



INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

Trabajo Terminal I.

**Generador de versos musicales en el idioma
inglés por medio de procesamiento de
lenguaje natural y redes neuronales**

TT2021-AXXX.

Integrantes:

Espinosa de los Monteros Lechuga Jaime Daniel
Nava Romo Edgar Adrián
Salgado Gómez Alfredo Emilio

Directores:

Kolesnikova Olga
López Rojas Ariel

Índice.

I Trabajo Terminal I	11
1. Introducción	12
1.1. Objetivos	15
1.1.1. Objetivo general	15
1.1.2. Objetivos particulares	15
1.2. Trabajo previo	15
1.2.1. Trabajo Previo 1	16
1.2.2. Trabajo Previo 2	16
1.3. Justificación	17
1.3.1. Limites y alcances	18
1.4. Metodología	21
1.5. Organización del documento	30
1.5.1. Capítulo 2. Marco teórico	30
1.5.2. Capítulo 3. Análisis	31
1.5.3. Capítulo 4. Diseño	31
1.5.4. Capítulo 5. Desarrollo	31
1.5.5. Capítulo 6. Avances y trabajo por hacer	31
2. Marco teórico	32
2.0.1. Github Student Pack	32
2.1. Redes neuronales	34
2.1.1. Tipos de Redes Neuronales	35
2.2. Base de datos	41
2.2.1. Tipos de bases de datos	42
2.3. ¿Qué es una canción?	43
2.3.1. Elementos de una canción	43
2.3.2. Introducción	43
2.3.3. Verso	43
2.3.4. Pre-estribillo	43
2.3.5. Estribillo	43
2.3.6. Puente	44

2.3.7.	Cierre	44
2.3.8.	Estructura de una canción	44
2.4.	Cadenas de Markov	45
2.5.	Flask	46
2.6.	Apache	50
2.7.	Servidor Web	51
2.8.	Certificado SSL	52
2.8.1.	Objetivos de la criptografía.	52
2.8.2.	Usos de la criptografía.	53
2.8.3.	Criptografía simétrica y asimétrica.	53
2.8.4.	Criptografía a nivel de aplicación.	56
2.9.	Tema 6	58
2.9.1.	Historia	58
2.9.2.	¿Qué es Tema 6?	58
2.9.3.	Objetivo de Tema 6	60
2.9.4.	¿Cómo funciona?	61
2.9.5.	Propiedades de Tema 6	63
2.9.6.	All-or-Nothing and the Package Transform (AONT) . .	63
2.9.7.	Comparando Chaffing and Winnowing contra Cifrado y Esteganografía	66
Sprint 1.		69
3. Análisis.		70
3.1.	Estudio de Factibilidad.	70
3.1.1.	Factibilidad Técnica	70
3.1.2.	Factibilidad Operativa	72
3.1.3.	Factibilidad Económica	73
3.2.	Herramientas a usar.	74
3.2.1.	Software.	74
3.2.2.	Hardware.	84
3.3.	Arquitectura del sistema.	85
3.3.1.	Descripción de la arquitectura del sistema.	85
3.4.	Diagrama BPMN	87
3.4.1.	Diagrama BPMN Proceso: Inicio de Sesión	88
3.5.	Diagrama de casos de uso general.	89
3.6.	Componente I. Extensión.	89
3.6.1.	Descripción.	89
3.6.2.	Estudio de requerimientos.	90
3.6.3.	Reglas del negocio.	92

3.7.	Componente II: Servidor autentificador.	94
3.7.1.	Descripción.	94
3.7.2.	Estudio de requerimientos.	94
3.7.3.	Reglas del negocio.	95
3.8.	Componente III: API.	96
3.8.1.	Descripción.	96
3.8.2.	Estudio de requerimientos.	97
3.8.3.	Reglas del negocio.	98
4.	Diseño.	100
4.1.	Componente I: Extensión.	100
4.1.1.	Diagrama de casos de uso	100
4.1.2.	Diagrama de flujo.	113
4.1.3.	Diagrama de flujo de datos.	115
4.1.4.	Diagrama de clases.	116
4.1.5.	Diagramas de secuencia.	119
4.1.6.	Diagrama de actividades	126
4.1.7.	Interfaz de usuario.	127
4.1.8.	Requisitos de diseño.	137
4.1.9.	Algoritmos.	138
4.2.	Componente II: Servidor Autentificador.	141
4.2.1.	Diagrama de casos de uso.	141
4.2.2.	Diagrama de flujo.	149
4.2.3.	Diagrama de flujo de datos.	150
4.2.4.	Diagrama de clases.	151
4.2.5.	Diagramas de secuencias.	154
4.2.6.	Diagrama de actividades.	158
4.3.	Componente III: API.	159
4.3.1.	Diagrama de casos de uso.	159
4.3.2.	Diagrama de flujo.	164
4.3.3.	Diagrama de flujo de datos.	165
4.3.4.	Diagrama de clases.	166
4.3.5.	Diagramas de secuencias.	169
4.3.6.	Diagrama de actividades	172
4.3.7.	Interfaz de usuario.	172
4.3.8.	Algoritmos.	176
5.	Desarrollo.	180
5.1.	Componente I. Extensión.	180
5.1.1.	Archivo popup.html	180
5.1.2.	Archivo background.js	181

5.2.	Componente II. Servidor Autentificador.	182
5.2.1.	Manejador de paquetes de Node (npm)	182
5.2.2.	Componentes.	183
5.3.	Componente III. API.	184
5.3.1.	API	185
5.3.2.	Servicio Web.	187
6.	Pruebas.	189
6.1.	Pruebas de integración.	189
6.1.1.	Extensión y Servidor autentificador.	189
6.1.2.	Extensión y Servicio Web.	190
6.1.3.	Servidor Autentificador y API.	190
6.2.	Tiempos de ejecución.	190
6.2.1.	Algoritmo de Chaffing.	191
6.2.2.	Algoritmo de Winnowing.	191
6.2.3.	Inicio de sesión.	191
Trabajo Terminal II		192
Sprint 2.		193
7.	Análisis.	194
7.1.	Arquitectura del sistema.	194
7.1.1.	Descripción de la arquitectura del sistema.	194
7.2.	Componente I. Extensión.	195
7.2.1.	Descripción.	195
7.2.2.	Cambios planteados.	195
7.2.3.	Cambios al estudio de requerimientos y reglas de negocio.	196
7.3.	Componente II: Servidor autentificador.	196
7.3.1.	Descripción.	196
7.3.2.	Cambios Planteados.	197
7.3.3.	Cambios al estudio de requerimientos y reglas de negocio.	197
7.4.	Componente III: API.	198
7.4.1.	Descripción.	198
7.4.2.	Cambios planteados.	198
7.4.3.	Cambios al estudio de requerimientos y reglas de negocio.	198

8. Diseño.	200
8.1. Componente I: Extensión.	200
8.1.1. Cambios en el algoritmo.	200
8.2. Componente II: Servidor Autentificador.	202
8.2.1. Cambios realizados.	202
8.2.2. Diagrama de Actividades.	202
8.2.3. Diagrama de Secuencia.	202
8.3. Componente III: API.	203
8.3.1. Cambios realizados.	203
8.3.2. Cambio en el algoritmo.	203
8.3.3. Diagrama de Flujo.	205
9. Desarrollo.	206
9.1. Componente I: Extensión.	206
9.1.1. Cambios en el manifest.	206
9.1.2. Implementación del Chaffing.	206
9.2. Componente II: Servidor Autentificador.	207
9.2.1. Servidor de accesos FTP.	207
9.3. Componente III: API.	208
9.3.1. Nuevo Proceso de Winnowing y eliminación del descifrado AES.	208
9.3.2. Guardado del certificado	208
10. Pruebas.	209
10.1. Pruebas de integración.	209
10.1.1. Extensión y Servidor autentificador	209
10.1.2. Extensión y Servicio Web	210
10.1.3. Servidor Autentificador y API	210
10.2. Pruebas funcionales.	210
10.3. Tiempos de ejecución.	211
10.3.1. Algoritmo de Chaffing.	212
10.3.2. Algoritmo de Winnowing.	212
10.3.3. Inicio de sesión.	212
11. Conclusiones	214
12. Trabajo a futuro.	215

Índice de figuras.

1.1.	Fases de la metodología por prototipos	22
2.1.	Github realacionado con el correo institucional	33
2.2.	Tiempo de espera de aprobación de la solicitud	34
2.3.	Activación del students pack	34
2.4.	Diagrama Red Neuronal Feedforward	36
2.5.	Diagrama Red Neuronal Radial Basis Function	37
2.6.	Diagrama Red Neuronal Multilayer Perception	38
2.7.	Diagrama Red Neuronal Convolucional	38
2.8.	Diagrama Red Neuronal Recurrente	40
2.9.	Diagrama Red Neuronal Modular	41
2.10.	Diagrama De Estados finitos De Una Cadena De Markov	46
2.11.	Esquema del protocolo de criptografía simétrica.	54
2.12.	Esquema del protocolo de criptografía asimétrica.	55
2.13.	Charles agrega los paquetes inválidos.	59
2.14.	Charles no agrega los paquetes pero multiplexa los flujos.	60
2.15.	Secuencia de Chaffing después del proceso de autentificación.	61
2.16.	Formas del proceso de chaff	62
2.17.	Visión general del proceso Chaffing and Winnowing.	63
2.18.	Proceso de Chaffing and Winnowing junto con AONT.	65
3.1.	Arquitectura General del Sistema	85
3.2.	Diagrama de Modelo y Notación de Procesos de Negocio	87
3.3.	Diagrama BPMN del proceso Iniciar Sesión	88
3.4.	Diagrama de casos de uso general del sistema	89
4.1.	Diagrama de casos de uso del Componente I.	100
4.2.	Diagrama de flujo del Componente I.	113
4.3.	Diagrama de flujo de datos del Componente I.	115
4.4.	Diagrama de clases de Componente I.	116
4.5.	Diagrama de secuencia 1 del Componente I	119
4.6.	Diagrama de secuencia 2 del Componente I	120

4.7.	Diagrama de secuencia 3 del Componente I	121
4.8.	Diagrama de secuencia 4 del Componente I	122
4.9.	Diagrama de secuencia 5 del Componente I	123
4.10.	Diagrama de secuencia 6 del Componente I	124
4.11.	Diagrama de secuencia 7 del Componente I	125
4.12.	Diagrama de actividades del Prototipo 2.	126
4.13.	Pantalla de interfaz de la extensión.	127
4.14.	Pantalla inicial.	128
4.15.	Pantalla de aviso.	129
4.16.	Tab de la extensión. Permite inicio de sesión	130
4.17.	Mensaje de éxito	130
4.18.	Mensaje de error	131
4.19.	Tab de la extensión. Permite registrarse	132
4.20.	Mensaje de éxito	133
4.21.	Mensaje de error	133
4.22.	Mensaje de error	134
4.23.	Mensaje de error	134
4.24.	Mensaje de error	135
4.25.	Interfaz	135
4.26.	Mensaje de éxito	136
4.27.	Mensaje de error	136
4.28.	Arreglo del patrón de chaffing final	140
4.29.	Diagrama de casos de uso del Componente II.	141
4.30.	Diagrama de flujo de Componente II.	149
4.31.	Diagrama de flujo de datos del Componente II.	150
4.32.	Diagrama de clases de Componente II.	151
4.33.	Diagrama de secuencia 1 del Componente II	154
4.34.	Diagrama de secuencia 2 del Componente II	155
4.35.	Diagrama de secuencia 3 del Componente II	156
4.36.	Diagrama de secuencia 4 del Componente II	157
4.37.	Diagrama de actividades de Componente II.	158
4.38.	Diagrama de caso de uso del componente III	159
4.39.	Diagrama de flujo de Componente III.	164
4.40.	Diagrama de flujo de datos de Componente III.	165
4.41.	Diagrama de Clases de componente 3.	166
4.42.	Diagrama de Secuencia de Caso de Uso 1. Comprobar certificado.	169
4.43.	Diagrama de Secuencia de Caso de Uso 2. Enviar petición.	170
4.44.	Diagrama de Secuencia de Caso de Uso 3. Redirigir página.	171
4.45.	Diagrama de Actividades de componente 3.	172
4.46.	Mensaje de éxito	173
4.47.	Interfaz	174

4.48. Interfaz	174
4.49. Interfaz	175
4.50. Interfaz	175
4.51. Mensaje de error	176
5.1. Estructura del desarrollo de la Autoridad Certificadora.	183
5.2. Estructura del desarrollo de la API.	186
6.1. Sesión iniciada	190
6.2. Resultados de la API al recibir petición de la extensión.	190
6.3. Resultados de la comunicación entre la API y el Servidor Autentificador.	190
7.1. Arquitectura General del Sistema	194
8.1. Ajustes al Diagrama de Actividades del Componente II.	202
8.2. Ajustes al Diagrama de Secuencia crear nuevo usuario del componente II.	202
8.3. Ajustes al Diagrama de Secuencia revocar certificado del componente II.	202
8.4. Ajustes al Diagrama de Secuencia obtener certificado del componente II.	203
8.5. Ajustes al Diagrama de Secuencia verificar certificado del componente II.	203
8.6. Ajustes al Diagrama de Flujo Componente III.	205
10.1. Respuesta de la Autoridad Certificadora hacia la Extensión . .	209
10.2. Sesión iniciada con éxito (Obtención de certificado de la Autoridad certificadora.)	210
10.3. Analizador de protocolos Wireshark de la petición dada por el usuario hacia servicio web.)	210
10.4. Analizador de protocolos Wireshark de la petición dada por la API hacia la autoridad certificadora.)	210
10.5. Inicio de sesión en la extensión	210
10.6. Certificado del cliente devuelto por el Servidor Autentificador	211
10.7. Página de inicio del Servicio Web	211
10.8. Inicio de sesión con la extensión habilitada.	211
10.9. Inicio de sesión exitoso a la cuenta con el certificado asignado.	211

Índice de cuadros.

1.1. Resumen de productos similares comparados con nuestra pro- puesta	14
2.1. Aplicación de los métodos de autentificación	48
2.2. Seguridad en los métodos de autentificación	49
2.3. Tabla comparativa de los distintos servidores web contemplados	51
3.1. Herramientas de Software	71
3.2. Equipo de Hardware 1	71
3.3. Equipo de Hardware 2	71
3.4. Equipo de Hardware 3	72
3.5. Horas de trabajo	73
4.1. DCU: CI_CU1	101
4.2. DCU: CI_CU2	102
4.3. DCU: CI_CU3	103
4.4. DCU: CI_CU4	104
4.5. DCU: CI_CU5	107
4.6. DCU: CI_CU6	108
4.7. DCU: CI_CU7	111
4.8. DCU: CII_CU1	142
4.9. DCU: CII_CU2	144
4.10. DCU: CII_CU3	146
4.11. DCU: CII_CU4	148
4.12. DCU: CIII_CU1	160
4.13. DCU: CIII_CU2	162
4.14. DCU: CIII_CU3	163

Parte I

Trabajo Terminal I

Capítulo 1

Introducción

En la actualidad la industria musical obtiene ganancias a través de la creación y divulgación de la música de manera física y digital (Bourreau and Gensollen 2006 [1]), dejando que aficionados y emprendedores de la música no tengan oportunidad de avanzar en su carrera por falta de creatividad, tiempo y/o recursos, haciendo que la creación de nuevas letras para sus canciones sea un gran obstáculo, nuestra propuesta implica la utilización de nuevas tecnologías que permitan la generación de letras para integrar con sus canciones. La generación de texto es una de las tareas más populares y desafiantes en el área de procesamiento del lenguaje natural. Recientemente, hay gran cantidad de trabajos (Generating Text with Recurrent Neural Networks [2], Convolutional Neural Networks for Sentence Classification [3]) los cuales propusieron generar texto utilizando redes neuronales recurrentes y redes neuronales convolucionales. Sin embargo, la mayoría de los trabajos actuales solo se enfocan en generar una o varias oraciones, ni siquiera un párrafo largo y mucho menos una canción completa.

Las letras de canciones, como un tipo de texto, tienen algunas características propias, estas se constituyen de rimas, versos, coros y en algunos casos, patrones de repetición. Coro se refiere a la parte de una canción que se repite sin modificaciones dentro de la misma después de un verso. Mientras que en el verso suelen cambiar una o varias líneas que lo componen. Estas características hacen que generar letras musicales sea mucho más difícil que generar textos normales.

La mayoría de las investigaciones actuales sobre generación de letras vienen con muchas condiciones, como dar una pieza de melodía (Automatic Generation of Melodic Accompaniments for Lyrics [4]), o solo generar un tipo específico de letra (Conditional Rap Lyrics Generation with Denoising Autoencoders [5]). Sin embargo, la generación de letras por medio de inteligencia artificial dado un estilo y un tema es un asunto poco trabajado. Por

lo tanto, planeamos centrarnos en este nuevo problema. Estamos interesados en ver si el modelo propuesto puede aprender diferentes características en un género musical y generar letras que sean acorde a este. Actualmente, en el mercado se encuentran cuatro aplicaciones web que tienen una funcionalidad similar a la propuesta en este Trabajo Terminal:

- These lyrics do not exist.
- Bored humans - lyrics_generator.
- DeepBeat.
- Premium Lyrics.

En el cuadro 1 que se presenta a continuación, se muestran las características de aplicaciones web similares y comparándolas con nuestra propuesta:

Cuadro 1.1: Resumen de productos similares comparados con nuestra propuesta

Software	Características	Precio en el mercado
These Lyrics do not Exist	Aplicación web que genera letras completamente originales de varios temas, hace uso de IA para generar coros y versos originales; se puede escoger el tema principal de la letra, género musical e incluso el estado de ánimo al que iría dirigido.	Gratis (Contiene Anuncios)
Boredhumans_lyrics-generator	Aplicación web en el que la IA fue entrenada con una base de datos con miles de letras para generar letras de canciones totalmente nuevas. La letra que crea es única y no una copia de alguna que exista actualmente, sin embargo, no permite customizar la letra.	Gratis
DeepBeat	Aplicación web que por medio de IA genera letras de música enfocada principalmente en el género rap. Si una línea no es del agrado se puede sustituir por alguna de las otras propuestas de las que ofrece.	Gratis
Premium Lyrics	Aplicación web que proporciona versos compuestos en distintos idiomas por artistas independientes que se escogen manualmente de acuerdo a su originalidad y calidad.	3\$ a 75\$ Dólares por letra musical
Nuestra propuesta	Aplicación web que haciendo uso de una IA va a generar letras musicales originales a partir de un género musical en exclusivo, lo que asegurará un resultado natural con coros y versos distintos cada vez que se utilice.	Gratis

1.1. Objetivos.

1.1.1. Objetivo general.

Se pretende crear una herramienta para estudiantes, aficionados o cualquier persona interesada en este rubro que se le dificulte componer nuevas letras musicales para que puedan interpretarlas y crear nuevas canciones. Haciendo uso de un conjunto de datos (dataset) y herramientas alojadas en la nube (Google Cloud Platform o Amazon Web Services) se van a procesar estos datos, se pretende crear un módulo que encuentre patrones por medio de redes neuronales para analizar la semántica mediante técnicas y herramientas ya existentes de procesamiento de lenguaje natural. Se van a realizar pruebas y experimentos con estas herramientas antes de la implementación (Bert [6], spaCy [7]) para poder generar versos. A su vez se va a desarrollar una interfaz web intuitiva en versión prototipo donde el usuario va a poder utilizar esta herramienta donde se le va a mostrar la letra musical que se va a generar en ese momento. Esta herramienta pretende ayudar a los usuarios mencionados anteriormente con el fin de impulsar la carrera de futuros artistas en la industria musical que no tengan los suficientes recursos para poder contratar servicios particulares de compositores.

1.1.2. Objetivos particulares.

- Generar un conjunto de datos (dataset) con letras musicales de un género musical para efecto de entrenamiento de la red semántica.
- Hacer uso de alguna herramienta de aprendizaje automático (machine learning) e implementar su uso en la nube para ayudar a procesar las letras musicales de un género en específico.
- Implementar un módulo analizador de semántica para entrenar redes neuronales.
- Desarrollar una interfaz web intuitiva en versión prototipo que utilice una aplicación web alojada en la nube para la visualización del verso musical generado a partir de un género.

1.2. Trabajo previo.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum

massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

1.2.1. Trabajo Previo 1

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

1.2.2. Trabajo Previo 2

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

- Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper

vestibulum turpis. Pellentesque cursus luctus mauris.

- Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

1.3. Justificación.

El crear nuevas composiciones musicales puede llegar a ser muy difícil, estresante e incluso agotador para cualquier aficionado o incluso algunos expertos en este medio, esto se debe a la falta de creatividad y/o tiempo de quien lo quiera realizar [8]. En ocasiones se pueden contratar servicios particulares para la producción de la letra de una canción, sin embargo, puede ser muy costoso y en ocasiones el resultado final no alcanza a llenar las expectativas de la inversión que se hace; por ende, se pretende crear una herramienta para estudiantes, aficionados o cualquier persona interesada en este rubro que se les dificulte componer nuevas letras musicales. Normalmente las letras musicales creadas por el humano, tienden a estar compuestas por patrones de acuerdo al género musical [9]. Algunos ejemplos de estos patrones pueden ser las rimas, enunciados, frases cortas y que tengan una semántica correcta, estos pueden ser encontrados por medio de procesamiento de lenguaje natural y una investigación profunda en la composición de letras de estos géneros. Se eligió el idioma inglés debido a que existe una gran cantidad de bases de datos para procesar, al igual que herramientas y documentación para este idioma. Nos proponemos orientar esta solución en un entorno de nube, donde la información de configuración, servicios y datos necesarios pueden mantenerse

de forma independiente a la implementación, facilitando la adaptación y flexibilidad de la plataforma. Nuestro proyecto ayudará al usuario utilizando herramientas como el procesamiento de lenguaje natural, redes neuronales, aprendizaje de máquina (machine learning) y servidores en la nube. A diferencia de los proyectos señalados en la Tabla 1 nuestra propuesta es generar letras musicales con métodos y tecnologías distintas a los que se usaron, esto es, aunque se utilicen los mismos géneros musicales, se tendrán resultados completamente diferentes con propuestas distintas. En el desarrollo de este proyecto haremos uso de los conocimientos adquiridos durante el transcurso de la carrera. Se van a utilizar técnicas de diseño de proyectos aprendidas en el curso de Ingeniería de Software , se van a aplicar los conocimientos de programación adquiridos en unidades de aprendizaje como Inteligencia Artificial , Procesamiento de Lenguaje Natural , Web Application Development , Programación Orientada a Objetos , Análisis de Algoritmos , así como técnicas de construcción de documentos y análisis de semántica vistas en Análisis y Diseño orientado a Objetos y Comunicación Oral y Escrita . [18].

1.3.1. Limites y alcances

Los límites y alcances que tienen este trabajo terminal son los que se enlistan a continuación:

Límites

- Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.
- Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat

a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

- Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.
- Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetur.
- Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Alcances

- Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit.

Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

- Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Donec odio elit, dictum in, hendrerit sit amet, egestas sed, leo. Praesent feugiat sapien aliquet odio. Integer vitae justo. Aliquam vestibulum fringilla lorem. Sed neque lectus, consectetur at, consectetur sed, eleifend ac, lectus. Nulla facilisi. Pellentesque eget lectus. Proin eu metus. Sed porttitor. In hac habitasse platea dictumst. Suspendisse eu lectus. Ut mi mi, lacinia sit amet, placerat et, mollis vitae, dui. Sed ante tellus, tristique ut, iaculis eu, malesuada ac, dui. Mauris nibh leo, facilisis non, adipiscing quis, ultrices a, dui.
- Morbi luctus, wisi viverra faucibus pretium, nibh est placerat odio, nec commodo wisi enim eget quam. Quisque libero justo, consectetur a, feugiat vitae, porttitor eu, libero. Suspendisse sed mauris vitae elit sollicitudin malesuada. Maecenas ultricies eros sit amet ante. Ut venenatis velit. Maecenas sed mi eget dui varius euismod. Phasellus aliquet volutpat odio. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Pellentesque sit amet pede ac sem eleifend consectetur. Nullam elementum, urna vel imperdiet sodales, elit ipsum pharetra ligula, ac pretium ante justo a nulla. Curabitur tristique arcu eu metus. Vestibulum lectus. Proin mauris. Proin eu nunc eu urna hendrerit faucibus. Aliquam auctor, pede consequat laoreet varius, eros tellus scelerisque quam, pellentesque hendrerit ipsum dolor sed augue. Nulla nec lacus.
- Suspendisse vitae elit. Aliquam arcu neque, ornare in, ullamcorper quis, commodo eu, libero. Fusce sagittis erat at erat tristique mollis. Maecenas sapien libero, molestie et, lobortis in, sodales eget, dui. Morbi ultrices rutrum lorem. Nam elementum ullamcorper leo. Morbi dui. Aliquam sagittis. Nunc placerat. Pellentesque tristique sodales est. Maecenas imperdiet lacinia velit. Cras non urna. Morbi eros pede, suscipit ac, varius vel, egestas non, eros. Praesent malesuada, diam id pretium elementum, eros sem dictum tortor, vel consectetur odio sem sed wisi.
- Sed feugiat. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Ut pellentesque augue sed urna. Ves-

tibulum diam eros, fringilla et, consectetuer eu, nonummy id, sapien. Nullam at lectus. In sagittis ultrices mauris. Curabitur malesuada erat sit amet massa. Fusce blandit. Aliquam erat volutpat. Aliquam euismod. Aenean vel lectus. Nunc imperdiet justo nec dolor.

1.4. Metodología.

Para el desarrollo de este trabajo terminal se utilizará la metodología ágil Scrum, debido a que este es un proceso de gestión el cual reduce la complejidad en el desarrollo de productos para satisfacer las necesidades de los clientes. Además, permite trabajar de manera eficiente colaborativamente, es decir, en equipo, para obtener el mejor resultado posible. Ken Schwaber y Jeff Sutherland [10] explican Scrum de una manera clara y simple. Dicen que no es una colección de partes y/o componentes definidos de manera prescriptiva, sino que está basado en un modelo de proceso empírico, basado en la autoorganización de los equipos los cuales logran lidiar con lo imprevisible, resolviendo los problemas complejos inspeccionándolos y adaptándose continuamente. Scrum contiene los siguientes eventos:

- Planificación del Sprint (Sprint Planning)
- Scrum Diario (Daily Scrum)
- Revisión del Sprint (Sprint Review)
- Retrospectiva del Sprint (Sprint Retrospective)

Estos eventos existen con el fin de establecer una regularidad y minimizar la necesidad de reuniones no definidas en Scrum. Estos eventos son bloques de tiempo (time boxes), de tal forma que todos cuentan con una duración máxima. También se definen los siguientes artefactos:

- Lista de Producto (Product Backlog)
- Lista de Pendientes del Sprint (Sprint Backlog)
- Incremento (Increment)

Los artefactos en Scrum se definen para así fomentar la transparencia de la información de tal manera que todos los involucrados tengan el mismo entendimiento de que es lo que se está llevando a cabo, además de que nos crean oportunidades para realizar inspecciones y adaptaciones. Se nos permite crear Sprints, los cuales son ciclos breves de un mes o menos con diferentes

fases, en las cuales al final de cada ciclo se define la fecha para la entrega de una versión del producto deseado. Debido a que se trata de una versión, no se indica la finalización del proyecto, sino que habrá un mantenimiento constante para que se obtenga un producto final óptimo.

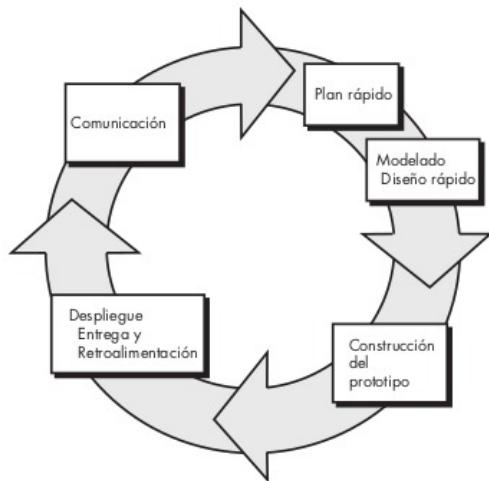


Figura 1.1: Diagrama de los fases de la metodología de prototipos evolutivos.

Nulla in ipsum. Praesent eros nulla, congue vitae, euismod ut, commodo a, wisi. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Aenean nonummy magna non leo. Sed felis erat, ullamcorper in, dictum non, ultricies ut, lectus. Proin vel arcu a odio lobortis euismod. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Proin ut est. Aliquam odio. Pellentesque massa turpis, cursus eu, euismod nec, tempor congue, nulla. Duis viverra gravida mauris. Cras tincidunt. Curabitur eros ligula, varius ut, pulvinar in, cursus faucibus, augue.

A continuación se indica cada una de las etapas de la metodología con lo que se realizará en cada una de estas:[39].

1. Nulla mattis luctus nulla. Duis commodo velit at leo. Aliquam vulputate magna et leo. Nam vestibulum ullamcorper leo. Vestibulum condimentum rutrum mauris. Donec id mauris. Morbi molestie justo et pede. Vivamus eget turpis sed nisl cursus tempor. Curabitur mollis sapien condimentum nunc. In wisi nisl, malesuada at, dignissim sit amet, lobortis in, odio. Aenean consequat arcu a ante. Pellentesque porta elit sit amet orci. Etiam at turpis nec elit ultricies imperdiet. Nulla facilisi. In hac habitasse platea dictumst. Suspendisse viverra aliquam risus.

Nullam pede justo, molestie nonummy, scelerisque eu, facilisis vel, arcu.

- Curabitur tellus magna, porttitor a, commodo a, commodo in, tortor. Donec interdum. Praesent scelerisque. Maecenas posuere sodales odio. Vivamus metus lacus, varius quis, imperdiet quis, rhoncus a, turpis. Etiam ligula arcu, elementum a, venenatis quis, sollicitudin sed, metus. Donec nunc pede, tincidunt in, venenatis vitae, faucibus vel, nibh. Pellentesque wisi. Nullam malesuada. Morbi ut tellus ut pede tincidunt porta. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam congue neque id dolor.
- Donec et nisl at wisi luctus bibendum. Nam interdum tellus ac libero. Sed sem justo, laoreet vitae, fringilla at, adipiscing ut, nibh. Maecenas non sem quis tortor eleifend fermentum. Etiam id tortor ac mauris porta vulputate. Integer porta neque vitae massa. Maecenas tempus libero a libero posuere dictum. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Aenean quis mauris sed elit commodo placerat. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Vivamus rhoncus tincidunt libero. Etiam elementum pretium justo. Vivamus est. Morbi a tellus eget pede tristique commodo. Nulla nisl. Vestibulum sed nisl eu sapien cursus rutrum.
 - Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.
 - Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.
- Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consec-

tetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

- Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.
- Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.
- Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet

orci dignissim rutrum.

- Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.
- Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.
- Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.
- Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

2. Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.
 - Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.
 - Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.
 - Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.
 - Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet,

consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

- Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.
- Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.
 - Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.
 - Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

- Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.
 - Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.
3. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

▪ Componente 1

- Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

- Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.
- Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

■ Componente 2

- Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.
- Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

■ Componente 3

- Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed

gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

- Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.
4. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

1.5. Organización del documento

Para dar inicio a este trabajo terminal, presentamos de manera breve la estructura de este reporte, con el objetivo de que el lector pueda tener un mejor entendimiento del trabajo.

1.5.1. Capítulo 2. Marco teórico.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

1.5.2. Capítulo 3. Análisis.

Dentro de este capítulo se analiza el estudio de factibilidad tanto técnico como operativo y económico con la finalidad de conocer los recursos necesarios para la elaboración de este trabajo terminal. Se mencionan resumidamente las herramientas a utilizar y se explica de manera general la arquitectura del sistema y el diagrama de casos de uso general. Finalmente se muestra el análisis de los 3 componentes que tenemos en el sistema.

1.5.3. Capítulo 4. Diseño.

En el tercer capítulo, no adentraremos en el desarrollo de los prototipos, es decir, se encuentran los diagramas pertinentes para poder modelar nuestro trabajo terminal y proceder a la etapa de codificación. En este capítulo se desarrollan y explican los siguientes diagramas: 'Casos de uso', 'Flujo', 'Flujo de datos', 'Clases', 'Secuencia' y 'Actividades' y se muestra la interfaz de usuario propuesta junto con los requisitos de diseño.

1.5.4. Capítulo 5. Desarrollo.

En este capítulo, mostraremos lo que hemos desarrollado para este TT1 (Componente I). Se muestra que el prototipo cumpla los requerimientos que se le impusieron en la sección de análisis.

1.5.5. Capítulo 6. Avances y trabajo por hacer.

En el último capítulo de este reporte, hablaremos de los avances que hemos logrado a lo largo de la asignatura de trabajo terminal I y además, exponemos el trabajo esperado para la asignatura de trabajo terminal II.

Capítulo 2

Marco teórico

2.0.1. Github Student Pack

Se solicitó Github Students Pack para obtener el dominio donde se hospeda la aplicación web por un periodo de doce meses sin ningún costo por medio de namecheap.com, para ello primero se ingresa a la página principal del servicio de Github Students Pack <https://education.github.com/pack>, se crea una cuenta junto con el correo proporcionado por el Instituto Politécnico Nacional.

ACOMODAR FOTOS DEL PROCESO CON EL PROCEDIMIENTO

Una vez que la cuenta se activó, en la página web a un lado de este nos aparecerá el nombre de la institución educativa, en este caso, el Instituto Politécnico Nacional es la institución educativa que nos respalda. Y por último se nos pide agregar el que vamos a hacer en la plataforma.

To qualify for student benefits, you must:

- Be currently enrolled in a degree or diploma granting course of study such as a high school, secondary school, college, university, homeschool, or similar educational institution.
- Have a verifiable school-issued email address or upload documents that prove your current student status.
- Have a GitHub user account.
- Be at least 13 years old.

What e-mail address do you use for school?
Note: Selecting a school-issued email address gives you the best chance of a speedy review.

alfredoe.sgomez@gmail.com

asalgadog1300@alumno.ipn.mx ✓ National Polytechnic Institute (Instituto Politécnico Nacional) (IPN)

[+ Add an email address](#)

What is the name of your school?
Note: If your school is not listed, then enter the full school name and continue. You will be asked to provide further information about your school on the next page.

National Polytechnic Institute (Instituto Politécnico)

We chose this school based on your email. If this isn't your school, please use a different email.

How do you plan to use GitHub?

Please note, your request cannot be edited once it has been submitted, so please verify your details for accuracy before sending them to us.

Submit your information

Figura 2.1: Github relacionado con el correo institucional

Ya cuando enviamos la solicitud con nuestra información nos aparece un mensaje en el cual nos dicen que obtendremos una respuesta sobre nuestra solicitud en menos de 22 días.

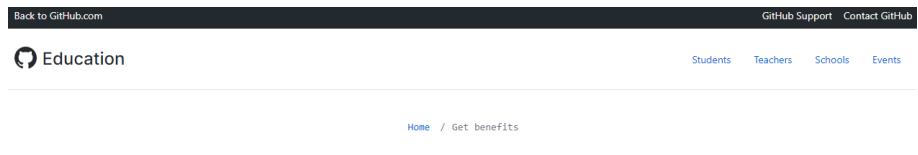


Figura 2.2: Tiempo de espera de aprobación de la solicitud

Nunca se obtuvo una respuesta en forma de correo electronico por parte de Github, sino que simplemente a la cuenta con la que solicitamos este paquete de estudiante se le otorgo, este es posible saberlo, ya que entrando a la cuenta si checamos nuestro perfil, en la parte inferior aparecerá un icono de pro, con el cual se nos da el acceso solicitado.



Figura 2.3: Activación del students pack

2.1. Redes neuronales

El aprendizaje profundo (deep learning) es, de hecho, un nuevo nombre o enfoque de la inteligencia artificial llamada redes neuronales. Las redes neuronales son una forma de hacer que las computadoras aprendan, en donde la computadora Aprende a realizar alguna tarea analizando ejemplos de entrenamiento. Por lo general, estos ejemplos han sido etiquetados previamente.

Modelado vagamente en el cerebro humano, una red neuronal consiste

en miles de millones de nodos de procesamiento los cuales están densamente interconectados. En la actualidad las redes neuronales están organizadas como capas de nodos, y estas capas están “alimentadas hacia delante” (feed-forward), es decir, la información que se mueve a través de ellas solo fluye en una sola dirección. Un solo nodo puede estar conectado a muchos nodos en capas inferiores de las cuales recibe información y a su vez, puede estar conectado a nodos en capas superiores a los cuales envía información.

Cuando una red neuronal está siendo entrenada, la información que de entrenamiento pasa a alimentar las capas inferiores (capas de entrada) la cual será procesada y pasada a posteriores capas donde será transformada hasta llegar a las capas superiores (capas de salida).[5]

2.1.1. Tipos de Redes Neuronales

Los diferentes tipos de redes neuronales usan diferentes principios para determinar sus reglas. Cada uno de los tipos de redes neuronales tienen sus propias debilidades y fortalezas, a continuación, describiremos algunas de las más utilizadas y sus aplicaciones.

2.1.1.1. Feedforward Neural Network – Artificial Neuron

Una de las redes neuronales más simples, donde la información pasa a través de diferentes nodos de entrada, hasta alcanzar los nodos de salida. En otras palabras, la información solo fluye en una sola dirección, generalmente se hace uso de esta red para tecnologías que requieran Reconocimiento de patrones o clasificación de objetos.[8]

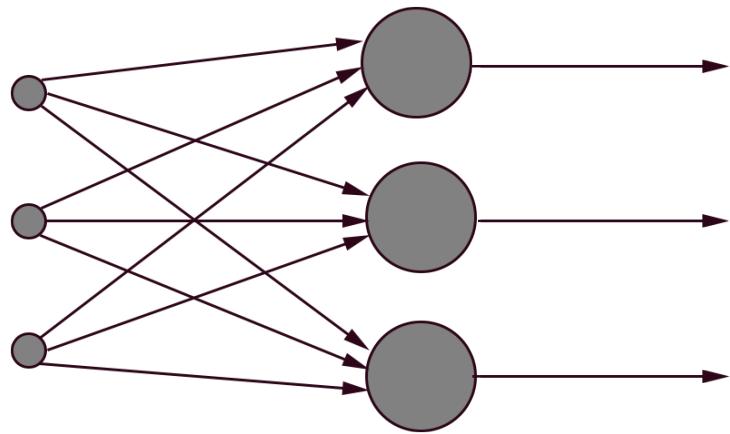


Figura 2.4: Diagrama Red Neuronal Feedforward

2.1.1.2. Radial Basis Function Neural Network

Este tipo de red considera la distancia de cualquier punto en relación con el centro. Estas redes suelen estar constituidas por 2 capas. En la capa interior, donde la información se combina con la función de base radial, para luego, las salidas de esta capa se tomen en cuenta para calcular la próxima salida. Este tipo de redes suele implementarse en sistemas que requieran una restauración.[7]

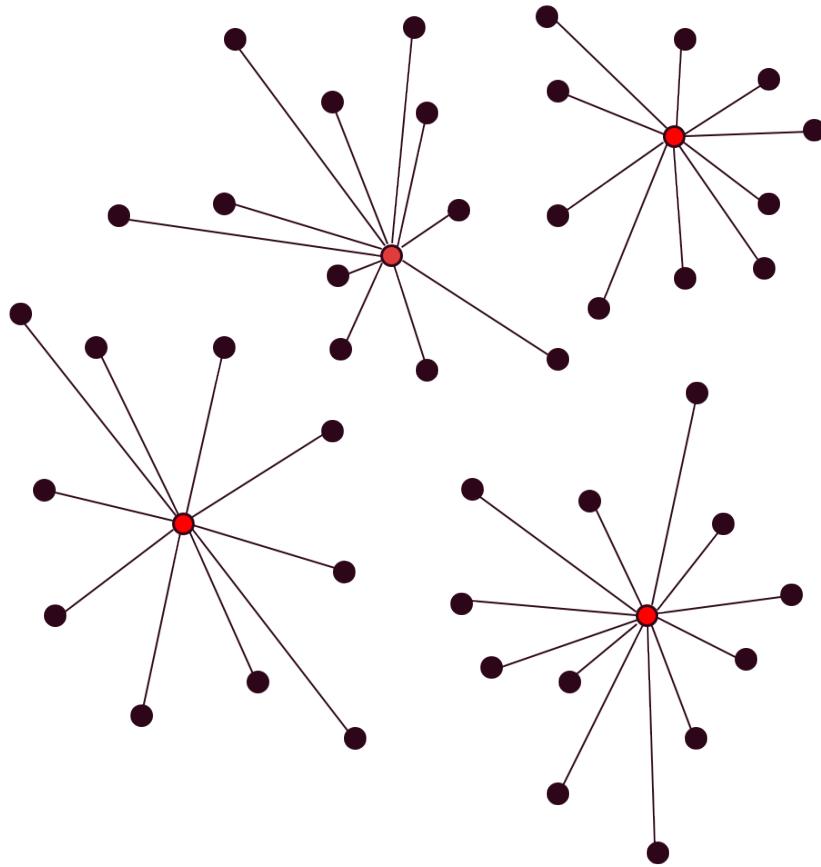


Figura 2.5: Diagrama Red Neuronal Radial Basis Function

2.1.1.3. Multilayer Perception

Esta red cuenta con tres o más capas, las cuales están completamente conectadas, ya que cada uno de los nodos de cada capa se encuentra conectado con todos los nodos de la capa siguiente. Esta red es usada para clasificar información que no pueda ser separada de forma lineal como lo son el Reconocimiento de voz o las traducciones automáticas.[7]

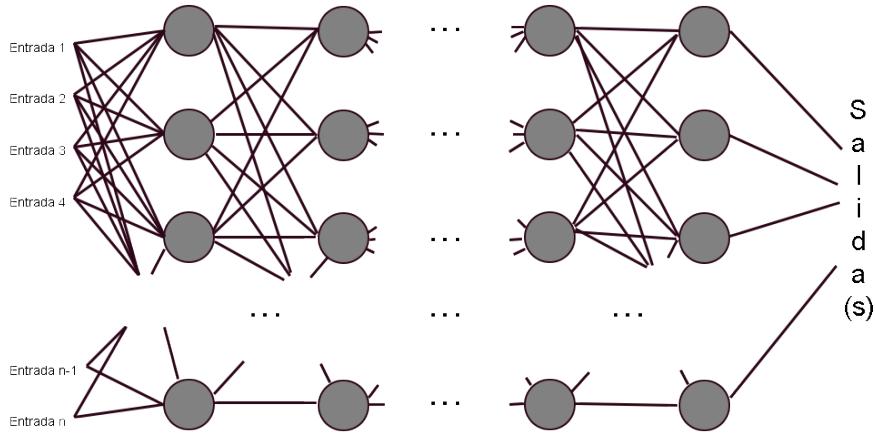


Figura 2.6: Diagrama Red Neuronal Multilayer Perception

2.1.1.4. Convolutional Neural Network

Es una variación de la Multilayer Perception. Contiene una o más capas convolucionales, las cuales pueden estar completamente interconectadas o agrupadas. Antes de pasar el resultado a la siguiente capa, la capa Convolutional hace uso de una operación convolucional en la entrada. Como resultado de esta operación convolucional la red puede ser más profunda, pero con menos parámetros.

