 ensayos clave sobre cómo internet está cambiando nuestras vidas* (pp. 314-326). Madrid: Artes Gráficas Palermo.

Mamani, S. M. (2019). *Aplicación de SCRUM y UML para el desarrollo de un sistema de ventas.* Cochabamba, Bolivia.

MDN Web Dcos. (2021, Mayo 30). *Uso de la Web Speech API*. Retrieved from Uso de la Web Speech API: https://developer.mozilla.org/es/docs/Web/API/Web_Speech_API/Using_the_Web_Speech_API

MDN Web Docs. (2021, Octubre 24). *Declaraciones de CSS*. Retrieved from https://developer.mozilla.org/es/docs/Web/CSS/Syntax

MDN Web Docs. (2021, Octubre 24). *Empezar con HTML*. Retrieved from 2021

Mihalcin, T. (2007). *Web Frameworks Comparison Concerning the Efficiency of Development.* Praga.

Muñoz , M. P., & Rodriguez, S. P. (2015). *Diseño e implementación de un agente animado interactivo para informar sobre el calendario académico de la Universidad de Córdoba mediante reconocimiento de voz basado en un entorno web.* Trabajo de grado para optar al título de ingeniero de sistemas, Universidad de Córdoba, Facultad de Ingenierías, Santa Cruz de Lorica. Retrieved 11 10, 2020, from https://repositorio.unicordoba.edu.co/bitstream/handle/ucordoba/290/DISE%c3%91O%20E%20IMPLEMENTACI%c3%93N%20DE%20UN%20AGENTE%20ANIMADO%20INTERACTIVO%20PARA%20INFORMAR%20SOBRE%20EL%20CALENDARIO%20ACAD%c3%89MICO%20%20DE%20LA%20UNIVERSIDAD%20DE%20C%c3%93RDOB

Murillo, J. (2013). *Métodos de Investigación de Enfoque Experimental.* Lima.

Nagilla, R. (2012). *Comparison of Web Development Technologies - ASP.NET & PHP.* Vasteras y Eskilstuna.

No Magic, Inc. (2010). *Magicdraw Architecture Made Simple.* No Magic, Inc.

Otzen, T., & Manterola, C. (2017). *Técnicas de muestreo sobre una población a estudio.* Temuco.

Palacio, M. (2020). *Guía de Scrum Master.* Iubaris Info 4 Media SL.

Pilamunga, E. M. (2012). *El maquetado a base de scripts y hojas de estilo en cascada (CSS) y su incidencia en la optimización de un sitio web.* Trabajo de investigación Previa a la obtención del Título de Especialista en Diseño y Animación Web, Universidad técnica de Ambato, Ambato - Ecuador. Retrieved 10 10, 2020

Roig, A. (2019). *Desarrollo de una aplicación para la web utilizando el Web Speech API.* Valencia.

Sajjadi, S. (2012). *Adapting the Website Design Method WSDM towards Recent Common Practices in Web Development.* Bruselas.

Szigeti, S. (2012). *The challenge of web design guidelines: Investigating issues of awarenes, interpretation, and efficacy.* Toronto.

Torres, I. (2013). *Framework multiplataforma para reconocimiento de voz en aplicaciones open rich-client para dispositivos móviles.* México D.F.

U.S. Department of Health and Human Services. (2006). *Research-Based Web Design and Usability Guidelines.* Washington, D.C: U.S. Government Printing Office.

Vinuesa, P. (2016). *Tema 8 - Correlación: teoría y práctica.* México.

Vivancos, P. (2016). *Plataforma inteligente de diseño para todos para control de teléfonos móviles mediante habla en lenguaje natural.* Murcia.

Wikipedia. (2020, Mayo 03). *Framework de CSS*. Retrieved from https://es.wikipedia.org/wiki/Framework_de_CSS

ANEXOS

ANEXO 1. Archivo excel enviado a cada experto conteniendo los indicadores y las dimensiones del instrumento de recolección de datos para su evaluación

EVALUACIÓN DEL INSTRUMENTO DE RECOLECCIÓN DE DATOS - TESIS DE POSGRADO EN INGENIERÍA DE SISTEMAS E INFORMÁTICA - UNMSM

SECCIÓN I.	
TÍTULO DE LA TESIS	FRAMEWORK PARA LA AUTOMATIZACIÓN DEL MAQUETADO DE PÁGINAS WEB APLICANDO PROCESAMIENTO DE LENGUAJE NATURAL
VARIABLE INDEPENDIENTE	FRAMEWORK BASADO EN PROCESAMIENTO DE LENGUAJE NATURAL
VARIABLE DEPENDIENTE	AUTOMATIZACIÓN DEL MAQUETADO DE PÁGINAS WEB

SECCIÓN II.	
OBJETIVO DE LA EVALUACIÓN	CONSIDERANDO SU AMPLIA EXPERIENCIA EN TEMAS DE DISEÑO WEB, SE SOLICITA SU GENTIL PARTICIPACIÓN, EN LA EVALUACIÓN DEL INSTRUMENTO DE VALIDACIÓN DE TESIS SIGUIENTE.

SECCIÓN III.

VARIABLE	DIMENSIONES	INDICADORES	ÍTEMS	ESCALA	¿El ítem presenta claridad y coherencia? (Marcar aspa (X) en un ítem) Opciones de respuesta: 1 Completamente en desacuerdo	2 En desacuerdo	3 Indiferente	4 De acuerdo	5 Completamente de	Relación entre variables y dimensiones (Inicar SI / NO)	Relación entre dimensiones e indicadores (Inicar SI / NO)	Relación entre ítems y opciones de respuesta (Inicar SI / NO)	Obserbaciones sobre el ítem (Texto descriptivo, opcional, según su evaluación)
MAQUETADO DE PÁGINA WEB	Animación	Tecnología	El maquetado mantiene un diseño de animación y movimiento estandar	ORDINAL									
			El maquetado evita el uso de gifs animados	ORDINAL									
			El maquetado usa animaciones creadas con CSS, evitando el uso de tecnologías desfasada	ORDINAL									
		Coherencia	El maquetado mantiene la coherencia entre las animaciones y el contenido para su mejor	ORDINAL									
	Botones	Comprensión	El maquetado mantiene un diseño de botones e íconos estándar	ORDINAL									
			El maquetado utiliza colores para indicar la acción a realizar (verde para grabar, rojo para	ORDINAL									
			El maquetado utiliza iconos según la funcionalidad a realiza (basurero para eliminar, flech	ORDINAL									
			El maquetado evita el uso innecesario de iconos y gráficos	ORDINAL									
		Interacción	El maquetado cambia de color o forma para enfatizar la interacción entre el elemento y e	ORDINAL									
	Contenido	Estructura	El maquetado mantiene un diseño de contenido estándar	ORDINAL									
			El maquetado permite el movimiento vertical dinámico del encabezado de las columnas d	ORDINAL									
		Uso de imagen	El maquetado evita el uso innecesario de fotografías	ORDINAL									
			El maquetado evita o limita el uso de mapas de imágenes	ORDINAL									
		Información complementaria	El maquetado brinda una descripción por escrito de cualquier información crítica contenid	ORDINAL									
			El maquetado utiliza los mismos colores para agrupar elementos relacionados (menús de	ORDINAL									
	Fondo	Estandarización	El maquetado mantiene un diseño de fondo estándar, evitando tener fondos de colores d	ORDINAL									
			El maquetado asegura diseños de fondo sencillos para mejorar la legibilidad del texto	ORDINAL									
			El maquetado asegura que las imágenes de fondo no hagan imposible la lectura y compre	ORDINAL									
			El maquetado cuenta con colores de fondo para proporcionar el máximo contraste, evitar	ORDINAL									
			El maquetado cuenta con el color blanco o colores claros, para el fondo	ORDINAL									
	Imagen	Calidad	El maquetado mantiene un diseño de imágenes estándar	ORDINAL									
			El maquetado asegura que todas las imágenes sean nitidas y faciles de visualizar	ORDINAL									
			El maquetado mantiene la cantidad de colores en sus imágenes al mínimo	ORDINAL									
			El maquetado evita las imágenes peige o marrones	ORDINAL									
			El maquetado utiliza imágenes entrelazadas o progresivas	ORDINAL									
	Tipografía	Interpretación	El maquetado mantiene un diseño de tipografia estándar	ORDINAL									
			El maquetado cuenta con textos fáciles de leer e interpretar por el usuario	ORDINAL									
		Escritura	El maquetado evitar texto de colores amarilla o blanco, ya que son difíciles de leer e impr	ORDINAL									
			El maquetado evita el uso de letras mayúsculas en todo el texto	ORDINAL									
			El maquetado evita el texto parpadeante	ORDINAL									
		Alcance	El maquetado utiliza solo un número limitado de fuentes cuidadosamente seleccionadas	ORDINAL									

ANEXO 2. Formulario único de trámite para solicitar apoyo sobre la implementación de tesis en la Escuela de Ingeniería de Sistemas e Informática de la UNASAM

UNIVERSIDAD NACIONAL
SANTIAGO ANTÚNEZ DE MAYOLO
"Una Nueva Universidad para el Desarrollo"
Av. Centenario N° 200 - Central telefónica: (043) 640020
HUARAZ - ANCASH