Gracias a esto las redes neuronales convolucionales han demostrado buenos resultados en el Reconocimiento de imágenes y video, procesamiento de lenguaje natural, así como para realizar recomendaciones.[6]

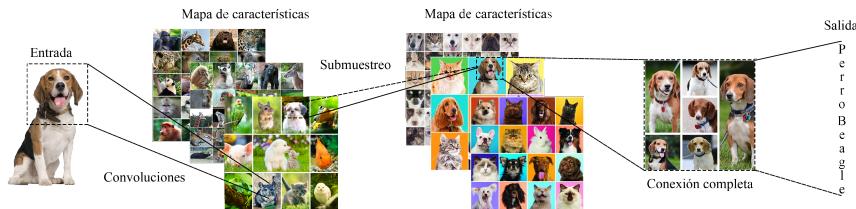


Figura 2.7: Diagrama Red Neuronal Convolucional

2.1.1.5. Recurrent Neural Network – Long Short Term Memory

Una red neuronal Recurrente es un tipo de red neuronal artificial donde la salida de alguna capa en particular es salvada y sirve para retroalimentar la entrada de esta capa, lo cual ayuda a predecir futuras salidas de esta.

La primera capa esta forma de la misma manera que la Feedforward Neural Network, es decir, solo pasa la información que entra a la siguiente capa inmediata, posteriormente la siguiente capa con el paso del tiempo esta capa comenzara a retroalimentarse, pero manteniendo la propagación frontal. Haciendo uso de esta retroalimentación la capa en futuras operaciones puede realizar predicciones, si estas predicciones no son los resultados esperados, el Sistema Aprende y trabaja para corregir sus futuras predicciones.[6]

Este tipo de red neuronal es muy efectiva en tecnologías que requieran conversión de texto a voz o generación de texto.



Texto obtenido hasta el momento

Oigo una



Texto obtenido hasta el momento

Oigo una voz



Texto obtenido hasta el momento

Oigo una voz que



Texto obtenido hasta el momento

Oigo una voz que dice

Figura 2.8: Diagrama Red Neuronal Recurrente

2.1.1.6. Modular Neural Network

Esta red está constituida por un numero variado de redes, las cuales funcionan de manera independiente y realizan subtareas. Las diferentes redes, en realidad no interactúan entre si durante la ejecución del proceso. Trabajan de manera independiente para lograr una misma salida.

Como resultado, un largo y complejo proceso puede ser ejecutado rápido, separándolo en componentes individuales.[7]

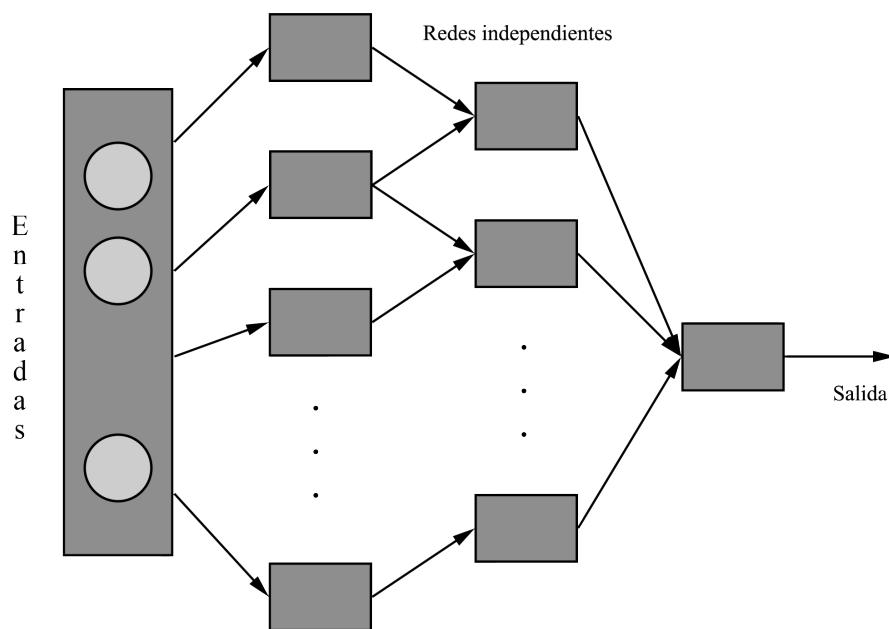


Figura 2.9: Diagrama Red Neuronal Modular

2.2. Base de datos

Una base de datos es una colección organizada de información o datos, los cuales pertenecen a un mismo contexto, se encuentran almacenados de forma física o digital, con la finalidad de realizar alguna consulta futura, ingreso de nuevos datos, actualización o eliminación de estos.

Las bases de datos se componen de una o más tablas las cuales son las encargadas de guardar un conjunto de datos, estas se dividen en columnas y filas.

Una base de datos generalmente es manejada por un Sistema de gestión de base de datos (DBMS). En conjunto, los datos y el DBMS, junto con

las aplicaciones asociadas a ellos, se les conoce como un sistema de base de datos.[9]

2.2.1. Tipos de bases de datos

2.2.1.1. Bases de datos estáticas

Estas son de sólo lectura, utilizadas principalmente para almacenar datos históricos los cuales posteriormente se pueden utilizar para estudiar el comportamiento de un conjunto datos a través del tiempo, realizar proyecciones o tomar decisiones.

2.2.1.2. Bases de datos dinámicas

Estas son las que la información almacenada puede ser modificada con el tiempo, permitiendo operaciones como adición, actualización y eliminación de datos.

2.2.1.3. Bases de datos jerárquicas

Como su nombre su dice, almacenan la información en una estructura jerárquica, en este modelo la organización es similar a un árbol visto al revés, de donde un nodo padre se pueden tener varios nodos hijos, el nodo padre es llamado nodo raíz, y a los nodos que no tiene hijos se les conoce como hojas.

Estas bases de datos son útiles cuando se anejan grandes cantidades de información y datos que requieren compartirse.

2.2.1.4. Bases de datos relacionales

En esta se almacenan y se proporcionan acceso a puntos de datos relacionados entre sí. Esto gracias a que cada fila de la tabla es un registro con un ID único, las columnas de la tabla contienen atributos de los datos, y cada registro generalmente cuenta con un valor para cada atributo, lo que facilita el establecimiento de las relaciones entre los puntos de datos.

2.2.1.5. Base de datos NoSQL o no relacionales

Estas bases permiten que los datos no estructurados y/o semiestructurados se almacenen y manipulen, a diferencia de la base de datos relacional donde se define como deben de componerse todos los datos insertados en esta. Generalmente los registros de este tipo de base de datos suelen almacenarse como un documento de tipo JSON.

2.2.1.6. Base de datos orientada a objetos

En esta se tratan de almacenar los objetos completos (Estado y comportamiento), incorpora todos los conceptos del paradigma orientado a objetos (encapsulamiento, herencia, polimorfismo)

2.3. ¿Qué es una canción?

De manera resumida podemos decir que una canción es una composición literaria, generalmente en verso, a la que se le puede acompañar con música para poder ser cantada.[10]

2.3.1. Elementos de una canción

Los elementos que conforman una canción son los siguientes:

2.3.2. Introducción

Generalmente es una parte única la cual aparece al inicio de una canción, acompañada de una Armonía o Melodía compuesta solo para este inicio. El objetivo principal de la introducción es captar la atención y generar un ambiente.[11]

2.3.3. Verso

En este se comienza a desarrollar la idea a transmitir, trata de contarnos de que va a ser la canción, y ya cuenta con una Armonía bien establecida.

2.3.4. Pre-estribillo

Es un arreglo que permite realizar una transición, su función principal es conectar el verso con el estribillo. También ayuda a evitar que el estribillo se estanque en la monotonía.

2.3.5. Estribillo

Un estribillo es una Estrofa la cual se repite varias veces dentro de una composición. La función principal del estribillo es destacar la idea de la canción tanto en la letra como en lo musical. El estribillo es considerado la parte más importante de la canción y en algunas ocasiones es repetido al inicio y al final de la canción.

2.3.6. Puente

Es un interludio la cual conecta dos partes de una canción, permitiendo construir una Armonía entre ellas, suele ser usado para llevar a la canción a su clímax, para prepararlo para el desarrollo final de la canción.

2.3.7. Cierre

Hay distintas formas de terminar o concluir una canción, puede ser un quiebre brusco generado por un silencio repentino o por una sucesión de Acordes. Pero la forma más común es haciendo uso de la repetición de un estribillo.

2.3.8. Estructura de una canción

La estructura mínima de una canción está compuesta de:

- Verso
- Estribillo
- Verso
- Estribillo

La estructura más usada en una canción es la siguiente

- Introducción
- Verso
- Pre-estribillo
- Estribillo
- Verso
- Estribillo
- Puente
- Cierre

2.4. Cadenas de Markov

Las cadenas de Markov son un Sistema matemático la cuales experimenta con las transiciones de un Estado a otro de acuerdo con ciertas reglas probabilísticas. La característica que definen a una cadena de Markov es que no importa cómo llegó el proceso a su estado actual, y sus posibles estados futuros son fijos. En otras palabras, la probabilidad de pasar a cualquier otro estado depende únicamente del estado actual y del tiempo transcurrido. El espacio de estados o conjunto de todos los posibles estados, pueden ser cualquier cosa: letras, números, puntuaciones de un partido, acciones, etc.

Son procesos Estocásticos, pero con la diferencia de que estos deben ser “sin memoria”, es decir, la probabilidad de las acciones futuras no depende, ni se ve afectada de los pasos que la condujeron al estado actual. A esto se le denomina una propiedad de Markov.

En la teoría de probabilidad, el ejemplo mas inmediato es el de una cadena de Markov homogénea en el tiempo, en la que la probabilidad de que cualquier transición de estado es independiente del tiempo.

En el lenguaje de probabilidad condicional y variables aleatorias, una cadena de Markov es una secuencia X_0, X_1, X_2, \dots de variables aleatorias que satisfacen la regla de independencia condicional. En otras palabras, el conocimiento del estado anterior es todo lo que se necesita para determinar la distribución de probabilidad del estado actual. Esta definición es más amplia que la explorada anteriormente, ya que permite probabilidades de transición no estacionarias y, por lo tanto, cadenas de Markov no homogéneas en el tiempo; es decir, a medida que pasa el tiempo los pasos aumentan y la probabilidad de pasar de un estado a otro puede cambiar. [12]

Las cadenas de Markov pueden ser modeladas mediante Máquinas de Estados Finitos.

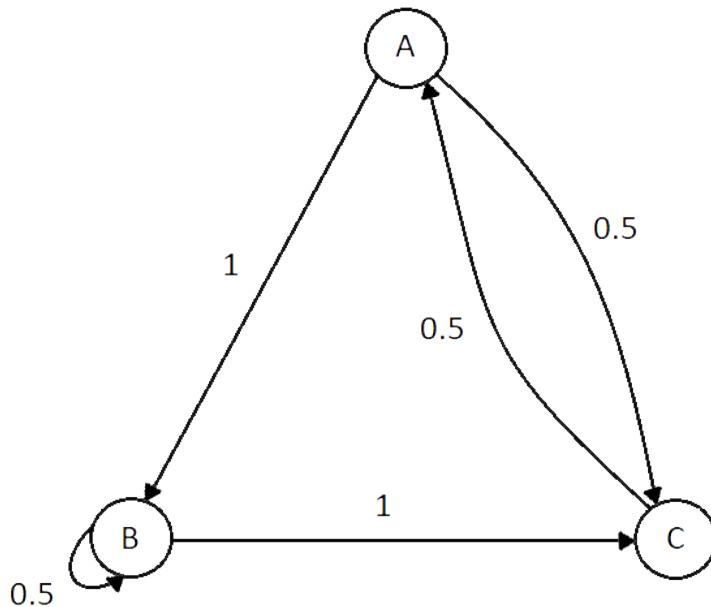


Figura 2.10: Diagrama De Estados finitos De Una Cadena De Markov

2.5. Flask

Flask es un marco (framework) web, el cual es un modulo de Python el cual permite desarrollar aplicaciones web. No cuenta con un Manejador de Objetos Realacionales u ORM por sus siglas en inglés (Object Relational Manager), pero si cuenta con características como el enrutamiento de URLs y un motor de plantillas. En general es un marco de aplicación web WSGI.[15]

WSGI son las siglas de Web Server Gateway Interface (Interfaz de Puerta de enlace del Servidor Web) la cual es una especificación que describe como se va a comunicar un servidor web con una aplicación web, y como se pueden llegar a enlazar distintas aplicaciones web para procesar una solicitud o una petición.[14]

Para poder empezar a trabajar con un proyecto o aplicación web Flask lo primero que hacemos es usar virtualenv para poder manejar un ambiente virtual separado para el proyecto y así evitar conflictos entre dependencias del proyecto. Virtualenv es una herramienta que nos ayuda a crear ambientes Python aislados (en forma de un directorio).

Para poder hacer esto, en la terminal escribimos algo como:

```
virtualenv -no-site-packages myapp
```

Al usar ese comando creara un directorio con la siguiente estructura:

- myapp/
 - bin/
 - include/
 - lib/

Si se trabaja en Windows la estructura es la siguiente:

- myapp/
 - lib/
 - Scripts/

Para poder trabajar con este ambiente en la terminal debemos acceder a la carpeta bin o Scripts y activar este ambiente de la siguiente manera:

```
source myapp/bin/activate  
source myapp/Scripts/activate
```

Luego se procede a escribir el código de una aplicación, por ejemplo, el siguiente extraído de la documentación de flask.[13]

```
from flask import Flask  
app = Flask(__name__)  
  
@app.route('/')  
def hello_world():  
    return 'Hello, World!'
```

Se guarda como cualquier otra aplicación de Python y se ejecuta de la misma manera que una aplicación de Python. Una vez ejecutada te aparecerá los siguiente en terminal:

```
* Running on http://127.0.0.1:5000/
```

Si copiamos esa URL y la abrimos en algún navegador podremos observar nuestra aplicación web en ejecución, en este caso solo se mostrará en pantalla “Hello, World!”

Como se había mencionado al inicio, una de las características de Flask es que podemos manejar las rutas de la aplicación web usando el método route(), siempre que un usuario visita esa ruta se ejecuta el método adjuntado a este.

Flask soporta diferentes tipos de estructuras de variables. Se pueden incluir cadenas personalizadas, números enteros y otros tipos de caracteres en las URLs.

Si queremos manejar solicitudes, Flask tiene soporte para solicitudes de tipo GET y POST, por defecto Flask solo admite peticiones de tipo GET, si necesitamos hacer uso de peticiones tipo POST estas debe de estar especificada como parámetro de la función route(), por ejemplo:

```
@app.route('/login', methods=['POST'])
```

Aun me falta mas por escribir sobre Flask...

A continuación en el cuadro 2.1, mostraremos algunas tablas comparativas que servirán para tener una mejor perspectiva de las ventajas, desventajas.

	Recordar	Otros dispositivos	Acciones	Facilidad	Tiempo	Errores	Recuperación
Contraseñas	1	3	2	3	3	2	3
Otros recursos	2	3	3	3	3	3	2
Contraseñas gráficas	1	1	2	3	3	2	3
Contraseñas dinámicas	1	3	2	2	3	2	2
Tokens	3	1	1	2	2	3	1
Multivariación	1	1	1	3	2	2	1
Cryptograffia	3	1	1	1	1	2	1
Biométricos	3	3	2	3	2	2	1

Cuadro 2.1: Tabla comparativa de la aplicación en los distintos métodos de autentificación

La tabla anterior concentra las siguientes características:

- Recordar: Hace referencia a que tan complicado es que un usuario se acuerde de los datos necesarios para la autentificación.
- Otros dispositivos: El usuario usa una entidad externa para facilitar su autentificación.
- Acciones: Hace referencia a que tantas acciones adicionales se deben de realizar para autenticarse.
- Facilidad: Simplicidad de tecnología.
- Tiempo: Cantidad de recursos temporales que consume el método de autentificación.

- Errores: Posibles errores durante la autentificación.
- Recuperación: Denota la dificultad de recuperar la clave de acceso en caso de pérdida.

En el cuadro 2.2 se muestra una tabla comparativa del nivel de seguridad en los distintos métodos de autentificación, donde 1 - baja seguridad, 2 – media seguridad y 3 – alta seguridad.

	Ataque por fuerza bruta	Observación	Hackeo indirecto	Phishing
Contraseñas	1	1	1	1
Otros recursos	2	2	3	3
Contraseñas gráficas	1	1	2	2
Contraseñas dinámicas	2	3	2	2
Tokens	3	3	3	3
Multivariación	1	1	3	3
Criptografía	3	3	3	3
Biométricos	3	3	1	1

Cuadro 2.2: Tabla comparativa de la seguridad en los distintos métodos de autentificación

La tabla se enfoca principalmente en los siguientes problemas de seguridad:

- Ataque por fuerza bruta: Se descifra el método de autentificación con una gran cantidad de intentos, usualmente generados por un programa.
- Observación: Cuando se intenta ver directamente los datos necesarios para la autentificación desde una distancia cercana hasta incluso usando binoculares, cámaras o algún otro dispositivo.
- Hackeo indirecto: El usuario confía sus datos del método de autentificación a terceros quienes pueden ser atacados.
- Phishing: Hace referencia a programas que se hacen pasar por entidades confiables para interceptar los datos que desean.

Seguridad en internet En la actualidad, el incremento constante de internet ha impactado directamente en la seguridad de la información que se maneja cotidianamente y por la mayoría de usuarios. Existen infinidad de sitios donde es aplicada la seguridad, ya que sin ésta, se verían afectados

todos los usuarios en sus cuentas, pudiendo verse afectados desde un posible Palabra1 (Robo de identidad), hasta la perdida de dinero real dado que la base de algunas de éstas paginas son E-Commerce, estas paginas implican el manejo de tarjetas de crédito, paypal, etc.

Uno de los puntos más críticos de la seguridad en Internet son las herramientas que interactúan de forma directa con los usuarios. Es común escuchar sobre fallas en los sistemas de protección de los servidores más frecuentemente utilizados, por ejemplo Apache, NGINX, IIS, etc. O en los lenguajes de programación en que son escritas las aplicaciones [41]. Sin embargo, la vulnerabilidad más grande dentro de un sistema, son los ataques directos a los usuarios finales durante la autentificación.

Una de las técnicas de autentificación que actualmente se usa es "recordar la sesión" usando las "cookies", en la siguiente sección, nos adentraremos en definir qué son las cookies y exponer sus vulnerabilidades.

2.6. Apache

Apache es un servidor web gratuito y de código abierto que permite que los dueños de sitios web mostrar contenido en ellas, no es un servidor físico, sino más bien un software que corre en un servidor. Su trabajo es establecer la conexión entre un servidor y los navegadores de los visitantes del sitio web mientras se mandan archivos de ida y regreso entre ellos (estructura cliente-servidor). El servidor y el cliente se comunican mediante el protocolo HTTP, y el software de Apache es responsable por la comunicación fluida y segura entre las 2 máquinas. Apache trabaja sin problemas con muchos otros sistemas de gestión de contenido (Joomla, Drupal, etc.), marcos de trabajo web (Django, Laravel, etc.), y lenguajes de programación. Todo esto en conjunto lo vuelve una opción sólida al momento de escoger entre todos los tipos de plataformas de alojamiento web, como serían VPS o alojamiento compartido. Palabra1.

En el cuadro 1 que se presenta a continuación, se muestran las características de aplicaciones web similares y comparándolas con nuestra propuesta:

Cuadro 2.3: Tabla comparativa de los distintos servidores web contemplados

Apache	NGINX	Tomcat
Apache viene con ventajas útiles sobre Nginx, como sería su fácil configuración, muchos módulos, y un entorno amigable con los primerizos.	Nginx fue creado para resolver el llamado “problema c10k”, significando que un servidor web que usa hilos para manejar solicitudes del usuario es incapaz de gestionar más de 10,000 conexiones al mismo tiempo.	Tomcat fue creado específicamente para aplicaciones Java, mientras que Apache es un servidor HTTP de propósito general.
De código abierto y gratuito, incluso para su uso comercial. Software estable y confiable. Frecuentemente actualizado incluyendo actualizaciones regulares a los parches de seguridad.	Nginx es uno de los servidores web que soluciona el problema c10k y probablemente el más exitoso al hacerlo, no crea un nuevo proceso por cada solicitud y en su lugar, maneja toda solicitud entrante en un único hilo.	Tomcat es menos configurable comparado a otros servidores web. Por ejemplo, para correr WordPress, la mejor elección es un servidor HTTP de propósito general como Apache o Nginx.
Funciona de forma intuitiva, con sitios web hechos con WordPress. Comunidad grande y posibilidad de contactar con soporte disponible de manera sencilla en caso de cualquier problema.	El modelo basado en eventos de Nginx distribuye solicitudes de usuario entre procesos trabajadores de una manera eficiente, llevando con ello a una mejor escalabilidad.	
Problemas de rendimiento en sitios web con tráfico extremadamente pesado. Tantas opciones de configuración pueden llevar a vulnerabilidades de seguridad.	Si necesitas manejar un sitio web con un alto tráfico, Nginx es una excelente opción, puesto que puede hacerlo con un mínimo uso de los recursos.	

2.7. Servidor Web

El trabajo de un servidor web es mostrar sitios web en internet. Para conseguir este objetivo, actúa como un intermediario entre el servidor y las máquinas del cliente. Jala contenido del servidor en cada petición del usuario y lo entrega al sitio web.

Los servidores web procesan archivos escritos en diferentes lenguajes de programación como PHP, Python, Java, y otros.

Cuando escuchas la palabra ‘servidor web’, piensa en eso como la herramienta responsable por la correcta comunicación entre el cliente y el servidor.

2.8. Certificado SSL

Conocido como (Secure Sockets Layer) o (Capa de Conexión Segura) es un estándar de seguridad global que fue originalmente creado por Netscape in los 1990. SSL crea una conexión encriptada entre tu servidor web y el navegador web de tu visitante permitiendo que la información privada sea transmitida sin los problemas de espionaje, manipulación de la información, y falsificación del mensaje. Básicamente, la capa SSL permite que dos partes tengan una conversación”privada.

Para establecer esta conexión segura, se instala en un servidor web un certificado SSL (también llamado “certificado digital”) que cumple dos funciones:

- Autentificar la identidad del sitio web, garantizando a los visitantes que no están en un sitio falso.
- Cifrar la información transmitida.

Hay varios tipos de certificados SSL según la cantidad de nombres de dominio o subdominios que se tengan, como por ejemplo:

- Único: Asegura un nombre de dominio o subdominio completo (FQDN por sus siglas en inglés).
- Comodín: Cubre un nombre de dominio y un número ilimitado de sus subdominios.
- Multidominio: Asegura varios nombres de dominio.

2.8.1. Objetivos de la criptografía.

Algunos de los objetivos que se busca con la implementación de la criptografía para la seguridad de la información son los siguientes:

- **Confidencialidad:** este objetivo, también conocido como privacidad de la información, implica mantener en secreto una determinada información, por tanto, sólo aquellas personas que estén autorizadas tendrán acceso a ella.
- **Autenticación:** este objetivo, implica hablar de la corroboración de la identidad de una entidad, por tanto, asegura que la entidad de donde la información es originada pueda ser identificada.

- **Integridad:** este objetivo asegura que determinada información no haya sido alterada por personas no autorizadas o por cualquier otro medio no conocido.
- **No repudio:** este objetivo previene que una entidad niegue un envío de información de un acuerdo preestablecido.

2.8.2. Usos de la criptografía.

Los usos que tiene la criptografía son muy variadas, dependiendo del ámbito donde se está aplicando. Las siguientes usos, son sólo algunas de los tantos que hay y provienen de distintos objetivos que tiene la criptografía aplicada a la seguridad de la información [34].

- **Autorización:** permiso concreto, de una parte o entidad, para el acceso o la realización de una tarea específica.
- **Validación:** proveer una autorización para el uso o la manipulación de información o recursos.
- **Control de acceso:** restricción de acceso a la información o recurso.
- **Certificación:** respaldo de información por una entidad externa de confianza.
- **Confirmación:** acuse de recibo a servicios que han sido dados.
- **Anonimato:** ocultamiento de la identidad de una entidad.

2.8.3. Criptografía simétrica y asimétrica.

Anteriormente en la historia de la criptografía, se hizo una rápida mención del nacimiento de estos dos métodos de cifrado, en esta sección se explicará más profundamente sus funciones con el fin de que el lector conozca un poco más y entienda porque hemos decidido utilizar determinado método para cumplir con los objetivos de este trabajo.

2.8.3.1. Criptografía simétrica.

En la criptografía simétrica, tanto el emisor como el receptor comparten una única llave secreta para cifrar y descifrar la información que se deseé transmitir. Esto implica que ambas partes de la comunicación deben tener un acuerdo antes de que se realice la comunicación. La seguridad de este tipo

de algoritmos radica en mantener segura la llave secreta, por tanto, si ésta es revelada, cualquiera con acceso a ella puede descifrar el mensaje. Por estas razones, este tipo de criptografía puede ser visto como 'criptografía de llave privada'. Algunos de los algoritmos más famosos de criptografía simétrica son: DES (Data Encryption Standard), TripleDES, AES (Advanced Encryption Standard) y IDEA (International Data Encryption Algorithm) [34].

En el esquema de la Figura 2.11, se muestran los pasos que sigue un protocolo de criptografía simétrica. Definamos 'A' como una entidad que pretende enviar un mensaje a otra entidad llamada 'B'. Luego entonces, ambas partes acordarán una 'Llave Secreta' con la que 'A' cifrará el mensaje utilizando un algoritmo establecido, mandando el resultado (Texto Cifrado) a 'B' que descifrará el mensaje con la misma llave y algoritmo que 'A'.



Figura 2.11: Esquema del protocolo de criptografía simétrica.

2.8.3.2. Criptografía asimétrica.

La criptografía de clave pública fue inventada en 1976 por los matemáticos Whitfield Diffie y Martin Hellman y es la base de la criptografía moderna. En los algoritmos de criptografía asimétrica, el receptor posee una llave pública y una llave privada para poder descifrar los mensajes. Las claves privadas deben ser conocidas únicamente por su propietario, mientras que la correspondiente clave pública puede ser dada a conocer abiertamente. Si un usuario quiere enviar a otro un mensaje de forma que sólo el receptor pueda entenderlo, lo codificará con la clave pública del receptor y éste utilizará su clave privada, que solo él tiene, para poder leerlo. Dicho esto, podemos llamar llave de cifrado a la llave pública y llave de descifrado a la llave privada, siendo ambas totalmente diferentes una de la otra. Algunos de los algoritmos más famosos de criptografía asimétrica son: RSA y ElGamal [34].

La criptografía asimétrica está basada en la utilización de números primos muy grandes. Si multiplicamos entre sí dos números primos muy grandes, el

resultado obtenido no puede descomponerse eficazmente, es decir, utilizando los métodos aritméticos más avanzados en las computadoras más avanzadas sería necesario utilizar durante miles de millones de años tantas computadoras como átomos existen en el universo. El proceso será más seguro cuanto mayor sea el tamaño de los números primos utilizados.

En el esquema de la Figura 2.12, se muestran los pasos que sigue un protocolo de criptografía asimétrica. Definamos 'A' como una entidad que desea enviar información a otra llamada 'B'. Luego entonces, 'B' enviará a 'A' su llave pública para que 'A' cifre la información utilizándola. Cuando la información (Texto Cifrado) haya viajado a través del canal inseguro para que 'B' la reciba, 'B' descifrará el Texto Cifrado con su llave privada.



Figura 2.12: Esquema del protocolo de criptografía asimétrica.

Algoritmo RSA.

RSA es el algoritmo de cifrado asimétrico más popular en la actualidad. Creado por Ron Rivest, Adi Shamir y Leonard Adleman y publicado en el año 1977. El algoritmo es considerado seguro, en tanto sean utilizadas llaves de longitud suficientemente seguras (se siguen utilizando llaves de 1024 bits, pero ya se recomienda al menos una longitud de 2048). El algoritmo sirve tanto para cifrar y descifrar, así como también para la generación de firmas digitales. Es, en la actualidad, ampliamente utilizado en protocolos de comercio electrónico. La seguridad *RSA* está basada en la dificultad de realizar el *factoreo* de números grandes. La llave privada y la pública son generadas o calculadas en función de un par de números primos, del orden de los 200 dígitos o más grandes aún [34].

Al describir ya concretamente al algoritmo, se establece que para la generación del par de llaves (llave pública y privada) se deberán seleccionar dos números primos grandes aleatorios, p y q , y se calculará n como su producto:

$$n = pq$$

La llave de cifrado, e , será elegida también de manera aleatoria, tal que e y $(p - 1)(q - 1)$ sean primos relativos.

La llave de descifrado d será obtenida despejando la ecuación:

$$ed = 1 \bmod (p - 1)(q - 1)$$

En otras palabras, o mejor dicho, en otros símbolos:

$$d = e^{-1} \bmod ((p - 1)(q - 1))$$

Los números e y n componen la llave privada; el número d corresponde a la llave privada; p y q serán descartados pero no revelados.

A la hora de cifrar un mensaje m , éste deberá ser dividido en bloques más pequeños que n y cada parte del texto-cifrado, c , será obtenida mediante:

$$c_i = m_i^e \bmod n$$

Para el descifrado, cada parte o bloque del texto-cifrado se tomará para calcular:

$$m_i = c_i^d \bmod n$$

El patrón con el que se genera el código *Chaffing* no puede viajar a través de la red visible para cualquier entidad que quiera obtener dicha petición, por lo que, debemos cifrarlo con algún algoritmo confiable.

El algoritmo RSA nos proporciona esa seguridad, debido a que es un cifrado asimétrico con el cual podemos cifrar el patrón con la llave pública del servicio. Una vez que la petición cifrada llegue al servicio, este será capaz de descifrar dicho mensaje con su llave privada, para así poder ver el mensaje original.

2.8.4. Criptografía a nivel de aplicación.

La criptografía a nivel de aplicación se entiende como aplicaciones o programas informáticos que hacen uso de diferentes técnicas criptográficas. Al hablar de criptografía en el nivel de aplicación, también podría considerarse a los protocolos criptográficos de alto nivel, los cuales determinan como han de comunicarse ambas partes, que algoritmos usar, define formatos, etc [34].

2.8.4.1. SSL/TLS.

Secure Sockets Layer (SSL) provee servicios de seguridad entre la capa TCP y las aplicaciones que hacen uso de esa capa. Actualmente, la sucesora de SSL es TLS (Transport Layer Service), sin embargo, lo acuñado que está el término SSL hace que se use indistintamente para referirse a TLS, aunado a ello, las diferencias entre la última versión de SSL (SSL3.0) y la primera versión de TLS (TLSv1) son menores, por lo que en el desarrollo de este reporte, utilizaremos el término SSL/TLS.

SSL/TLS provee entonces confidencialidad, lográndola con criptografía asimétrica y controlando la integridad de los datos utilizando un MAC (Message Authentication Code).

El proceso de comunicación del protocolo establece, como primer paso, la negociación de ambas partes de los algoritmos a utilizar. Luego, procede al intercambio de llaves públicas y a la autenticación basada en certificados digitales para, finalmente, cifrar de manera simétrica los datos o información a transferir [34].

En nuestro trabajo, usaremos SSL/TLS para brindar confidencialidad al canal de comunicación entre la extensión y el servidor autenticador. Esto se explicará más a detalle en el análisis del Componente II.

2.8.4.2. OpenSSL

OpenSSL es un proyecto de código abierto que implementa funciones criptográficas sin limitaciones dentro de una librería y que provee diversas herramientas útiles. OpenSSL actualmente implementa SSL2.0, SSL3.0 y TLSv [34].

Dentro de las herramientas que tiene se encuentran:

- Crear y manejar llaves privadas, públicas y parámetros.
- Realizar operaciones criptográficas de llave pública.
- Calcular hash de algún mensaje.
- Cifrar y descifrar con algoritmos simétricos.
- Crear certificados X.509 (CSRs y CRLs).

Esta última herramienta, es la que usaremos para este trabajo terminal. Se creará un certificado de cada usuario a partir de sus datos de inicio de sesión para autenticarlo en los servicios web en donde quiera acceder. Esto se explicará más a detalle en el análisis del Componente II.

Ahora que se ha expuesto el uso de la criptografía en este trabajo, procedemos a platicar acerca del método *Chaffing and Winnowing* y cómo se implementará en este trabajo.

2.9. Tema 6

2.9.1. Historia

Chaffing and Winnowing es una técnica que logra confidencialidad sin usar ningún proceso de cifrado para el envío de datos sobre un canal inseguro. El nombre **Chaffing and Winnowing** el nombre proviene de la agricultura: Después de que el grano ha sido cosechado y trillado es mezclado con paja fibrosa no comestible. La paja y el grano son separados por el movimiento de las hojas y la paja es descartada. Ésta técnica fue creada por Ron Rivest y fue publicada en un artículo en línea el 18 de Marzo de 1998 [40]. Aunque parece ser similar a un cifrado tradicional o esteganografía, chaffing and winnowing no puede ser clasificado como uno de ellos.

Ésta técnica permite el envío de datos evitando la responsabilidad del cifrado de su contenido. Cuando se usa chaffing and winnowing, el emisor transmite el mensaje sin cifrar (texto plano). Aunque el emisor y el receptor comparten una llave, ellos la usan sólo para autenticar. Sin embargo, una tercera parte puede hacer su comunicación confidencial durante el envío simultáneo de mensajes especialmente mensajes diseñados a través del mismo canal.

2.9.2. ¿Qué es Tema 6?

Chaffing and Winnowing es un nuevo esquema establecido por Rivest en 1998. Este esquema ofrece confidencialidad para el contenido de un mensaje sin involucrarse con cifrado ni estenografía [34].

El proceso **Chaffing** no hace uso de un cifrado por lo que no tiene una "clave de cifrado". Este proceso consiste en agregar paquetes inválidos (Información innecesaria) al mensaje a enviar, haciendo que el mensaje viaje seguro a la vista de todos los posibles "atacantes".

El proceso de **Winnowing** no emplea algún tipo de cifrado, por lo que al

igual que el proceso chaff no tiene una "clave de descifrado". Intentando regular la confidencialidad que provee un cifrado damos paso a la esteganografía y el proceso de winnowing [40].

Existen dos partes en el envío de mensajes con winnowing: Autentificación (Agregando MACs) y agregando paquetes chaff. Nosotros nos enfocaremos mas al uso de paquetes chaff para el envío seguro de información, ya que, el receptor es quien remueve los paquetes chaff para obtener el mensaje original.

Los siguientes esquemas explican como es que se lleva a cabo el proceso de **Chaffing and Winnowing** en diferentes escenarios.

Escenario 1: Alice se está comunicando con Bob en un solo camino de comunicación sobre un canal inseguro y Charles agrega los paquetes de Chaff.

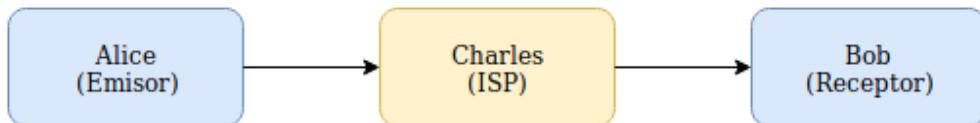


Figura 2.13: Charles agrega los paquetes inválidos.

En el escenario anterior Alice y Bob se están comunicando mutuamente por un canal de comunicación no seguro, en donde son enviados paquetes no cifrados. Alice y Bob comparten la llave de autentificación la cual será usada para el proceso de autentificación. Cuando Alice envía un mensaje a Bob, su mensaje es autenticado de su lado y es enviado a Charles antes de ser enviado a Bob. Charles agrega los paquetes chaff a la secuencia transmitida por Alice, al agregar los paquetes chaff, Charles provee confidencialidad para la comunicación entre Alice y Bob. Pero donde Charles no conoce la llave secreta compartida entre Alice y Bob. Por lo que el proceso de chaffing no necesita ningún conocimiento de la llave secreta de autentificación compartida.

Escenario 2: Alice se comunica con Bob en un camino de comunicación inseguro y en el cual Charles no agrega los paquetes chaff si no que multiplexa los flujos de las dos partes (David y Alice). Este escenario es diferente al anterior, ya que se multiplexa el flujo de datos de Alice y Bob con el flujo de

datos de David y Jane, y cuando el paquete llega a Bob el flujo de paquetes de David hacia Jane es el chaff de Bob y es descartado y vice versa para Jane.

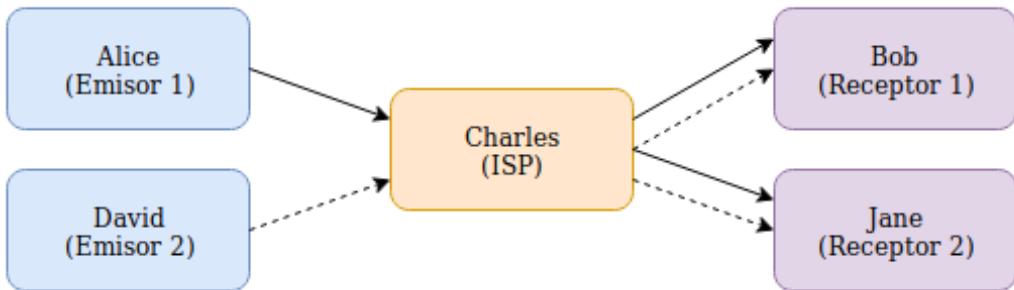


Figura 2.14: Charles no agrega los paquetes pero multiplexa los flujos.

Escenario 3: Alice se comunica con Bob en un canal de comunicación inseguro y Alice no agrega los paquetes chaff. En este escenario, Alice desarrolla la autentificación de sus mensajes, por lo que Alice aplica chaffing para autenticar los mensajes y producir una secuencia de paquetes que serán transmitidos a Bob por la vía de Charles. Bob lleva a cabo el proceso de winnowing para recuperar el mensaje original.