FORMULARIO ÚNICO DE TRÁMITE DIGITAL

1. Dirigido a:

Señor: Mag. Henry Angel Garrido Angulo Cargo: Decano	Facultad: Ciencias

2. Datos del solicitante:

Nombres: Julio Cesar	Apellidos: Prudencio Nieves

3. Identificación:

DNI: N° 46610089	Código de alumno: N° 071.0125.532

4. Carrera Profesional:

Ingeniería de Sistemas e Informática

5. Domicilio:

Av. / Jr. / Pje. y Nro.: Av. Pedro Pablo Atusparia Mz. 172 Lt. 2	Lugar / Ciudad: Huaraz – Huaraz – Ancash

6. Otros Datos:

Correo electrónico: Institucional ⇨ @unasam.edu.pe Personal ⇨ prudencio11@hotmail.com	Celular N°: 950048198 Teléfono fijo N°: 043-784707

7. Motivo o Asunto:

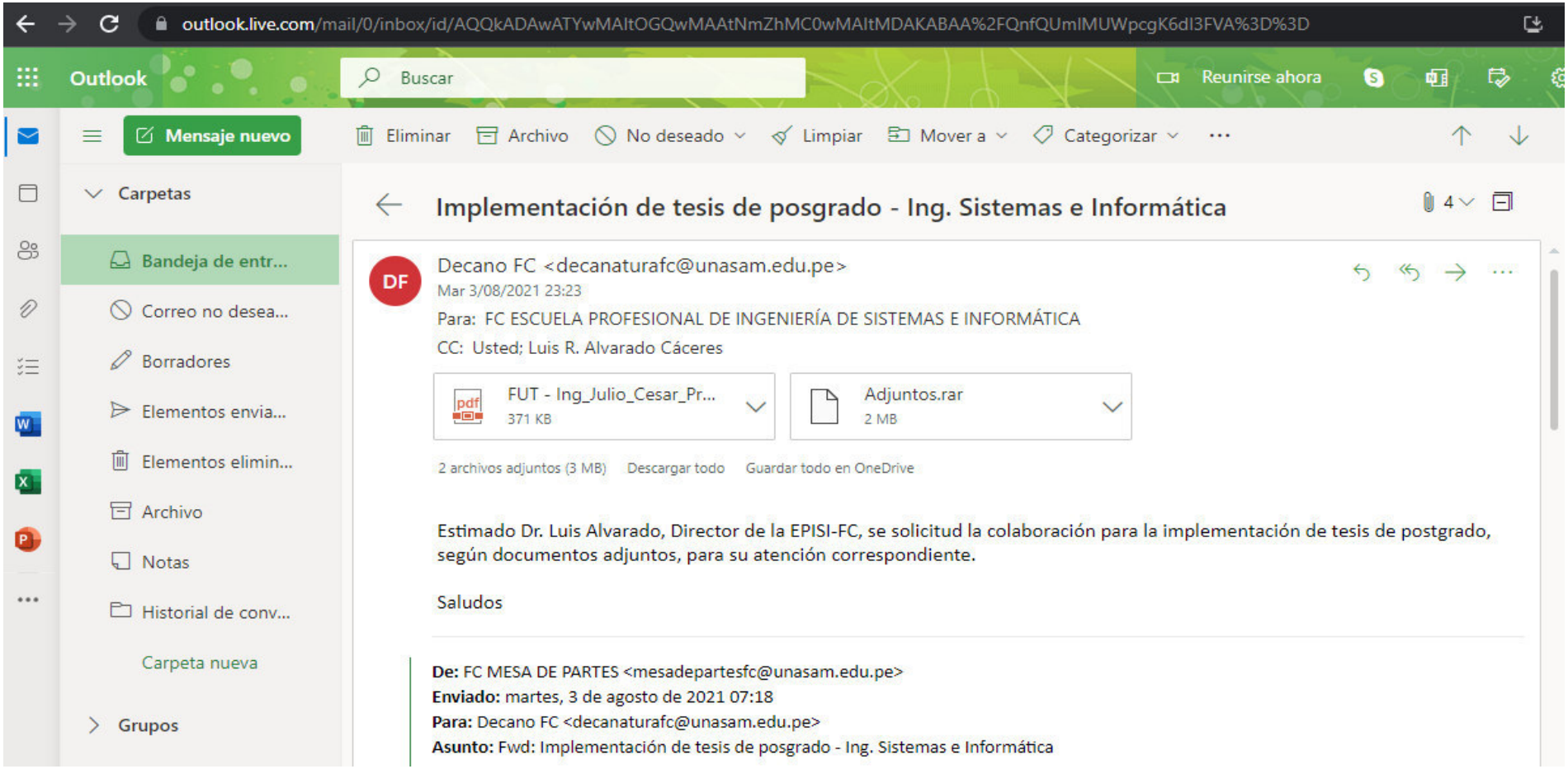
Solicita:

Apoyo para la implementación de mi tesis de posgrado realizada en la Universidad Nacional Mayor de San Marcos (UNMSM), con la finalidad de evaluar el sistema editor CSS 3 con reconocimiento de voz para la generación automatizada de hojas de estilo, con los alumnos del noveno (9°) ciclo de la carrera de Ing. de Sistemas e Informática de la UNASAM. Para ello requiero contar con la participación de treinta (30) alumnos del ciclo mencionado durante siete (07) días calendarios con una duración exacta de treinta (30) minutos por día.

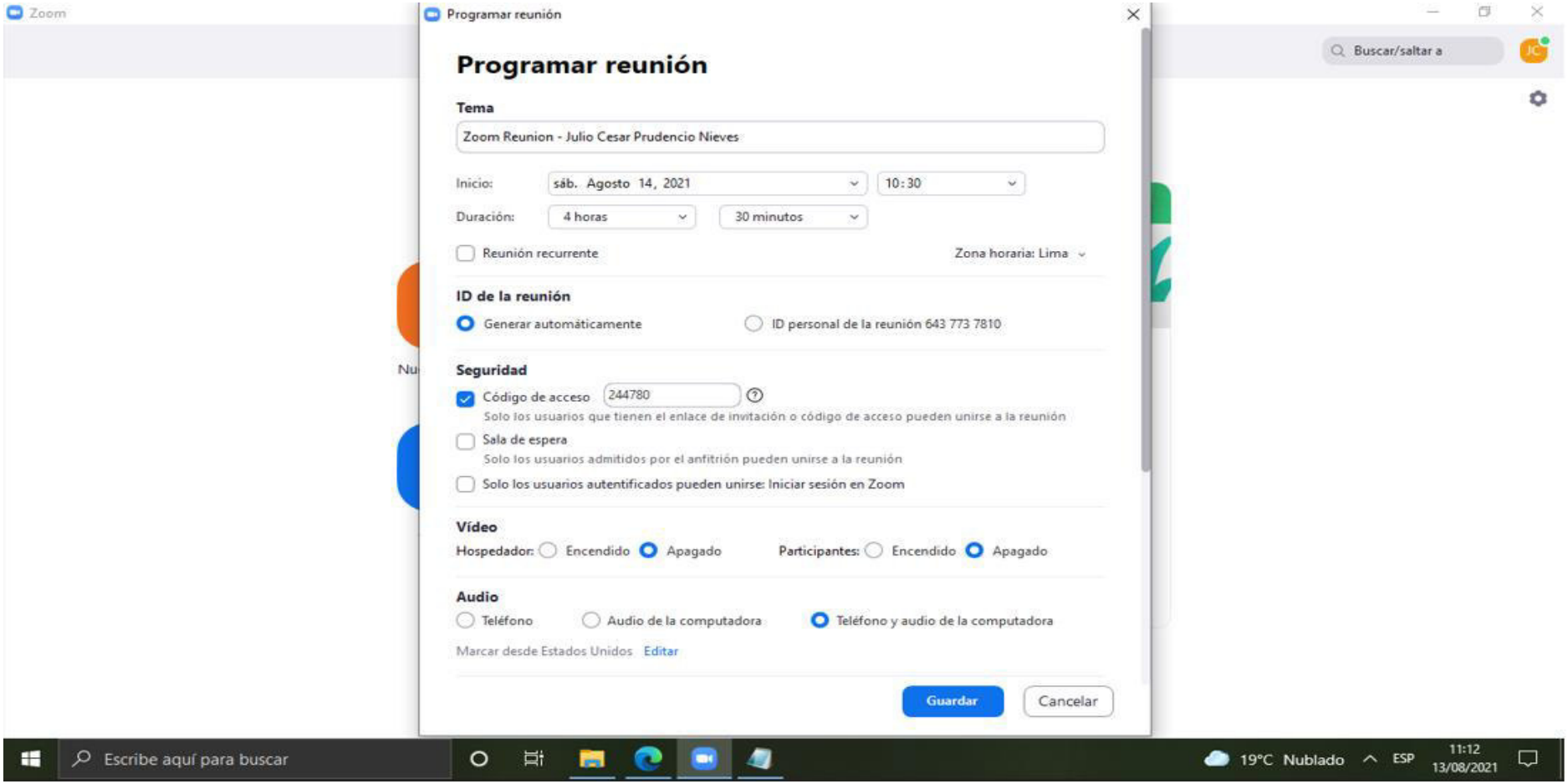
8. Anexos Adjuntos:

Copia de DNI
Constancia de Egreso - posgrado Ing. de Sistemas e Informática Mención Ing. Software - UNMSM
Temario a tratar durante la implementación del editor CSS 3 con reconocimiento de voz
Link del grupo de WhatsApp para la implementación del editor CSS 3 con reconocimiento de voz
Link del manual y del sistema editor CSS 3 con reconocimiento de voz

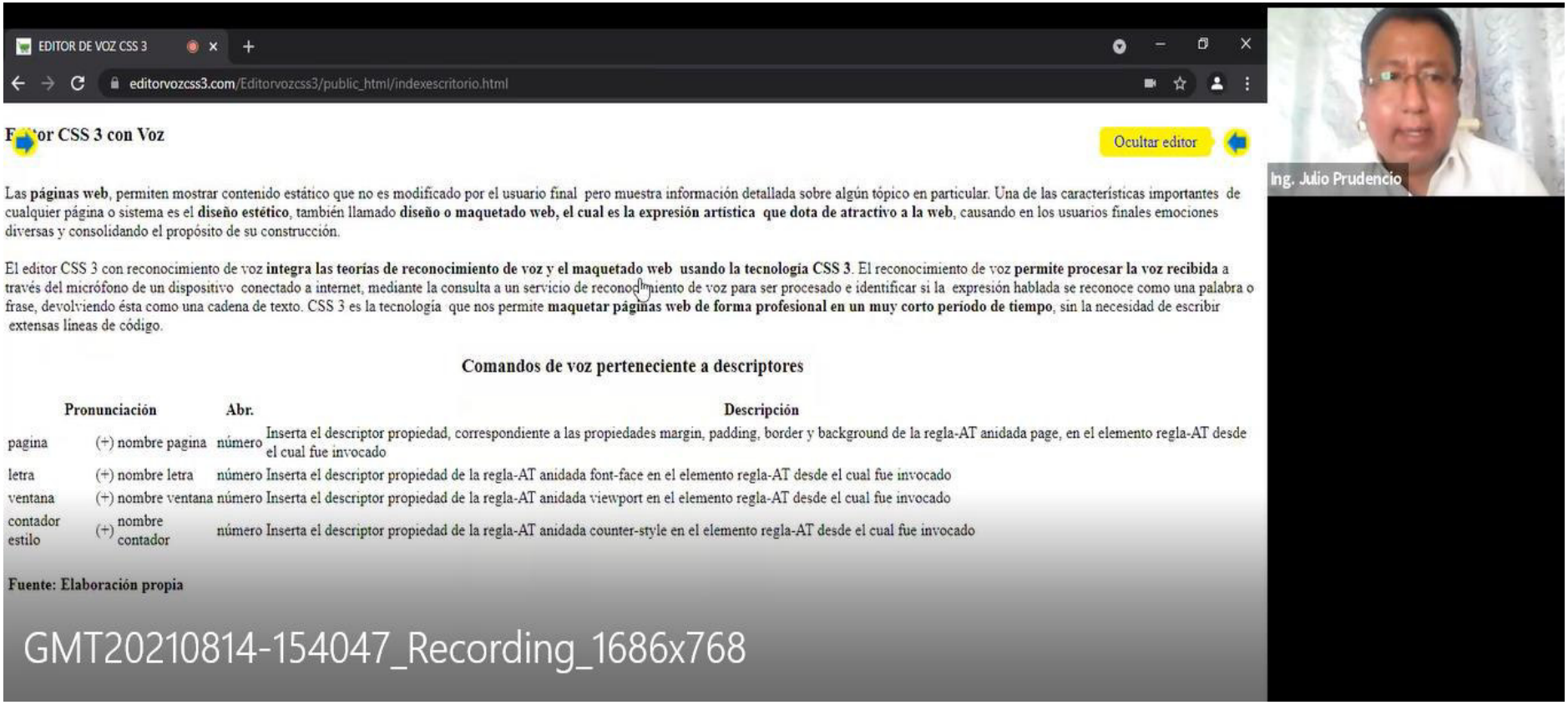
ANEXO 3. Email dirigido a la Decanatura de la Facultad de Ciencias de la UNASAM para solicitar apoyo sobre la implementación de tesis



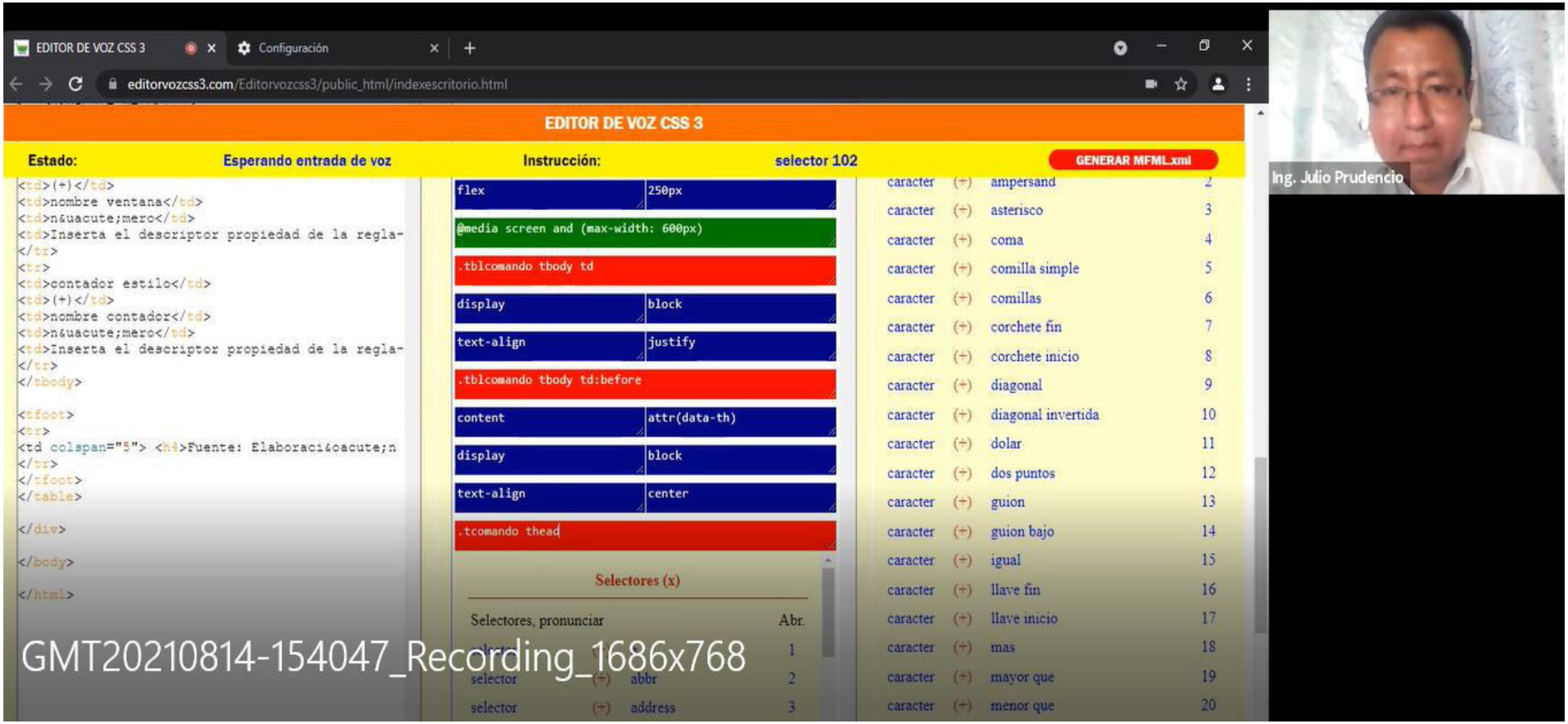
ANEXO 4. Programación, mediante el programa Zoom, de la primera reunión de implementación de tesis con los alumnos del noveno ciclo de la Escuela de Ingeniería de Sistemas e Informática de la UNASAM