2.9.3. Objetivo de Tema 6

El objetivo de seguridad del esquema de chaffing-and-winnowing es proporcionar privacidad en un entorno simétrico. Desde un punto de vista de seguridad, este esquema debe tratarse simplemente como un esquema de cifrado simétrico. Hay algunos procesos de "cifrado" que toman un mensaje y crean un "texto cifrado", y algún proceso de "descifrado" toma el texto cifrado y recupera el mensaje, ambos operando bajo una clave secreta en común. (Para el esquema Chaffing and Winnowing es la clave para el MAC). Estos procesos no se implementan de manera "habitual", pero, de manera abstracta, deben existir, de lo contrario no se logra la privacidad [18] [26].

"No es una propiedad de seguridad novedosa, sino un conjunto novedoso de restricciones en los procesos dirigidos a lograr una propiedad de seguridad estándar"

"Encontrar-luego-adivinar". Extensión más directa al caso simétrico de la noción de indistinguibilidad.

Haciendo uso de **Chaffing and Winnowing** se asegura que los adversarios no obtengan información del mensaje transmitido a lo largo de un canal de comunicación inseguro entre dos partes.

Rivest propone un esquema, el cual cuenta con tres partes principales [40].

1. **Autentificación** Es el proceso de descomponer el mensaje original en un paquete más pequeño y complementar cada paquete con un código de autentificación de mensaje (MAC).
2. **Chaffing** Es el proceso de agregar paquetes inválidos (Chaff packets).
3. **Winnowing** Es el proceso de remover paquetes Chaff para obtener el mensaje original en texto plano.

2.9.4. ¿Cómo funciona?

El esquema de Chaffing and Winnowing deja que cada paquete conste de:

- Un número de serie
- Contenido del paquete
- Código de autentificación del mensaje

Cuando son enviados los paquetes, el mensaje con el texto plano se descompone en pequeños paquetes los cuales contienen datos y el tamaño del paquete original. Entonces, el emisor (Alice) usa el algoritmo de **código de autentificación de mensaje** (MAC) para generar el valor MAC para ser agregado al paquete y el cual se basa en el número de serie, contenido del paquete y la llave autentificación. La Figura 2.15, muestra la salida del paquete después del proceso de autentificación.

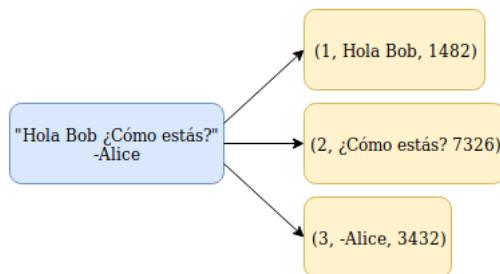


Figura 2.15: Secuencia de Chaffing después del proceso de autentificación.

Esta secuencia de paquetes es enviada a Charles (ISP) para llevar a cabo el proceso de Chaffing. Charles agrega paquetes chaff a la secuencia de paquetes antes de ser enviados por medio del canal de comunicación y ser recibidos por Bob.

Existen dos maneras donde Charles puede enviar la secuencia de chaff hacia Bob. La primera es enviando aleatoriamente mezclados los paquetes chaff para formar una secuencia y la otra manera es enviarlos de manera ordenada por el número de serie seguido del contenido del mensaje. En la Figura 2.16, se muestra cómo es el proceso de chaff en esta secuencia.

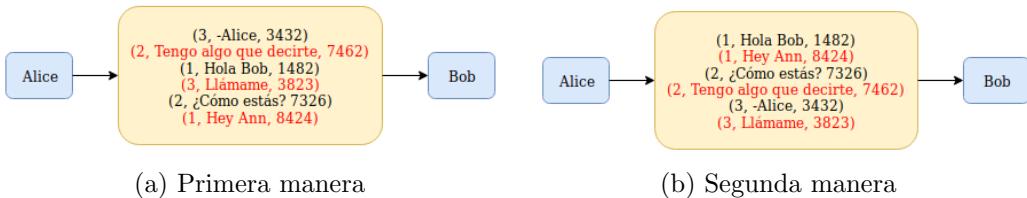


Figura 2.16: Las dos maneras para el proceso de chaff pueden ser utilizadas. Los paquetes chaff son los mensajes de color rojo.

Una vez que la secuencia de chaff llega a Bob, el último proceso es Winnowing. Bob determina la secuencia del mensaje que es válida del paquete chaff usando una función hash para el contenido de cada paquete y la llave de autentificación para re-calcular el MAC y compararlo contra el MAC del paquete recibido, si la comparación falla, el paquete chaff es descartado. Si la comparación es valida, entonces el paquete es parte del mensaje original. En la Figura 2.17, se muestra el proceso completo de Chaffing and Winnowing [40].

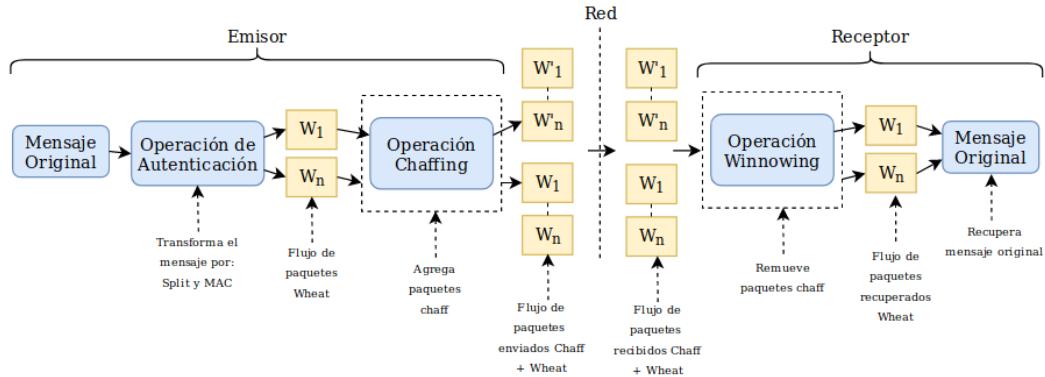


Figura 2.17: Visión general del proceso Chaffing and Winnowing.

2.9.5. Propiedades de Tema 6

- La técnica de Chaffing y Winnowing no depende de la fuerza del esquema de cifrado para proporcionar confidencialidad debido al hecho de que es muy difícil distinguir la información útil de los paquetes chaff sin la clave secreta. Por lo tanto, la dificultad de distinguir la información útil del chaff proporciona confidencialidad al esquema.
- La operación de Chaffing puede ser realizada por un tercero, ya que la clave secreta compartida no es necesaria en el proceso del mismo.
- Los paquetes de Chaff no tienen que contener datos aleatorios, ya que uno podría usar un mensaje válido con una clave secreta diferente para hacer el paquete de Chaff. Cuando el receptor recibe esos paquetes de Chaff, se verán como paquetes de Chaff, ya que la clave que se usa para volver a calcular el Chaff es diferente de la que los hace.

2.9.6. All-or-Nothing and the Package Transform (AONT)

All-or-Nothing and the Package Transform es una variación dentro de la técnica Chaffing and Winnowing, donde se mejora la eficiencia de su esquema original. AONT es la transformación de pre-procesamiento que permite a las partes enviar más datos (en términos de bit) por paquete en lugar de solo uno. Este pre-procesamiento es una transformación sin cifrado que toma el mensaje de texto sin formato y produce un mensaje empaquetado que

luego se procesa de la manera normal de Chaffing and Winnowing [33]. Las definiciones de la transformación AONT son las siguientes:

1. El algoritmo de transformación es **reversible**: Dado el bloque de mensaje transformado, el receptor puede obtener el mensaje de texto sin formato original.
2. El algoritmo de transformación y su inverso son **computables** de manera eficiente: Lo que significa que es computacionalmente factible recrear el texto original dada la llave privada y recibir todos los paquetes con éxito.
3. La transformación no es **computacionalmente factible**: Esto significa que si se ha recibido parte del paquete de la transmisión, cualquiera que esté intentando leer el mensaje no puede hacerlo ya que la transformación **AONT** requiere que se reciba todo el mensaje, de lo contrario no entrega nada.
4. La transformación es una **técnica sin cifrado**: La técnica de preprocesamiento no tiene llaves y no hay una llave secreta compartida involucrada en la operación. Cualquier persona que haya recibido todos los mensajes transformados del paquete puede recuperar el mensaje de texto original.

¿Cómo funciona AONT?

Supongamos que el mensaje de entrada es el siguiente: m_1, m_2, \dots, m_n . Seleccionamos una llave aleatoria K' el cual se usará para la función del paquete de transformación.

Se calcula la secuencia transformada m'_1, m'_2, \dots, m'_s para $s' = s + 1$ como se muestra a continuación:

Tenemos:

$$m_i \otimes E(K', i) \text{ for } i = 1, 2, 3, \dots, s$$

También:

$$m'_{s'} = K' \otimes h_1 \otimes h_2 \otimes \dots \otimes h_s$$

Donde:

$$h_i = E(K_0, m'_i \otimes i) \text{ for } i = 1, 2, \dots, s$$

Donde K_0 es una llave conocida pública fija.

Para que el receptor en el otro extremo obtenga el K_0 , el cual es la llave para el uso de **AONT**, el receptor realiza el siguiente cálculo:

$$K' = m'_s \otimes h_1 \otimes h_2 \otimes \dots \otimes h_s$$

$$m_i = m'_i \otimes E(K', i) \text{ for } i = 1, 2, \dots, s$$

AONT toma el mensaje de texto sin formato de entrada y los transforma, luego crea un bloque para almacenar los mensajes transformados antes de pasar al proceso de autenticación. Después, se genera el paquete Chaff (la cantidad de paquetes Chaff no tiene que ser igual a los paquetes de la información útil).

Esta técnica produce una menor sobrecarga que la sugerencia número 1. El AONT ofrece más confidencialidad al esquema de Chaffing and Winnowing, ya que el adversario debe recibir todo el bloque de mensajes de transformación e identificar correctamente todo el paquete de la información útil para obtener el mensaje de texto original. La Figura 2.18, muestra la descripción general de Chaffing y Winnowing si se agrega la función AONT [33].

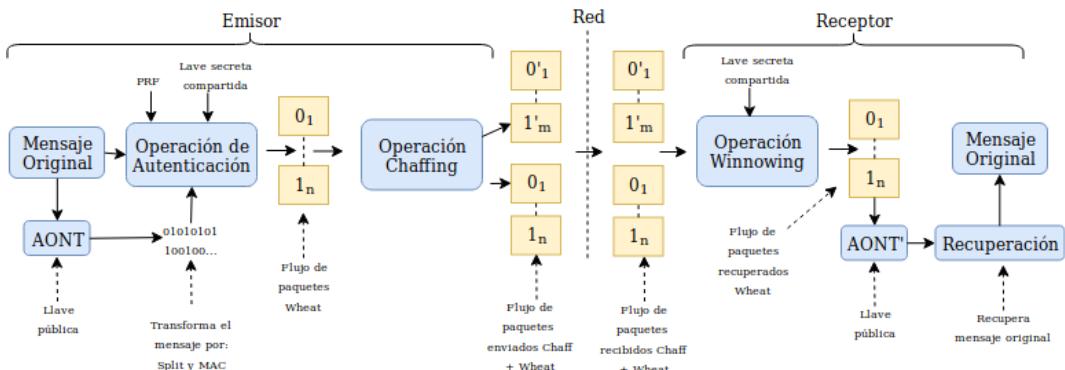


Figura 2.18: Proceso de Chaffing and Winnowing junto con AONT.

¿Cómo AONT puede hacer la diferencia?

1. Requiere menos ancho de banda al transferir paquetes, ya que se pueden transferir más bits en un paquete en lugar de un bit por paquete.
2. Los paquetes Chaff son más fáciles de generar, ya que AONT transforma el mensaje de texto plano en bits aleatorios.

3. La distinción entre Chaffing and Winnowing es más difícil: Si el adversario va a ejercer fuerza bruta en los paquetes, la tarea se ralentizará por el factor del número de bloque de mensajes. Dado que el bloque de mensaje de información adicional se mezcla aleatoriamente dentro de los flujos de paquetes de Chaffing and Winnowing, sin saber que es muy difícil que el bloque adicional proporcione la posibilidad de elegir el bloque de mensaje correcto de los paquetes para obtener el texto plano original.

2.9.7. Comparando Chaffing and Winnowing contra Cifrado y Esteganografía

En esta sección explicaremos porque Chaffing and Winnowing no puede ser clasificado como una técnica de cifrado o Estenografía.

2.9.7.1. Chafing and Winnowing vs Cifrado

Nosotros podríamos clasificar Chaffing and Winnowing como un método de cifrado, pero volvamos a recordar el principio de un Cifrado. El principal objetivo de un cifrado es ocultar el mensaje en texto plano de tal manera que oculta su contenido con el uso de una clave de cifrado para el texto cifrado. Por otro lado, en el esquema original de Chaffing and Winnowing, una llave compartida es usada con el fin de autentificar la validación de los paquetes ya sea del emisor o del receptor. Además, Chaffing and Winnowing no hace uso de ninguna técnica de cifrado para ocultar el contenido de un mensaje y que nadie pueda ver dicho mensaje, solo aquellos con la llave correspondiente pueden determinar que paquetes contienen la información valida. En la figura 2.18, se muestra como se puede ver el esquema Chaffing and Winnowing como una técnica de cifrado.

Chaffing y Winnowing pueden verse como **un tipo especial de esquema de cifrado simétrico**, ya que la operación **chaffing** es similar al "proceso de cifrado". En la operación de chaffing, el texto cifrado se crea para producir un paquete de la información útil no válido que se envía al receptor. Luego, el receptor realiza el "proceso de descifrado", que implica descartar el paquete de desperdicios y recuperar el mensaje original. Ambas operaciones operan bajo una llave secreta común que se usa para derivar el valor MAC.

Pero la diferencia es, *Chaffing y Winnowing* dos partes que no buscan lograr la confidencialidad. El emisor comparte una clave secreta con el re-

ceptor para que el receptor pueda usar la clave secreta para autenticarse (si se afirma que el mensaje recibido proviene del remitente deseado). Pero la ganancia de confidencialidad proviene de la dificultad de distinguir el paquete Chaff del paquete de la información útil. Mientras que en el cifrado, la clave se utiliza para lograr la confidencialidad mediante la creación de texto cifrado que oculta el contenido del mensaje de personas [40].

Chaffing y Winnowing junto con el esquema AONT, el esquema en sí es muy parecido al cifrado, excepto que la clave que se usa en la transformación AONT, se elige aleatoriamente cada vez en lugar de fijarla. Además, el último bloque de mensajes es exclusivo o de la clave y todo el hash del bloque de mensajes está allí para garantizar que cualquier modificación en el bloque de mensajes cambiará la clave K' calculado por el receptor. Por lo tanto, el último bloque de mensajes $m'_{s'}$ está allí solo con el propósito de autenticación. Por lo tanto, Chaffing y Winnowing con el esquema AONT no pueden ser clasificados bajo cifrado [33].

2.9.7.2. Chaffing and Winnowing vs Esteganografía

La esteganografía trata de cómo ocultar, dentro de un mensaje público, información secreta. La agregación de información 'extra' es también un secreto, es decir, nadie externo a la comunicación sabe que en ese mensaje público se ha agregado información [34]. Un ejemplo de esta técnica de cifrado, se puede visualizar claramente en la utilización de tinta invisible en cartas. El emisor escribe una carta común y corriente con tinta visible, pero además, escribe información secreta con tinta invisible. Luego entonces, el receptor, aplicando el método correspondiente, descubrirá la información mandada con tinta invisible, pero agentes externos, ni siquiera saben la existencia de dicha información. En la actualidad, la esteganografía es muy utilizada para ocultar información en archivos de imágenes.

Para algunas personas, Chaffing and Winnowing puede ser clasificado como una técnica esteganografía. Sin embargo, el objetivo principal de la esteganografía, como se mencionó antes, es el de ocultar el mensaje original dentro de otro mensaje, por lo tanto, nadie aparte del emisor y el receptor sabrá que hay un mensaje oculto. Contrario a Chaffing and Winnowing, en donde cualquiera puede ver el contenido del mensaje, ya que este método no trata de esconderlo de los posibles atacantes. Otra diferencia es que en esteganografía el emisor tiene que ocultar el mensaje el mismo, mientras que en Chaffing and Winnowing no necesariamente es así, ya que una "tercera

parte” puede hacerlo [35].

Por lo tanto, Chaffing and Winnowing no puede ser considerado como esteganografía.

Para nuestro trabajo terminal usaremos *Chaffing and Winnowing*, proponiendo así un nuevo método de autentificación para servicios web.

Sprint 1.

Capítulo 3

Análisis.

3.1. Estudio de Factibilidad.

El estudio de factibilidad es un instrumento que sirve para orientar la toma de decisiones en la evaluación de un proyecto y corresponde a la última fase de la etapa pre-operativa dentro del ciclo del proyecto. Se formula con base en información que tiene la menor incertidumbre posible para medir las posibilidades de éxito o fracaso de un proyecto, apoyándose en él se tomará la decisión de proceder o no con su implementación. Este estudio establecerá la viabilidad, si existe, del trabajo.

- Factibilidad Técnica: Hace referencia a los recursos como herramientas, conocimientos, habilidades, experiencia, etc. que son necesarios para efectuar las actividades del trabajo terminal.
- Factibilidad Operativa: Se refiere a los recursos necesarios para llevar a cabo los procesos de forma eficiente , depende de los recursos humanos.
- Factibilidad Económica: Consiste en los recursos financieros necesarios para llevar a cabo la elaboración de este trabajo.

3.1.1. Factibilidad Técnica

En esta parte explicaremos detalladamente las tecnologías que usaremos. Para la elección de estas herramientas fue necesario investigar las tecnologías que más se usan en la actualidad, además de ver las características y equipos de cómputo con los que contamos actualmente.

Factibilidad Técnica	
Sistema Operativo	Multiplataforma
Navegador Web	Google Chrome
Lenguaje de Programación	JavaScript
Servidor	Apache 2.0

Cuadro 3.1: Herramientas de Software a utilizar

Además de las herramientas de software a utilizar, es necesario mencionar el equipo de hardware que utilizaremos tanto para desarrollar como para probar e implementar cada uno de los prototipos que se mencionarán a lo largo de este trabajo terminar, el cual es:

Equipo de hardware [1]	
Marca	DELL
Modelo	Inspiron 5567
Procesador	Intel Core i7 7gen
Tarjeta de video	Radeon (TM) R7 M445
Memoria RAM	16 GB
Disco Duro	256 GB SSD y 1 TB HDD

Cuadro 3.2: Equipo de hardware a utilizar [1]

Equipo de hardware [2]	
Marca	Asus
Modelo	FX505GM-BN061T
Procesador	Intel Core i5 8th Gen
Tarjeta de video	NVidia GeForce GTX 1060
Memoria RAM	8 GB
Disco Duro	256 GB SSD y 1 TB HDD

Cuadro 3.3: Equipo de hardware a utilizar [2]

Equipo de hardware [3]	
Marca	HP
Modelo	Pavilion g4
Procesador	Intel Core i3
Tarjeta de video	Intel Sandybridge Mobile
Memoria RAM	6 GB
Disco Duro	500 GB

Cuadro 3.4: Equipo de hardware a utilizar [3]

Junto con las herramientas de hardware y software a utilizar es necesario mencionar una serie de servicios básicos que son relevantes para el desarrollo de este trabajo terminal como lo son:

- Luz Eléctrica
- Agua Potable
- Internet
- Papelería en general

Estos servicios forman parte de la factibilidad técnica ya que sin ellos no se podría realizar este trabajo terminal y por eso mismo generan un costo, dicho costo se menciona en la Factibilidad Económica.

3.1.2. Factibilidad Operativa

Los recursos operativos de este trabajo terminal se calcularon con base en los recursos humanos con los que se cuenta y un análisis de las horas que el personal estará en operación trabajando sobre éste, el cual se muestra a continuación:

Horas a trabajar en el desarrollo del trabajo terminal						
Mes	No. de Días	Sábado y Domingo	Días hábiles	Horas de trabajo por día	Horas Totales	Días laborables (8 hr.)
Enero	31	8	9	2	18	2
Febrero	28	8	19	2	38	4
Marzo	31	10	20	2	40	5
Abril	30	10	15	2	30	3
Mayo	31	8	18	2	36	4
Junio	30	10	8	2	16	2
Agosto	31	9	12	2	24	3
Septiembre	30	10	16	2	32	4
Octubre	31	10	20	2	40	5
Noviembre	31	8	18	2	36	4

Cuadro 3.5: Relación de horas de trabajo estimadas para la realización de este trabajo terminal

Con esto podemos concluir que contamos suficiente tiempo para el desarrollo de este trabajo terminal, ya que las horas totales de trabajo están contempladas para cada uno de los integrantes del equipo

3.1.3. Factibilidad Económica

Luego de haber realizado el estudio de factibilidad técnica así como el operacional es necesario tomar en cuenta un estudio de factibilidad económica el cuan desglosará todo el gasto económico realizado para la elaboración de este trabajo terminal:

- Capital Humano: Se tienen contemplados aproximadamente 36 días laborales, es decir 288 horas para la elaboración de este trabajo terminal en el cual participaremos los cuatro integrantes
- Capital Técnico: Se cuentan con las instalaciones de la escuela, así como las viviendas de cada uno de los integrantes y los equipos de cómputo correspondientes.

En cuanto a los costos monetarios de todo el trabajo terminal se tiene lo siguiente:

- Servicios
En cuando a los servicios se considera un gasto mensual aproximado

de \$1,600.00 que al multiplicarlo por todo el tiempo de elaboración tenemos \$ 16,000.00.

- Software

En este caso durante todo el trabajo terminal usaremos herramientas gratuitas y la mayoría de software libre por lo que no dedicaremos una parte monetaria en el gasto de este tipo.

- Hardware

En este caso y como se mencionó anteriormente utilizaremos los equipos de cómputo personales de cada integrante lo que da un costo aproximado total de \$ 35,000.00.

- Recursos Humanos

Estamos estimando un gasto de \$80,000.00 por cada integrante para la elaboración de este trabajo terminal por lo que se genera un gasto total de \$320,000.00

Por lo que el costo final del desarrollo de este trabajo terminal es:

\$371,000.00

Conclusión Tras analizar todo este trabajo terminal y cada una de las partes del estudio de factibilidad es pertinente decir que los integrantes no contarán con el apoyo financiero antes mencionado y que el hardware actualmente ya es propiedad de los integrantes, por lo que el trabajo terminal se califica como "*Viable*" iniciando de esta manera su implementación acorde con las fechas mencionadas.

3.2. Herramientas a usar.

3.2.1. Software.

Para el desarrollo de software de este prototipo, es necesario hacer mención de algunas de las siguientes herramientas, para tener una idea clara sobre qué herramientas estamos utilizando y porque es que las estamos utilizando:

HTML5. HTML comenzó mucho tiempo atrás con una simple versión propuesta para crear la estructura básica de páginas web, organizar su contenido y compartir información, todo esto tenía la intención de comunicar información por medio de texto. El limitado objetivo de html motivó a varias

compañías a desarrollar nuevos lenguajes y programas para agregar características a la web nunca antes implementadas.

Dos de las opciones propuestas fueron Java y Palabra1; ambas fueron muy aceptadas y consideradas como el objetivo de la internet, sin embargo, con el crecimiento exponencial del internet, éste dejó de ser únicamente para los aficionados de los computadores y pasó a ser usado como un campo estratégico para los negocios y para la interacción social, ciertas limitaciones presentes en ambas tecnologías probaron ser una sentencia de muerte. Esta falta de integración resultó ser crítica y preparó el camino para la evaluación de un lenguaje del cual hablaremos un poco más a detalle después: JavaScript. Sin embargo, pese a su gran impacto, el mercado no terminó de adoptarlo plenamente y rápidamente su popularidad fue declinando, y el mercado terminó enfocando su atención a Flash. No fue hasta que los navegadores mejoraron su intérprete para JavaScript y la gente se empezaba a dar cuenta de las limitaciones que ofrecía Flash, que JavaScript fue implementado y comenzó a innovar la forma en la que se programaba la web. Al cabo de unos años, JavaScript, html y css eran considerados como la más perfecta combinación para evolucionar la Web.

HTML5 es una mejora de esta combinación, lo que unió todos estos elementos. HTML5 propone estándares para cada aspecto de la Web y también un propósito claro para cada una de las tecnologías involucradas. A partir de esto, html provee los elementos estructurales, CSS se concentra en volver esta estructura utilizable y atractiva a la vista, y JavaScript tiene todo lo necesario para brindar dinamismo y construir aplicaciones web completamente funcionales. Cabe mencionar que HTML5 funciona diferente dependiendo del navegador y la versión en la que se esté trabajando, algunos soportan más características o diferentes funcionalidades que otros.

CSS3.

Ya se ha mencionado anteriormente como es que HTML5 fue evolucionando a un grado de combinación de estructura y diseño, sin embargo, la web demanda diseño y funcionalidad, no solamente organización estructural o definición de secciones, la función de CSS se concentra en volver la estructura de HTML utilizable y atractivo a la vista.

Oficialmente CSS no tiene nada que ver con HTML4, no es parte de la especificación, es de hecho, un complemento desarrollado para superar las limitaciones y reducir la complejidad de HTML. Al principio, atributos den-

tro de las etiquetas HTML proveían estilos esenciales para cada elemento, pero a medida que HTML evolucionó, la escritura de códigos se volvió más compleja y HTML por sí mismo no pudo satisfacer más las demandas de los diseñadores. En consecuencia a esta demanda, CSS fue adoptado como la forma de separar la estructura de la presentación, y ha ido creciendo y ganando importancia, pero siempre desarrollado en paralelo enfocado en las necesidades de los diseñadores y apartado de la estructura de HTML.

La versión 3 de CSS sigue el mismo camino, pero esta vez con un mayor compromiso. La especificación de HTML5 fue desarrollada considerando CSS a cargo del diseño. Debido a esta consideración, la integración entre HTML y CSS es ahora vital para el desarrollo web y esta razón por la que cada vez que mencionamos HTML5 también estamos haciendo referencia a CSS3, aunque oficialmente se trate de dos tecnologías completamente separadas. Las nuevas características incorporadas en CSS3 están siendo implementadas e incluidas junto al resto de la especificación en navegadores compatibles con HTML5 [25].

Frameworks CSS.

Un framework de CSS es una biblioteca de estilos genéricos que puede ser usada para implementar diseños web. Aportan una serie de utilidades que pueden ser aprovechadas frecuentemente en los distintos diseños web. Un framework de CSS, si está bien diseñado e implementado, proporciona las siguientes ventajas [50]:

- Proporcionar una forma fácil y por tanto rápida de implementar diseños web.
- Nos aseguran que el diseño va a funcionar en una amplia gama de navegadores.
- Nos aseguran que su código cumple cierta norma estándar.
- Nos aseguran cierto grado de fiabilidad en la eficacia de las utilidades que nos aportan. El framework se supone que está bien probado para asegurarnos que no hay errores.

Sin embargo, un framework de CSS puede llevar aparejado las siguientes desventajas[50]:

- La importación de código del framework que no es necesario en nuestro diseño web concreto. Esto provoca un incremento innecesario del consumo del ancho de banda y del tiempo de descarga.

- Hay un menor control por parte del maquetador de lo que realmente está sucediendo en la visualización de la página web. Esto suele ser un problema cuando se tiene que corregir algún efecto indeseado.
- Al diseñar con código prehecho, podemos estar limitándonos en cuanto las posibilidades de elección del diseño web.

Bootstrap.

Bootstrap es framework de código abierto que contiene HTML, CSS y JS, en la actualidad, Bootstrap se ha convertido en uno de los frameworks de front-end más importantes en la actualidad. [51]

Materialize.

Materialize es un framework CSS creado y diseñado por Google. Combina los principios clásicos de diseño con las tecnologías e innovación moderna de Material Design. El objetivo de materialize es desarrollar un sistema de diseño que permita la unificación de la experiencia del usuario a través de los productos de google.

JavaScript.

JavaScript es considerado como el lenguaje de programación de html y de la web. Es un lenguaje de programación fácil de usar y muy versátil para el ámbito de la comunicación en redes. Los programas, llamados "scripts", se ejecutan en el navegador (Mozilla, Google Chrome, Internet Explorer, etc.) normalmente consisten en unas funciones que son llamadas desde el propio html cuando algún evento sucede.

Su primera aproximación a un uso real, fue en mayor parte para "dar vida a una página web", como dar animaciones a un botón, interacciones en tiempo real, entre otras más. JavaScript fue desarrollado por Palabra1, a partir del lenguaje Java, que en ese momento tenía mucho auge y popularidad, y su principal diferencia es que JavaScript sólo "funciona" dentro de una página html.

JavaScript fue declarado como estándar del European Computer Manufacturers Association (ECMA) en 1997, y poco después, también fue estandarizado por ISO [37].

JavaScript es un lenguaje interpretado, usado mayormente como complemento de ciertos objetivos específicos, sin embargo, uno de las innovaciones que ayudó a JavaScript fue el desarrollo de nuevos motores de interpretación, creados para acelerar el procesamiento del código. La clave de los motores más exitosos fue transformar el código de Javascript en código máquina para

obtener una velocidad de ejecución mejor que antes. Esto a la vez permitió superar viejas limitaciones de rendimiento y confirmar el lenguaje JavaScript como la mejor opción para la Web.

Para aprovechar esta prometedora plataforma de trabajo ofrecida por los nuevos navegadores, JavaScript fue expandido en cuestión de portabilidad e integración, a la vez, interfaces de programación de aplicaciones (APIs) fueron incorporando por defecto con cada navegador para asistir a JavaScript en funciones elementales. El objetivo de esto, fue principalmente hacer disponible funciones a través de técnicas de programación sencillas y estándares, expandiendo el alcance del lenguaje y facilitando la creación de programas útiles para la Web [25].

JQuery.

Antes de continuar con JQuery, debemos saber que es un framework. Un framework es un producto que sirve como base para la programación avanzada de aplicaciones, la cual aporta una serie de funciones o códigos para realizar tareas habituales. Dicho de otra manera, un framework son librerías de código que contienen procesos o rutinas listos para hacer uso. Habitualmente los programadores utilizan los frameworks para no tener que desarrollar ellos mismos las tareas más básicas, puesto que en el propio framework ya hay implementaciones que están probadas, funcionan y no se necesitan volver a programar.

Una vez comprendido que es un framework podemos continuar con jQuery. Este framework (para el lenguaje Javascript), es un producto que nos simplifica la vida para programar en este lenguaje. jQuery implementa una serie de clases (programación orientada a objetos) que nos permiten programar sin preocuparnos del navegador con el que nos está visitando el usuario, ya que funcionan de forma exacta en todas las plataformas más habituales. Así pues, este framework Javascript, nos ofrece una infraestructura con la que tendremos mucha mayor facilidad para la creación de aplicaciones complejas del lado del cliente. Por ejemplo, con jQuery obtendremos ayuda en la creación de interfaces de usuario, efectos dinámicos, aplicaciones que hacen uso de Ajax, etc. Cuando se programa Javascript con jQuery se tiene a su disposición una interfaz para programación que nos permitirá hacer cosas con el navegador que estemos seguros que funcionarán para todos los visitantes de la página.[27]

CryptoJs.

CryptoJs es una colección estándar y segura de algoritmos criptográficos implementados para JavaScript usando las mejores prácticas y patrones. En este trabajo usaremos CryptoJS en el desarrollo del Componente I, específicamente, utilizaremos los algoritmos de cifrado SHA-256 y AES, ya implementados con ella.[28]

NodeJs.

NodeJS es un framework de ejecución de JavaScript orientado a eventos asíncronos para construir aplicaciones en red escalables. Cabe destacar que a diferencia de la mayoría del código JavaScript, no se ejecuta en el navegador, sino en un servidor. [52]

NodeJs tiene una amplia cantidad de librerías para el desarrollo web, una herramienta de la cual nosotros haremos uso es la librería ó paquete de OpenSSL, el cual nos permite crear certificados SSL (x509) con código más limpio, ya que este paquete se encarga de hacer las llamadas al sistema para la creación de los certificados, haciendo llamadas a sus funciones nos permite crear ó revocar, entre muchas otras acciones, las cuales nos ayudarán a la creación del componente 2 (Autoridad certificadora). [53]

Java.

Java es un lenguaje de programación y una plataforma informática comercializada por primera vez en 1995 por Sun Microsystems. El lenguaje de programación Java fue originalmente desarrollado por James Gosling, de Sun Microsystems (constituida en 1983 y posteriormente adquirida el 27 de enero de 2010 por la compañía Oracle), y publicado en 1995 como un componente fundamental de la plataforma Java de Sun Microsystems. Su sintaxis deriva en gran medida de C y C++, pero tiene menos utilidades de bajo nivel que cualquiera de ellos. Las aplicaciones de Java son compiladas a bytecode (clase Java), que puede ejecutarse en cualquier máquina virtual Java (JVM) sin importar la arquitectura de la computadora subyacente [54].

Spring Framework.

Spring es un framework para el desarrollo de aplicaciones y contenedor de inversión de control, de código abierto para la plataforma Java. Si bien las características fundamentales de Spring Framework pueden ser usadas en cualquier aplicación desarrollada en Java, existen variadas extensiones para la construcción de aplicaciones web sobre la plataforma Java EE. A pesar de que no impone ningún modelo de programación en particular, este framework

se ha vuelto popular en la comunidad al ser considerado una alternativa, sustituto, e incluso un complemento al modelo EJB (Enterprise JavaBean) [55]. En este trabajo utilizaremos Spring Framework 5 para el desarrollo del Componente III, específicamente la API.

Apache Tomcat.

Apache Tomcat (también llamado Jakarta Tomcat o simplemente Tomcat) funciona como un contenedor de servlets desarrollado bajo el proyecto Jakarta en la Apache Software Foundation. Tomcat implementa las especificaciones de los servlets y de JavaServer Pages (JSP) de Oracle Corporation (aunque creado por Sun Microsystems).

Tomcat es un contenedor web con soporte de servlets y JSPs. Tomcat no es un servidor de aplicaciones, como JBoss o JOnAS. Incluye el compilador Jasper, que compila JSPs convirtiéndolas en servlets. El motor de servlets de Tomcat a menudo se presenta en combinación con el servidor web Apache. Tomcat puede funcionar como servidor web por sí mismo. En sus inicios existió la percepción de que el uso de Tomcat de forma autónoma era sólo recomendable para entornos de desarrollo y entornos con requisitos mínimos de velocidad y gestión de transacciones. Hoy en día ya no existe esa percepción y Tomcat es usado como servidor web autónomo en entornos con alto nivel de tráfico y alta disponibilidad. Dado que Tomcat fue escrito en Java, funciona en cualquier sistema operativo que disponga de la máquina virtual Java [56].

En este trabajo terminal, estaremos usando Apache Tomcat 9.

OpenSSL.

OpenSSL es un proyecto de software libre basado en SSLeay, desarrollado por Eric Young y Tim Hudson.

Consiste en un robusto paquete de herramientas de administración y bibliotecas relacionadas con la criptografía, que suministran funciones criptográficas a otros paquetes como OpenSSH y navegadores web (para acceso seguro a sitios HTTPS). Estas herramientas ayudan al sistema a implementar el Secure Sockets Layer (SSL), así como otros protocolos relacionados con la seguridad, como el Transport Layer Security (TLS). OpenSSL también permite crear certificados digitales que pueden aplicarse a un servidor, por ejemplo Apache [44].

Para este trabajo terminal utilizaremos OpenSSL para generar los certificados de los usuarios, así como los certificados autofirmados para el Componente II y el Componente III.

MongoDB.

MongoDB es un sistema de base de datos multiplataforma orientado a documentos, de esquema libre, esto significa que cada entrada o registro puede tener un esquema de datos diferente, con atributos o "columnas" que no tienen por qué repetirse de un registro a otro.

Las características más destacadas son su velocidad y su sencillo sistema de consulta de los contenidos de la base de datos. Alcanzando así un balance perfecto entre rendimiento y funcionalidad. MongoDB utiliza un modelo NoSQL el cual es un modelo de agregación que se basan en la noción de agregado, entendiendo el agregado como una colección de objetos relacionados que se desean tratar de forma semántica e independiente [43].

Las ventajas que ofrece MongoDB como herramienta de desarrollo de base de datos no relacionales son:

- La base de datos no tiene un esquema de datos predefinido.
- El esquema puede variar para instancias de datos que pertenecen a una misma entidad.
- En ocasiones el gestor de la base de datos no es consciente del esquema de la base de datos.
- Permite reducir los problemas de concordancia entre estructuras de datos usadas por las aplicaciones y la base de datos.
- Frecuentemente se aplican técnicas de desnormalización de los datos.

MySQL.

Es un sistema de gestión de bases de datos relacional desarrollado bajo licencia dual: Licencia pública general/Licencia comercial por Oracle Corporation y está considerada como la base de datos de código abierto más popular del mundo, y una de las más populares en general junto a Oracle y Microsoft SQL Server, sobre todo para entornos de desarrollo web [57].

El modelo relacional, para el modelado y la gestión de bases de datos, es un modelo de datos basado en la lógica de predicados y en la teoría de conjuntos. Su idea fundamental es el uso de relaciones. Estas relaciones podrían considerarse en forma lógica como conjuntos de datos llamados tuplas. Es el modelo más utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente [58].

Las principales ventajas son:

- Provee herramientas que garantizan evitar la duplicidad de registros.
- Garantiza la integridad referencial, así, al eliminar un registro elimina todos los registros relacionados dependientes.
- Favorece la normalización por ser más comprensible y aplicable.

Mientras que las principales desventajas son:

- Presentan deficiencias con datos gráficos, multimedia, CAD y sistemas de información geográfica.
- No se manipulan de forma eficiente los bloques de texto como tipo de dato.

JSEncrypt.

Generada por Travis Tidwell, JSEncrypt es una librería para JavaScript que permite el cifrado, descifrado y la generación de llaves de RSA con OpenSSL.

VSFTPD.

VSFTPD son las siglas para Very Secure File Transport Protocol Daemon y es un servidor de FTP que nos da la ventaja de garantizarnos seguridad, estabilidad y desempeño al momento de montarlo, además de que nos permite configurar su control de acceso mediante una serie de criterios, un factor muy útil y necesario que utilizaremos más adelante para el desarrollo de nuestro segundo prototipo. [60] La herramienta se escogió debido a la comparación de los diferentes protocolos de internet que permiten el intercambio de archivos, los cuales son:

- **FTP:** *File Transfer Protocol* ó *Protocolo de transferencia de archivos* es un protocolo para transferencia de archivos entre sistemas conectados a una red. Desde un equipo cliente se puede conectar a un servidor para descargar archivos desde él o para enviarle archivos, independientemente del sistema operativo utilizado en cada equipo. FTP nos permite tener un control de accesos de las diferentes carpetas, dando así diferentes prioridades. Al subir archivos al servidor FTP si se remplaza alguno, éste no podrá recuperarse [62].
- **HTTP:** *Hypertext Transfer Protocol* ó *Transferencia de HiperTexto* es un protocolo cliente-servidor que establece los intercambios de información entre los clientes web y los servidores HTTP. La versión mas

reciente de HTTP es la 1.1, y su especificación se encuentra recogida en el documento RFC 2616. HTTP se establece sobre la capa TCP/IP [63]. Las tres operaciones más usadas que permiten a un cliente dialogar con el servidor son [62]:

1. GET: Recoger un objeto
2. POST: Enviar información al servidor
3. HEAD: Solicitar las características de un objeto

- **TFTP:** *Protocolo Trivial de Transferencia de Archivos* es un protocolo que proporciona la transferencia de archivos sin autenticación de usuario. TFTP está más enfocado a las aplicaciones que no necesitan las interacciones sofisticadas que proporciona el protocolo de transferencia de archivos (FTP) [64].
- **SMTP:** *Simple Mail Transfer Protocol* ó *Protocolo Simple de Transferencia de Correo* es protocolo que funciona en línea, encapsulado en una trama TCP/IP. El correo se envía directamente al Servidor de Correo del destinatario. El protocolo SMTP funciona con comandos de textos enviados al servidor SMTP. A cada comando enviado por el Cliente le sigue una respuesta del servidor SMTP compuesta por un número y un mensaje descriptivo [65].
- **HTTPS:** *HyperText Transfer Protocol Secure* ó *Protocolo Seguro de Transferencia de Hipertexto* es un protocolo de comunicación de Internet que protege la integridad y la confidencialidad de los datos de los usuarios entre sus ordenadores y el sitio web. Los datos que se envían mediante HTTPS están protegidos con el protocolo Seguridad en la capa de transporte (TLS), que da estas tres capas clave de seguridad [66]:
 1. Cifrado: Se cifran los datos intercambiados.
 2. Integridad de los datos: Los datos no pueden modificarse ni dañarse durante las transferencias.
 3. Autenticación: Demuestra que los usuarios se comunican con el sitio web previsto. Proporciona protección frente a los ataques *man-in-the-middle*.

Wireshark.

Wireshark es una analizadora de paquetes de red. Un analizador captura paquetes de red y muestra los datos del paquete con el mayor detalle posible.

Ésta herramienta es software libre y es el mejor analizador de paquetes hoy en día [45]. Utilizaremos Wireshark para analizar los paquetes que viajan a través de la red, en este caso la petición modificada la cual contiene el certificado y el patrón ocultados mediante el método *Chaffing*.

3.2.2. Hardware.

En el ámbito del hardware, utilizaremos los equipos de cómputo con los cuales contamos actualmente los integrantes de este equipo, los cuales se especificarán a continuación:

Equipo de hardware utilizado. [1]	
Marca	Asus
Modelo	FX505GM-BN061T
Procesador	Intel Core i5 8th Gen
Tarjeta de video	NVidia GeForce GTX 1060
Memoria RAM	8 GB
Disco duro	256 GB SSD y 1 TB HDD

Equipo de hardware utilizado. [2]	
Marca	HP
Modelo	Pavilion g4
Procesador	Intel Core i3
Tarjeta de video	Intel Sandybridge Mobile
Memoria RAM	6 GB
Disco duro	500GB

Equipo de hardware utilizado. [3]	
Marca	DELL
Modelo	Inspiron 5567
Procesador	Intel Core i7
Tarjeta de video	Radeon (TM) R7 M445
Memoria RAM	16 GB
Disco duro	256 GB SSD y 1 TB HDD

Equipo de hardware utilizado. [4]	
Marca	HP
Modelo	Pavilion 15-p000ns
Procesador	AMD A A8-5545M
Tarjeta de video	Radeon R8 M540
Memoria RAM	8 GB
Disco duro	1TB

3.3. Arquitectura del sistema.

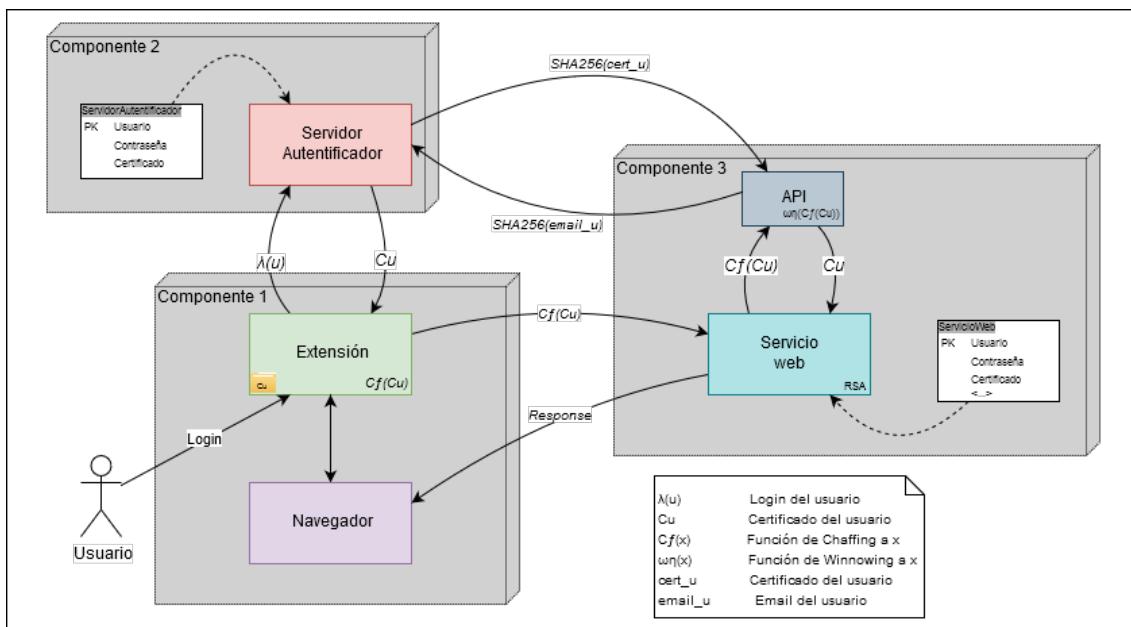


Figura 3.1: Arquitectura General del Sistema

3.3.1. Descripción de la arquitectura del sistema.

El sistema se compone de 3 grandes bloques los cuales se comunicarán vía red:

1. **Navegador Chrome con la Extensión instalada:** Este primer bloque es el que se encuentra interactuando directamente con el usuario de nuestro sistema, consiste en la extensión creada por nosotros y el

navegador en el que el usuario realiza peticiones a diferentes servicios en la web.

2. **Servidor autentificador:** Este bloque va ser el encargado de generar los certificados para cada usuario que se registre en la extensión y enviarlos a la extensión. Para la generación de dichos certificados utilizaremos una autoridad certificadora con lo que garantizamos la seguridad de estos mismos. Por otro lado para almacenar los datos de nuestros usuarios contaremos con una tabla que contenga como principales campos:

- Usuario
- Contraseña
- Certificado

3. **Servidor web con API instalada:** En este módulo el servicio web contará con una API, que se encargará de reconocer las peticiones que se reciban con nuestro método de autenticación y será la encargada de interpretar los datos y facilitarle la información de autenticación al servicio. Es importante destacar que el servicio almacenará el certificado en cuestión para que el usuario pueda autenticarse la próxima vez de forma automática.

Es importante mencionar que la comunicación entre cada uno de los bloques se realizará mediante técnicas que permitan la confidencialidad de los datos, por un lado la comunicación del certificado que viajará entre la extensión y el servicio web se encontrará oculto mediante Chaffing and Winnowing y el patrón necesario para el método se encontrará cifrado mediante RSA. La comunicación entre el Servidor autentificador y la Extensión se encontrará oculto mediante un socket seguro.

3.4. Diagrama BPMN

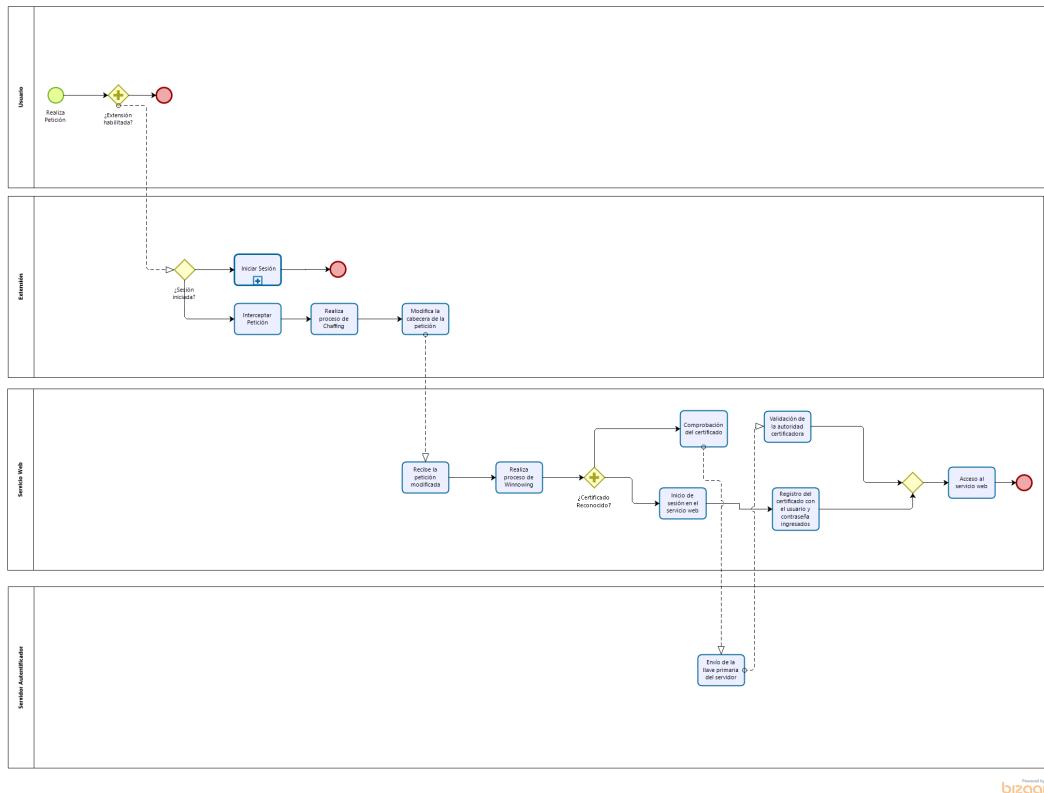
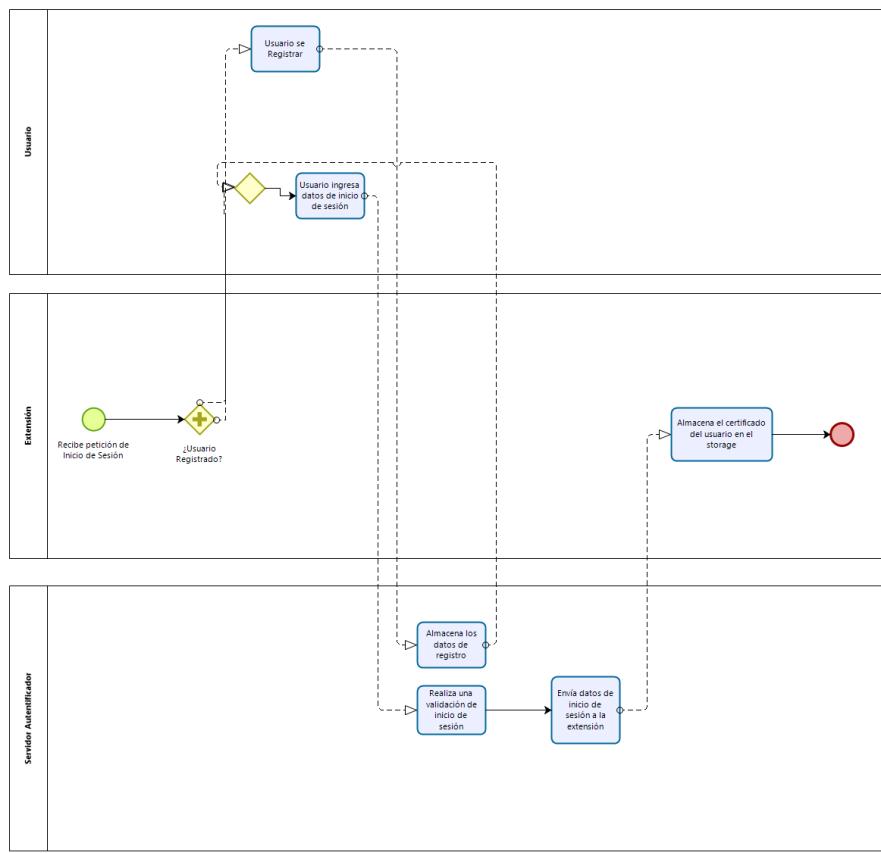


Figura 3.2: Diagrama de Modelo y Notación de Procesos de Negocio

3.4.1. Diagrama BPMN Proceso: Inicio de Sesión



Powered by
bizagi
Modeler

Figura 3.3: Diagrama BPMN del proceso Iniciar Sesión

3.5. Diagrama de casos de uso general.

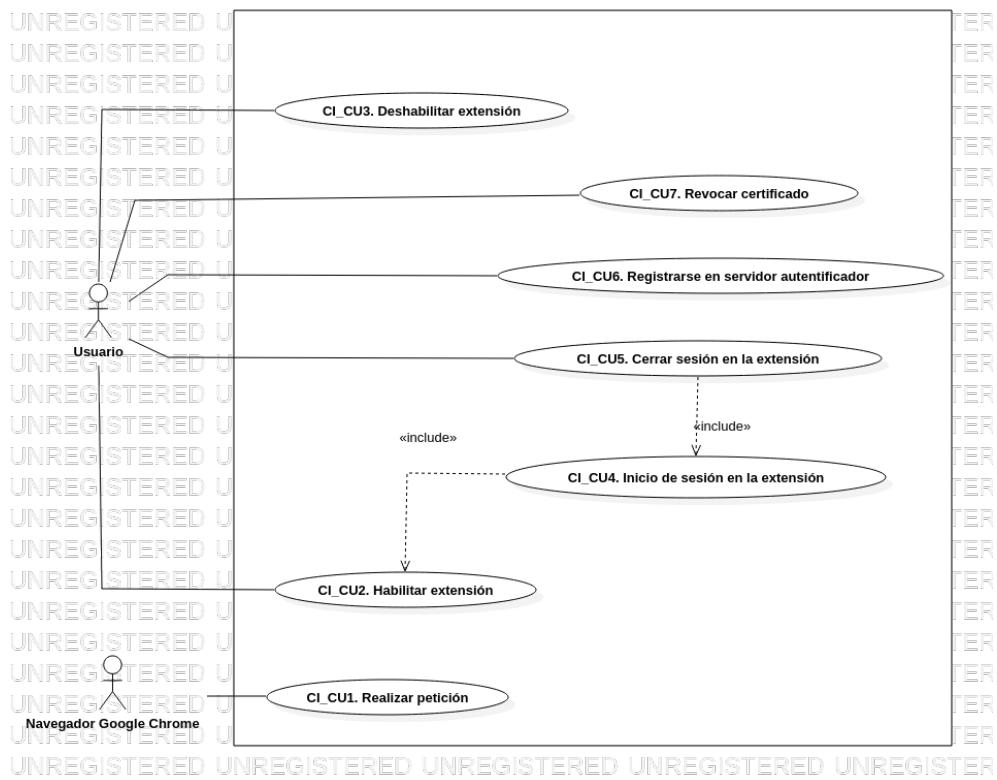


Figura 3.4: Diagrama de casos de uso general del sistema

NOTA: En la sección 4 "Diseño", cada prototipo describirá sus casos de uso correspondientes para su funcionamiento.

3.6. Componente I. Extensión.

3.6.1. Descripción.

Este componente permite a la extensión poder interceptar peticiones hechas por el usuario a través del navegador de Google Chrome.

Una vez que se intercepta la petición, ésta podrá ser modificada. La modificación se hará sólo mientras la extensión esté habilitada, y tiene como objetivo injectar el certificado autenticador en el encabezado del protocolo una vez que se haya llevado a cabo el proceso de Chaffing. Cuando dicho certificado

sea injectado, la extensión deberá liberar la petición para que salga a red. Este certificado será único por cada usuario y será obtenido desde el Componente II cuando el usuario inicie sesión en la extensión. Si el usuario no puede iniciar sesión debido a que no tiene una cuenta, desde la extensión se podrá registrar para poder obtener su certificado.

Además, el usuario podrá cerrar la sesión en la extensión si es que así lo desea, lo que eliminaría el certificado de la máquina local del usuario. Por otro lado, si el usuario desea revocar su certificado, lo podrá hacer también desde este componente.

El propósito de este prototipo es utilizar la técnica de *Chaffing and Winnowing* en este nuevo método de autentificación, para evitarle al usuario la tediosa tarea de ingresar sus credenciales cada vez que accede al servicio y brindarle la seguridad necesaria al iniciar sesión.

Para injectar el certificado autentificador, es necesario crear un "*patrón de chaffing*", este patrón lo generaremos aleatoriamente para después mandarlo junto con la petición HTTP. Dicho patrón irá cifrado con la clave pública del Componente III (API). El objetivo de mandar el patrón junto con el protocolo HTTP, es que el servidor pueda descifrar el patrón con su clave privada y con él realizar la etapa de *winnowing* para extraer el certificado.

3.6.2. Estudio de requerimientos.

3.6.2.1. Requerimientos Funcionales.

CI_RF1. Interceptar petición . La extensión deberá interceptar la petición del navegador, en cuanto el usuario realice alguna a través de éste.

CI_RF2. Deshabilitar extensión. El usuario podrá deshabilitar la extensión, para que ésta no vigile su actividad en el navegador.

CI_RF3. Habilitar extensión. El usuario podrá habilitar la extensión, para que ésta vigile las peticiones .

CI_RF4. Validar petición. La extensión deberá analizar la petición previamente interceptada, y validar si ésta es HTTP(S) o no.

CI_RF5. Interceptar petición. La extensión deberá evitar que la petición salga a red, deteniéndola para aplicar la etapa de *Chaffing*.

CI_RF6. Inicio de sesión en la extensión. La extensión contará con una interfaz para el ingreso de datos del usuario, donde ingresará un usuario y contraseña.

CI_RF7. Obtención del certificado autentificador. La extensión, mediante el inicio de sesión del usuario, se conectará al Componente II (Servidor autentificador) para obtener el certificado autentificador.

CI_RF8. Almacenamiento del certificado autentificador. La extensión deberá almacenar el certificado autentificador devuelto por la autoridad certificadora.

CI_RF9. Generación de patrón de *Chaffing*. La extensión generará un patrón para poder implementar el método de *Chaffing*. Este patrón será generado al azar, y es aquel que se usará para introducir el código autentificador en el protocolo HTTP.

CI_RF10. Etapa de *Chaffing*. Por medio del método *Chaffing* se agregará al encabezado HTTP el certificado del usuario, utilizando el patrón de *Chaffing* del requerimiento funcional CI_RF9. Generación de patrón de *Chaffing*

CI_RF11. Liberación de Petición. Se liberará el bloqueo a la petición HTTP impuesto por el requerimiento funcional CI_RF5. Interceptar petición.

CI_RF12. Cierre de sesión en la extensión. El usuario podrá cerrar sesión en la extensión para que ésta no siga guardando su certificado.

CI_RF13. Registro en servidor autentificador. La extensión contará con una interfaz para el ingreso de datos del usuario, para que éste se registre en el servidor y pueda obtener un certificado. Los datos a ingresar serán: correo electrónico (email), y contraseña.

CI_RF14. Revocar certificado. El usuario podrá, mediante la interfaz de la extensión, revocar su certificado en caso de que así lo deseé. La interfaz requerirá que se introduzca el email correspondiente al usuario y la contraseña para validar la identidad del usuario.

CI_RF15. Ver contraseña. Si el usuario así lo desea, podrá habilitar la función ver contraseña cuando escribe. Este requerimiento se usara en los

requerimientos CI_RF6. Inicio de sesión en la extensión y CI_RF13. Registro en servidor autenticador.

3.6.2.2. Requerimientos no Funcionales.

CI_RNF1. Plataforma de implementación. La extensión será implementada en el navegador Google Chrome Desktop.

CI_RNF2. Versión del navegador La extensión funcionará a partir de la versión 28.0.

CI_RNF3. Tecnologías para la interfaz de usuario Para el sistema se hará uso de HTML5, JavaScript, CSS3, JSON.

CI_RNF4. Codificación en base64 del Chaffing. Se codificará en base64 la cadena de chaffing para permitir el envío de los datos, debido a que el algoritmo nos devuelve ciertos caracteres no permitidos para enviar.

CI_RNF5. Conexión a internet. Para el funcionamiento de la extensión, no es necesario que se tenga conexión a internet.

CI_RNF6. Tamaño del código autenticador. El tamaño del código autenticador es de aproximadamente 10496 bits pero puede variar dependiendo de la información introducida para su elaboración.

CI_RNF7. Almacenado de archivo en la extensión. Se necesita tener almacenado el archivo en la extensión de Google Chrome. Específicamente, utilizamos el **Storage** para almacenarlo.

3.6.3. Reglas del negocio.

CI_RN1. Extensión habilitada. En cuanto el usuario lo indique por medio de la Significado, la extensión deberá vigilar la actividad que éste realice en el navegador para interceptar una petición.

CI_RN2. Extensión deshabilitada. En cuanto el usuario lo indique por medio de la Significado, la extensión deberá dejar de vigilar la actividad que éste realice en el navegador.

CI_RN3. Petición válida. La extensión modificará la petición siempre y cuando se trate de una petición válida HTTP .

CI_RN4. Inicio de sesión de extensión por usuario. Cada usuario que desee utilizar la extensión sólo deberá tener una cuenta con un correo electrónico y una contraseña respectiva a este usuario.

CI_RN5. Acceso a internet. Se debe de contar con acceso a internet para que la extensión pueda enviar al servidor la petición modificada.

CI_RN6. Longitud de código autentificador. La longitud del código autentificador es de aproximadamente 10496 bits pero puede variar dependiendo de la información introducida para su elaboración.

CI_RN7. Longitud del campo de usuario. La longitud del usuario no debe pasar de 20 caracteres, y mínimo será de 3 caracteres.

CI_RN8. Longitud del campo contraseña. La longitud de la contraseña no debe pasar de 16 caracteres, y mínimo sera de 8 caracteres.

CI_RN9. Caracteres permitidos en campo usuario. Los caracteres permitidos en el campo de usuario son únicamente símbolos alfanuméricos y guion bajo.

CI_RN10. Caracteres permitidos en campo contraseña. Los caracteres permitidos en el campo de contraseña son únicamente símbolos alfanuméricos así como los símbolos _ @ / # ? .

CI_RN11. Formato de contraseña. La contraseña debe tener al menos un número y una letra mayúscula.

CI_RN12. Longitud del campo email. La longitud del email no debe pasar de 100 caracteres, y mínimo sera de 7 caracteres.

CI_RN13. Caracteres permitidos en campo email. Los caracteres permitidos en el campo email son únicamente símbolos alfanuméricos así como los símbolos ' ' '@' ' _ ' - ' , ' () { }] [' .

3.7. Componente II: Servidor autentificador.

3.7.1. Descripción.

Para este componente, implementaremos un servidor autentificador en el cual se crearán y almacenarán los certificados de los usuarios. En este servidor, los usuarios tendrán registrada una cuenta a la cual accederán desde el Componente I. Una vez que el usuario inicie sesión en la extensión, este servidor autentificador regresará como respuesta el certificado generado para que la extensión lo almacene. Para la creación del certificado autentificador se utilizará la herramienta OpenSSL, además, la comunicación entre extensión y servidor se hará bajo SSL/TLS.

La principal función de este componente es gestionar las cuentas y certificados en una base de datos, de tal forma que el Componente I se pueda comunicar con este componente ya sea para que le genere un certificado a una cuenta (en el caso de que el usuario se esté registrando en el servidor autentificador por primera vez), para mandarle su certificado si es que ya se encuentra registrado en el servidor, o bien, para revocar un certificado. Así mismo, este componente también estará comunicado con el Componente III, específicamente la API, para controlar la revocación de certificados.

El propósito de este componente es poder crear y utilizar un certificado real, generado por una autoridad certificadora de confianza y con el cual se pueda autenticar a un usuario en un servicio web.

3.7.2. Estudio de requerimientos.

3.7.2.1. Requerimientos funcionales.

CII_RF1. Creación de nuevo usuario. La autoridad certificadora podrá crear una nueva instancia en la base de datos de acuerdo a los datos recuperados por la extensión (Usuario y contraseña), si es que estos no se encuentran guardados en la base de datos.

CII_RF2. Generar certificado. La autoridad certificadora deberá generar un certificado diferente para cada usuario que se registre en la extensión con los datos proporcionados por la misma.

CII_RF3. Asignar certificado. La autoridad certificadora deberá asignar el certificado generado al usuario correspondiente en la base de datos.

CII_RF4. Devolver certificado. La autoridad certificadora deberá enviar el certificado correspondiente al usuario quien realiza la solicitud para obtener el certificado.

CII_RF5. Actualizar certificado. La autoridad certificadora deberá actualizar el certificado cuanto éste haya caducado.

CII_RF6. Revocar certificado. La autoridad certificadora deberá revocar un certificado de acuerdo a la petición dada por el usuario, si esta petición tiene los datos correctos del usuario, su certificado ligado al mismo se eliminará, se creará uno nuevo y se le asignará a dicho usuario.

CII_RF7. Comprobar certificado certificado. La autoridad certificadora deberá poder comprobar la existencia de un certificado basándose en el código hash recibido por parte del Componente III.

3.7.2.2. Requerimientos no funcionales.

CII_RNF1. Plataforma de implementación. La autoridad certificadora será implementada en NodeJs.

CII_RNF2. Version de SSL/TLS. La autoridad certificadora usará como conexión, así como la generación de certificado, OpenSSL con la versión 1.1.1.

CII_RNF3. Base de datos. La autoridad certificadora se conectará a una base de datos MongoDB 4.0.10.

CII_RNF4. Formato de certificado. El servidor autenticador deberá crear certificados X509.

3.7.3. Reglas del negocio.

CII_RN1. Acceso a internet. Se debe de contar con acceso a internet para que la autoridad certificadora pueda enviar a la extensión el certificado generado.

CII_RN2. Conexión a la base de datos. La autoridad certificadora cuenta con un pull de conexiones a la base de datos para poder insertar u

obtener certificados de los usuarios.

CII_RN3. Conexión SSL. La autoridad certificadora cuenta con un certificado auto firmado, el cual le permitirá tener una conexión SSL con la extensión para poder así enviar el certificado de dicho usuario de manera segura y cifrada.

CII_RN4. Parámetros POST. La autoridad certificadora recibirá mediante método POST peticiones las cuales tendrán como datos: Correo electrónico (email) y password (Con hash 256).

CII_RN5. Parámetros GET. La autoridad certificadora enviará una pagina como respuesta si el usuario aprueba el uso del certificado de la autoridad certificadora.

CII_RN6. Respuestas a peticiones. La autoridad certificadora podrá devolver un certificado creado por ella misma. Así como también devolver códigos de respuestas, como lo son: 200 (Se realizo petición correctamente) ó 404 (No se encontró usuario).

3.8. Componente III: API.

3.8.1. Descripción.

Para el componente III, se utilizará un servicio web de prueba para mostrar la funcionalidad del inicio de sesión por este método propuesto, por lo cual dicho servicio no se ve reflejado en el análisis desarrollo del Componente III.

Se realizará la etapa de *winnowing* a la petición HTTP para obtener el certificado. Esto último lo realizará una API dedicada exclusivamente a ello, con conexión al Componente II para checar el estatus y validez del certificado.

El servicio web sólo tendrá que conectarse al API para obtener el certificado y su validez, con base en ello realizar la lógica del negocio para iniciar sesión.

Así mismo, el API implementará un cifrado asimétrico (RSA), esto con la finalidad de que el Componente I pueda tomar la llave pública del servidor y cifrar el *"patrón de chaffing"* que se mandará en la petición. Una vez que el patrón y el chaffing llegue al API, sólo ella podrá descifrar el patrón con su llave privada, dandole la integridad necesaria al patrón para viajar a través

de la red.

El propósito de este componente es poder obtener el certificado que identificará a cada usuario, para poder validarla y saber si se debe de dar o negar el acceso. Cabe mencionar que, la primera vez que el servidor reciba el certificado autenticador en el servicio, éste pedirá al usuario que se inicie sesión con la finalidad de poder asociar este certificado a una cuenta del servicio; para las peticiones posteriores, el certificado ya contará con una cuenta asociada a la cual podrá dar acceso siempre y cuando el certificado sea válido.

3.8.2. Estudio de requerimientos.

3.8.2.1. Requerimientos funcionales.

CIII_RF1. Primer inicio de sesión con chaffing. El servicio deberá guardar el certificado en caso de que sea la primera vez que le llega y si es válido, asociandolo a un usuario por medio de un inicio de sesión.

CIII_RF2. Inicio de sesión con chaffing. El servicio web, en caso de que el certificado sea válido, deberá de dar acceso a la cuenta del usuario sin la necesidad de tener que pedir las credenciales del mismo, a excepción de lo establecido en CIII_RF1. Primer inicio de sesión con chaffing.

CIII_RF3. Negación del inicio de sesión. En caso de que el certificado de usuario no sea válido, el servicio deberá negar el acceso al usuario.

CIII_RF4. Envío de chaffing y patrón de chaffing. El servicio web deberá mandar, en caso de que los reciba, el chaffing y el patrón de chaffing al API para que ésta pueda realizar el winnowing sobre el chaffing.

CIII_RF5. Comunicación con autoridad certificadora. El API desarrollada en este componente, deberá tener comunicación con el Componente II. Servidor Autentificador para verificar la validez del certificado.

CIII_RF6. Generación de llaves. El API deberá tener su par de llaves, pública y privada, para poder realizar el cifrado asimétrico al '*patrón de chaffing*'.

CIII_RF7. Descifrado del patrón de chaffing. El API deberá ser capaz de descifrar el patrón de chaffing con su llave privada, para poder

realizar el winnowing sobre el chaffing.

CIII_RF8. Realización de la etapa de winnowing. El API deberá de poder realizar la etapa de winnowing para poder obtener el certificado.

CIII_RF9. Validación de firma. Posteriormente a la etapa de winnowing del CIII_RF8, este tiene que validar que la firma del certificado, para saber si este fue emitido por nuestro mismo servidor autenticador.

CIII_RF10. Retorno de certificado. La API deberá de retornarle el certificado y su estatus al servicio para que este pueda realizar la lógica del negocio necesaria para el inicio de sesión.

CIII_RF11. Descifrado AES al patrón. La API deberá de descifrar el patrón utilizando AES.

3.8.2.2. Requerimientos no funcionales.

CIII_RNF1. Plataforma de implementación. Las pruebas del servidor web serán realizadas en el servidor Apache Tomcat.

CIII_RNF2. Base de datos. Se utilizará una base de datos en MySQL para guardar la información de los usuarios.

CIII_RNF3. Tamaño de llaves. Se utilizarán llaves de un tamaño de 2048 bits para el cifrado de tipo RSA.

CIII_RNF4. Framework de desarrollo. Se utilizará Spring Framework Core 5 para implementar la API con JAVA 8.

3.8.3. Reglas del negocio.

CIII_RN1. El servicio debe contar con una correcta conexión. Se debe de contar con una conexión estable tanto a internet para que los usuarios finales puedan acceder como al servidor autenticador necesario para el correcto funcionamiento.

CIII_RN2. Conexión a base de datos. El servidor debe tener una correcta conexión a la base de datos tanto del negocio como la modificación

necesaria para que el inicio de sesión implementado funcione correctamente.

CIII_RN3. API en el servidor. El servidor debe de contar con la API para que pueda realizar el procesos de *winnowing*.

CIII_RN4. Inicio de Sesión. El servidor debe contar con dos inicios de sesión, uno propio del negocio y otro propio del inicio de sesión con *Chaffing and Winnowing*.

Capítulo 4

Diseño.

4.1. Componente I: Extensión.

4.1.1. Diagrama de casos de uso

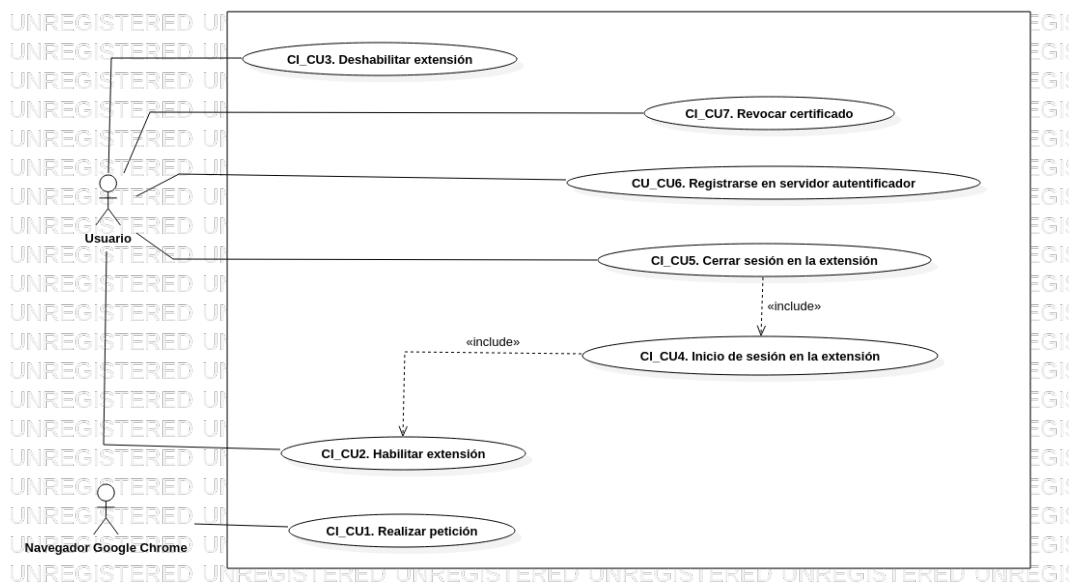


Figura 4.1: Diagrama de casos de uso del Componente I.

4.1.1.1. Descripción de casos de uso.

Caso de uso: CI_CU1. Realizar petición.	
Concepto	Descripción
Actor	Navegador de Google Chrome.
Propósito	Este caso de uso permite al navegador realizar una petición , ordenada por el usuario o un sistema externo.
Entradas	URL del servicio web solicitado.
Salidas	Petición .
Pre-condiciones	Algún agente externo (Sistema o usuario) ha ordenando al navegador mandar una petición .
Post-condiciones	Creación de la petición HTTP.
Reglas del negocio	-
Errores	La petición no se pudo realizar. La petición no es tipo .

Cuadro 4.1: Descripción CU: CI_CU1

... Trayectoria Principal ...

1. **El Usuario o El Sistema Externo** realiza una petición en el navegador Google Chrome.
2. **El navegador** realiza la petición.

... Fin de la Trayectoria Principal ...

... Trayectoria Alternativa 1 ...

1. **El Usuario o El Sistema Externo** realiza una petición que no es en el navegador Google Chrome.
2. **El navegador** realiza la petición.

... Fin de la Trayectoria Alternativa 1 ...

Caso de uso: CI_CU2. Habilitar extensión.	
Concepto	Descripción
Actor	Usuario.
Propósito	Este caso de uso, permite al usuario habilitar la extensión, para que ésta sea capaz de ver todas las peticiones que realiza el navegador.
Entradas	Indicación de habilitar extensión, mediante interfaz de usuario.
Salidas	-
Pre-condiciones	CI_CU3.
Post-condiciones	-
Reglas del negocio	CI_RN1.
Errores	No se puede habilitar la extensión.

Cuadro 4.2: Descripción CU: CI_CU2

... Trayectoria Principal ...

1. **El usuario** da click en el ícono de la extensión.
2. **El usuario** da click en el botón 'Activar'.
3. **La extensión** empieza a vigilar las peticiones que se realicen a través del navegador.

... Fin de la Trayectoria Principal ...

... Trayectoria Alternativa 1 ...

1. **La extensión** no muestra el botón 'Activar', por ende el usuario no puede dar click.

... Fin de la Trayectoria Alternativa 1 ...

Caso de uso: CI_CU3. Deshabilitar extensión.	
Concepto	Descripción
Actor	Usuario.
Propósito	Este caso de uso permite al usuario deshabilitar la extensión, para que ésta ignore todas las peticiones que se realicen por medio del navegador.
Entradas	Indicación de deshabilitar extensión, mediante interfaz de usuario.
Salidas	-
Pre-condiciones	CI_CU2.
Post-condiciones	-
Reglas del negocio	CI_RN2.
Errores	No se puede deshabilitar la extensión.

Cuadro 4.3: Descripción CU: CI_CU3

... Trayectoria Principal ...

1. **El usuario** da click en el ícono de la extensión.
2. **El usuario** da click en el botón.
3. **La extensión** deja de vigilar las peticiones que se realicen a través del navegador.

... Fin de la Trayectoria Principal ...

... Trayectoria Alternativa 1 ...

1. **La extensión** no muestra el botón, por ende el usuario no puede dar click.

... Fin de la Trayectoria Alternativa 1 ...

Caso de uso: CI_CU4. Inicio de sesión en la extensión	
Concepto	Descripción
Actor	Usuario
Propósito	<p>Este caso de uso permite al usuario poder iniciar sesión en la extensión para posteriormente obtener un código autentificador, es decir, el certificado del usuario, al cual se le aplicará <i>Chaffing</i>.</p> <p>Sólo se requerirá iniciar sesión una sola vez ya que después de ello el certificado se guardará en la extensión.</p>
Entradas	Usuario y Contraseña
Salidas	Certificado del usuario que acaba de iniciar sesión.
Pre-condiciones	Haber instalado la extensión en el navegador Google Chrome y haberla habilitado.
Post-condiciones	Por medio de los datos introducidos por el usuario, se obtendrá el certificado del mismo para que se pueda realizar la etapa de chaffing al momento que deseé identificarse en un servicio web.
Reglas del negocio	CI_RN1. CI_RN4. CI_RN7. CI_RN8. CI_RN9. CI_RN10. CI_RN11.
Errores	No se encuentra usuario registrado. No se puede obtener el código autentificador. No se pudo guardar el código autentificador. Contraseña no válida. Email no válido.

Cuadro 4.4: Descripción CU: CI_CU4

... Trayectoria Principal ...

1. ***El usuario*** da click en el ícono de la extensión.
2. ***EL usuario*** da click en el botón 'Iniciar sesión'.
3. ***La extensión*** abre en una nueva pestaña del navegador la página principal de la extensión.

4. ***El usuario*** llena correctamente los campos del formulario, los cuales son: 'Email' y 'Contraseña'.
5. ***EL usuario*** da click en el botón 'Iniciar sesión'.
6. ***La extensión*** despliega un popup con la información de la obtención del certificado.
7. ***El usuario*** da click en el botón 'Aceptar'.

... Fin de la Trayectoria Principal ...

... Trayectoria Alternativa 1 ...

1. ***El usuario*** da click en el ícono de la extensión.
2. ***EL usuario*** da click en el botón 'Iniciar sesión'.
3. ***La extensión*** abre en una nueva pestaña del navegador la página principal de la extensión.
4. ***El usuario*** da click en el botón 'Iniciar sesión'.
5. ***La extensión*** abre la página de inicio de sesión.
6. ***El usuario*** llena correctamente los campos del formulario, los cuales son: 'Email' y 'Contraseña'.
7. ***EL usuario*** da click en el botón 'Iniciar sesión'.
8. ***La extensión*** despliega un popup con la información de la obtención del certificado.

... Fin de la Trayectoria Alternativa 1 ...

... Trayectoria Alternativa 2 ...

1. ***El usuario*** da click en el ícono de la extensión.
2. ***EL usuario*** da click en el botón 'Iniciar sesión'.
3. ***La extensión*** abre en una nueva pestaña del navegador la página principal de la extensión.
4. ***El usuario*** llena incorrectamente al menos un campo del formulario.

5. ***EL usuario*** da click en el botón 'Iniciar sesión'.
6. ***La extensión*** despliega un popup en donde informa cuál campo está llenado de forma incorrecta.
7. ***El usuario*** da click en el botón 'Aceptar'.
8. ***La extensión*** retorna al formulario sin realizar ninguna otra acción.

... Fin de la Trayectoria Alternativa 2 ...

... Trayectoria Alternativa 3 ...

1. ***El usuario*** da click en el ícono de la extensión.
2. ***EL usuario*** da click en el botón 'Iniciar sesión'.
3. ***La extensión*** abre en una nueva pestaña del navegador la página principal de la extensión.
4. ***El usuario*** da click en el botón 'Iniciar sesión'.
5. ***La extensión*** abre la página de inicio de sesión.
6. ***El usuario*** llena incorrectamente al menos un campo del formulario.
7. ***EL usuario*** da click en el botón 'Iniciar sesión'.
8. ***La extensión*** despliega un popup en donde informa cuál campo está llenado de forma incorrecta.
9. ***El usuario*** da click en el botón 'Aceptar'.
10. ***La extensión*** retorna al formulario sin realizar ninguna otra acción.

... Fin de la Trayectoria Alternativa 3 ...

Caso de uso: CI_CU5. Cerrar sesión en la extensión.	
Concepto	Descripción
Actor	Usuario
Propósito	Este caso de uso permite al usuario cerrar su sesión en el ordenador que se encuentre en ese momento, siempre y cuando haya iniciado sesión anteriormente. El objetivo es que se elimine su certificado de la extensión.
Entradas	-
Salidas	Sesión cerrada, en este momento la extensión se encuentra sin usuario logueado.
Pre-condiciones	CI_CU4.
Post-condiciones	La extensión debe mantenerse habilitada para su correcto funcionamiento.
Reglas del negocio	CI_RN1. CI_RN4. CI_RN5.
Errores	No se puede cerrar sesión.

Cuadro 4.5: Descripción CU: CI_CU5

... Trayectoria Principal ...

1. **El usuario** da click en el ícono de la extensión.
2. **El usuario** da click en el botón 'Cerrar sesión'.
3. **La extensión** cierra la sesión del usuario.