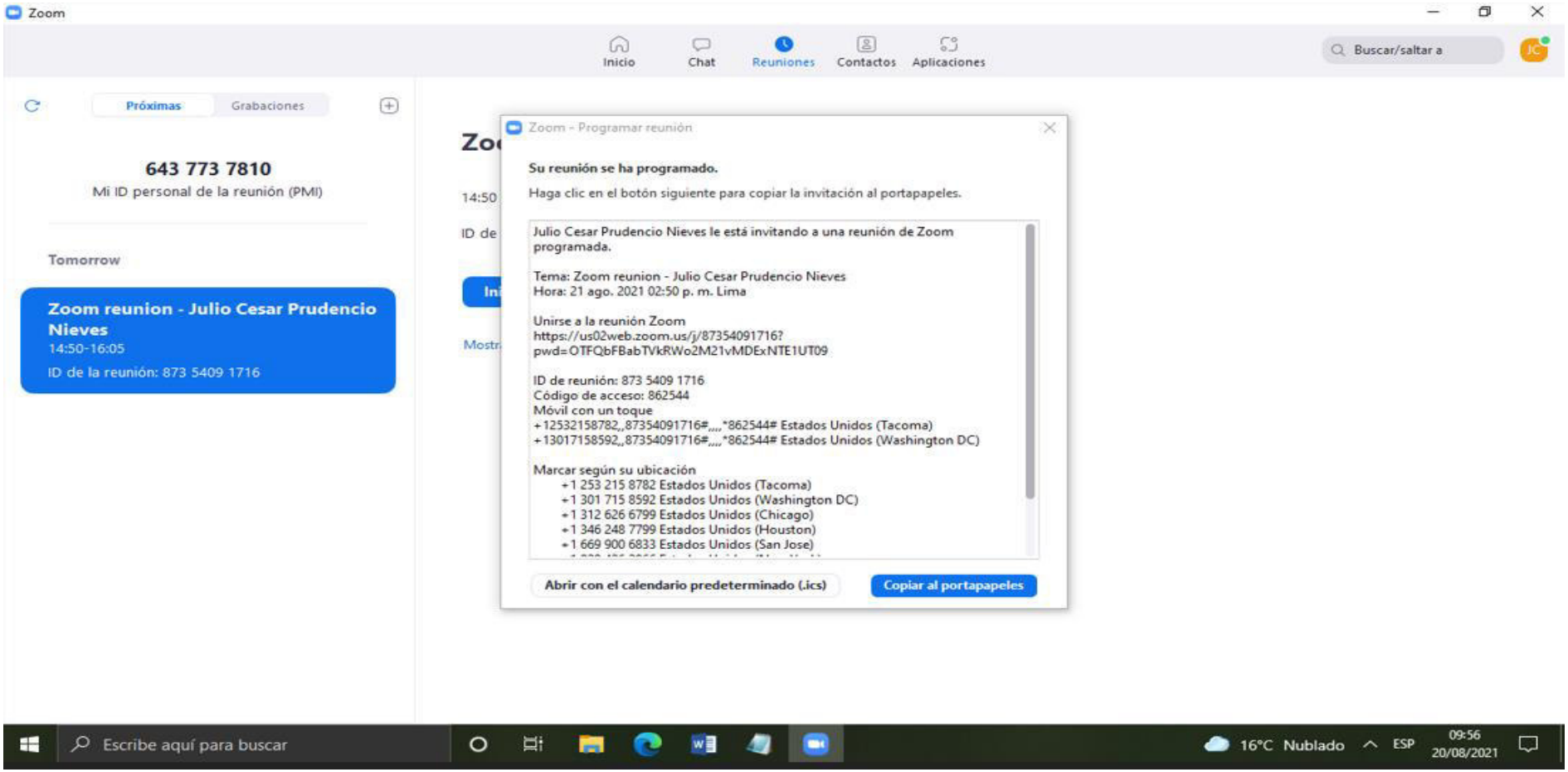
ANEXO 5. Ejecución, mediante el programa Zoom, de la primera reunión de implementación de tesis con los alumnos del noveno ciclo de la Escuela de Ingeniería de Sistemas e Informática de la UNASAM, donde se obtiene el pre-test maquetando la página web de prueba



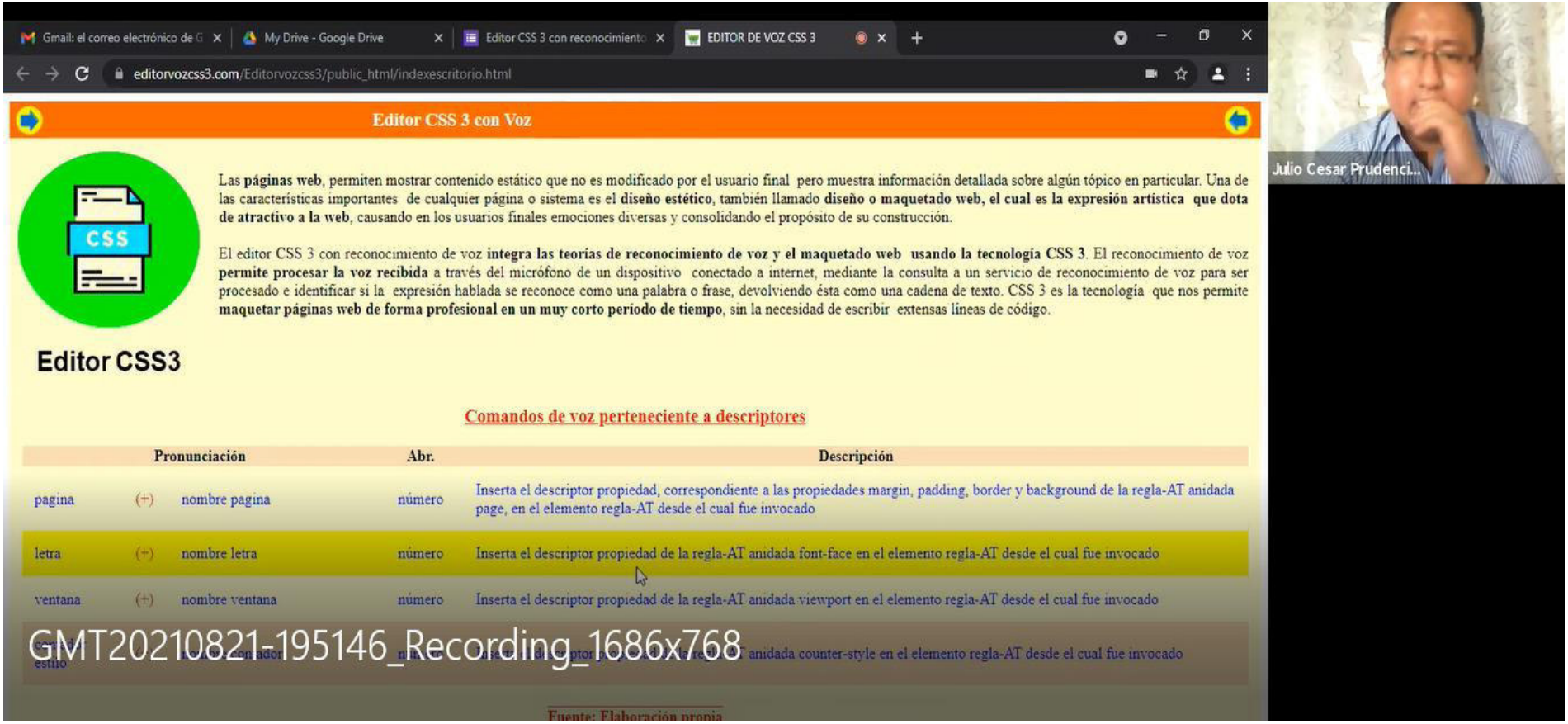
ANEXO 6. Ejecución, mediante el programa Zoom, de la primera reunión de implementación de tesis con los alumnos del noveno ciclo de la Escuela de Ingeniería de Sistemas e Informática de la UNASAM, donde se explica el funcionamiento del framework propuesto



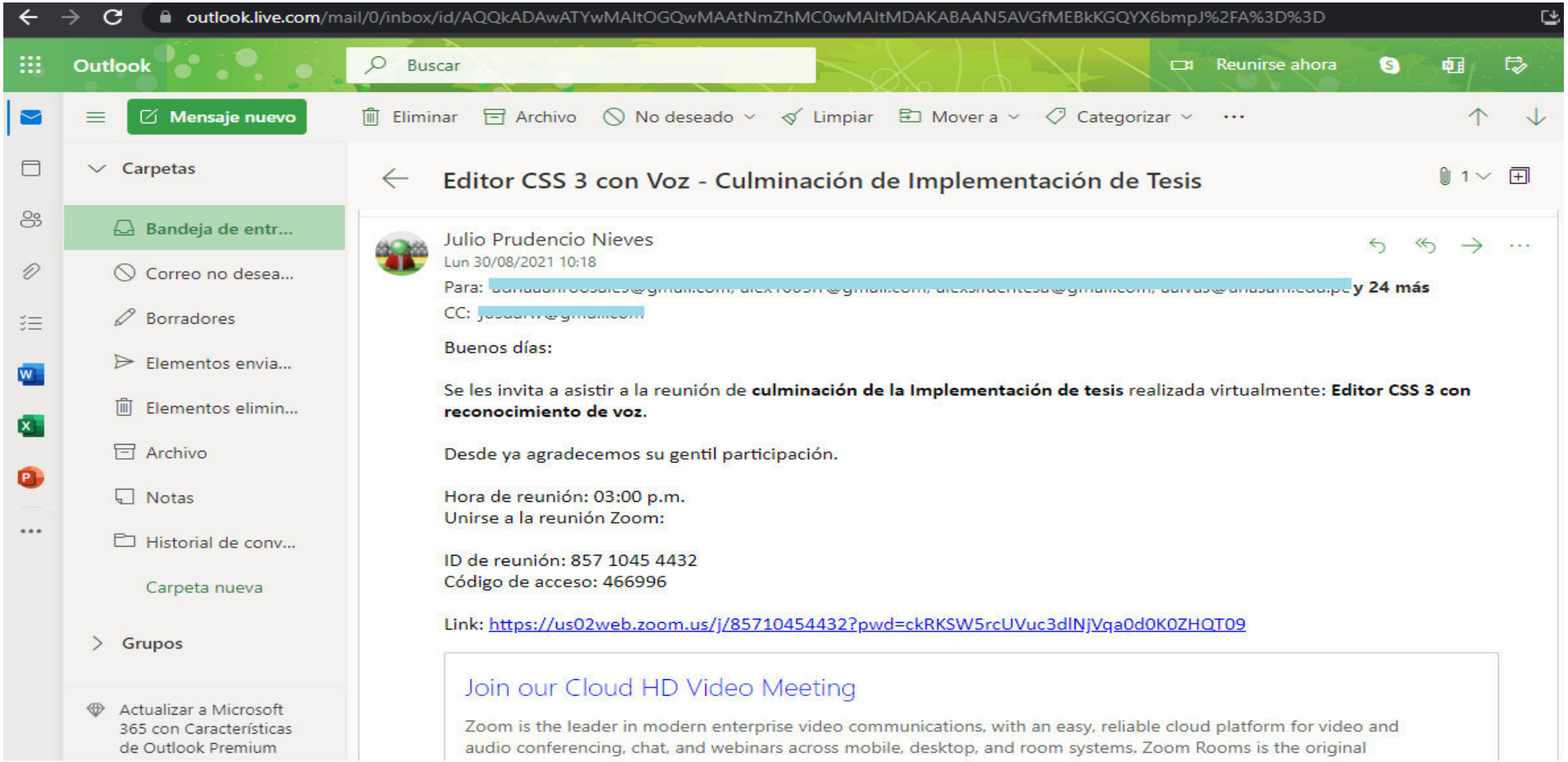
ANEXO 7. Programación, mediante el programa Zoom, de la segunda reunión de implementación de tesis con los alumnos del noveno ciclo de la Escuela de Ingeniería de Sistemas e Informática de la UNASAM