... Fin de la Trayectoria Principal ...

... Trayectoria Alternativa 1 ...

1. **El usuario** da click en el ícono de la extensión.
2. **El usuario** da click en el botón 'Cerrar sesión'.
3. **La extensión** no puede cerrar la sesión y lo informa al usuario por medio de un popup.

... Fin de la Trayectoria Alternativa 1 ...

Caso de uso: CI_CU6. Registrarse en servidor autentificador.	
Concepto	Descripción
Actor	Usuario
Propósito	Este caso de uso permitirá al usuario poder registrarse en el servidor autentificador si es que no tiene una cuenta con la cual poder iniciar sesión.
Entradas	email y contraseña ingresados por el usuario en la extensión.
Salidas	Estatus de operación.
Pre-condiciones	-
Post-condiciones	-
Reglas del negocio	CI_RN5. CI_RN7. CI_RN8. CI_RN9. CI_RN10. CI_RN11. CI_RN12. CI_RN13.
Errores	Error en el nombre de usuario. Error en el email. Error en la contraseña. No se pudo registrar al usuario.

Cuadro 4.6: Descripción CU: CI_CU6

... Trayectoria Principal ...

1. **El usuario** da click en el ícono de la extensión.
2. **La extensión** despliega un popup.
3. **El usuario** da click en el botón 'Iniciar Sesión'.
4. **La extensión** abre una página en otra pestaña del navegador.
5. **El usuario** da click en el botón 'Registrarse'.
6. **La extensión** muestra la página para registrarse.
7. **El usuario** llena los datos del formulario correctamente, los cuales son: 'Email', 'Contraseña' y 'Repetir contraseña'.

8. ***El usuario*** da click en el botón 'Crear Usuario'.
9. ***La extensión*** despliega un mensaje de confirmación.
10. ***El usuario*** da click en el botón '!Si, continuar!'.
11. ***La extensión*** despliega el estado de la operación.
12. ***El usuario*** da click en el botón 'Iniciar sesión'.
13. ***La extensión*** redirige a la página de inicio de sesión.

... Fin de la Trayectoria Principal ...

... Trayectoria Alternativa 1 ...

1. ***El usuario*** da click en el ícono de la extensión.
2. ***La extensión*** despliega un popup.
3. ***El usuario*** da click en el botón 'Iniciar Sesión'.
4. ***La extensión*** abre una página en otra pestaña del navegador.
5. ***El usuario*** da click en el botón 'Registrarse'.
6. ***La extensión*** muestra la página para registrarse.
7. ***El usuario*** llena al menos un campo del formulario incorrectamente.
8. ***El usuario*** da click en el botón 'Crear Usuario'.
9. ***La extensión*** despliega un popup en donde informa cuál campo está llenado de forma incorrecta.
10. ***El usuario*** da click en el botón 'Aceptar'.
11. ***La extensión*** retorna al formulario sin realizar ninguna otra acción.

... Fin de la Trayectoria Alternativa 1 ...

... Trayectoria Alternativa 2 ...

1. **El usuario** da click en el ícono de la extensión.
2. **La extensión** despliega un popup.
3. **El usuario** da click en el botón 'Iniciar Sesión'.
4. **La extensión** abre una página en otra pestaña del navegador.
5. **El usuario** da click en el botón 'Registrarse'.
6. **La extensión** muestra la página para registrarse.
7. **El usuario** llena los datos del formulario correctamente, los cuales son: 'Nombre de usuario', 'Email', 'Contraseña' y 'Repetir contraseña'.
8. **El usuario** da click en el botón 'Crear Usuario'.
9. **La extensión** despliega un popup de confirmación.
10. **El usuario** da click en el botón 'Cancelar'.
11. **La extensión** retorna al formulario sin realizar ninguna otra acción.

... Fin de la Trayectoria Alternativa 2 ...

Caso de uso: CI_CU7. Revocar certificado.	
Concepto	Descripción
Actor	Usuario
Propósito	El usuario tendrá la opción de actualizar su certificado generando uno nuevo, con la finalidad de que el usuario tenga un mejor control de sus credenciales.
Entradas	Certificado autentificador.
Salidas	Certificado autentificador.
Pre-condiciones	CI_CU4, CI_CU6.
Post-condiciones	CI_CU4
Reglas del negocio	CLRN4. CLRN5. CLRN6.
Errores	No se puede almacenar el certificado autentificador.

Cuadro 4.7: Descripción CU: CI_CU7

... Trayectoria Principal ...

1. ***La extensión*** ha creado el certificado autentificador.
2. ***La extensión*** actualiza las credenciales de su nuevo certificado.

... Fin de la Trayectoria Principal ...

... Trayectoria Alternativa 1 ...

1. ***La extensión*** no ha creado el certificado autentificador.
2. ***La extensión*** no guarda el certificado autentificador en storage. 4.18

... Fin de la Trayectoria Alternativa 1 ...

... Trayectoria Alternativa 2 ...

1. ***La extensión*** ha creado el certificado autentificador.
2. ***La extensión*** no puede guardar el certificado autentificador en storage.
3. ***La extensión*** muestra al usuario el siguiente mensaje "No se pudo guardar el certificado en el Storage" en la figura 4.18

... Fin de la Trayectoria Alternativa 2 ...

... Trayectoria Alternativa 3 ...

1. *La extensión* ha creado el certificado autentificador.
2. *La extensión* no puede eliminar el certificado anterior.
3. *La extensión* muestra al usuario el siguiente mensaje "No se pudo guardar el certificado en el Storage" en la figura 4.18

... Fin de la Trayectoria Alternativa 2 ...

4.1.2. Diagrama de flujo.

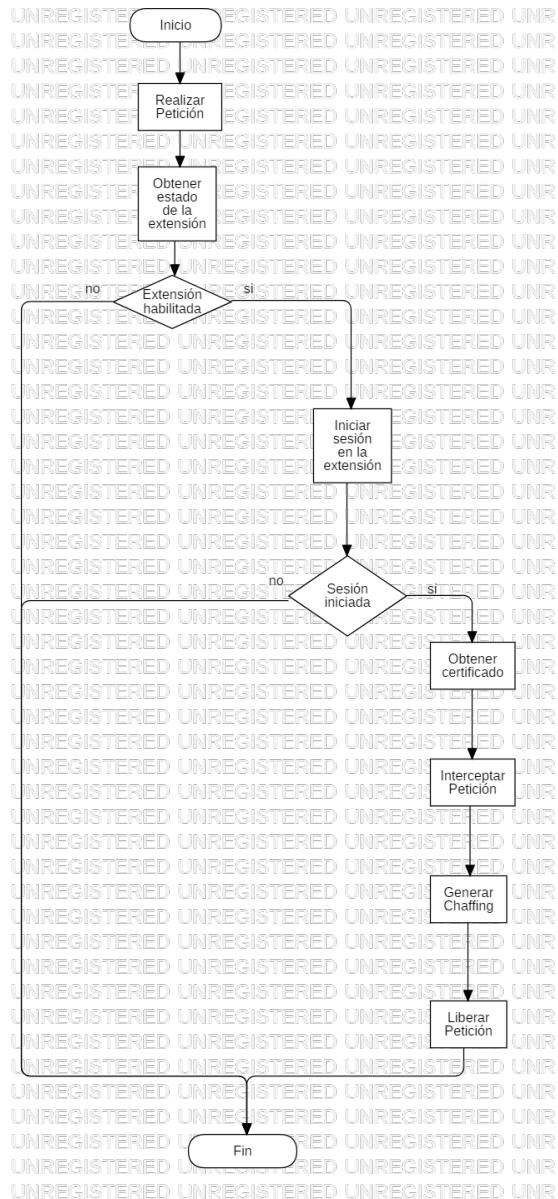


Figura 4.2: Diagrama de flujo del Componente I.

4.1.2.1. Descripción diagrama de flujo.

Para el caso de este diagrama se inicia con una petición realizada por el navegador web para luego analizar si la extensión se encuentra activada, en

caso de que no se realiza ninguna acción, pero si sí se encuentra, se analizará si ya se inició sesión, en caso de que no ya no se realiza ninguna acción pero en caso afirmativo se obtiene el certificado guardado en storage para luego interceptar el patrón y realizar el proceso de chaffing para finalmente liberar la petición modificada.

4.1.3. Diagrama de flujo de datos.

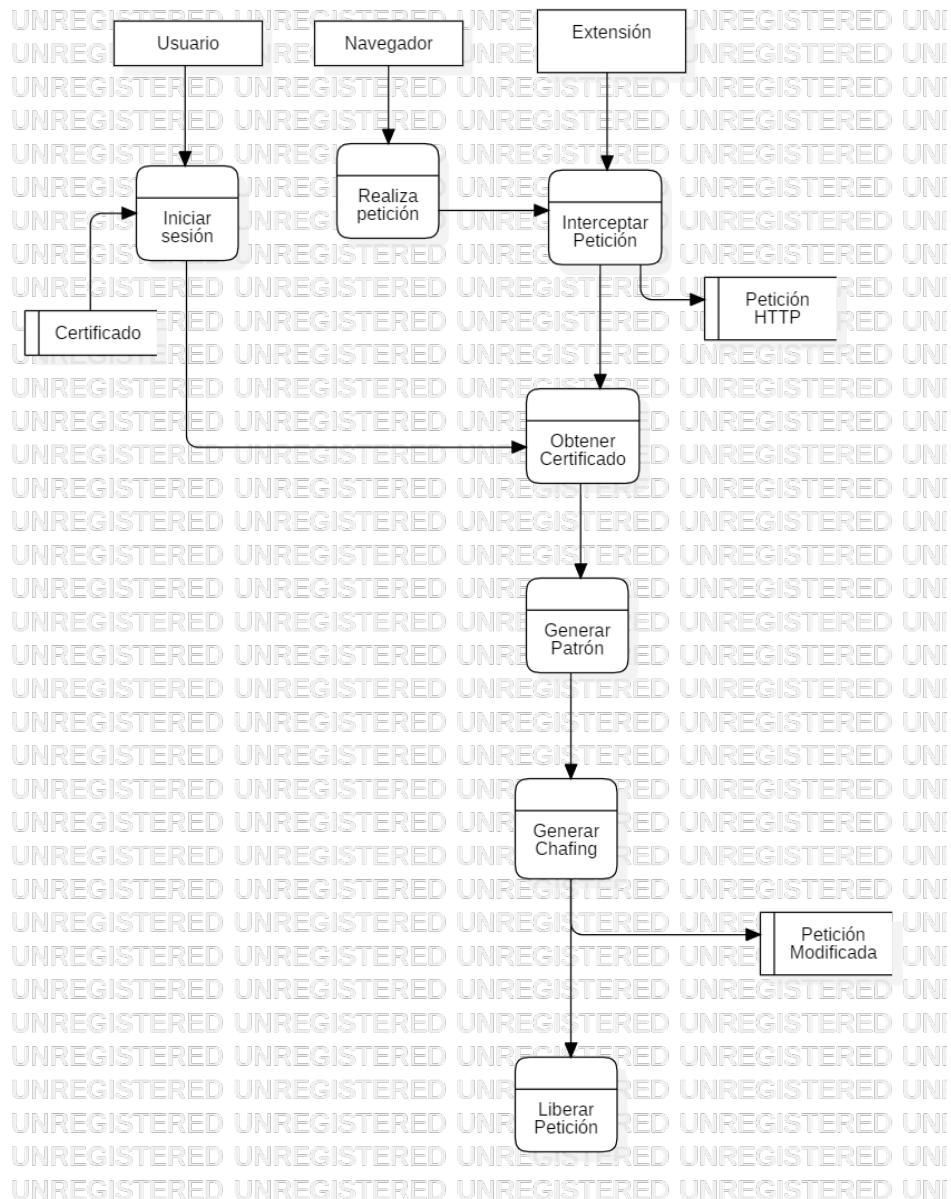


Figura 4.3: Diagrama de flujo de datos del Componente I.

4.1.3.1. Descripción diagrama de flujo de datos.

En este caso contamos con dos entidades externas, el usuario por una parte debe iniciar sesión para que se pueda generar un **Certificado** el cual

se utilizará para generar el chaffing y por otro lado el Navegador web que intercepta una **petición HTTP** que de igual manera se utilizará para generar el chaffing. Como resultado de la mezcla de estos dos se genera una **petición modificada** la cuál mas tarde se liberará para que pueda viajar hacia el servidor.

4.1.4. Diagrama de clases.

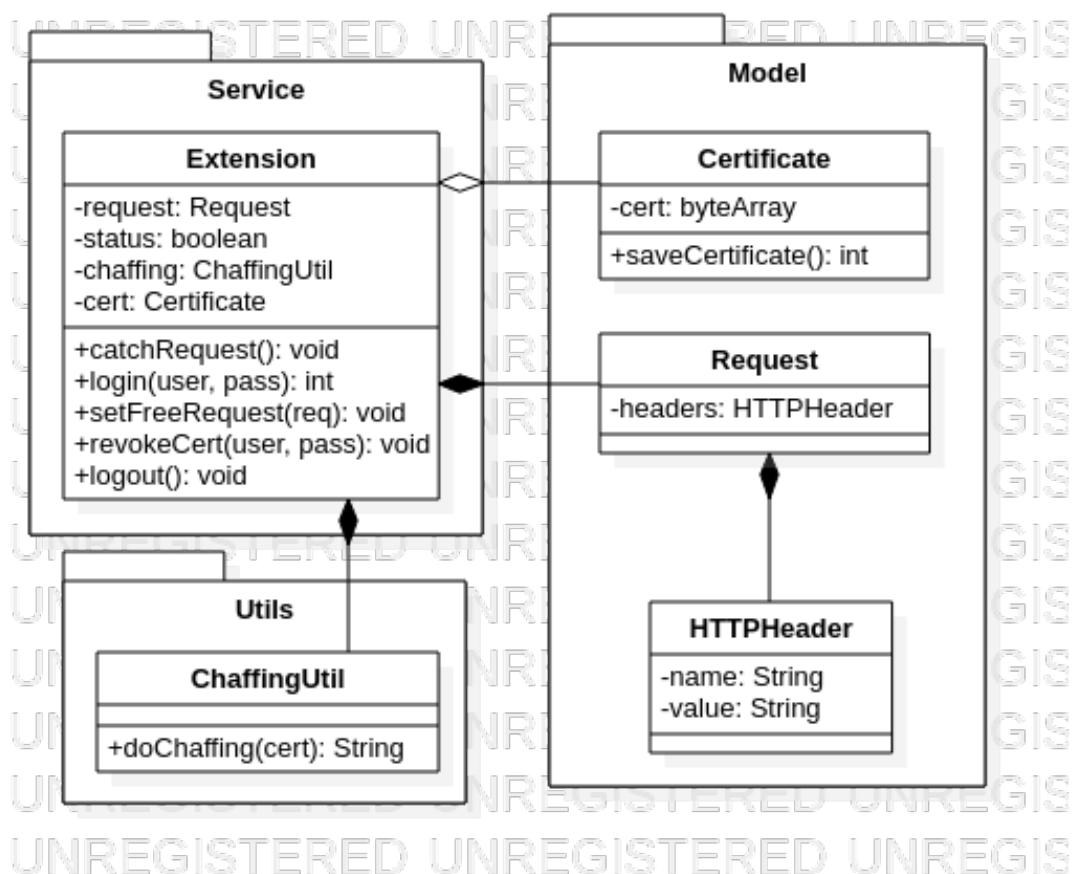


Figura 4.4: Diagrama de clases de Componente I.

4.1.4.1. Descripción de diagrama de clases

Clase: *Extension*

Atributos

1. **request** : Variable que almacena la petición HTTP.
 - Tipo de dato: **Request**.
2. **status** : Variable para saber si la extensión está activada o no.
 - Tipo de dato: **boolean**.
3. **chaffing** : instancia para ejecutar el proceso de Chaffing.
 - Tipo de dato: **ChaffingUtil**.
4. **cert** : Variable para almacenar el certificado del usuario.
 - Tipo de dato: **Certificate**.

Métodos

1. **catchRequest()**: Este método permite interceptar las peticiones del usuario.
 - Tipo de dato de retorno: **void**
2. **login(String user, String pass)**: Este método permite iniciar sesión para obtener el certificado del usuario.
 - Tipo de dato de retorno: **int**
3. **logout(String pass, String email)**: Este método permite cerrar sesión en la extensión y eliminar el certificado almacenado.
 - Tipo de dato de retorno: **void**
4. **revokeCert(String user, String pass)**: Este método permite revocar el certificado del usuario.
 - Tipo de dato de retorno: **void**
5. **setFreeRequest(Request req)**: Este método permite liberar la petición nueva con chaffing.
 - Tipo de dato de retorno: **void**

Clase: *ChaffingUtil*

Métodos

1. **doChaffing(Certificate certificate)**: Este método permite realizar la etapa de chaffing.

- Tipo de dato de retorno: **String**

Clase: *HTTPHeader*

Atributos

1. **name**: variable para guardar el nombre del header el protocolo.
 - Tipo de dato de retorno: **String**
2. **value**: variable para guardar el contenido del header el protocolo.
 - Tipo de dato de retorno: **String**

Clase: *HTTPRequest*

Atributos

1. **headers**: variable para guardar los header de una petición HTTP.
 - Tipo de dato de retorno: **HTTPHeader**

Clase: *Certificate*

Atributos

1. **cert**: variable para guardar el certificado del usuario.
 - Tipo de dato de retorno: **ByteArray**

Métodos

1. **saveCertificate()**: variable para guardar el certificado del usuario en el Storage.
 - Tipo de dato de retorno: **ByteArray**

4.1.5. Diagramas de secuencia.

4.1.5.1. Diagrama de secuencia: Realizar petición.

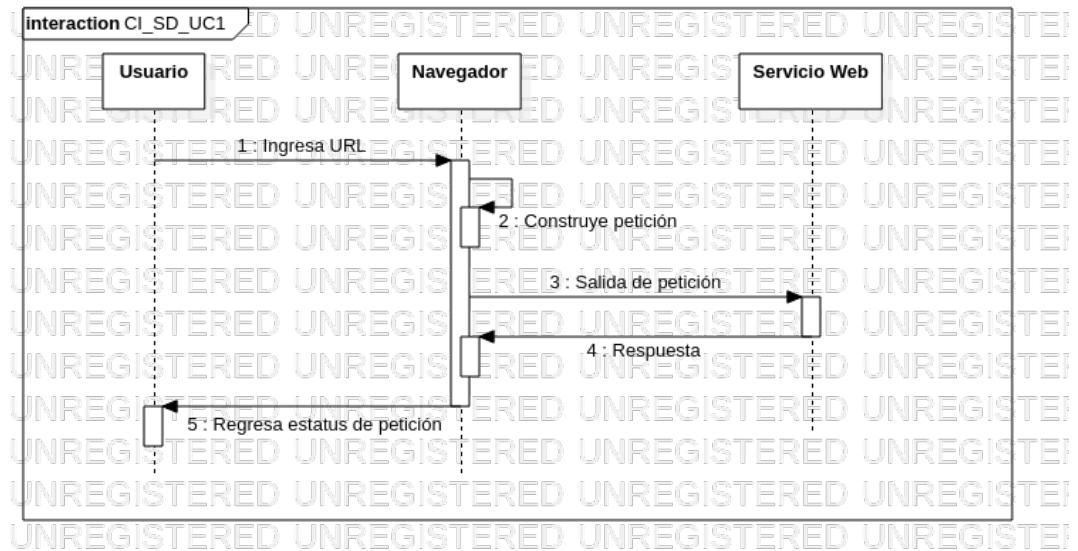


Figura 4.5: Diagrama de secuencia del CI_CU1. Realizar petición.

Descripción: Para realizar una petición, es necesario seguir la siguiente secuencia. Primeramente, el usuario ingresa un URL al navegador, para que este construya la petición que mandará. Una vez que ha construido la petición realiza la petición, es decir sale a red la petición.

4.1.5.2. Diagrama de secuencia: Habilitar extensión.

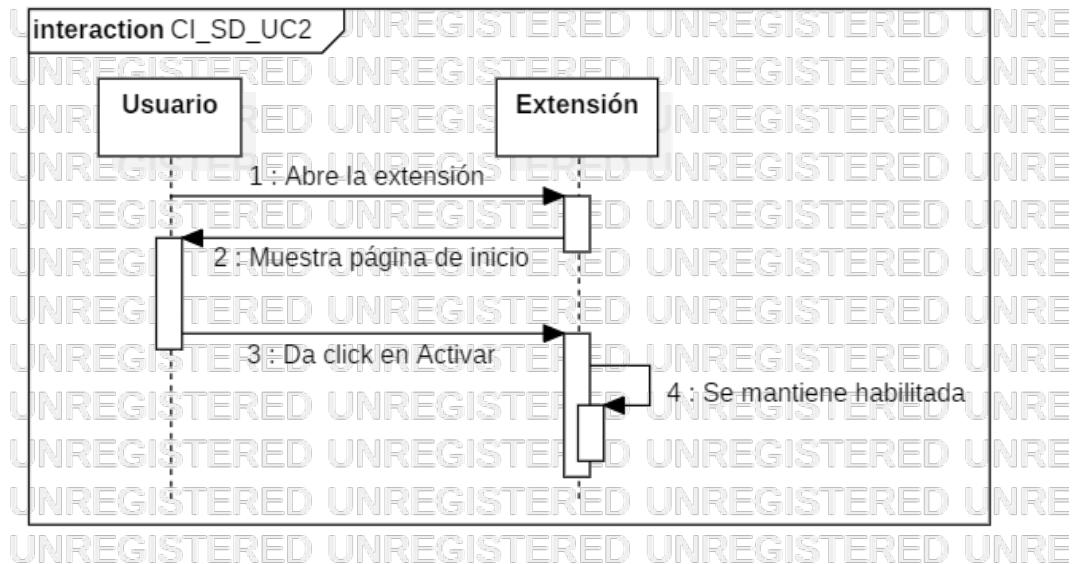


Figura 4.6: Diagrama de secuencia del CI_CU2. Habilitar extensión.

Descripción: Para habilitar la extensión, el usuario necesita primero abrir la extensión. Una vez abierta, debe de dar click en el botón 'Activar' para que la extensión ejecute la orden y se mantenga habilitada.

4.1.5.3. Diagrama de secuencia: Deshabilitar extensión.

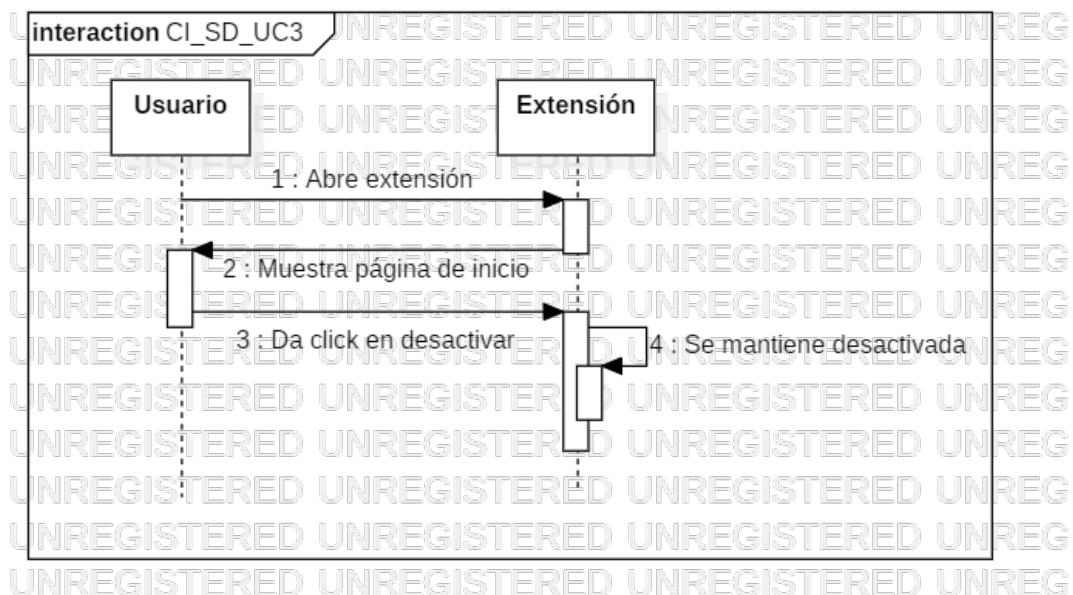


Figura 4.7: Diagrama de secuencia del CI_CU3. Deshabilitar extensión.

Descripción: Para deshabilitar la extensión, el usuario necesita primero abrir la extensión. Una vez abierta la extensión, debe de dar click en el botón 'Desactivar', para que la extensión ejecute la orden y se mantenga deshabilitada.

4.1.5.4. Diagrama de secuencia: Inicio de sesión

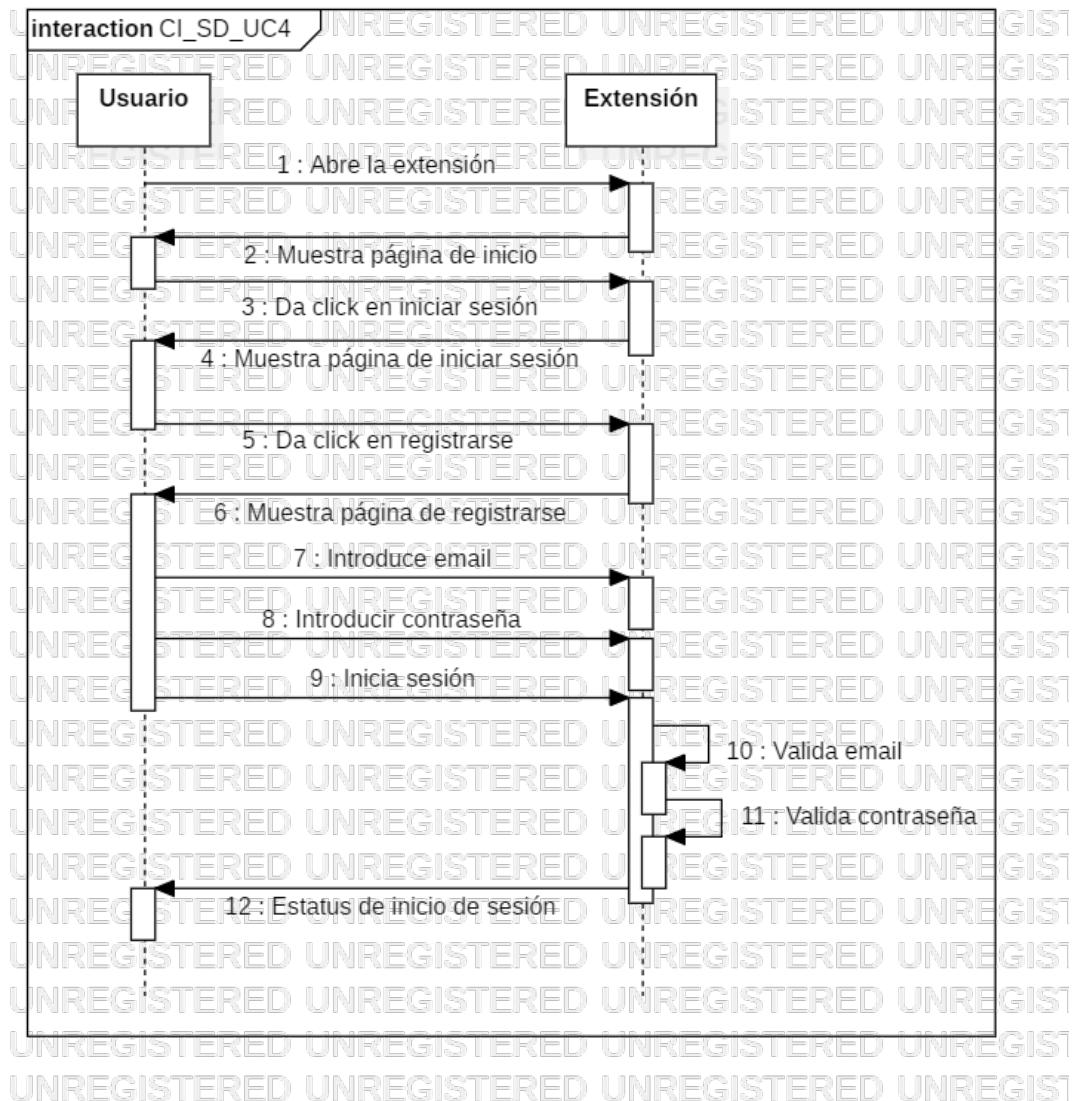


Figura 4.8: Diagrama de secuencia del CI_CU4. Inicio de sesión en la extensión.

Descripción: Como primeros dos pasos de la secuencia para iniciar sesión en la extensión, el usuario ingresa su nombre de usuario y su contraseña, para después iniciar sesión en la extensión. Una vez que el usuario ha dado la orden de iniciar sesión, la extensión valida el nombre de usuario y la contraseña, retornando un mensaje en caso de que alguno de estos dos valores esté mal,

si no es el caso, la extensión retornará un mensaje con el estatus del inicio de sesión.

4.1.5.5. Diagrama de secuencia: Cerrar sesión.

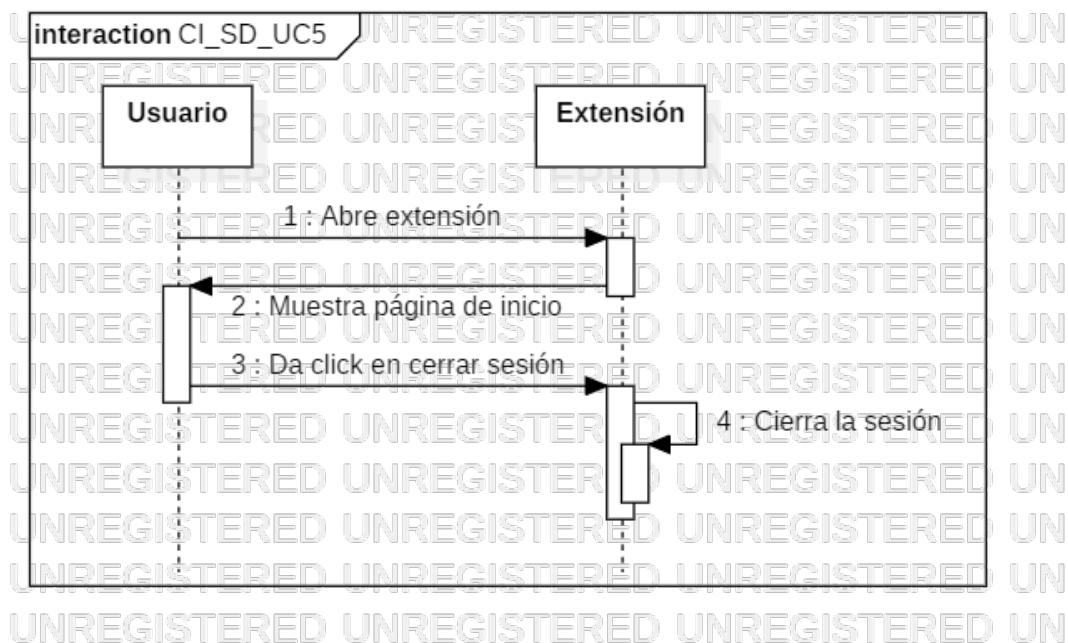


Figura 4.9: Diagrama de secuencia del CI_CU5. Cerrar sesión en la extensión.

Descripción: Como primer paso de este diagrama de secuencia se tiene que abrir la extensión para después dar click en el botón de 'Cerrar Sesión'.

4.1.5.6. Diagrama de secuencia: Registrarse en servidor autenticador.

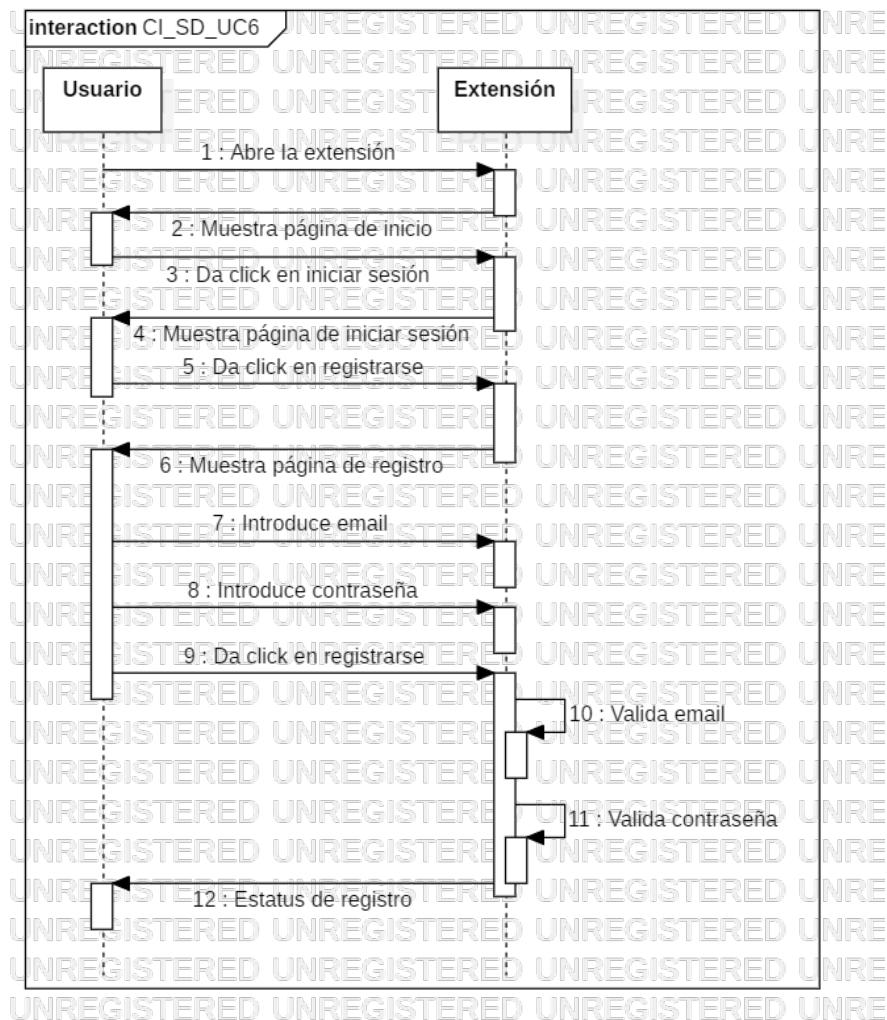


Figura 4.10: Diagrama de secuencia del CI_CU6. Registrarse en servidor autenticador.

Descripción: Como primeros pasos de este diagrama de secuencia, la extensión se tiene que abrir para después dar click en el botón de 'Iniciar sesión' para desplegar la página principal y, una vez ahí, dar click en el botón de 'Registrarse'. Una vez en la página de registro, el usuario, como siguiente paso, llena los datos requeridos por el formulario para después dar click en el

botón 'Crear Usuario'. La extensión validará los campos retornando un mensaje de error en caso de que alguno se llene mal, si no es el caso, retornará un aviso con el estatus del registro.

4.1.5.7. Diagrama de secuencia: Revocar certificado.

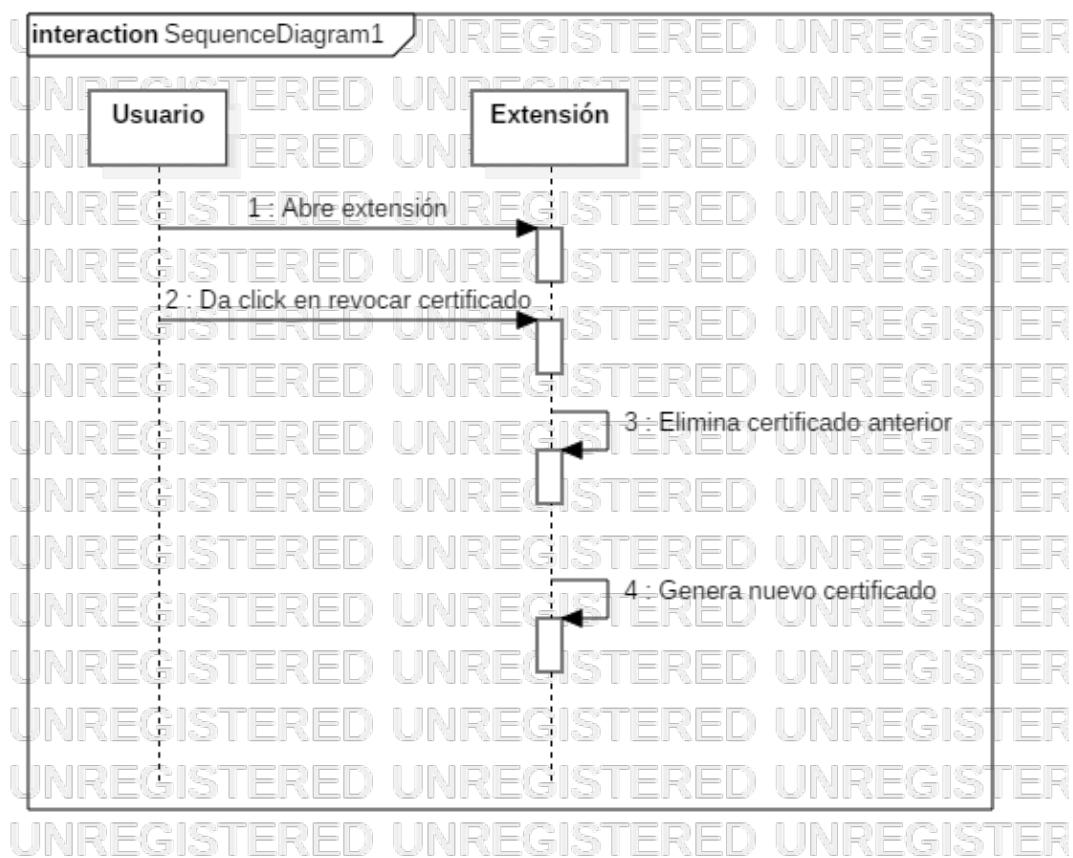


Figura 4.11: Diagrama de secuencia del CI_CU7. Revocar certificado.

Descripción:

4.1.6. Diagrama de actividades

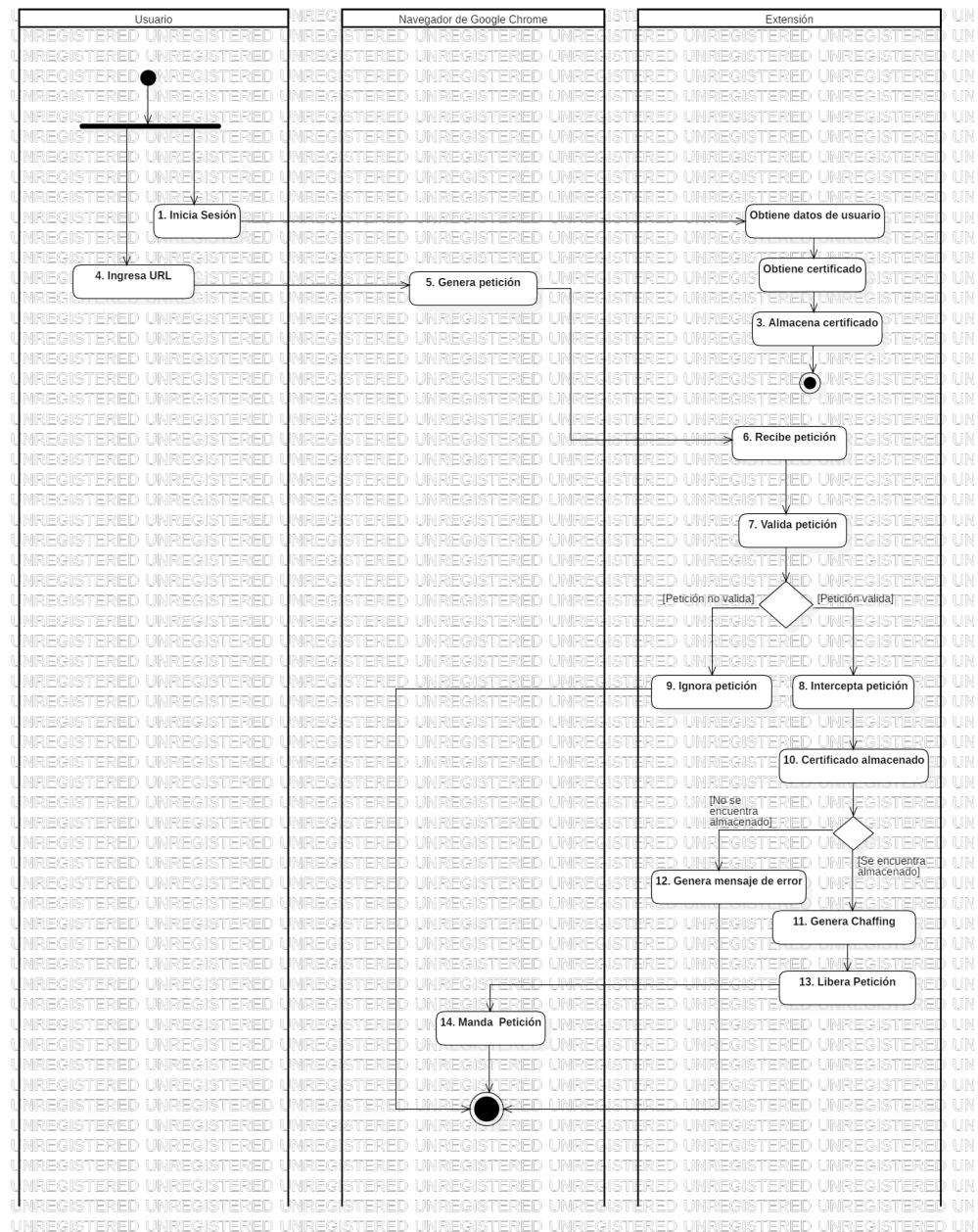


Figura 4.12: Diagrama de actividades del Prototipo 2.