ANEXO 8. Ejecución, mediante el programa Zoom, de la segunda reunión de implementación de tesis con los alumnos del noveno ciclo de la Escuela de Ingeniería de Sistemas e Informática de la UNASAM, donde se realiza el maquetado de la página web de prueba usando el framework propuesto

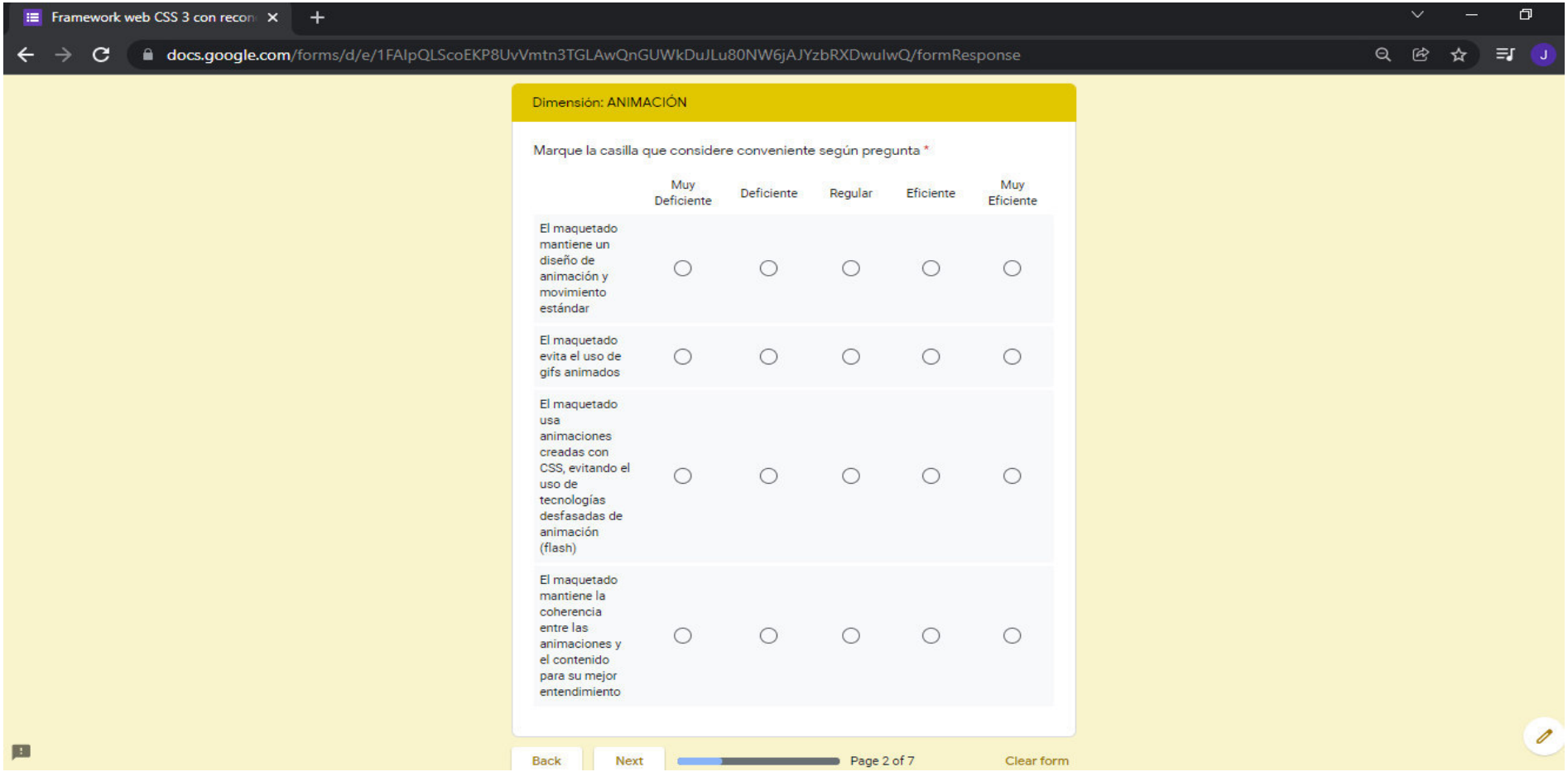


ANEXO 9. Email dirigido a los alumnos del noveno ciclo de la Escuela de Ingeniería de Sistemas e Informática de la UNASAM para invitarlos formalmente a la culminación de la implementación de tesis



ANEXO 10. Presentación del cuestionario, elaborado con el programa Google Forms, ante los alumnos del noveno ciclo de la Escuela de Ingeniería de Sistemas e Informática de la UNASAM para comenzar el llenado del Post-test

ANEXO 11. Contenido del cuestionario, elaborado con el programa Google Forms, para ser llenado por los alumnos del noveno ciclo de la Escuela de Ingeniería de Sistemas e Informática de la UNASAM como parte del proceso de realización del Post-test



ANEXO 12. Comandos para el maquetado del archivo ejemplodisenio.html

Pronunciación			Abr.	Descripción
comando	(+)	insertar selector	1	Inserta el selector en el modelo físico desde el cual fue invocado
class	(+)	bl	-	Inserta la clase ".b l" en el selector desde el cual fue invocado
comando	(+)	Atrás	9	Retrocede el cursor un espacio en el elemento desde el cual fue invocado
comando	(+)	Atrás	9	Retrocede el cursor un espacio en el selector o declaración desde el cual fue invocado
comando	(+)	Atrás	9	Retrocede el cursor un espacio en el selector o declaración desde el cual fue invocado
abecedario	(+)	Ti	5	Inserta la letra "t" en el selector o declaración desde el cual fue invocado
comando	(+)	Atrás	9	Retrocede el cursor un espacio en el selector o declaración desde el cual fue invocado
comando	(+)	Atrás	9	Retrocede el cursor un espacio en el selector o declaración desde el cual fue invocado
comando	(+)	seleccionar	20	Selecciona parte del contenido del selector o declaración desde el cual fue invocado
comando	(+)	Borrar	11	Borra el contenido seleccionado del selector o declaración desde el cual fue invocado
comando	(+)	Avanzar	10	Avanza el cursor un espacio en el selector o declaración desde el cual fue invocado
texto	(+)	Comando	-	Inserta la palabra "comando" en el selector o declaración desde el cual fue invocado
comando	(+)	insertar declaracion	2	Inserta la declaración debajo del selector o declaración desde el cual fue invocado
propiedad	(+)	Width	324	Inserta la propiedad en la declaración desde el cual fue invocado
texto	(+)	95	-	Inserta el texto en el selector o declaración desde el cual fue invocado

caracter	(+)	porcentaje	25	Inserta el caracter "%" indicado en la declaración desde el cual fue invocado
comando	(+)	insertar declaracion	2	Inserta la declaración debajo del selector o declaración desde el cual fue invocado
propiedad	(+)	margin	187	Inserta la propiedad en la declaración desde el cual fue invocado
texto	(+)	0	-	Inserta el texto en el selector o declaración desde el cual fue invocado
caracter	(+)	porcentaje	25	Inserta el caracter "%" indicado en la declaración desde el cual fue invocado
comando	(+)	espacio	15	Inserta un espacio en blanco " " en el selector o la declaración desde el cual fue invocado
texto	(+)	2	-	Inserta el texto en el selector o declaración desde el cual fue invocado
caracter	(+)	porcentaje	25	Inserta el caracter "%" indicado en la declaración desde el cual fue invocado
comando	(+)	insertar declaracion	2	Inserta la declaración debajo del selector o declaración desde el cual fue invocado
propiedad	(+)	border-collapse	45	Inserta la propiedad en la declaración desde el cual fue invocado
valor	(+)	collapse	1	Inserta el valor en la declaración desde el cual fue invocado
comando	(+)	insertar selector	1	Inserta un nuevo selector en el modelo físico
comando	(+)	selector 1	-	Asigna el cursor en el primer selector
comando	(+)	seleccionar todo	21	Selecciona todo el contenido del selector o declaración desde el cual fue invocado
comando	(+)	copiar	13	Guarda el contenido seleccionado del selector o declaración desde el cual fue invocado
comando	(+)	selector 2	-	Asigna el cursor en el segundo selector
comando	(+)	pegar	18	Inserta el contenido guardado, en la posición actual del cursor, del selector o declaración desde el cual fue invocado

comando	(+)	final	16	Asigna el cursor al final del contenido del selector o declaración desde el cual fue invocado
comando	(+)	espacio	15	Inserta un espacio en blanco " " en el selector o la declaración desde el cual fue invocado
selector	(+)	caption	17	Inserta el contenido del selector en el selector desde el cual fue invocado
comando	(+)	insertar declaracion	2	Inserta la declaración debajo del selector o declaración desde el cual fue invocado
propiedad	(+)	color	98	Inserta la propiedad en la declaración desde el cual fue invocado
valor	(+)	red	122	Inserta el valor en la declaración desde el cual fue invocado
comando	(+)	insertar declaracion	2	Inserta la declaración debajo del selector o declaración desde el cual fue invocado
propiedad	(+)	text-decoration	291	Inserta la propiedad en la declaración desde el cual fue invocado
valor	(+)	underline	13	Inserta el valor en la declaración desde el cual fue invocado
comando	(+)	insertar selector	1	Inserta un nuevo selector en el modelo físico
comando	(+)	pegar	18	Inserta el contenido guardado, en la posición actual del cursor, del selector o declaración desde el cual fue invocado
comando	(+)	final	16	Asigna el cursor al final del contenido del selector o declaración desde el cual fue invocado
comando	(+)	espacio	15	Inserta un espacio en blanco " " en el selector o la declaración desde el cual fue invocado
selector	(+)	thead	102	Inserta el contenido del selector en el selector desde el cual fue invocado
comando	(+)	espacio	15	Inserta un espacio en blanco " " en el selector o la declaración desde el cual fue invocado

selector	(+)	th	101	Inserta el contenido del selector en el selector desde el cual fue invocado
comando	(+)	insertar declaracion	2	Inserta la declaración debajo del selector o declaración desde el cual fue invocado
propiedad	(+)	text-align	288	Inserta la propiedad en la declaración desde el cual fue invocado
valor	(+)	center	2	Inserta el valor en la declaración desde el cual fue invocado
comando	(+)	insertar declaracion	2	Inserta la declaración debajo del selector o declaración desde el cual fue invocado
propiedad	(+)	background-color	19	Inserta la propiedad en la declaración desde el cual fue invocado
texto	(+)	w	-	Inserta el texto en el selector o declaración desde el cual fue invocado
texto	(+)	h	-	Inserta el texto en el selector o declaración desde el cual fue invocado
vocal	(+)	e	2	Inserta la vocal en el selector o declaración desde el cual fue invocado
vocal	(+)	a	1	Inserta la vocal en el selector o declaración desde el cual fue invocado
abecedario	(+)	ti	5	Inserta el abecedario en el selector o declaración desde el cual fue invocado
comando	(+)	insertar selector	1	Inserta un nuevo selector en el modelo físico
comando	(+)	pegar	18	Inserta el contenido guardado, en la posición actual del cursor, del selector o declaración desde el cual fue invocado
comando	(+)	final	16	Asigna el cursor al final del contenido del selector o declaración desde el cual fue invocado
comando	(+)	espacio	15	Inserta un espacio en blanco " " en el selector o la declaración desde el cual fue invocado
selector	(+)	tbody	97	Inserta el contenido del selector en el selector desde el cual fue invocado

comando	(+)	espacio	15	Inserta un espacio en blanco " " en el selector o la declaración desde el cual fue invocado
selector	(+)	tr	105	Inserta el contenido del selector en el selector desde el cual fue invocado
comando	(+)	espacio	15	Inserta un espacio en blanco " " en el selector o la declaración desde el cual fue invocado
selector	(+)	td	98	Inserta el contenido del selector en el selector desde el cual fue invocado
comando	(+)	insertar declaracion	2	Inserta la declaración debajo del selector o declaración desde el cual fue invocado
propiedad	(+)	padding	229	Inserta la propiedad en la declaración desde el cual fue invocado
texto	(+)	1	-	Inserta el texto en el selector o declaración desde el cual fue invocado
caracter	(+)	porcentaje	25	Inserta el caracter "%" indicado en la declaración desde el cual fue invocado
comando	(+)	insertar selector	1	Inserta un nuevo selector en el modelo físico
comando	(+)	pegar	18	Inserta el contenido guardado, en la posición actual del cursor, del selector o declaración desde el cual fue invocado
comando	(+)	final	16	Asigna el cursor al final del contenido del selector o declaración desde el cual fue invocado
comando	(+)	espacio	15	Inserta un espacio en blanco " " en el selector o la declaración desde el cual fue invocado
selector	(+)	tbody	97	Inserta el contenido del selector en el selector desde el cual fue invocado
comando	(+)	espacio	15	Inserta un espacio en blanco " " en el selector o la declaración desde el cual fue invocado
selector	(+)	tr	105	Inserta el contenido del selector en el selector desde el cual fue invocado
clase	(+)	nth-child	35	Inserta la pseudoclase en el selector desde el cual fue invocado

caracter	(+)	parentesis inicio	24	Inserta el caracter "(" en el selector desde el cual fue invocado
texto	(+)	2	-	Inserta el texto en el selector o declaración desde el cual fue invocado
abecedario	(+)	n	3	Inserta el abecedario en el selector o declaración desde el cual fue invocado
caracter	(+)	parentesis fin	23	Inserta el caracter ")" en el selector desde el cual fue invocado
comando	(+)	insertar declaracion	2	Inserta la declaración debajo del selector o declaración desde el cual fue invocado
propiedad	(+)	background-color	19	Inserta la propiedad en la declaración desde el cual fue invocado
texto	(+)	w	-	Inserta el texto en el selector o declaración desde el cual fue invocado
texto	(+)	h	-	Inserta el texto en el selector o declaración desde el cual fue invocado
vocal	(+)	e	2	Inserta la vocal en el selector o declaración desde el cual fue invocado
vocal	(+)	a	1	Inserta la vocal en el selector o declaración desde el cual fue invocado
abecedario	(+)	ti	5	Inserta el abecedario en el selector o declaración desde el cual fue invocado
comando	(+)	insertar selector	1	Inserta un nuevo selector en el modelo físico
comando	(+)	pegar	18	Inserta el contenido guardado, en la posición actual del cursor, del selector o declaración desde el cual fue invocado
comando	(+)	final	16	Asigna el cursor al final del contenido del selector o declaración desde el cual fue invocado
comando	(+)	espacio	15	Inserta un espacio en blanco " " en el selector o la declaración desde el cual fue invocado
selector	(+)	tbody	97	Inserta el contenido del selector en el selector desde el cual fue invocado

comando	(+)	espacio	15	Inserta un espacio en blanco " " en el selector o la declaración desde el cual fue invocado
selector	(+)	tr	105	Inserta el contenido del selector en el selector desde el cual fue invocado
clase	(+)	hover	23	Inserta la pseudoclase en el selector desde el cual fue invocado
comando	(+)	insertar declaracion	2	Inserta la declaración debajo del selector o declaración desde el cual fue invocado
propiedad	(+)	background-color	19	Inserta la propiedad en la declaración desde el cual fue invocado
texto	(+)	yellow	-	Inserta el texto en el selector o declaración desde el cual fue invocado
comando	(+)	insertar selector	1	Inserta un nuevo selector en el modelo físico
comando	(+)	pegar	18	Inserta el contenido guardado, en la posición actual del cursor, del selector o declaración desde el cual fue invocado
comando	(+)	final	16	Asigna el cursor al final del contenido del selector o declaración desde el cual fue invocado
comando	(+)	espacio	15	Inserta un espacio en blanco " " en el selector o la declaración desde el cual fue invocado
selector	(+)	tbody	97	Inserta el contenido del selector en el selector desde el cual fue invocado
comando	(+)	espacio	15	Inserta un espacio en blanco " " en el selector o la declaración desde el cual fue invocado
selector	(+)	td	98	Inserta el contenido del selector en el selector desde el cual fue invocado
clase	(+)	nth-child	35	Inserta la pseudoclase en el selector desde el cual fue invocado
caracter	(+)	parentesis inicio	24	Inserta el caracter "(" en el selector desde el cual fue invocado
texto	(+)	1	-	Inserta el texto en el selector o declaración desde el cual fue invocado