4.1.6.1. Descripción del diagrama de actividades.

El componente cuenta con los siguientes pasos. Uno de los primeros pasos que debe hacer el usuario es iniciar sesión, esto con el fin de generar un certificado de acuerdo a los datos del usuario. Dicho certificado se almacena en el **Storage** de Google Chrome.

El otro camino a seguir por el usuario es realizar una búsqueda en el navegador, en la cual el navegador realiza la petición y la extensión recibe la misma. Modificándola siempre y cuando se encuentre un certificado guardado en el Storage. Si este proceso es válido, se genera el proceso de Chaffing y se libera la petición.

4.1.7. Interfaz de usuario.

4.1.7.1. Pantalla Inicial.

Esta pantalla es la primer vista que el usuario tiene el sistema. Aparece al darle click a la extensión en su ícono correspondiente.



Figura 4.13: Pantalla de interfaz de la extensión.

En ella, el usuario puede activar y desactivar la extensión para empezar a interceptar peticiones, ademas cuenta con un botón de inicio de sesión (*Iniciar Sesión*), el cual al darle click abrirá la pantalla 'Tab de la extensión' (Figura 4.14). La cual se muestra a continuación.

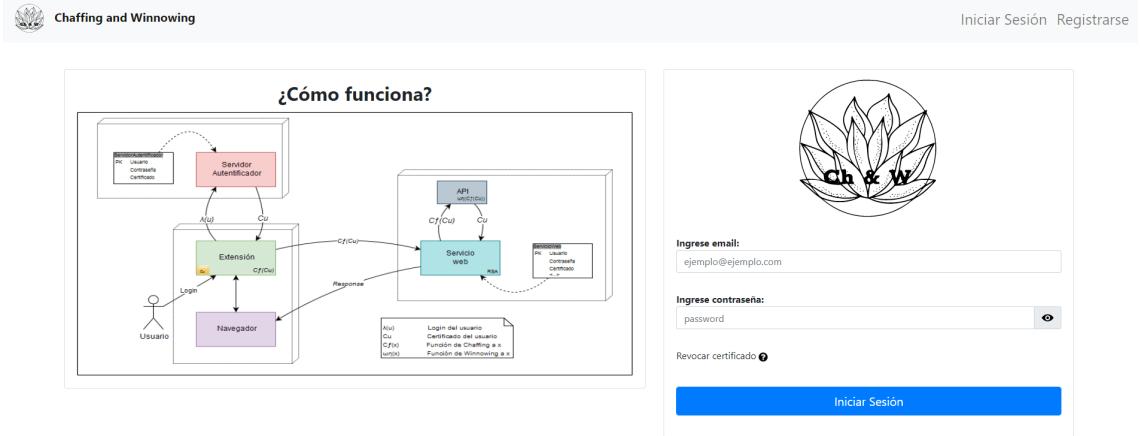


Figura 4.14: Pantalla inicial.

El navegador Google Chrome tiene ciertas restricciones con los certificados que no son firmados por una autoridad certificadora de confianza, y debido a que nuestros certificados son autofirmados. nos aparecerá un mensaje de alerta por parte del navegador, para evitar que esto sea un problema, en la extensión vamos a seleccionar la opción de *Aviso*, donde nos abrirá una pestaña nueva de un aviso de seguridad, vamos a escoger la opción de *opciones avanzadas*, y se desplegará información adicional con la opción de *Proceder a la ip x.x.x.x*, como se muestra en la siguiente imagen:



Your connection is not private

Attackers might be trying to steal your information from **172.16.140.79** (for example, passwords, messages, or credit cards). [Learn more](#)

NET::ERR_CERT_AUTHORITY_INVALID

Help improve Safe Browsing by sending some [system information](#) and [page content](#) to Google.
[Privacy policy](#)

[Hide advanced](#)

[Back to safety](#)

This server could not prove that it is **172.16.140.79**; its security certificate is not trusted by your computer's operating system. This may be caused by a misconfiguration or an attacker intercepting your connection.

[Proceed to 172.16.140.79 \(unsafe\)](#)

Figura 4.15: Pantalla de aviso.

4.1.7.2. Tab de la extensión.

Esta pantalla es la interfaz que se le brinda al usuario para que inicie sesión y obtenga su certificado autenticador (Figura 4.16). En ella se aprecian únicamente dos campos para introducir texto ('Ingrrese email' e 'Ingrrese contraseña'), y un botón 'Iniciar Sesión'.

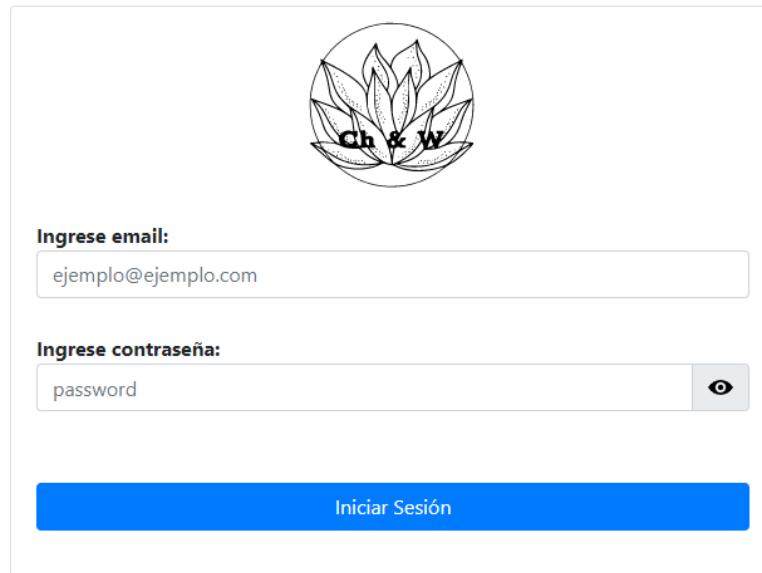


Figura 4.16: Tab de la extensión. Permite inicio de sesión

Durante el inicio de sesión el usuario puede visualizar mensajes de éxito o error. La Figura 4.17 se le muestra al usuario tras haber obtenido el certificado desde la autoridad y guardado con éxito en el Storage de Google Chrome. La Figura 4.18 notifica al usuario que existió un error al obtener el certificado de la autoridad certificadora.

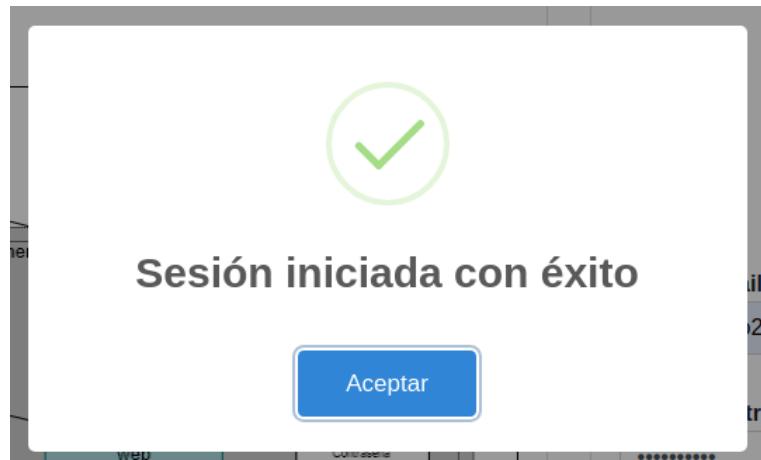


Figura 4.17: Mensaje de éxito tras obtener el certificado de la autoridad y guardarlo en el storage de Google Chrome.

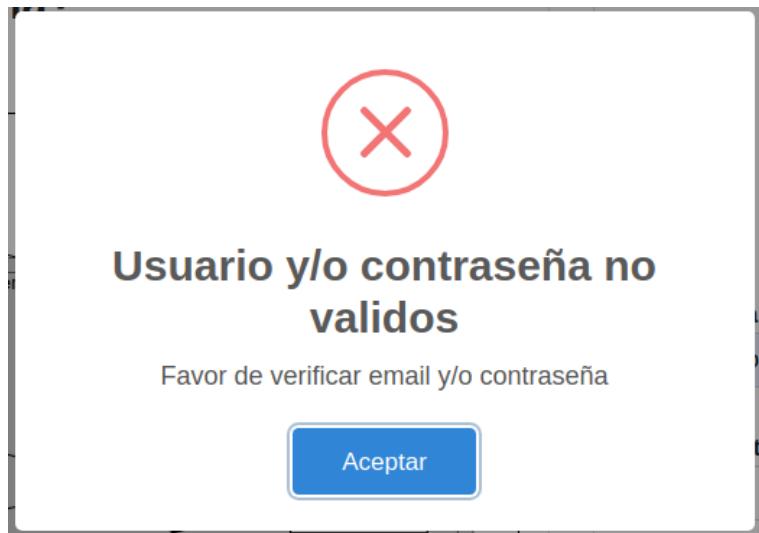


Figura 4.18: Mensaje de error al tratar de obtener certificado de la autoridad (Usuario y/o contraseña incorrectos).

Por otra parte, se le brinda al usuario una interfaz donde pueda registrarse, para ello basta con escoger la opción de *Registrarse* en la parte superior de la interfaz, al acceder a esta interfaz, aparecerán cuatro campos los cuales el usuario puede llenar para crear una cuenta y registrarse en la extensión. Estos campos son correo electrónico y contraseña.



Ingrese email:
ejemplo@ejemplo.com

Ingrese contraseña: ⓘ
password

Repetir contraseña:
password

Crear Usuario

Figura 4.19: Tab de la extensión. Permite registrarse

Al llenar los campos y enviar la petición a la autoridad certificadora de registrarse, se le notificará al usuario si pudo generarse el registro del usuario con éxito u ocurrió un error tras crear el registro. Cabe resaltar que la autoridad creará un certificado para dicho usuario, el cual el usuario deberá obtener posteriormente con un inicio de sesión, éste proceso se explicará en el componente 2.

Como mensaje de éxito se muestra la figura 4.20 la cual significa que el usuario fue creado con éxito y se le generó un certificado para dicho usuario.

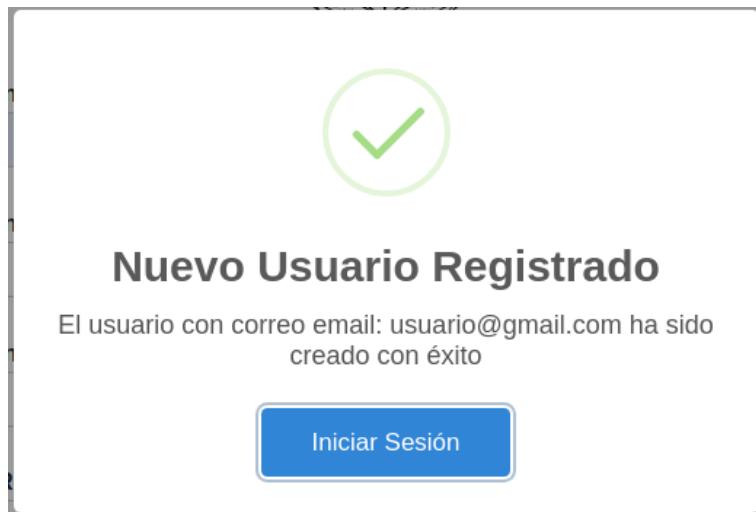


Figura 4.20: Mensaje de éxito tras obtener el certificado de la autoridad y guardarlo en el storage de Google Chrome.

Como mensajes de errores se muestran distintos tipos de mensajes para diferentes situaciones, las cuales se explican a continuación.

La figura 4.21 notifica al usuario que el email introducido no es valido, es decir, no cuenta con una estructura valida de un correo electrónico (email).

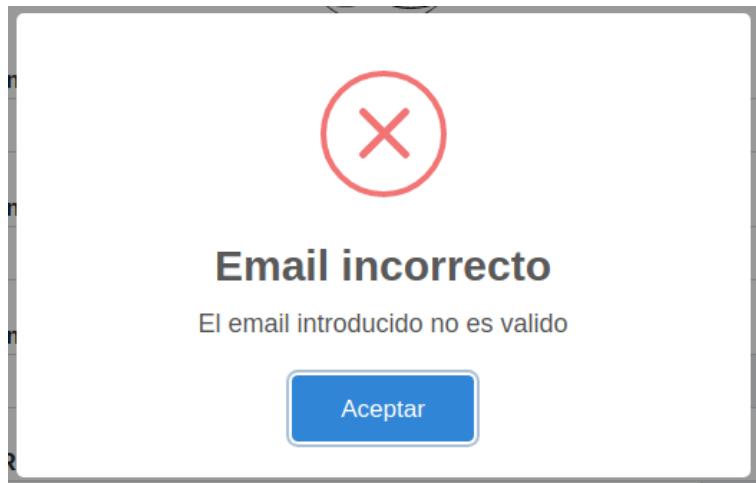


Figura 4.21: Mensaje de error tras detectar que no se cumplen los requisitos para el correo ingresado.

La figura 4.22 muestra un error tras no cumplir los requisitos con la

contraseña introducida.

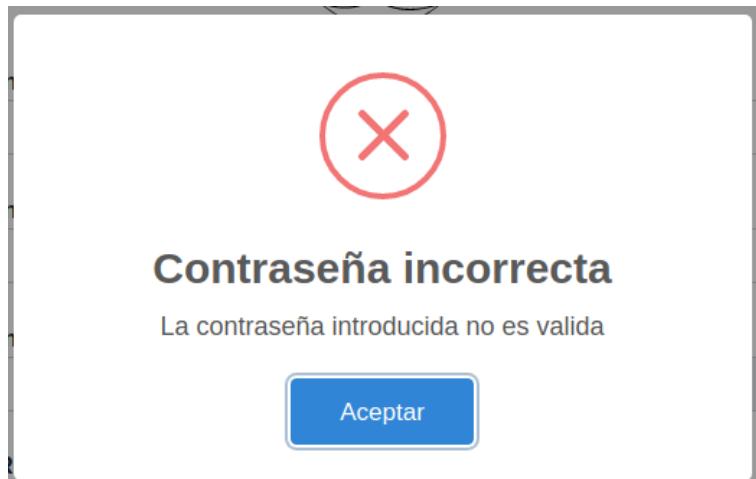


Figura 4.22: Mensaje de error tras detectar que no se cumplen los requisitos para la contraseña ingresada.

La figura 4.23 muestra un error tras no coincidir las contraseñas introducidas.

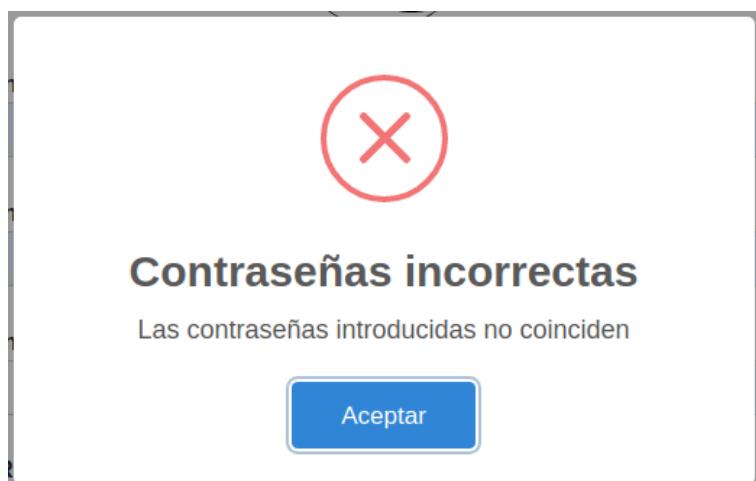


Figura 4.23: Mensaje de error tras detectar que no coinciden el par de contraseñas ingresadas.

Y por ultimo se muestra en la figura 4.24 el error que se obtiene al tratar de registrarse con un usuario que ya tiene una cuenta en la extensión.

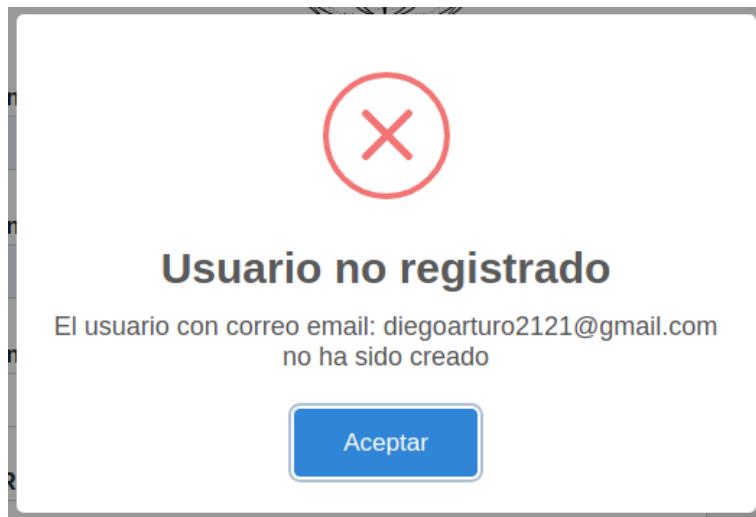


Figura 4.24: Mensaje de error tras detectar que el usuario ya se encuentra registrado.

El usuario tendrá también la posibilidad de revocar su certificado, permitiéndole así generar uno por la autoridad certificadora. La figura 4.25 muestra la interfaz que se le proporciona al usuario para realizar la revocación de su certificado.

A screenshot of a login form for certificate revocation. At the top center is a circular logo featuring a stylized plant or flower with the letters "Ch & W" in the center. Below the logo is a text field labeled "Ingrese email:" containing the placeholder "ejemplo@ejemplo.com". Below that is a password field labeled "Ingrese contraseña:" containing the placeholder "password" and a small eye icon to toggle visibility. At the bottom is a large blue rectangular button with the text "Revocar Certificado" in white.

Figura 4.25: Interfaz para revocar certificado.

La figura 4.26 muestra un mensaje de éxito, el cual se le da a conocer al usuario que su certificado fue revocado exitosamente.

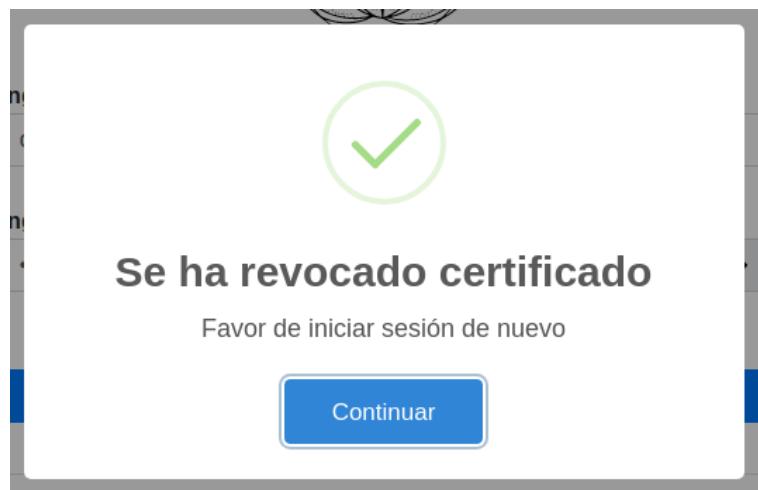


Figura 4.26: Mensaje de éxito al revocar el certificado de un usuario.

Si el usuario ingresa sus credenciales (correo electrónico y contraseña) incorrectas, la interfaz despliega la información de error de la figura 4.27.

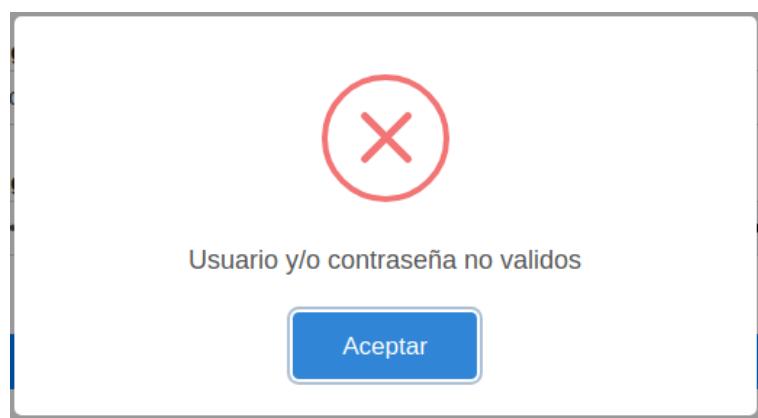


Figura 4.27: Mensaje de error al tratar de revocar el certificado de un usuario con credenciales incorrectas.

El usuario visualizará y tendrá interacción con este componente para obtener el certificado y llevar a cabo el proceso de Chaffing. Este proceso se lleva a cabo en segundo plano de la extensión (background) el cual cacha la

petición e inyecta el resultado del Chaffing en el header de la petición hecha previamente. Una vez injectado el código, la extensión libera la petición y ésta viaja en red al servidor de prueba.

4.1.8. Requisitos de diseño.

En este apartado, se especificarán los requisitos de diseño para que el prototipo opere de forma correcta, de igualmanera se tiene por entendido que son necesarios los Requisitos del diseño del primer prototipo.

4.1.8.1. Requisitos de ejecución de la extensión

Para poder ejecutar el prototipo dos es necesario contar con todo lo requerido para el prototipo uno y agregarle la interacción con **jQuery 3.3.1** y **Bootstrap** que son dos frameworks que corren sobre JavaScript y que nos permiten interactuar un poco mejor con el usuario haciendo la interfaz de la extensión más amigable y entendible, estos ya se encuentran insertados en la extensión por lo que no es necesario que el usuario realice acción alguna. Otros de los requisitos para su ejecución es contar con un **usuario y contraseña** válidos ya que serán de suma importancia para el correcto funcionamiento del prototipo dos. Para deshabilitar la extensión, el usuario necesita primero abrir la extensión. Una vez abierta la extensión, debe de dar click en el botón "deshabilitar", para que la extensión ejecute la orden y se mantenga deshabilitada.

4.1.8.2. Requisitos para el envío del código autentificador

Para este prototipo se considera necesaria los siguientes requisitos de diseño:

- **Código Autentificador** Archivo indispensable en este prototipo debido a que sin este es imposible completar el proceso.
- **Iniciar Sesión** Este botón generará un certificado de prueba y se almacenará en el Storage del navegador Google Chrome. Esto con el fin, que al momento de interceptar la petición, la extensión de encargará de inyectar el certificado almacenado modificado con el método *Chaffing* en el encabezado de protocolo HTTP.

4.1.9. Algoritmos.

Los algoritmos necesarios para hacer el *chaffing* son expuestos a continuación:

```
Data: lenCert, lenAccept, Cert  
Result: patternChaffing  
1 lenPattern  $\leftarrow$  lenCert + lenAccept;  
2 Pattern[lenPattern * 8]  $\leftarrow$  0;  
3 unosCert  $\leftarrow$  getOnes(Cert);  
4 for i = 1 to unosCert do  
5   repeat  
6     | random  $\leftarrow$  secure_random(0, lenPattern);  
7     | until Pattern[random]! = 0;  
8     | Pattern[random]  $\leftarrow$  '1';  
9 end
```

Algoritmo 1: getPattern: Generación de patrón de chaffing

```

Data: Cert, Accept
Result: Chaffing
1 lenCert  $\leftarrow$  length(Cert);
2 lenAccept  $\leftarrow$  length(Accept);
3 Pattern  $\leftarrow$  getPattern(lenCert, lenAccept, Cert);
4 Chaffing  $\leftarrow$  "";
5 countCert  $\leftarrow$  0;
6 countAccept  $\leftarrow$  0;
7 certBits  $\leftarrow$  getBits(Cert);
8 acceptBits  $\leftarrow$  getBits(Accept);
9 foreach i in Pattern do
10   if i ==' 1' then
11     Chaffing  $\leftarrow$  Chaffing + certBits[countCert];
12     countCert  $\leftarrow$  countCert + 1;
13   else
14     Chaffing  $\leftarrow$  Chaffing + acceptBits[countAccept];
15     countAccept  $\leftarrow$  countAccept + 1;
16   end
17 end
18 Chaffing  $\leftarrow$  getBytes(Chaffing);

```

Algoritmo 2: getChaff: Generación de chaffing

Primero vamos a analizar el algoritmo del *patrón de chaffing* con un pequeño ejemplo (utilizaremos datos de variables cortos); supongamos que nuestro certificado es el siguiente:

$$C_k = MITZ057abZ251$$

y utilizaremos el campo *Accept* del header de la petición como *chaff*, lo cual tendrá lo siguiente:

$$P_{HTTP} = Mozilla5,5/Chrome8,1/Safari$$

Entonces, nuestro patrón de chaffing terminará teniendo un tamaño de la longitud de los caracteres de nuestro certificado más la longitud de los datos de la petición HTTP, esta variable se llamará *lenPattern*. A continuación vamos a utilizar un arreglo de banderas de ese tamaño multiplicado por 8, para que así podamos llenar de ceros y unos este arreglo y cada una de las posiciones nos represente en el chaffing un bit, este arreglo inicia con 0 todos sus valores, y que al final será nuestro patrón, esta variable será *Pattern*.

Este algoritmo nos permite generar posiciones aleatorias dentro del rango del tamaño de *lenPattern*, y tantas veces como se tengan caracteres en el certificado, se pondrá un 1 en dicha posición si es que hay un 0, en caso de que no se encuentre un 0 se repetirá el procedimiento obteniendo una posición aleatoria diferente, esto nos generará nuestro patrón para el Chaffing. Utilizaremos un contador para conseguir esto, aumentándolo cada vez que se obtenga un número aleatorio válido. En nuestro algoritmo, nuestra variable *random* contiene la posición aleatoria donde se validará si en esa posición se puede poner un 1 o no.

Este algoritmo nos permite llenar de unos el mismo número de veces que el número de unos contenga nuestro certificado en bits, lo que nos asegura que cada posición le corresponde a un byte ya sea del certificado o de la petición a enviar. Supongamos que al terminar este algoritmo, nuestro arreglo queda algo parecido a lo siguiente:



Figura 4.28: Arreglo del patrón de chaffing final

El segundo algoritmo realizará la etapa de Chaffing, utilizando el patrón generado en el algoritmo anterior, recorriendo cada posición de este arreglo, contaremos con dos contadores para saber que carácter se debe de poner a continuación en el proceso del algoritmo, *countCert* y *countAccept*, contador para el certificado y para el encabezado Accept respectivamente.

Cuando se detecte un 1 en el arreglo del patrón, vamos a proceder a colcoar el byte siguiente correspondiente al arreglo del certificado. En caso análogo, si se encuentra con un 0, entonces se colocará el siguiente carácter del encabezado Accept, se llevará el control de ambos mediante sus respectivos contadores.

Por último, es importante mencionar que se mandará el Chaffing en base64 para evitar problemas con los caractéres no válidos que se puedan generar. Seguido del Chaffing en base64, vamos a enviar el patrón con un cifrado híbrido, cifrando primero el patrón con AES, la llave que utilizaremos para realizar este cifrado la cifraremos con RSA utilizando la llave pública del servidor.

4.1.9.1. Complejidad computacional.

Haciendo un análisis de los algoritmos anteriormente mostrados, deducimos que la complejidad del algoritmo de *chaffing* es: $O(n)$ Donde n es la longitud de caracteres del certificado.

4.2. Componente II: Servidor Autentificador.

4.2.1. Diagrama de casos de uso.

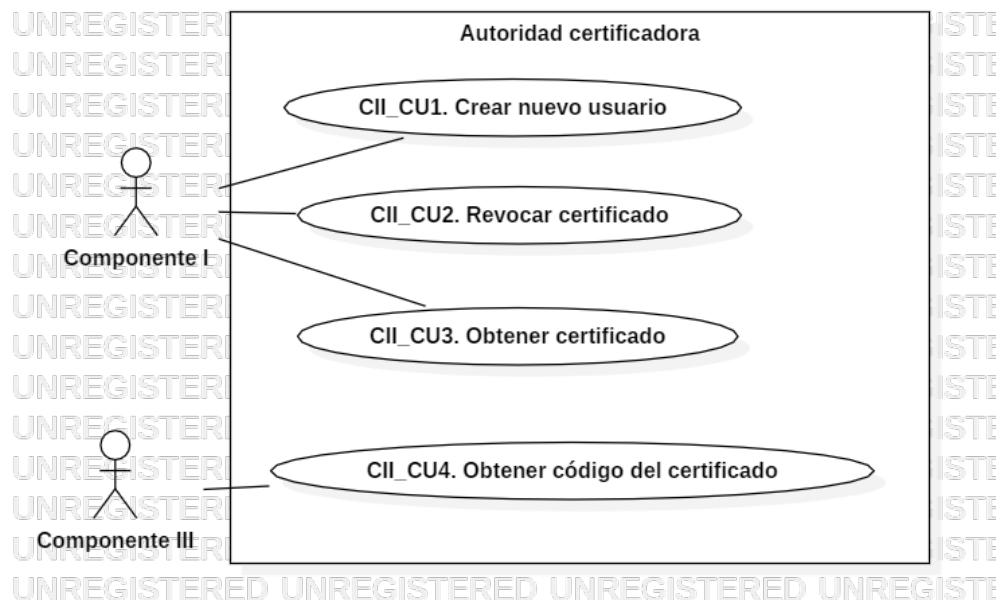


Figura 4.29: Diagrama de casos de uso del Componente II.

4.2.1.1. Descripción de casos de uso.

Caso de uso: CII_CU1. Crear nuevo usuario.	
Concepto	Descripción
Actor	Componente I. Extensión.
Propósito	Guardar un nuevo usuario en el Componente I para que éste pueda hacer uso de este método de autenticación.
Entradas	Correo electrónico y contraseña.
Salidas	Status de operación.
Pre-condiciones	-
Post-condiciones	Creación de un certificado.
Reglas del negocio	CII_RN1 CII_RN2
Errores	No se pudo registrar el usuario en la base de datos.

Cuadro 4.8: Descripción CU: CII_CU1

... Trayectoria Principal ...

1. **La extensión** solicita al servidor autenticador registrar un nuevo usuario enviando los datos necesarios.
2. **El servidor autenticador** recibe los parámetros que le envió la extensión.
3. **El servidor autenticador** guarda al usuario en la base de datos.
4. **El servidor autenticador** genera un certificado de acuerdo a los datos dados por el usuario.
5. **El servidor autenticador** comunica a la extensión que el usuario ha sido creado.
6. **La extensión** despliega un mensaje de confirmación al usuario.

... Fin de la Trayectoria Principal ...

... Trayectoria alternativa 1 ...

1. **La extensión** solicita al servidor autenticador a registrar un nuevo usuario enviando los datos necesarios.

2. *El servidor autentificador* no recibe correctamente los datos.

3. *La extensión* despliega un mensaje de error.

... Fin de Trayectoria alternativa 1 ...

... Trayectoria alternativa 2 ...

1. *La extensión* solicita al servidor autentificador a registrar un nuevo usuario enviando los datos necesarios.

2. *El servidor autentificador* recibe los parámetros que le envió la extensión.

3. *El servidor autentificador* no puede registrar el nuevo usuario en su base de datos.

4. *La extensión* despliega un mensaje de error.

... Fin de Trayectoria alternativa 2 ...

Caso de uso: CII_CU2. Revocar certificado.	
Concepto	Descripción
Actor	Componente I. Extensión.
Propósito	Revocar el certificado que se encuentra vinculado a cierto usuario, cambiándolo por uno nuevo.
Entradas	Usuario y contraseña.
Salidas	Certificado nuevo.
Pre-condiciones	Usuario previamente registrado en el servidor autenticador.
Post-condiciones	-
Reglas del negocio	CII_RN1 CII_RN2 CII_RN4 CII_RN6
Errores	El certificado no se pudo crear correctamente. No se pudo asignar el certificado al usuario.

Cuadro 4.9: Descripción CU: CII_CU2

... Trayectoria principal ...

1. **La extensión** solicita revocar certificado al usuario actual enviando los datos necesarios.
2. **El servidor autenticador** valida si se encuentra registrado el usuario.
3. **El servidor autenticador** genera un nuevo certificado.
4. **El servidor autenticador** elimina el antiguo certificado asignado a ese usuario.
5. **El servidor autenticador** asigna el nuevo certificado al usuario.
6. **El servidor autenticador** envía un mensaje a la extensión de que se revocó correctamente el certificado.

... Fin Trayectoria principal ...

... Trayectoria Alternativa 1 ...

1. **La extensión** solicita revocar certificado al usuario actual.

2. *El servidor autentificador* no puede encontrar el usuario que solicitó revocar el certificado.
3. *El servidor autentificador* envía un código de error a la extensión.
4. *La extensión* despliega un mensaje de error.

... Fin Trayectoria Alternativa 1 ...

... Trayectoria Alternativa 2 ...

1. *La extensión* solicita revocar el certificado al usuario actual.
2. *El servidor autentificador* valida si se encuentra registrado el usuario.
3. *El servidor autentificador* no puede generar un nuevo certificado.
4. *El servidor autentificador* envía un código de error a la extensión.
5. *La extensión* despliega un mensaje de error.

... Fin Trayectoria Alternativa 2 ...

... Trayectoria Alternativa 3 ...

1. *La extensión* solicita revocar el certificado al usuario actual.
2. *El servidor autentificador* valida si se encuentra registrado el usuario.
3. *El servidor autentificador* genera un nuevo certificado.
4. *El servidor autentificador* elimina el antiguo certificado asignado a ese usuario.
5. *El servidor autentificador* asigna el certificado al usuario.
6. *El servidor autentificador* no puede enviarle el certificado al usuario.
7. *El servidor autentificador* envía un código de error a la extensión.
8. *La extensión* despliega un mensaje de error.

... Fin Trayectoria Alternativa 3 ...

Caso de uso: CII_CU3. Obtener certificado.	
Concepto	Descripción
Actor	Componente I. Extensión.
Propósito	Retornar el certificado del usuario a la extensión desde donde éste inició sesión.
Entradas	Usuario y contraseña.
Salidas	Certificado asignado a dicho usuario.
Pre-condiciones	CII_CU3.
Post-condiciones	-
Reglas del negocio	CII_RN1 CII_RN2 CII_RN3
Errores	El servidor autentificador no puede retornar el certificado a la extensión.

Cuadro 4.10: Descripción CU: CII_CU3

... Trayectoria principal ...

1. *La extensión* solicita el certificado asignado al usuario con sesión iniciada actualmente.
2. *El servidor autentificador* valida al usuario registrado en su base de datos.
3. *El servidor autentificador* envía a la extensión el certificado asociado a este usuario.

... Fin Trayectoria principal ...

... Trayectoria Alternativa 1 ...

1. *La extensión* solicita el certificado asignado al usuario con sesión iniciada actualmente.
2. *El servidor autentificador* no encuentra el usuario registrado en su base de datos.
3. *El servidor autentificador* envía un código de error a la extensión.
4. *La extensión* despliega un mensaje de error.

... Fin Trayectoria Alternativa 1 ...

... Trayectoria Alternativa 2 ...

1. *La extensión* solicita el certificado asignado al usuario con sesión iniciada actualmente.
2. *El servidor autenticador* valida el usuario registrado en su base de datos.
3. *El servidor autenticador* no puede enviar el certificado a la extensión.
4. *El servidor autenticador* envía un código de error a la extensión.
5. *La extensión* despliega un mensaje de error.

... Fin Trayectoria Alternativa 1 ...

Caso de uso: CII_CU4. Obtener código del certificado.	
Concepto	Descripción
Actor	Componente III. API.
Propósito	Retornar un SHA256 del certificado de un usuario especificado.
Entradas	Código SHA256 del nombre del usuario.
Salidas	Código SHA256 del certificado del usuario.
Pre-condiciones	Usuario registrado en el servidor autentificador.
Post-condiciones	-
Reglas del negocio	CII_RN1 CII_RN2 CII_RN6
Errores	El servidor autentificador no puede responder.

Cuadro 4.11: Descripción CU: CII_CU4

... Trayectoria principal ...

1. *El servidor autentificador* recibe un SHA256 del nombre de usuario enviado por *Componente III. API*.
2. *El servidor autentificador* busca el certificado de dicho usuario.
3. *El servidor autentificador* retorna un código SHA256 del certificado del usuario.

... Fin Trayectoria principal ...

... Trayectoria alternativa 1 ...

1. *El servidor autentificador* recibe un SHA256 del nombre de usuario enviado por *Componente III. API*.
2. *El servidor autentificador* busca el certificado de dicho usuario.
3. *El servidor autentificador* no encuentra el certificado debido a que el usuario no existe.
4. *El servidor autentificador* retorna un código de error.

... Fin Trayectoria alternativa 1 ...

4.2.2. Diagrama de flujo.

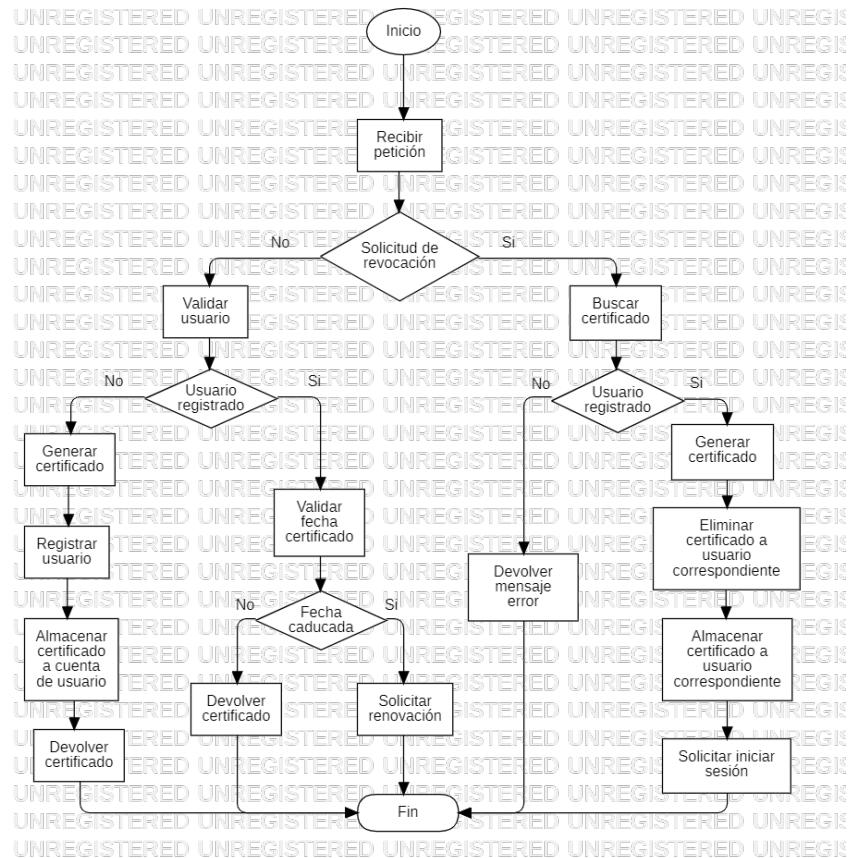


Figura 4.30: Diagrama de flujo de Componente II.