caracter	(+)	parentesis fin	23	Inserta el caracter ")" en el selector desde el cual fue invocado
comando	(+)	insertar declaracion	2	Inserta la declaración debajo del selector o declaración desde el cual fue invocado
propiedad	(+)	width	324	Inserta la propiedad en la declaración desde el cual fue invocado
texto	(+)	8	-	Inserta el texto en el selector o declaración desde el cual fue invocado
caracter	(+)	porcentaje	25	Inserta el caracter "%" indicado en la declaración desde el cual fue invocado
comando	(+)	insertar selector	1	Inserta un nuevo selector en el modelo físico
comando	(+)	pegar	18	Inserta el contenido guardado, en la posición actual del cursor, del selector o declaración desde el cual fue invocado
comando	(+)	final	16	Asigna el cursor al final del contenido del selector o declaración desde el cual fue invocado
comando	(+)	espacio	15	Inserta un espacio en blanco " " en el selector o la declaración desde el cual fue invocado
selector	(+)	tbody	97	Inserta el contenido del selector en el selector desde el cual fue invocado
comando	(+)	espacio	15	Inserta un espacio en blanco " " en el selector o la declaración desde el cual fue invocado
selector	(+)	td	98	Inserta el contenido del selector en el selector desde el cual fue invocado
clase	(+)	nth-child	35	Inserta la pseudoclase en el selector desde el cual fue invocado
caracter	(+)	parentesis inicio	24	Inserta el caracter "(" en el selector desde el cual fue invocado
texto	(+)	2	-	Inserta el texto en el selector o declaración desde el cual fue invocado
caracter	(+)	parentesis fin	23	Inserta el caracter ")" en el selector desde el cual fue invocado

comando	(+)	insertar declaracion	2	Inserta la declaración debajo del selector o declaración desde el cual fue invocado
propiedad	(+)	width	324	Inserta la propiedad en la declaración desde el cual fue invocado
texto	(+)	4	-	Inserta el texto en el selector o declaración desde el cual fue invocado
caracter	(+)	porcentaje	25	Inserta el caracter "%" indicado en la declaración desde el cual fue invocado
comando	(+)	insertar declaracion	2	Inserta la declaración debajo del selector o declaración desde el cual fue invocado
propiedad	(+)	text-align	288	Inserta la propiedad en la declaración desde el cual fue invocado
valor	(+)	center	2	Inserta el valor en la declaración desde el cual fue invocado
comando	(+)	insertar selector	1	Inserta un nuevo selector en el modelo físico
comando	(+)	pegar	18	Inserta el contenido guardado, en la posición actual del cursor, del selector o declaración desde el cual fue invocado
comando	(+)	final	16	Asigna el cursor al final del contenido del selector o declaración desde el cual fue invocado
comando	(+)	espacio	15	Inserta un espacio en blanco " " en el selector o la declaración desde el cual fue invocado
selector	(+)	tbody	97	Inserta el contenido del selector en el selector desde el cual fue invocado
comando	(+)	espacio	15	Inserta un espacio en blanco " " en el selector o la declaración desde el cual fue invocado
selector	(+)	td	98	Inserta el contenido del selector en el selector desde el cual fue invocado
clase	(+)	nth-child	35	Inserta la pseudoclase en el selector desde el cual fue invocado
caracter	(+)	parentesis inicio	24	Inserta el caracter "(" en el selector desde el cual fue invocado

texto	(+)	3	-	Inserta el texto en el selector o declaración desde el cual fue invocado
caracter	(+)	parentesis fin	23	Inserta el caracter ")" en el selector desde el cual fue invocado
comando	(+)	insertar declaracion	2	Inserta la declaración debajo del selector o declaración desde el cual fue invocado
propiedad	(+)	width	324	Inserta la propiedad en la declaración desde el cual fue invocado
texto	(+)	17	-	Inserta el texto en el selector o declaración desde el cual fue invocado
caracter	(+)	porcentaje	25	Inserta el caracter "%" indicado en la declaración desde el cual fue invocado
comando	(+)	insertar selector	1	Inserta un nuevo selector en el modelo físico
comando	(+)	pegar	18	Inserta el contenido guardado, en la posición actual del cursor, del selector o declaración desde el cual fue invocado
comando	(+)	final	16	Asigna el cursor al final del contenido del selector o declaración desde el cual fue invocado
comando	(+)	espacio	15	Inserta un espacio en blanco " " en el selector o la declaración desde el cual fue invocado
selector	(+)	tbody	97	Inserta el contenido del selector en el selector desde el cual fue invocado
comando	(+)	espacio	15	Inserta un espacio en blanco " " en el selector o la declaración desde el cual fue invocado
selector	(+)	td	98	Inserta el contenido del selector en el selector desde el cual fue invocado
clase	(+)	nth-child	35	Inserta la pseudoclase en el selector desde el cual fue invocado
caracter	(+)	parentesis inicio	24	Inserta el caracter "(" en el selector desde el cual fue invocado
texto	(+)	4	-	Inserta el texto en el selector o declaración desde el cual fue invocado

caracter	(+)	parentesis fin	23	Inserta el caracter ")" en el selector desde el cual fue invocado
comando	(+)	insertar declaracion	2	Inserta la declaración debajo del selector o declaración desde el cual fue invocado
propiedad	(+)	width	324	Inserta la propiedad en la declaración desde el cual fue invocado
texto	(+)	7	-	Inserta el texto en el selector o declaración desde el cual fue invocado
caracter	(+)	porcentaje	25	Inserta el caracter "%" indicado en la declaración desde el cual fue invocado
comando	(+)	insertar declaracion	2	Inserta la declaración debajo del selector o declaración desde el cual fue invocado
propiedad	(+)	text-align	288	Inserta la propiedad en la declaración desde el cual fue invocado
valor	(+)	center	2	Inserta el valor en la declaración desde el cual fue invocado
comando	(+)	insertar selector	1	Inserta un nuevo selector en el modelo físico
comando	(+)	pegar	18	Inserta el contenido guardado, en la posición actual del cursor, del selector o declaración desde el cual fue invocado
comando	(+)	final	16	Asigna el cursor al final del contenido del selector o declaración desde el cual fue invocado
comando	(+)	espacio	15	Inserta un espacio en blanco " " en el selector o la declaración desde el cual fue invocado
selector	(+)	tbody	97	Inserta el contenido del selector en el selector desde el cual fue invocado
comando	(+)	espacio	15	Inserta un espacio en blanco " " en el selector o la declaración desde el cual fue invocado
selector	(+)	td	98	Inserta el contenido del selector en el selector desde el cual fue invocado
clase	(+)	nth-child	35	Inserta la pseudoclase en el selector desde el cual fue invocado

caracter	(+)	parentesis inicio	24	Inserta el caracter "(" en el selector desde el cual fue invocado
texto	(+)	5	-	Inserta el texto en el selector o declaración desde el cual fue invocado
caracter	(+)	parentesis fin	23	Inserta el caracter ")" en el selector desde el cual fue invocado
comando	(+)	insertar declaracion	2	Inserta la declaración debajo del selector o declaración desde el cual fue invocado
propiedad	(+)	width	324	Inserta la propiedad en la declaración desde el cual fue invocado
texto	(+)	64	-	Inserta el texto en el selector o declaración desde el cual fue invocado
caracter	(+)	porcentaje	25	Inserta el caracter "%" indicado en la declaración desde el cual fue invocado
comando	(+)	insertar selector	1	Inserta un nuevo selector en el modelo físico
comando	(+)	pegar	18	Inserta el contenido guardado, en la posición actual del cursor, del selector o declaración desde el cual fue invocado
comando	(+)	final	16	Asigna el cursor al final del contenido del selector o declaración desde el cual fue invocado
comando	(+)	espacio	15	Inserta un espacio en blanco " " en el selector o la declaración desde el cual fue invocado
selector	(+)	tbody	97	Inserta el contenido del selector en el selector desde el cual fue invocado
comando	(+)	espacio	15	Inserta un espacio en blanco " " en el selector o la declaración desde el cual fue invocado
selector	(+)	td	98	Inserta el contenido del selector en el selector desde el cual fue invocado
clase	(+)	nth-child	35	Inserta la pseudoclase en el selector desde el cual fue invocado
caracter	(+)	parentesis inicio	24	Inserta el caracter "(" en el selector desde el cual fue invocado

texto	(+)	1	-	Inserta el texto en el selector o declaración desde el cual fue invocado
caracter	(+)	parentesis fin	23	Inserta el caracter ")" en el selector desde el cual fue invocado
caracter	(+)	coma	4	Inserta el caracter "," en el selector desde el cual fue invocado
comando	(+)	espacio	15	Inserta un espacio en blanco " " en el selector o la declaración desde el cual fue invocado
comando	(+)	pegar	18	Inserta el contenido guardado, en la posición actual del cursor, del selector o declaración desde el cual fue invocado
comando	(+)	final	16	Asigna el cursor al final del contenido del selector o declaración desde el cual fue invocado
comando	(+)	espacio	15	Inserta un espacio en blanco " " en el selector o la declaración desde el cual fue invocado
selector	(+)	tbody	97	Inserta el contenido del selector en el selector desde el cual fue invocado
comando	(+)	espacio	15	Inserta un espacio en blanco " " en el selector o la declaración desde el cual fue invocado
selector	(+)	td	98	Inserta el contenido del selector en el selector desde el cual fue invocado
clase	(+)	nth-child	35	Inserta la pseudoclase en el selector desde el cual fue invocado
caracter	(+)	parentesis inicio	24	Inserta el caracter "(" en el selector desde el cual fue invocado
texto	(+)	2	-	Inserta el texto en el selector o declaración desde el cual fue invocado
caracter	(+)	parentesis fin	23	Inserta el caracter ")" en el selector desde el cual fue invocado
caracter	(+)	coma	4	Inserta el caracter "," en el selector desde el cual fue invocado