4.2.2.1. Descripción diagrama de flujo.

El flujo de este componente empieza al recibir una petición, posteriormente analiza que es lo que se le solicitó, ya que este componente tiene múltiples propósitos, principalmente registrar usuarios, verificar el certificado que recibe y revocar certificados ya existentes. El componente recibirá una petición y si esta es para el inicio de sesión, entonces el servidor autenticador validará al usuario, revisando si se encuentra registrado o no, para el primer caso validará que la fecha de expiración de dicho certificado siga vigente, si es así entonces devolverá el certificado de este usuario, si no se encuentra vigente entonces se le mandará un mensaje al usuario para solicitar la renovación del mismo. Por el otro lado si no se encuentra registrado, se debe de crear un

nuevo registro logrando de esta manera la generación de un certificado nuevo el cual se almacenará con los datos ingresados y finalmente se devolverá para que *la extensión* pueda hacer uso del mismo.

Existe también el caso en el que se solicite al servidor autenticador revocar un certificado, para ello se va a buscar dicho certificado y si se encuentra quiere decir que el usuario está registrado y se puede empezar el proceso de revocarlo, primero generando un certificado nuevo para este usuario, después se elimina el certificado asociado a este usuario, y se le asigna el certificado previamente creado a este nuevo usuario, por último se le manda un mensaje al usuario para que inicie sesión y se pueda devolver el nuevo certificado. En el caso en el que se pida revocar un certificado y no se encuentre ninguna cuenta registrada a este usuario entonces se mandará un mensaje de error.

4.2.3. Diagrama de flujo de datos.

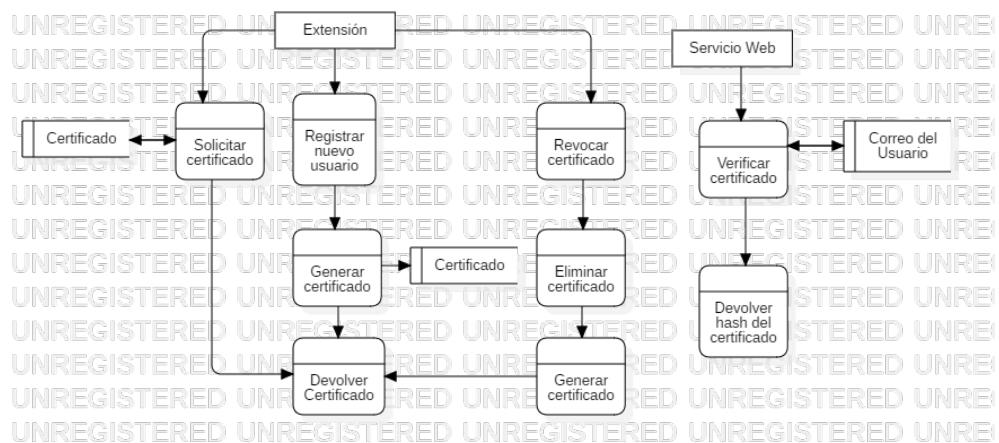


Figura 4.31: Diagrama de flujo de datos del Componente II.

4.2.3.1. Descripción diagrama de flujo de datos.

En este caso contamos con dos entidades externas, la extensión que por una parte puede solicitar un certificado mediante los datos solicitados por la autoridad, solicitar la generación de un nuevo usuario o bien solicitar la revocación de un certificado ligado a un usuario. La extensión le proporcionará los datos necesarios a la autoridad certificadora para que esta pueda crear y guardar al usuario con un certificado único para el mismo. Al solicitar un certificado la autoridad le devolverá a la extensión el certificado perteneciente al

usuario enviado. Si se solicita la creación de usuario, la autoridad solo devolverá como respuesta creación exitosa. Si la extensión solicita a la autoridad la revocación de un certificado, la autoridad necesita los datos del usuario a quien se le eliminará su certificado, para esto los datos del usuario deben ser correctos, si lo son, se procede a eliminar el certificado y a crear un nuevo, este certificado se guarda como certificado del usuario y se procede a devolver como respuesta dicho certificado creado previamente. Por otra parte, la API le solicitará a la autoridad la verificación de un certificado, esto con el fin de definir si el certificado fue generado por dicha autoridad y si ese certificado es valido.

4.2.4. Diagrama de clases.

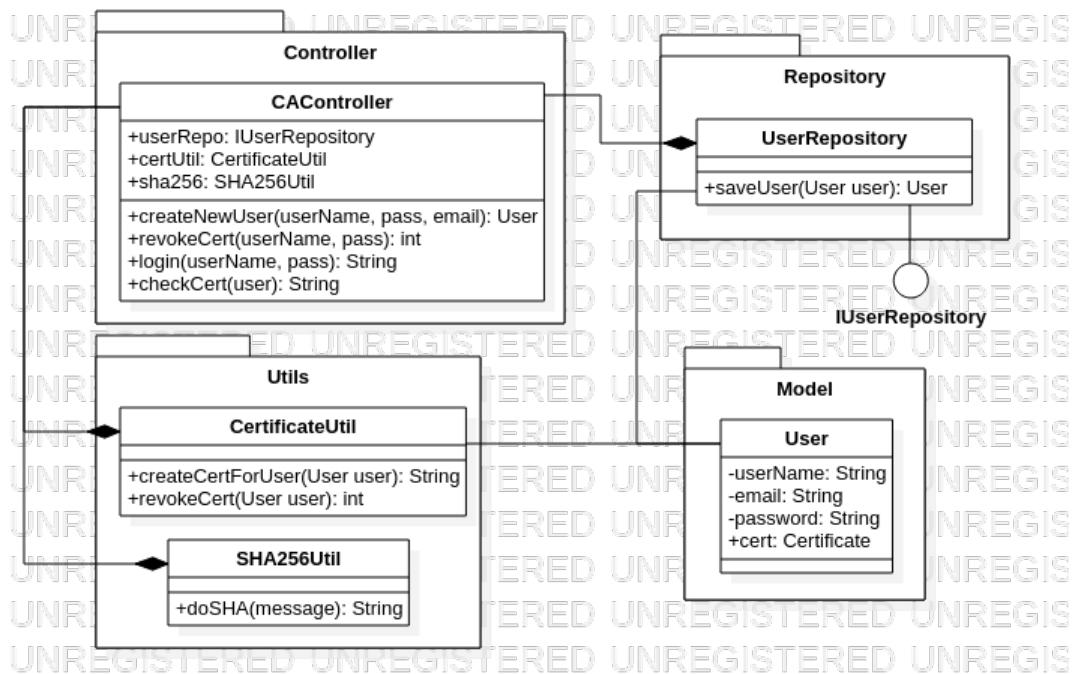


Figura 4.32: Diagrama de clases de Componente II.

4.2.4.1. Descripción de diagrama de clases

Clase: *CAController*

Atributos

1. **userRepo** : variable con la cual se pueden hacer operaciones con la base de datos.
 - Tipo de dato: **IUserRepository**.
2. **certUtil** : variable con la que se pueden hacer operaciones con y sobre los certificados, como crearlos o eliminarlos.
 - Tipo de dato: **CertificateUtil**.
3. **sha** : instancia para cifrar información con SHA256.
 - Tipo de dato: **SHA256Util**.

Métodos

1. **createNewUser(String pass, String email)**: Este método permite crear un nuevo usuario y guardar a éste en la base de datos .
 - Tipo de dato de retorno: **User**
2. **revokeCert(String email, String pass)**: Este método permite revocar el certificado actual del usuario con las credenciales recibidas.
 - Tipo de dato de retorno: **int**
3. **login(String pass, String email)**: Este método permite retornar el certificado del usuario de las credenciales recibidas.
 - Tipo de dato de retorno: **String**
4. **checkCert(String SHAuser)**: Este método permite retornar el sha256 del certificado del usuario recibido.
 - Tipo de dato de retorno: **String**

Clase: *CertificateUtil*

Métodos

1. **createCertForUser(User user)**: Este método permite crear un nuevo certificado con los datos del usuario.
 - Tipo de dato de retorno: **String**
2. **revokeCert(String userName, String pass)**: Este método permite revocar el certificado actual del usuario.

- Tipo de dato de retorno: **int**

Clase: *SHA256Util*

Métodos

1. **doSHA(String message)**: Este método permite cifrar el mensaje recibido con SHA256.

- Tipo de dato de retorno: **String**

Clase: *UserRepository*

Métodos

1. **saveUser(User user)**: Este método permite guardar en la base de datos el usuario recibido.

- Tipo de dato de retorno: **User**

Clase: *User*

Atributos

1. **email**: variable que almacena el email del usuario.

- Tipo de dato: **String**

2. **password**: variable que almacena la contraseña del usuario.

- Tipo de dato: **String**

3. **cert**: variable que almacena el certificado del usuario.

- Tipo de dato: **X509Certificate**

4.2.5. Diagramas de secuencias.

4.2.5.1. Diagrama de secuencia 1

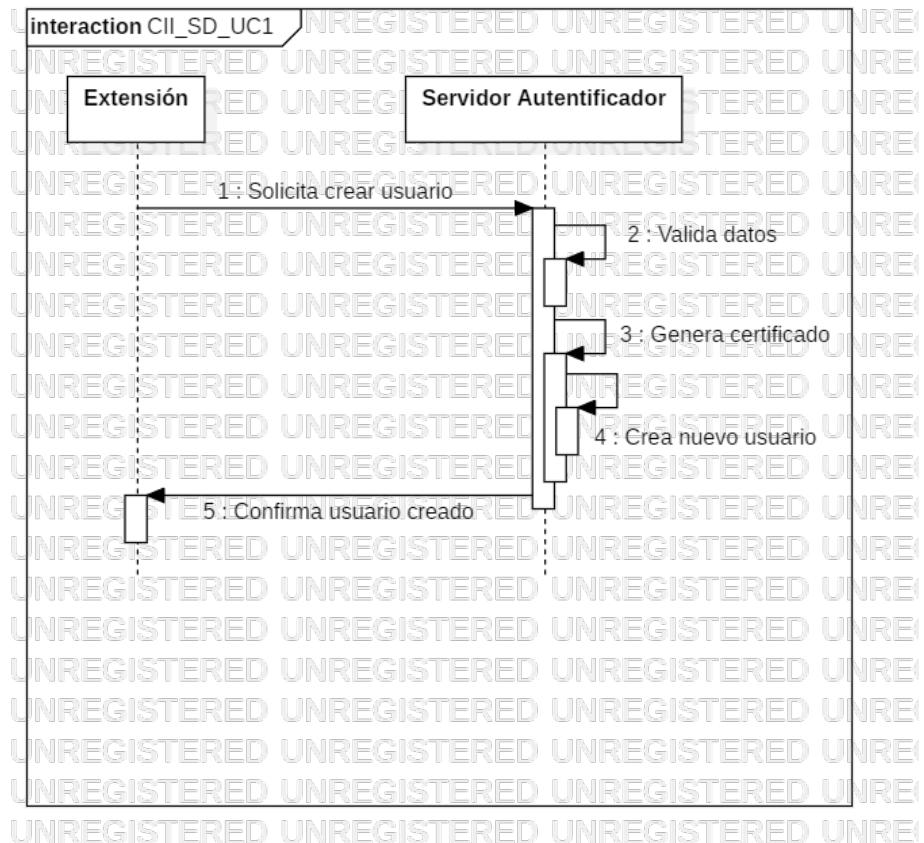


Figura 4.33: Diagrama de secuencia del CII_CU1. Crear Nuevo usuario.

Descripción: En este diagrama se explica el primer caso de uso, que es el de *crear un nuevo usuario*, donde el usuario solicita mediante la extensión el crear un nuevo usuario en el servidor autentificador, para que pueda empezar a utilizar nuestro servidor y asociar certificados a este usuario. Cabe resaltar que consideraremos cada correo que se reciba como un usuario.

4.2.5.2. Diagrama de secuencia 2

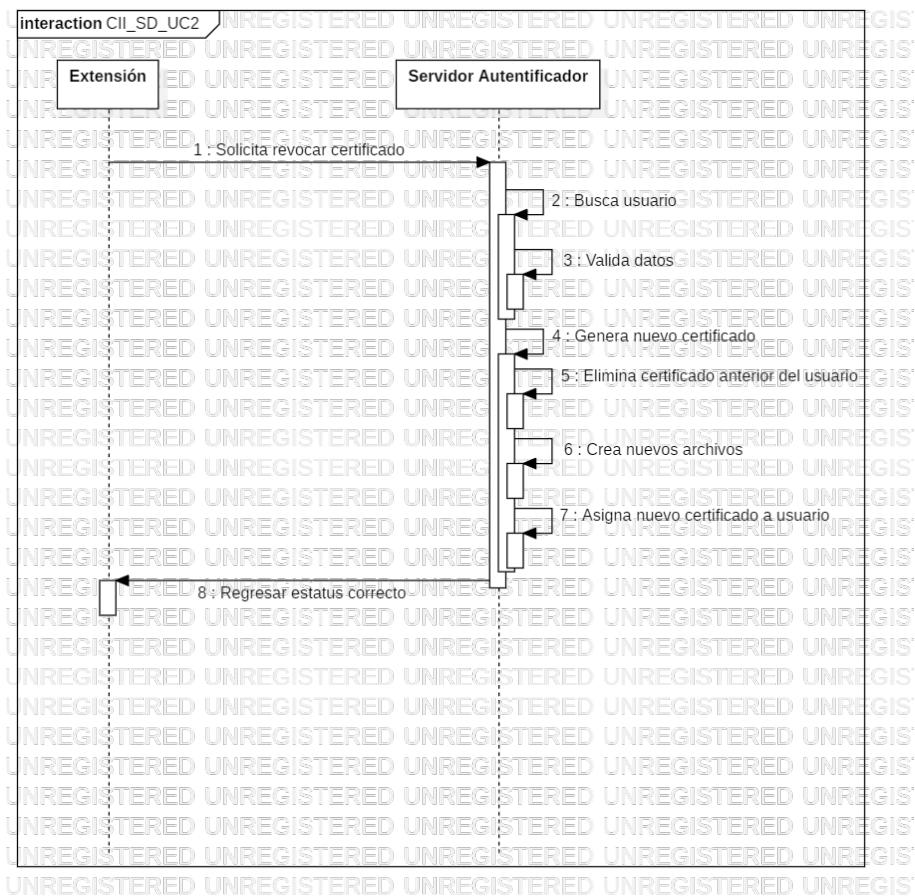


Figura 4.34: Diagrama de secuencia del CII_CU2. Revocar certificado

Descripción: En este diagrama, el usuario mediante la extensión solicita que se revoque su certificado, el servidor recibe esta solicitud y valida los datos, si son correctos genera un nuevo certificado y se lo asigna a este usuario.

4.2.5.3. Diagrama de secuencia 3

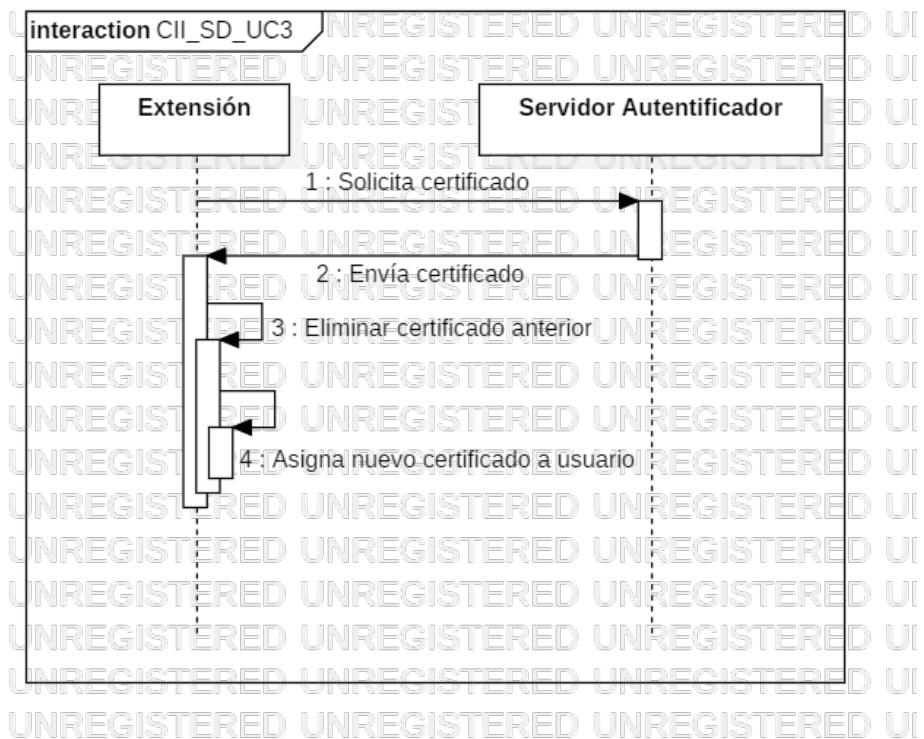


Figura 4.35: Diagrama de secuencia del CII_CU3. Obtener certificado.

Descripción: Para este diagrama, se obtienen un certificado desde el servidor a la extensión, esta después valida los datos y si son correctos, guarda en el storage de Chrome el certificado de ese usuario.

4.2.5.4. Diagrama de secuencia 4

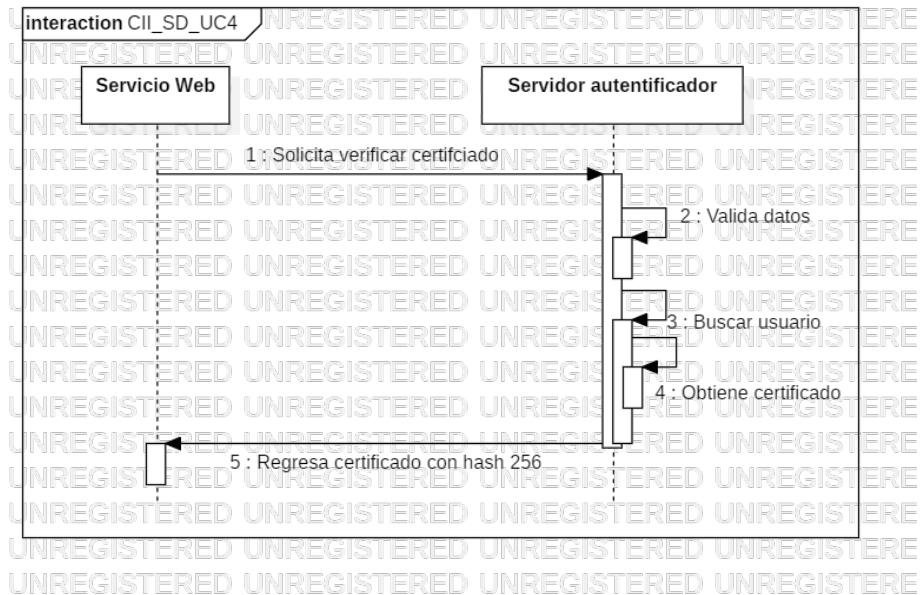


Figura 4.36: Diagrama de secuencia del CII_CU4. Verificar certificado.

Descripción: Este diagrama servira para verificar los certificados que obtiene el servidor desde la extensión, donde el servidor valida si el certificado que recibe existe en su base de datos, y le envía una respuesta a la extensión, en este caso se regresa como respuesta un hash 256 del certificado del usuario..

4.2.6. Diagrama de actividades.

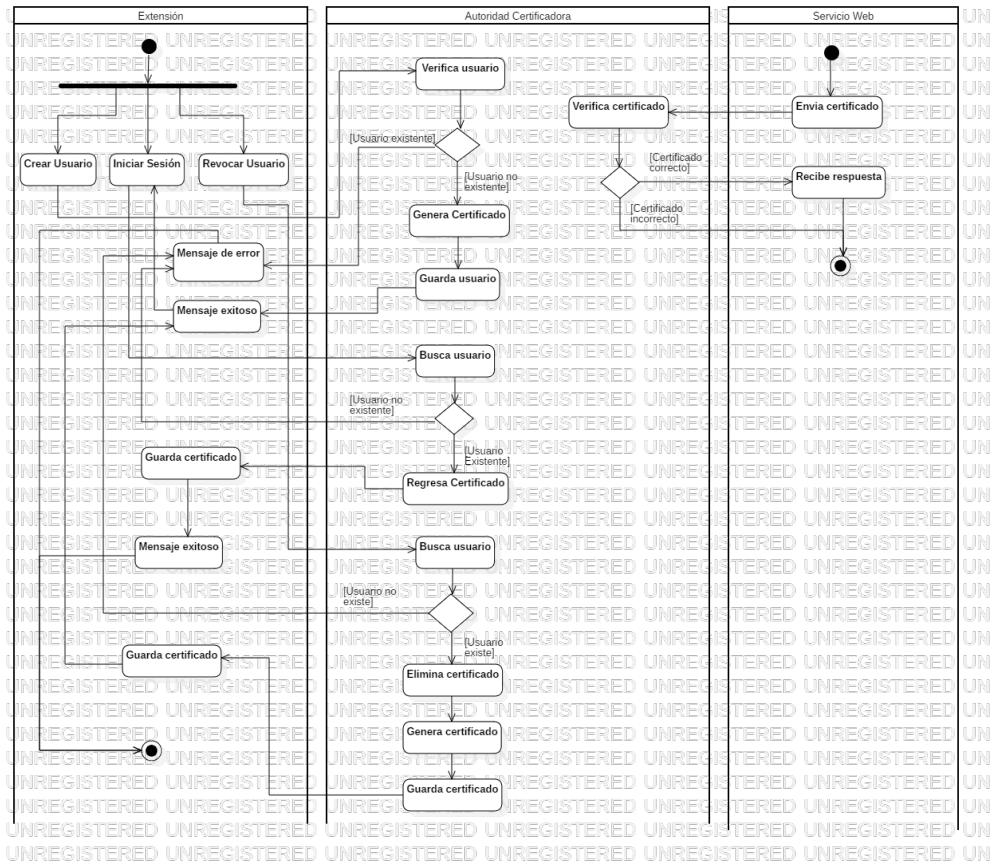


Figura 4.37: Diagrama de actividades de Componente II.

4.2.6.1. Descripción del diagrama de actividades.

El componente cuenta con los siguientes pasos. Por una parte la interacción entre la extensión y la autoridad certificadora, la cual la extensión podrá solicitar la creación de un nuevo usuario, iniciar sesión para obtener el certificado y la revocación de un certificado. Para el inicio de sesión la extensión debe de enviarle los datos de usuario para así la autoridad poder verificar dichos datos, si los datos son correctos y existen en la base de datos se procede a generar un certificado para el usuario y se registra en la base de datos. Si se solicita el inicio de sesión, es decir obtener un certificado, la autoridad al recibir dicha petición se procede a buscar el usuario solicitado, si existe la autoridad regresa como respuesta el certificado del usuario. Si la extensión solicita la revocación de un certificado, la autoridad debe buscar al

usuario en la base de datos, si existe un usuario con esos datos, se procede a eliminar su certificado y crear uno nuevo, devolviendo como respuesta el nuevo certificado. Ahora bien, si la API solicita la verificación de un certificado, la autoridad deberá verificar dicho certificado, si el certificado fue creado por la autoridad, este regresa respuesta satisfactoria.

4.3. Componente III: API.

4.3.1. Diagrama de casos de uso.

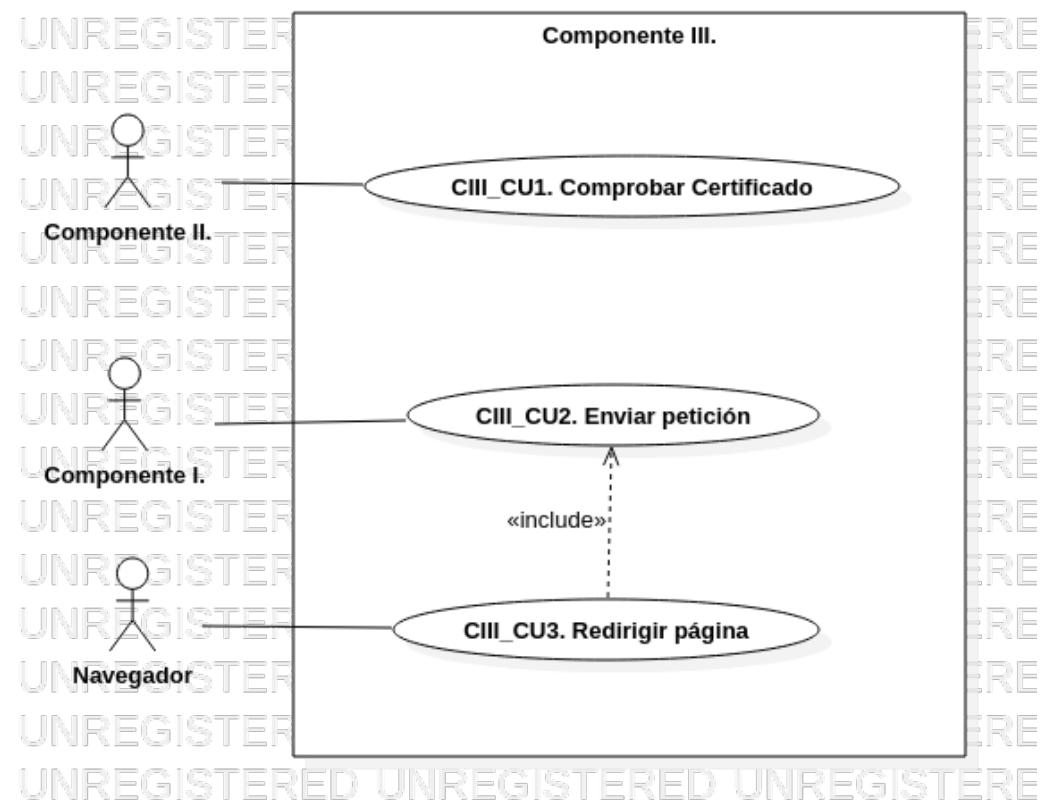


Figura 4.38: Diagrama de caso de uso del componente III

4.3.1.1. Descripción de diagrama de casos de uso.

Caso de uso: CIII_CU1. Comprobar certificado.	
Concepto	Descripción
Actor	Componente II. Servidor autentificador
Propósito	Saber si el certificado ha sido o no revocado por el usuario.
Entradas	SHA256 del usuario del certificado.
Salidas	SHA256 del certificado en el servidor autentificador del usuario.
Pre-condiciones	-
Post-condiciones	-
Reglas del negocio	CIII_RN3
Errores	No se encuentra al usuario.

Cuadro 4.12: Descripción CU: CIII_CU1

... Trayectoria Principal ...

1. *La API* envía un SHA256 del usuario del certificado al *Servidor autentificador*.
2. *El servidor autentificador* retorna un SHA256 del certificado actual en el servidor de ese usuario.
3. *La API* comprueba que el SHA256 del certificado obtenido de la petición es igual al SHA256 del certificado obtenido del *Servidor autentificador*
4. *La API* retorna al *Servicio web* que el certificado es válido

... Fin de la Trayectoria Principal ...

... Trayectoria Alternativa 1 ...

1. *La API* envía un SHA256 del usuario del certificado al *Servidor autentificador*.
2. *El servidor autentificador* retorna un SHA256 del certificado actual en el servidor de ese usuario.

3. *La API* comprueba que el SHA256 del certificado obtenido de la petición no es igual al SHA256 del certificado obtenido del *Servidor autenticador*
4. *La API* retorna al *Servicio web* que el certificado es inválido

... Fin de la Trayectoria Alternativa 1 ...

... Trayectoria Alternativa 2 ...

1. *La API* envía un SHA256 del usuario del certificado al *Servidor autenticador*.
2. *El servidor autenticador* no encuantra al usuario y retorna un código de error.
3. *La API* retorna al *Servicio web* que el certificado es inválido

... Fin de la Trayectoria Alternativa 1 ...

Caso de uso: CIII_CU2. Enviar petición.	
Concepto	Descripción
Actor	Componente I. Extensión
Propósito	Recibir la petición HTTP con el certificado inyectado.
Entradas	Petición HTTP con certificado inyectado para poder iniciar sesión.
Salidas	Acceso o no para el usuario al servicio web.
Pre-condiciones	-.
Post-condiciones	CIII CU4
Reglas del negocio	CIII RN4
Errores	La petición no tiene el certificado inyectado.

Cuadro 4.13: Descripción CU: CIII_CU2

... Trayectoria Principal ...

1. ***La extensión*** envía el certificado inyectado en la petición HTTP
2. ***El servicio web*** recibe la petición.

... Fin de la Trayectoria Principal ...

... Trayectoria Alternativa 1 ...

1. ***La extensión*** no envía el certificado inyectado en la petición HTTP
2. ***El servicio web*** recibe la petición.

... Fin de la Trayectoria Alternativa 1 ...

Caso de uso: CIII_CU3. Redirigir página.	
Concepto	Descripción
Actor	Navegador
Propósito	Redirigir la respuesta del servicio web para darle información al usuario acerca de su inicio de sesión.
Entradas	Respuesta del servicio web.
Salidas	Despliegue de la respuesta en el navegador.
Pre-condiciones	CIII_CU3.
Post-condiciones	-
Reglas del negocio	-
Errores	No se puede desplegar la respuesta del servicio web.

Cuadro 4.14: Descripción CU: CIII_CU3

... Trayectoria Principal ...

1. *El servicio web* envía la respuesta.
2. *El navegador* muestra la respuesta.

... Fin de la Trayectoria Principal ...

... Trayectoria Alternativa 1 ...

1. *El servicio web* envía la respuesta.
2. *El navegador* no muestra la respuesta.

... Fin de la Trayectoria Alternativa 1 ...

4.3.2. Diagrama de flujo.

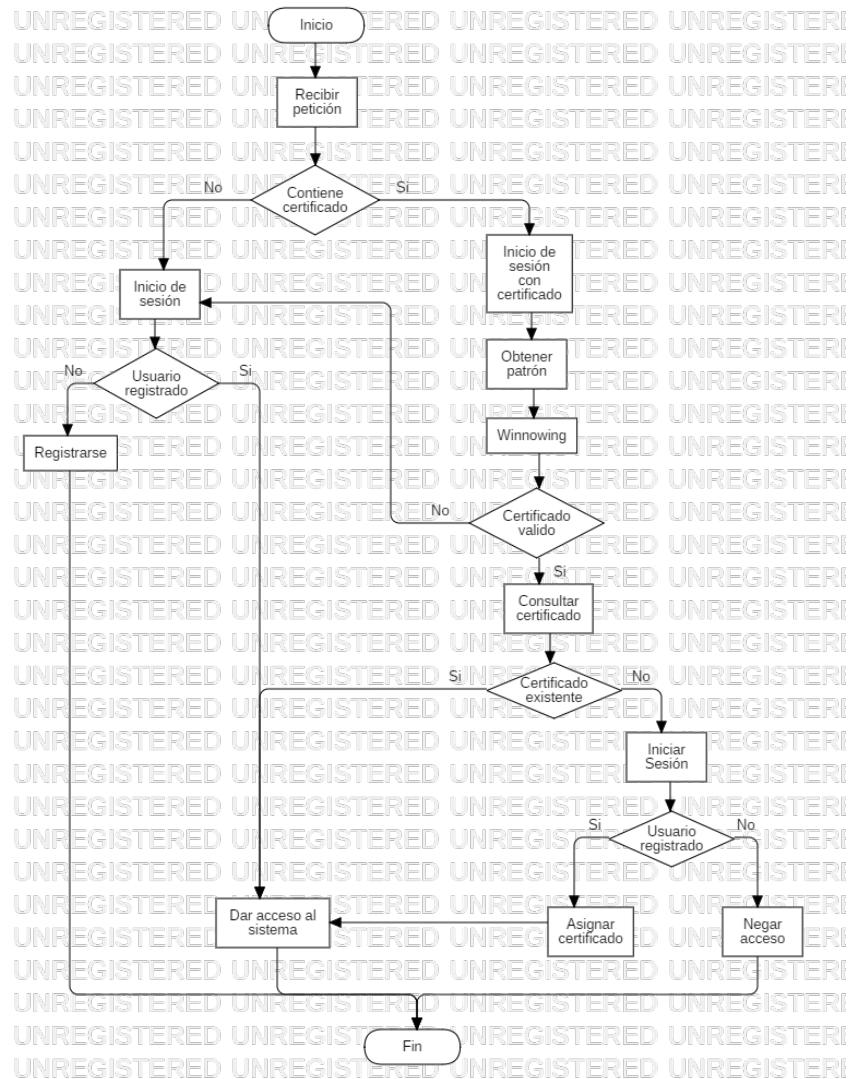


Figura 4.39: Diagrama de flujo de Componente III.

4.3.2.1. Descripción diagrama de flujo.

Para el caso de este diagrama se inicia con una petición, la cual es recibida por la API y analizada para verificar si el encabezado tiene el certificado. Si la petición no tiene el certificado, este componente ignora dicha petición. Si la petición contiene un certificado, se procede a iniciar sesión con certificado,

Si se realiza satisfactoriamente el winnowing, se inicia sesión donde puede o no estar registrado el usuario, de cualquier manera el usuario al registrarse o iniciar sesión se usara el certificado.

4.3.3. Diagrama de flujo de datos.

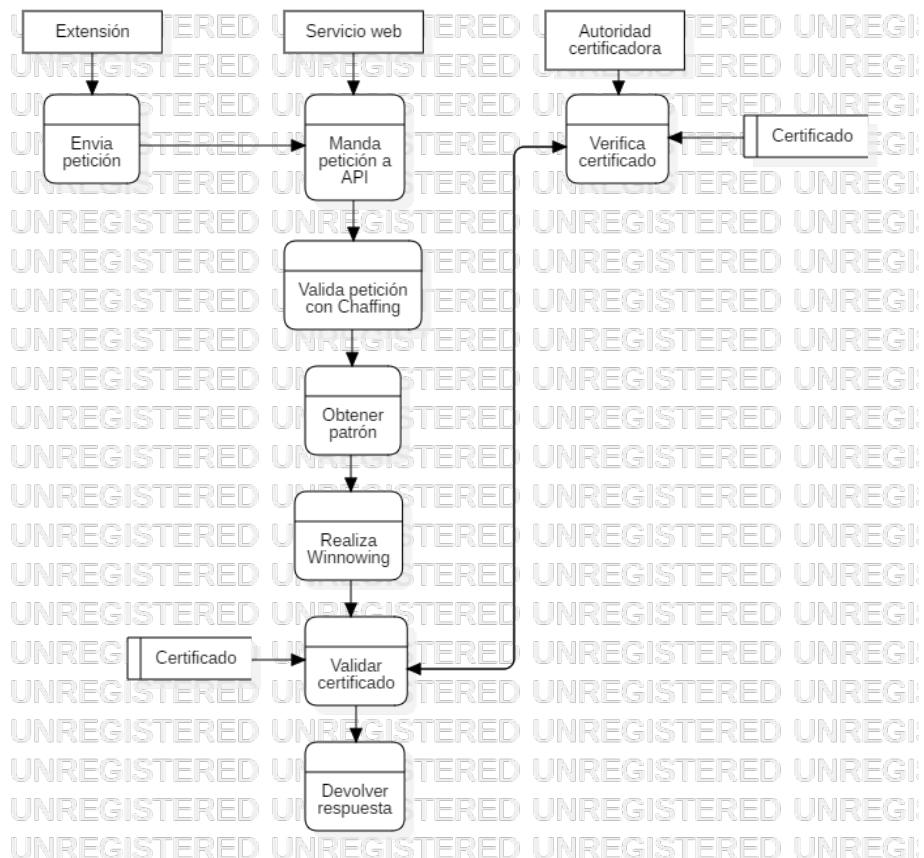


Figura 4.40: Diagrama de flujo de datos de Componente III.

4.3.3.1. Descripción diagrama de flujo de datos.

A continuación tenemos el diagrama de flujo de datos, donde podemos ver cláaramente como viaja la información principal a traves de este componente y con las entidades externas, primero la extensión envía una petición al servicio web, este lo recibe, la API lo intercepta y verifica si es una petición con Chaffing que debe de ser analizada, si es así realiza la etapa de winnowing descifrando el patrón con su llave privada y por último obtiene el certificado

dentro de esta petición y la compara con las que cuenta con el servicio web, para saber si debe de dar una respuesta de usuario o solicitar que inicie sesión en este mismo.

4.3.4. Diagrama de clases.

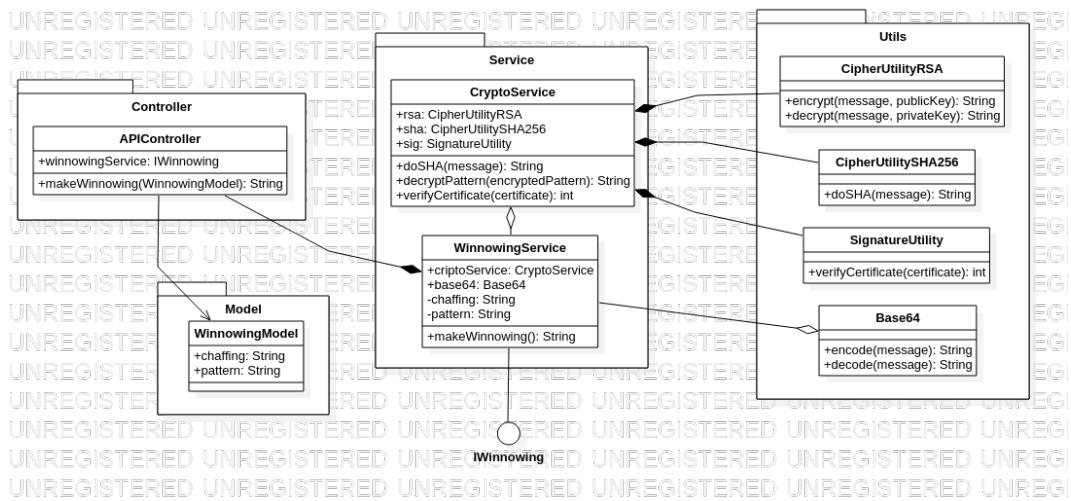


Figura 4.41: Diagrama de Clases de componente 3.

4.3.4.1. Descripción diagrama de clases.

Clase: *APIController*

Atributos

1. **winnowingService** : variable donde se tendrá acceso a los métodos necesarios para realizar la etapa de winnowing.
 - Tipo de dato: **IWinnowing**.

Métodos

1. **makeWinnowing(WinnowingModel wm)**: Este método realiza la etapa de winnowing, retorna el certificado.
 - Tipo de dato de retorno: **String**

Clase: *WinnowingModel*

Atributos

1. **chaffing** : variable donde se tiene guardado el chaffing de la petición.
 - Tipo de dato: **String**.
2. **pattern** : variable donde se tiene guardado el patrón de la petición.
 - Tipo de dato: **String**.

Clase: *CryptoService*

Atributos

1. **rsa** : instancia para descifrar información con RSA.
 - Tipo de dato: **CipherUtilityRSA**.
2. **sha** : instancia para cifrar información con SHA256.
 - Tipo de dato: **CipherUtilitySHA256**.
3. **sig** : instancia para validar información de un certificado.
 - Tipo de dato: **SignatureUtility**.

Métodos

1. **doSHA(String message)**: Este método calcula el sha256 de la cadena de texto message.
 - Tipo de dato de retorno: **String**
2. **decryptPattern(String encryptedPattern)**: Este método descifra el patrón de chaffing.
 - Tipo de dato de retorno: **String**
3. **verifyCertificate(String certificate)**: Este método verifica la autenticidad de un certificado.
 - Tipo de dato de retorno: **int**

Clase: *WinnowingService*

Atributos

1. **cryptoService** : instancia para acceder a todas las utilidades de cifrado.
 - Tipo de dato: **CryptoService**.
2. **base64** : instancia para decodificar información en formato BASE64.
 - Tipo de dato: **Base64**.
3. **chaffing** : variable para guardar el chaffing actual.
 - Tipo de dato: **String**.
4. **pattern** : variable para guardar el patron de chaffing actual.
 - Tipo de dato: **String**.

Métodos

1. **makeWinnowing()**: Este método realiza la etapa de winnowing.
 - Tipo de dato de retorno: **String**

Clase: *CipherUtilityRSA*

Métodos

1. **encrypt(String message, PublicKey publicKey)**: Este método cifra un mensaje con la llave pública especificada.
 - Tipo de dato de retorno: **String**
2. **decrypt(String message, PrivateKey privateKey)**: Este método descifra un mensaje con la llave privada especificada.
 - Tipo de dato de retorno: **String**

Clase: *CipherUtilitySHA256*

Métodos

1. **doSHA(String message)**: Este método cifra el mensaje que se le manda.
 - Tipo de dato de retorno: **String**

Clase: *Signature Utility*

Métodos

1. **verifyCertificado(String certificate)**: Este método verifica la validez de un certificado.
 - Tipo de dato de retorno: **int**

Clase: *Base64*

Métodos

1. **encode(String message)**: Este método codifica el mensaje recibido a base64.
 - Tipo de dato de retorno: **String**
2. **decode(String message)**: Este método decodifica el mensaje recibido de base64 a UTF-8.
 - Tipo de dato de retorno: **String**

4.3.5. Diagramas de secuencias.

Diagrama de Secuencia 1. Comprobar Certificado.