comando	(+)	espacio	15	Inserta un espacio en blanco " " en el selector o la declaración desde el cual fue invocado
comando	(+)	pegar	18	Inserta el contenido guardado, en la posición actual del cursor, del selector o declaración desde el cual fue invocado
comando	(+)	final	16	Asigna el cursor al final del contenido del selector o declaración desde el cual fue invocado
comando	(+)	espacio	15	Inserta un espacio en blanco " " en el selector o la declaración desde el cual fue invocado
selector	(+)	tbody	97	Inserta el contenido del selector en el selector desde el cual fue invocado
comando	(+)	espacio	15	Inserta un espacio en blanco " " en el selector o la declaración desde el cual fue invocado
selector	(+)	td	98	Inserta el contenido del selector en el selector desde el cual fue invocado
clase	(+)	nth-child	35	Inserta la pseudoclase en el selector desde el cual fue invocado
caracter	(+)	parentesis inicio	24	Inserta el caracter "(" en el selector desde el cual fue invocado
texto	(+)	3	-	Inserta el texto en el selector o declaración desde el cual fue invocado
caracter	(+)	parentesis fin	23	Inserta el caracter ")" en el selector desde el cual fue invocado
caracter	(+)	coma	4	Inserta el caracter "," en el selector desde el cual fue invocado
comando	(+)	espacio	15	Inserta un espacio en blanco " " en el selector o la declaración desde el cual fue invocado
comando	(+)	pegar	18	Inserta el contenido guardado, en la posición actual del cursor, del selector o declaración desde el cual fue invocado

comando	(+)	final	16	Asigna el cursor al final del contenido del selector o declaración desde el cual fue invocado
comando	(+)	espacio	15	Inserta un espacio en blanco " " en el selector o la declaración desde el cual fue invocado
selector	(+)	tbody	97	Inserta el contenido del selector en el selector desde el cual fue invocado
comando	(+)	espacio	15	Inserta un espacio en blanco " " en el selector o la declaración desde el cual fue invocado
selector	(+)	td	98	Inserta el contenido del selector en el selector desde el cual fue invocado
clase	(+)	nth-child	35	Inserta la pseudoclase en el selector desde el cual fue invocado
caracter	(+)	parentesis inicio	24	Inserta el caracter "(" en el selector desde el cual fue invocado
texto	(+)	4	-	Inserta el texto en el selector o declaración desde el cual fue invocado
caracter	(+)	parentesis fin	23	Inserta el caracter ")" en el selector desde el cual fue invocado
caracter	(+)	coma	4	Inserta el caracter "," en el selector desde el cual fue invocado
comando	(+)	espacio	15	Inserta un espacio en blanco " " en el selector o la declaración desde el cual fue invocado
comando	(+)	pegar	18	Inserta el contenido guardado, en la posición actual del cursor, del selector o declaración desde el cual fue invocado
comando	(+)	final	16	Asigna el cursor al final del contenido del selector o declaración desde el cual fue invocado
comando	(+)	espacio	15	Inserta un espacio en blanco " " en el selector o la declaración desde el cual fue invocado
selector	(+)	tbody	97	Inserta el contenido del selector en el selector desde el cual fue invocado

comando	(+)	espacio	15	Inserta un espacio en blanco " " en el selector o la declaración desde el cual fue invocado
selector	(+)	td	98	Inserta el contenido del selector en el selector desde el cual fue invocado
clase	(+)	nth-child	35	Inserta la pseudoclase en el selector desde el cual fue invocado
caracter	(+)	parentesis inicio	24	Inserta el caracter "(" en el selector desde el cual fue invocado
texto	(+)	5	-	Inserta el texto en el selector o declaración desde el cual fue invocado
caracter	(+)	parentesis fin	23	Inserta el caracter ")" en el selector desde el cual fue invocado
comando	(+)	insertar declaracion	2	Inserta la declaración debajo del selector o declaración desde el cual fue invocado
propiedad	(+)	color	98	Inserta la propiedad en la declaración desde el cual fue invocado
valor	(+)	blue	10	Inserta el valor en la declaración desde el cual fue invocado
comando	(+)	insertar selector	1	Inserta un nuevo selector en el modelo físico
comando	(+)	pegar	18	Inserta el contenido guardado, en la posición actual del cursor, del selector o declaración desde el cual fue invocado
comando	(+)	final	16	Asigna el cursor al final del contenido del selector o declaración desde el cual fue invocado
comando	(+)	espacio	15	Inserta un espacio en blanco " " en el selector o la declaración desde el cual fue invocado
selector	(+)	tbody	97	Inserta el contenido del selector en el selector desde el cual fue invocado
comando	(+)	espacio	15	Inserta un espacio en blanco " " en el selector o la declaración desde el cual fue invocado
selector	(+)	td	98	Inserta el contenido del selector en el selector desde el cual fue invocado

clase	(+)	nth-child	35	Inserta la pseudoclase en el selector desde el cual fue invocado
caracter	(+)	parentesis inicio	24	Inserta el caracter "(" en el selector desde el cual fue invocado
texto	(+)	2	-	Inserta el texto en el selector o declaración desde el cual fue invocado
caracter	(+)	parentesis fin	23	Inserta el caracter ")" en el selector desde el cual fue invocado
comando	(+)	insertar declaracion	2	Inserta la declaración debajo del selector o declaración desde el cual fue invocado
propiedad	(+)	color	98	Inserta la propiedad en la declaración desde el cual fue invocado
valor	(+)	red	122	Inserta el valor en la declaración desde el cual fue invocado
comando	(+)	insertar selector	1	Inserta un nuevo selector en el modelo físico
comando	(+)	pegar	18	Inserta el contenido guardado, en la posición actual del cursor, del selector o declaración desde el cual fue invocado
comando	(+)	final	16	Asigna el cursor al final del contenido del selector o declaración desde el cual fue invocado
comando	(+)	espacio	15	Inserta un espacio en blanco " " en el selector o la declaración desde el cual fue invocado
selector	(+)	tfoot	100	Inserta el contenido del selector en el selector desde el cual fue invocado
comando	(+)	insertar declaracion	2	Inserta la declaración debajo del selector o declaración desde el cual fue invocado
propiedad	(+)	text-align	288	Inserta la propiedad en la declaración desde el cual fue invocado
valor	(+)	center	2	Inserta el valor en la declaración desde el cual fue invocado
comando	(+)	insertar declaracion	2	Inserta la declaración debajo del selector o declaración desde el cual fue invocado

propiedad	(+)	color	98	Inserta la propiedad en la declaración desde el cual fue invocado
valor	(+)	red	122	Inserta el valor en la declaración desde el cual fue invocado
comando	(+)	insertar selector	1	Inserta un nuevo selector en el modelo físico
comando	(+)	pegar	18	Inserta el contenido guardado, en la posición actual del cursor, del selector o declaración desde el cual fue invocado
comando	(+)	final	16	Asigna el cursor al final del contenido del selector o declaración desde el cual fue invocado
comando	(+)	espacio	15	Inserta un espacio en blanco " " en el selector o la declaración desde el cual fue invocado
selector	(+)	tfoot	100	Inserta el contenido del selector en el selector desde el cual fue invocado
comando	(+)	espacio	15	Inserta un espacio en blanco " " en el selector o la declaración desde el cual fue invocado
selector	(+)	h4	42	Inserta el contenido del selector en el selector desde el cual fue invocado
comando	(+)	insertar declaracion	2	Inserta la declaración debajo del selector o declaración desde el cual fue invocado
propiedad	(+)	text-decoration	291	Inserta la propiedad en la declaración desde el cual fue invocado
valor	(+)	overline	11	Inserta el valor en la declaración desde el cual fue invocado
comando	(+)	insertar selector	1	Inserta un nuevo selector en el modelo físico
selector	(+)	div	29	Inserta el contenido del selector en el selector desde el cual fue invocado
comando	(+)	insertar declaracion	2	Inserta la declaración debajo del selector o declaración desde el cual fue invocado
propiedad	(+)	background-color	19	Inserta la propiedad en la declaración desde el cual fue invocado

caracter	(+)	numeral	21	Inserta el caracter "#" indicado en la declaración desde el cual fue invocado
Texto	(+)	f	-	Inserta el texto en el selector o declaración desde el cual fue invocado
texto	(+)	f	-	Inserta el texto en el selector o declaración desde el cual fue invocado
texto	(+)	f	-	Inserta el texto en el selector o declaración desde el cual fue invocado
texto	(+)	f	-	Inserta el texto en el selector o declaración desde el cual fue invocado
abecedario	(+)	si	4	Inserta el texto "c" en el selector o declaración desde el cual fue invocado
abecedario	(+)	si	4	Inserta el texto "c" en el selector o declaración desde el cual fue invocado