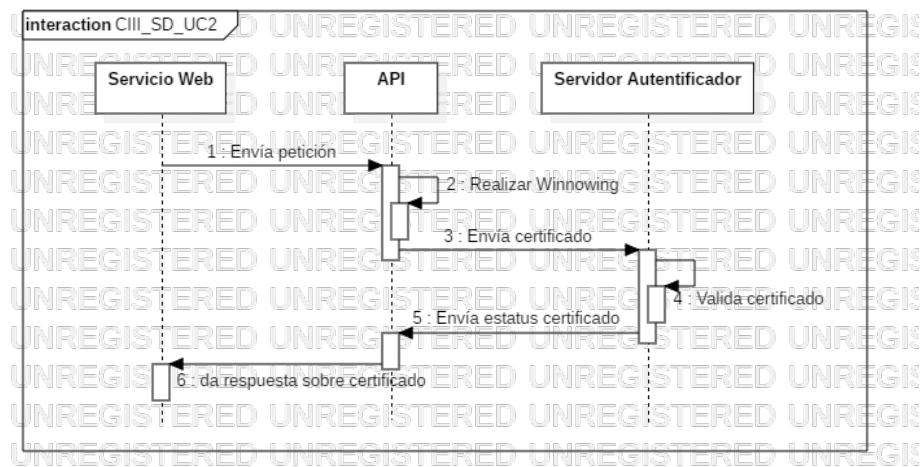


Figura 4.42: Diagrama de Secuencia de Caso de Uso 1. Comprobar certificado.

4.3.5.1. Descripción de diagrama de Secuencia CIII_SD_UC1.

Después de que el servicio web reciba una petición, la API la interceptará para realizar la etapa de Winnowing y posteriormente envíe el certificado al servidor autenticador, el Servidor Autentificador validará los datos del usuario y dependiendo si existe o no ese usuario registrado, con un certificado válido(actualizado) o uno inválido(revocado) le regresará un status a la API y está sabrá el como darle respuesta al Servicio Web basado en el estatus que recibió del Servidor Autentificador.

Diagrama de Secuencia 2. Enviar Petición.

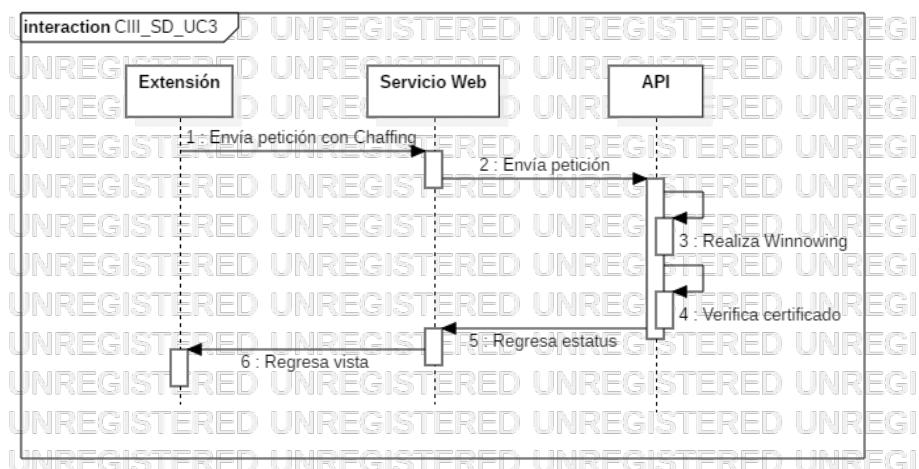


Figura 4.43: Diagrama de Secuencia de Caso de Uso 2. Enviar petición.

4.3.5.2. Descripción de diagrama de Secuencia CIII_SD_UC2.

La extensión envía una petición con Chaffing al Servicio Web, donde la API intercepta dicha petición para analizar si es una petición que contenga un Chaffing el cuál podamos analizar, posteriormente realizará la etapa de Winnowing si es una petición de nuestro interés y enviará el certificado al servicio web.

Diagrama de Secuencia 3. Redirigir Página.

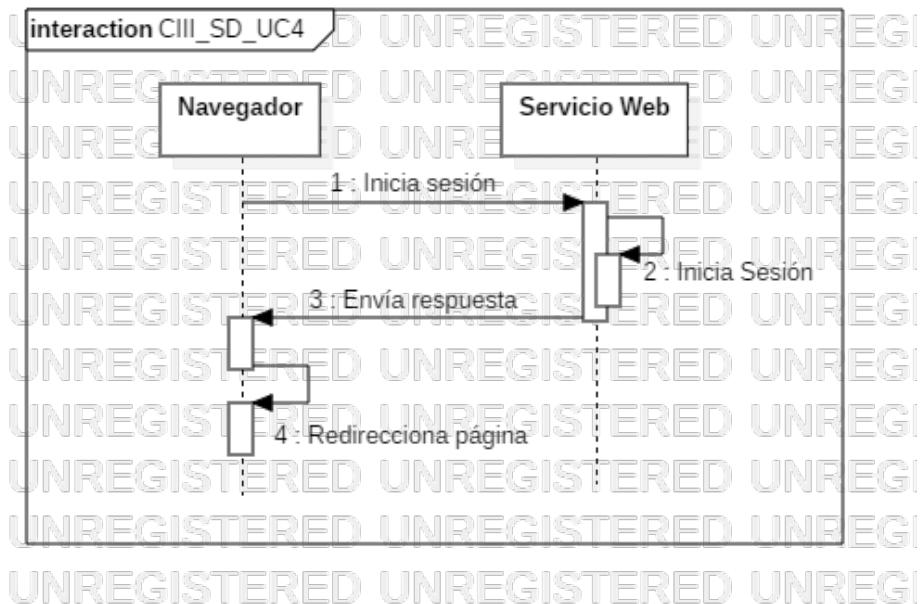


Figura 4.44: Diagrama de Secuencia de Caso de Uso 3. Redirigir página.

4.3.5.3. Descripción de diagrama de Secuencia CIII_SD_UC3.

El navegador teniendo lista la petición con Chaffing la envía al Servicio Web, donde éste mediante la API, valida dicha petición y elige el inicio de sesión para el usuario, posteriormente le envía una respuesta al navegador y por último este mismo redirige la página dependiendo es esta misma respuesta.

4.3.6. Diagrama de actividades

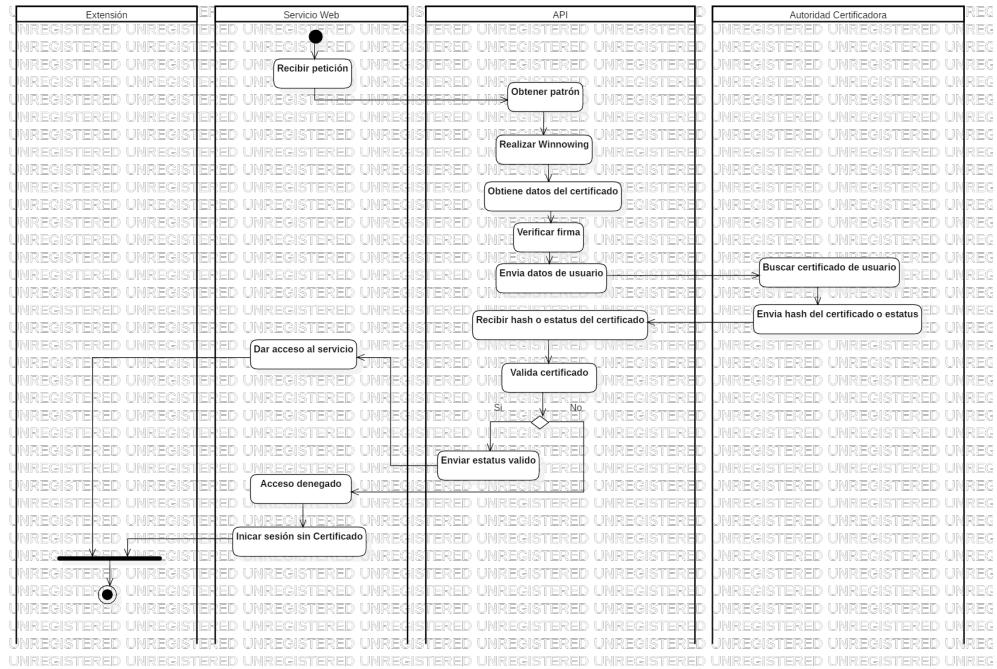


Figura 4.45: Diagrama de Actividades de componente 3.

4.3.6.1. Descripción diagrama de actividades.

Cuando el servicio web recibe una petición con chaffing, realiza la etapa de Winnowing para poder obtener el certificado, y posteriormente se comunica con la autoridad certificadora, este comparará el certificado, ya que existen 3 casos posibles: el primero es que el certificado sea válido, y simplemente le dará acceso al servicio web con los datos del certificado correspondiente, el segundo caso es que el usuario sea válido, pero el certificado haya sido revocado antes y necesite actualizar su certificado en ese ordenador y el tercero es que el certificado sea inválido, por lo cual le dará un mensaje al usuario de que el certificado no es el correcto.

4.3.7. Interfaz de usuario.

Una vez que el usuario tiene una cuenta y ha obtenido su certificado de la autoridad certificadora. El usuario puede proceder a navegar en la red para hacer uso de la extensión. En el componente 3, las interfaces que se muestran al usuario son las vistas del servicio web que se ha implementado para llevar

a cabo el proceso de *Winnowing*.

El servicio web de prueba que se utilizará para este trabajo es una pagina web para usuarios y doctores de una *veterinaria*, la cual se utilizará para la modificación correspondiente donde el servicio lleve a cabo la autenticación por *Chaffing and Winnowing*.

El servicio web de la veterinaria se muestra en la figura 4.46.



Figura 4.46: Interfaz principal del servicio web de veterinaria.

En esta interfaz dada por el servicio web, le permite al usuario iniciar sesión. Como la extensión esta activada, ésta realizará la tarea de bloquear la petición, para hacer el proceso de *Chaffing* y posteriormente inyectar el resultado en el encabezado de la petición (ver Componente 1).

Al recibir la petición en el servicio web, éste enviará la petición a la API que se implementa en el servicio web. La API es la encargada de realizar el proceso de *Winnowing* y verificará la autenticidad del certificado.

Una vez verificada la autenticidad del certificado, se muestra la interfaz de la figura 4.47 en el cual se le da a indicar al usuario que se ha detectado un certificado y esta listo para vincularlo a una cuenta existente en el servicio web.



Figura 4.47: Interfaz para iniciar sesión en el servicio web con certificado en el encabezado de la petición.

Una vez que el usuario ingrese sus credenciales correctas del servicio web, éste vinculará el certificado a este usuario y a su vez le dará acceso al servicio con su cuenta. La figura 4.48 muestra la página de inicio del servicio web con la cuenta del usuario la cual ya tiene un certificado vinculado.

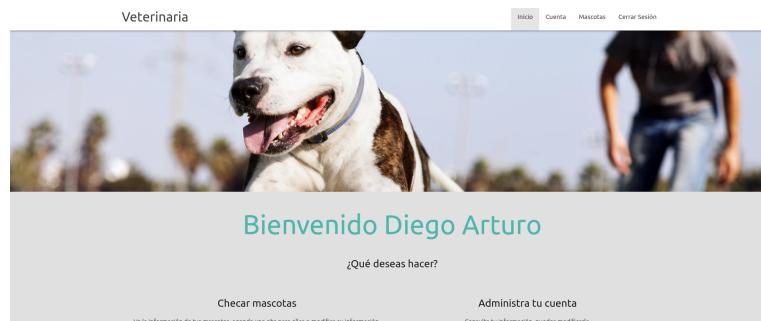


Figura 4.48: Interfaz del servicio web con acceso de una cuenta.

El servicio web le permite al usuario crear una nueva cuenta en el servicio, ya que es necesario para poder vincular el certificado. La figura 4.49 muestra la interfaz que se le presenta al usuario para la creación de una nueva cuenta en el servicio web.

Veterinaria

Iniciar Sesión Registrar

Completa el siguiente formulario

Información general

Nombre _____

Primer apellido _____

Segundo apellido _____

E-mail _____

Día de nacimiento Elige ▾

Mes de nacimiento Elige ▾

Año de nacimiento Elige ▾

Información de la cuenta

Soy veterinario

Figura 4.49: Interfaz del servicio web para crear nuevo usuario.

Otro caso a considerar es cuando el usuario no activa la extensión y quiere ingresar al servicio web, por lo que si el usuario realiza una petición al servicio con la extensión deshabilitada, el servicio web no encontrará ningún certificado con *Chaffing* inyectado en el encabezado, por lo que el servicio web pedirá realizar un inicio de sesión ordinario, el cual le pedirá usuario y contraseña para ingresar al servicio con su cuenta.

La figura 4.50 muestra la interfaz que se le muestra al usuario cuando se requiere un inicio de sesión ordinario.

Inicia sesión

Nombre de Usuario

Contraseña de Usuario

INICIAR SESIÓN >

Figura 4.50: Interfaz de inicio de sesión ordinario del servicio web.

Como sabemos, el usuario puede revocar un certificado lo cual quiere decir que puede generar uno nuevo. Esto le permite al usuario poder generar un

nuevo certificado si el usuario dejo su sesión iniciada en otra maquina publica. El tener la capacidad de generar otro certificado, le permite al servicio web no dar acceso si el certificado recibido se encuentra revocado, pidiendo asi iniciar sesión de nuevo con credenciales (usuario y contraseña) para vincular el nuevo certificado. La figura 4.51 es la interfaz que se le muestra al usuario si el certificado que el servicio web obtiene del encabezado se encuentra revocado.



Figura 4.51: Mensaje de inicio de sesión incorrecto por revocación de certificado ó credenciales inválidas.

4.3.8. Algoritmos.

Los algoritmos necesarios para hacer el *winnowing* así como la verificación de los certificados son expuestos a continuación:

```

Data: chaffing, patternCipher, aesCipher, privateKey
Result: cert,header
1 chaffingDecode[] ← base64.decode(chaffing);
2 aesKey ← rsa.decrypt(aesCipher,privateKey);
3 pattern[] ← aes.decrypt(patternCipher,aesKey);
4 cert[];
5 header[];
6 i ← 0;
7 while i < pattern.length do
8   | if pattern[i] == 1 then
9     |   | header.add(chaffingDecode[i]);
10    | else
11      |   | cert.add(chaffingDecode[i]);
12    | end
13    | i ← i + 1;
14 end

```

Algoritmo 3: Obtención de la cabecera y certificado mediante el proceso de winnowing.

En este algoritmo se toman en consideración los siguientes datos de la petición HTTP recibida:

- **chaffing**: es la cadena que se encuentra en base 64 que contiene el certificado y la basura mezclada.
- **aesCipher**: Se encuentra en el mismo campo del patrón, contiene la llave AES cifrada mediante el cifrado asimétrico RSA.
- **patternCipher**: es el otro apartado que contiene el patrón necesario para realizar el proceso de winnowing, el cual está cifrado con el algoritmo de cifrado simétrico AES.

Así como la llave privada necesaria para realizar el decifrado RSA **privateKey** la cual si bien no se envía en la petición, se encuentra almacenada localmente como dato estático. El proceso de winnowing se realiza analizando el patrón para dividir cada uno de los bits del chaffing en certificado y cabecera.

En cuanto a la salida de este algoritmo se hace mención de dos datos los cuales se despliegan a continuación:

- **cert**: Contiene el certificado del usuario completamente íntegro asumiendo que el proceso fue correcto.

- **header:** Contiene la información del campo *Accept* de la cabecera HTTP.

Por otro lado se cuenta con otro algoritmo para realizar la verificación del certificado buscando que no se incluya algún certificado no válido o que sea expedido por alguna AC externa a la desarrollada por nosotros:

```

Data: cert,publicKey
Result: certR,flag
1 if cert != null and cert.verify(publicKey) == 1 then
2   | dataCert ← getDataCert(cert);
3   | emailUser ← dataCert.getEmail();
4   | response ← CA.getValidation(sha256.doSha(emailUser));
5   | shaCert ← sha256.doSha(cert);
6   | if response == 0 then
7     |   | certR = 0;
8     |   | flag = 0;
9   | else
10    |   | if response == shaCert then
11      |   |   | certR = cert;
12      |   |   | flag = 1;
13    |   | else
14      |   |   | certR = 0;
15      |   |   | flag = 2;
16    |   | end
17  | end
18 else
19   |   | certR = 0;
20   |   | flag = 0;
21 end
```

Algoritmo 4: Algoritmo para la verificación del certificado.

4.3.8.1. Complejidad computacional.

Haciendo un análisis del algoritmo de winnowing anteriormente mostrado, deducimos que la complejidad del algoritmo de *winnowing* es: $O(n)$ donde n es el tamaño de caracteres del certificado.

Para el caso de la verificación se cuenta con los siguientes datos de entrada:

- **cert:** Contiene el certificado recibido en la petición HTTP proveniente del anterior algoritmo.

- **publicKey:** Contiene la public Key de la AC necesaria para la comprobación del certificado.

También es importante dejar en claro algunas funciones con las que se cuenta en el algoritmo como es el caso de **getDataCert(key)** que es el algoritmo que se encarga de comprobar tanto la fecha del certificado como validar que el mismo haya sido expedido por la AC correspondiente, por otro lado la función **CA.getValidation(sha256)** es una función que se encarga de comunicarse con la AC enviándole un sha del usuario, que en este caso es el email, para que busque en sus repositorios si el certificado aún se encuentra activo para el nombre de usuario dado, a lo que nos responderá *0* si el usuario no existe o *ShaCertificadoUsuario* si el usuario cuenta con un certificado válido para realizar una comparación con el recibido de la petición. La salida de este algoritmo se compone de lo siguiente:

- **certR:** Contiene el certificado después de validarse para que el servicio web pueda almacenarlo en su base de datos.
- **flag:** Contiene una bandera de control para saber el resultado de la validación cuyos valores posibles son los siguientes:
 - *0*: Se refiere a que el certificado recibido no es válido.
 - *1*: Se refiere a que el certificado recibido es válido.
 - *2*: Se refiere a que el certificado recibido es válido pero ya no se encuentra asignado al usuario en cuestión, es decir, fue revocado.

Por lo que de esta manera el servicio web puede decidir qué hacer en cada uno de los casos, tomando en cuenta la recomendación de no mostrar demasiada información al usuario cuidando siempre la seguridad de los sistemas.

Capítulo 5

Desarrollo.

5.1. Componente I. Extensión.

Como podemos ver hay dos archivos que acompañan al **manifest.json** llamados **popup.html** y **background.js** de los cuales hablaremos a continuación.

5.1.1. Archivo **popup.html**

Es un HTML que contiene la página que se mostrará al hacer click en el botón de la extensión, la idea de esta *mini página* es que nos permita gestionar las funciones de la extensión, por tal motivo contará con 4 botones necesarios para su funcionamiento:

- *btnActivar* : Botón necesario para activar o desactivar el funcionamiento de la extensión.
- *btnIniciarSesion* : Botón en el que el usuario podrá iniciar sesión o registrarse para empezar a utilizar la extensión.
- *btnCerrarSesion* : Botón que nos permite cerrar la sesión del usuario actual.
- *btnAviso* : Botón que nos redirige a una página en donde podremos aceptar el certificado, así como muestra algunos datos de cómo funciona el proceso de autentificación.

5.1.2. Archivo background.js

Es el archivo que contiene la mayor parte de la lógica del comportamiento de este componente, vamos a explicar alguna de las funciones más relevantes de su desarrollo

Lo primero que debe de realizar la extensión es interceptar la petición antes de que esta salga a red para poder realizar todo el proceso de Chaffing.

5.1.2.1. Interceptar petición.

Primero que nada, se necesita comprobar que la extensión se encuentre habilitada, checando la bandera de *btnActivar*, si

5.1.2.2. Creación del patrón de Chaffing

Esta función nos servirá para poder generar el patrón de

5.1.2.3. Generación del chaffing.

Esta función nos va a generar el Chaffing resultante

5.2. Componente II. Servidor Autentificador.

En esta sección se mostrará el proceso que se llevó a cabo para la creación del componente II. La implementación de este componente se desarrolló en **NodeJS** el cual es un entorno en tiempo de ejecución multiplataforma para la capa del servidor basado en lenguaje de programación *ECMAScript*.

5.2.1. Manejador de paquetes de Node (npm).

npm (*Node Package Manager*) es el sistema de gestión de paquetes por defecto para Node.js, un entorno de ejecución para

npm nos permite poder instalar una infinidad de librerías, para este componente haremos uso de algunas librerías como las que se describen a continuación:

- mongoose (versión 5.7.3): Nos permite de una manera más rápida tener una conexión con mongodb. Sólo se necesita especificar la base de datos a la que se desea conectar y el puerto. Mongoose se encarga de establecer la conexión para así poder manipular la base de datos.
- express (versión 4.17.1): Es una infraestructura de aplicaciones web que proporciona un conjunto de características como son:
 - Escritura de manejadores de peticiones.
 - Integración con motores de renderización de "vistas".
 - Añade procesamiento de peticiones "middleware".
- morgan (versión 1.9.1): Nos permite poder ver el estatus de las peticiones http que se hacen al servidor.
- nodemon (versión 1.19.3): Nos permite poder reiniciar el servidor automáticamente. Esta librería y la anterior no son necesarias en producción, mas sin embargo en desarrollo se recomienda.
- ejs (versión 2.7.1): Este paquete nos permite crear vistas en lenguaje *ejs*.
- crypto-js (versión 3.1.9): Este paquete tiene gran variedad de funciones criptográficas, desde hash hasta RSA, entre muchas otras funciones.
- node-openssl-cert (versión 0.0.98): Este paquete tiene gran variedad de funciones para la creación de certificados openSSL.
- Por si solo NodeJS tiene librerías tales como fs, path, https, entre otras, las cuales también haremos uso.

5.2.2. Componentes.

En la figura 5.1 se muestra los componentes que tendrá la autoridad certificadora.

Figura 5.1: Estructura del desarrollo de la Autoridad Certificadora.

Los componentes son los siguientes:

- node_modules: Son los módulos de NodeJs donde se encuentran todas las librerías que se estarán ocupando en el proyecto.
- src: En esta carpeta se almacena el código de las diferentes rutas, así como las vistas y los modelos para la base de datos.
- database.js: Es complemento de keys.js donde se establece la conexión a la base de datos con los datos obtenidos en keys.js
- index.js: Aquí se declaran las rutas y los *middlewares* del servidor, así como el puerto y la configuración https.
- keys.js: Aquí se especifica el puerto y la base de datos a la que queremos conectarnos.
- Usuarios_CRT: Esta carpeta contendrá todos los archivos .csr y .crt de los usuarios registrados
- package.json: Este es el paquete del proyecto, donde se especifica la versión, componentes, etc. (ver figura ??)

Antes que nada se debe de inicializar el servidor en un

Ahora necesitaremos declarar todas las rutas a las cuales la extensión (Componente I) podrá hacer peticiones. Una vez teniendo nuestro servidor corriendo con una conexión segura y con las rutas, debemos configurar el servidor para tener acceso a la base de datos de MongoDB, ya que necesitaremos guardar ciertos datos del usuario. Necesitamos especificar el puerto y nombre de la base de datos a la que queremos conectarnos. La figura ?? muestra los parámetros que se necesitan para conectarse a la base de datos, mientras que la figura ?? muestra Debemos crear nuestro modelo de la base de datos, ya que será de tipo JSON el modelo que debemos almacenar en la base de datos, por lo que se necesita especificar los atributos del modelo **usuario**. Gracias a la clase **Schema** que mongoose nos proporciona nos facilita el modelado del usuario para la base de datos. Los parámetros que tendrá nuestro esquema y nuestra b

5.2.2.1. Crear nuevo usuario.

Esta función en la autoridad certificadora permitirá al usuario poder crear una cuenta desde la extensión, para así poder obtener un certificado y hacer uso de la misma en peticiones futuras. Usando la librería *node-openssl-cert* nos permite crear certificados formato x509 de OpenSSL. El código

5.2.2.2. Obtener certificado.

Esta función le permite al usuario obtener su certificado. La autoridad certificadora se encargará de buscar que exista el usuario tanto en la base de datos como su certificado. Si existe el usuario y un certificado vinculado al mismo, su certificado se regresa como respuesta al usuario. Si no se encuentra el usuario o su certificado, se regresa como respuesta un estatus de error. El có

5.2.2.3. Revocar certificado.

Esta función será la encargada de revocar el certificado, esto con el fin de crear un nuevo certificado y reasignarlo al usuario quien realiza dicha revocación. Una vez terminado el proceso, el usuario necesitará hacer la petición correspondiente para obtener su certificado de nuevo, ya que el actual no tendrá v

5.2.2.4. Verificar certificado.

La verificación del certificado se hará por medio de peticiones que haga la API, la cual solicitará el hash 256 del usuario quien esta iniciando sesión en el servicio. Esta con el fin de verificar validez del certificado para saber si es valido o se ha revocado el certificado que esta llegando a través del encabezado de la petición. El código ?? muestra la implementación de esta función.

5.3. Componente III. API.

En esta sección se mostrarán los pasos que se realizaron para implementar el Componenete III. Dichos pasos constan de versiones de los softwares utilizados y configuraciones necesarias para el funcionamiento.

5.3.1. API

Para la realización de la API, utilizamos *Java JDK 8.0* junto con el framework *Spring Framework 5.0* con el objetivo de ahorrar tiempo de desarrollo gracias a las facilidades que nos brinda *Spring*.

5.3.1.1. Creación.

Utilizando una herramienta web llamada *Spring initializr* creamos una aplicación con los siguientes parámetros de configuración.

- Proyect: **Maven Projetc**
- Language: **Java**
- Spring Boot: **2.2.0**
- Proyect Metadata:
 - Group: **TT2018B003.comp3**
 - Artifact: **API**
 - Options:pretende
 - Name: **API**
 - Description: **API for winnowing**
 - Packaging: **JAR**
 - Java: **8**
- Dependencies:
 - **Spring Boot Actuator**
 - **Spring Web**
 - **Spring Boot DevTools**
 - **Lombok**

Una vez descargado el proyecto desde la página web, utilizamos *Spring Tools Suite*, el cual es un IDE desarrollado por Spring basado en Eclipse. Este IDE, nos brinda la posibilidad de poder cargar el proyecto como *Proyecto Maven Existente* para comenzar a implementar la solución.

5.3.1.2. Implementación de la solución.

Basándonos en los diagramas de la sección de Diseño para el Componente III, creamos los paquetes y clases necesarios para la solución. En la Figura 5.2 se muestra la organización del proyecto:

Figura 5.2: Estructura del desarrollo de la API.

ApiApplication.java es una clase creada automáticamente por Spring initializr. Dicha clase se encarga de correr todo lo necesario para la ejecución de la aplicación, es decir, creación de beans de Spring y su configuración.

APIController.java es una clase creada para este trabajo la cual contiene un método llamado *makeWinnowing*. Este método realiza lo necesario para devolver el certificado al Servicio Web.

Gracias a Spring, la invocación de este método se hace mediante un @RequestMapping y se leen los datos con @RequestBody, por lo que desde el servicio web sólo es necesario mandar un modelo de datos llamado WinnowingModel a la dirección especificada con el modelo de datos como el cuerpo de la petición HTTP.

El código ?? muestra el mapeo y el método implementado.

El paquete TT2018B003.comp3.API.Utils contiene 4 clases. En las clases *CipherUtilityAES.java*, *CipherUtilityRSA.java* y *CipherUtilitySHA256* se utilizó Java Security. Java Security proporciona las clases e interfaces para el framework de seguridad. Esto incluye clases que implementan una arquitectura de seguridad de control de acceso de fácilmente configurable. Este paquete también admite la generación y el almacenamiento de pares de claves públicas criptográficas, así como una serie de operaciones criptográficas exportables. Finalmente, este paquete proporciona clases que admiten objetos firmados o protegidos y generación segura de números aleatorios [59].

La clase *Base64u.java* es una implementación de la decodificación de Base64 realizada por nosotros para este Proyecto.

Finalmente, la clase *SignatureUtility* es una clase creada para poder recibir un certificado formato X509 y poder determinar si la firma es válida o no, utilizando *X509Certificate.java*.

El paquete TT2018B003.comp3.API.Service contiene 2 servicios y una interfaz. La clase *WinnowingService.java* implementa la interfaz *IWin-*

nowing. Dicha clase hace uso a su vez de la clase *CryptoService.java* encargada de realizar el decifrado AES y RSA y el cifrado SHA256, apoyándose de las clases del paquete Utils.

El código ?? muestra el contenido de esta clase para realizar el winnowing.

5.3.2. Servicio Web.

Para este trabajo terminal, utilizamos un sitio web de prueba desarrollado con *Spring Framework 5*, *Java 8* y montado sobre un servidor *Pivotal Server Developer Edition v4.0*.

Para este proyecto partimos bajo la premisa de que el servicio web está desarrollado, por lo que sólo se mostrarán los cambios necesarios para la integración de la API.

5.3.2.1. Modificaciones al inicio de sesión.

Para la modificación del servicio web, se ubicó el método encargado de desplegar la vista para el inicio de sesión, es decir, el formulario. Dicho método se halló en la clase *LoginController.java*. El código ??, muestra el contenido original en el cual se aprecia que en cuando solicita el recurso '*/login*' retorna una vista llamada *login*, la cual es un formulario.

En el código ?? se muestran todos los cambios realizados en el método *login()* para implementar el uso de este método de autentificación propuesto. Entre los elementos más destacables se encuentra la implementación de la anotación *@RequestHeader* perteneciente a *Spring Framework*. Esta anotación es capaz de obtener el header con el nombre especificado de la petición HTTP que acaba de recibir el servidor, es por ello que implementamos su uso para obtener el encabezado *Chaffing* y *Pattern* para poder realizar la etapa de *Winnowing*. Además, se hizo uso de *RestTemplate* para poder realizar la conexión con la API y poder obtener el certificado y la lógica de negocio correspondiente.

Por otro lado, también fue necesario modificar aquel método encargado de procesar el inicio de sesión por formulario. Este método se encuentra en la misma clase que el método login mostrado anteriormente, dicha clase es *LoginController.java*.

El código ?? muestra el método *loginByForm* original del servicio web, donde se aprecia que mapea la dirección '*/loginByForm*' y recibe los parámetros del formulario.

En el código ?? se muestran las modificaciones que se realizaron al método para poder implementar este método de autentificación propuesto. Entre los cambios más significativos se encuentra la operación *saveCertificate* la cual

se encarga de guardar el certificado para el usuario correspondiente en la base de datos.

Después de implementar estas modificaciones al código fuente del servicio web, éste puede hacer uso del método que proponemos para iniciar sesión automáticamente de manera segura y rápida.

Capítulo 6

Pruebas.

Las pruebas son un factor importante para comprobar la eficiencia y correcto funcionamiento de nuestro sistema. Las pruebas que aquí se presentan fueron realizadas con los equipos de la sección de Hardware en Herramientas a Usar, sobre una red local creada con un celular Huawei P20, cuya velocidad de conexión es de 76Mb/s, con una intensidad de señal catalogada como 'Excelente', es decir, los equipos estaban dentro de un radio de 3m del punto de conexión (Huawei P20).

6.1. Pruebas de integración.

Estas pruebas nos sirven para comprobar que los componentes de nuestro sistema estén interactuando de forma correcta entre ellos. Como sabemos, nuestro sistema en general se compone principalmente de 3 componentes, para este primer prototipo existe una comunicación entre el primer componente (Extensión) y el segundo componente (Servidor Autentificador), además de una comunicación entre el primer componente y el tercero (API).

6.1.1. Extensión y Servidor autentificador.

Vamos a empezar con la interacción entre el cliente y el servidor autentificador, uno de los casos en el que estos dos componentes interactúan es cuando el usuario inicia sesión en la extensión, aquí el Servidor Autentificador verifica si el usuario está registrado y si este es el caso, entonces le debe de devolver el certificado junto con un código diciendo que el usuario está registrado dentro del servidor.

Figura 6.1: Sesión iniciada

Si aparecen estos mensajes, quiere decir que la extensión se pudo comunicar correctamente con el servidor autentificador, y le envió correctamente el certificado correspondiente a este usuario, que se crea cuando se registra en el servidor.

6.1.2. Extensión y Servicio Web.

Ahora vamos a realizar las pruebas entre el componente de la extensión con el componente del Servicio Web, una vez que el usuario inicia sesión y tenga su certificado listo, vamos a ingresar al servicio web de prueba, con ello vamos a comprobar que la comunicación entre estos componentes esté funcionando. Para realizar estas pruebas funcionales vamos a utilizar un sniffer e interceptar la información cuando estas dos se comunican:

Figura 6.2: Resultados de la API al recibir petición de la extensión.

Aquí se muestra una salida de los datos con los que interactua la API del Servicio Web con la extensión, la respuesta en la que la API si encuentra al usuario registrado, podemos ver que entre la información se encuentra que el certificado recibido es un certificado válido.

6.1.3. Servidor Autentificador y API.

Cuando el usuario intenta iniciar una sesión en el servicio web, entonces este se debe de comunicar con el Servidor Autentificador para corroborar que el certificado que le ha llegado, vamos a utilizar un sniffer para comprobar que la información que le está llegando a uno de los dos servidores sea el correcto:

Figura 6.3: Resultados de la comunicación entre la API y el Servidor Autentificador.

6.2. Tiempos de ejecución.

En esta sección mostraremos el tiempo de ejecución de los algoritmos desarrollados para el Prototipo 1. Todas las pruebas fueron realizadas en

igualdad de condiciones, es decir, la computadora recién prendida (sin ningún programa abierto) y sólo ejecutando el algoritmo.

6.2.1. Algoritmo de Chaffing.

Se realizó una prueba al algoritmo de *Chaffing*, el cual incluye: creación de patrón, proceso de chaffing y cifrado de patrón. Para esta prueba se utilizó una función de JavaScript llamada *performance.now()*, la cual mide el tiempo con una precisión de milisegundos.

Tiempo de ejecución: 434.2925ms.

6.2.2. Algoritmo de Winnowing.

Se realizó una pruebaa al algoritmo de *Winnowing*, el cual incluye: des-cifrado del patrón y proceso de winnowing. para esta prueba se utilizó una función de Java llamada *System.currentTimeMillis()*, la cual mide el tiempo con una precisión de milisegundos.

Tiempo de ejecución: 53.99 ms.

6.2.3. Inicio de sesión.

Se realizaron un total de 100 pruebas del inicio de sesión por medio de este Trabajo Terminal. Este proceso incluye el tiempo que transcurre desde que el usuario da click en el botón de iniciar sesión en el Servicio Web, es decir, la interceptación de la petición por parte del Componente I) hasta la impresión de la respuesta del Servicio Web, es decir, el inicio de sesión completado exitosamente mostrando la pantalla de inicio del Servicio Web. Para medir el tiempo, se utilizó la función de JavaScript llamada *performance.now()*, al igual que en la medición del algoritmo de Chaffing, puesto que esta medición se realiza desde el Componente I. Extensión.

El tiempo que se muestra a continuación es el promedio de 100 inicios de sesión.

Tiempo promedio de ejecución: 1101.68 ms.

Trabajo Terminal II

Sprint 2.

Capítulo 7

Análisis.

7.1. Arquitectura del sistema.

Figura 7.1: Arquitectura General del Sistema

7.1.1. Descripción de la arquitectura del sistema.

El sistema seguirá manteniendo la misma arquitectura del prototipo 1, debido a que los cambios para el prototipo 2 son cambios internos que no afectan el flujo de la información.

Los cambios por componentes son los siguientes:

1. **Navegador Chrome con la Extensión instalada:** En este bloque la modificación que se lleva a cabo es en el proceso de *Chaffing* el cual ahora se inyectara *Chaffing* ó bien basura en el certificado. Recordemos que en el prototipo 1 se inyectaba el certificado en el apartado de *Accept* el cual ocupaba como *Chaffing* el valor del header *Accept*.
2. **Servidor autentificador:** Este componente tendrá pocas modificaciones en el aspecto de seguridad. Ya que en la base de datos del componente se guarda la ruta donde esta el certificado de cada usuario, por lo que, se necesitará tener un control de accesos para evitar cualquier robo de certificados. Éste control de accesos por FTP, sólo le dará acceso a la autoridad certificadora, para que la misma pueda obtener los certificados de los usuarios y mandárselos cuando lo necesiten.
3. **Servidor web con API instalada:** Las modificaciones para este componente son en el desarrollo del *Winnowing*, debido a que el componente

1 tiene cambios en el *Chaffing* por lo tanto, el proceso de *Winnowing* debe ser modificado también.

7.2. Componente I. Extensión.

7.2.1. Descripción.

Como sabemos, el componente 1 será el encargado de interceptar las peticiones de los usuarios para inyectar el *Chaffing* y enviar la petición modificada.

La modificación que tendrá el componente 1 es en el aspecto de seguridad, el cual se modificará en el patrón del *Chaffing* para tener un patrón repetido, y así tener menos caracteres en el mismo y por lo tanto, tener menos caracteres en el patrón.

En el Prototipo 1 por tener una gran cantidad de caracteres en el *Chaffing* se necesitaba cifrar con AES y después la llave cifrarla con RSA por lo que representaba un gasto computacional extra que podría ser eliminado. Para este prototipo se pretende reducir el patrón en tamaño de tal manera que se pueda repetir siguiendo el comportamiento de una *lista circular enlazada*.

7.2.2. Cambios planteados.

- **Eliminación del atributo usuario:** En este prototipo ya no se contará con un nombre de usuario ya que con el email y la contraseña del mismo eran suficientes para que el componente funcionara correctamente.
- **Chaffing sin caracteres inválidos:** Debido a la nueva implementación del Chaffing se consideró desde un inicio generar solamente caracteres imprimibles por lo que la aplicación del *Base 64* ya no será necesaria.
- **Reducción del tamaño del patrón:** Uno de los problemas que pudimos detectar en el primer componente fue que el patrón tenía la misma longitud del chaffing lo que representaba un envío de información un tanto grande a lo que planteamos modificar el algoritmo de generación de patrón así como el de generación del chaffing permitiendo reducir la cantidad de datos enviados manteniendo la confidencialidad que en un principio se planteó.

- **Eliminación del cifrado AES:** Como consecuencia del punto anterior se considerará dejar de un tamaño definido la longitud del patrón, logrando de esta manera que solo sea necesario cifrarlo mediante RSA sin la necesidad de AES con lo que se busca reducir el costo computacional tanto en este componente como en los que dependan del mismo.

7.2.3. Cambios al estudio de requerimientos y reglas de negocio.

Para este componente los cambios son un tanto mínimos los cuales se enlistan a continuación:

Requerimientos funcionales En cuanto a los requerimientos funcionales que se modificaron nos encontramos con el requerimiento funcional **CI RF13. Registro en servidor autentificador.** de la sección 3.6.2.1 en el cual se hace mención de un nombre de usuario que para este prototipo no es necesario por lo que los datos a ingresar serán: email y contraseña.

Requerimientos no funcionales Debido a que en este prototipo ya se consideran enviar caracteres válidos en las peticiones HTTP, el requerimiento no funcional **CI RNF4. Codificación en base64 del Chaffing.** de la sección 3.6.2.2 ya no será necesario.

Reglas del negocio Como consecuencia de la eliminación del atributo *nombre de usuario* igualmente ya no son necesarias las siguientes reglas del negocio: **CI RN7. Longitud del campo de usuario.** y **CI RN9. Caracteres permitidos en campo usuario.** de la sección 3.6.3.

El resto de los requerimientos y reglas del negocio quedan de la misma manera.

7.3. Componente II: Servidor autentificador.

7.3.1. Descripción.

Este componente es el encargado de guardar los certificados del usuario, para mandarlos a sus respectivos usuarios (siempre que los usuarios lo necesiten). Por lo que es importante que los certificados por ningún motivo estén vulnerables. Dando así lugar a la creación de un control de accesos por FTP para que la autoridad certificadora sólo tenga acceso a estos archivos

(Certificados) de los usuarios. Este control de accesos evitara que atacantes quienes quieran obtener el certificado de los usuarios, estos no podrán debido al control de accesos. Dando mayor seguridad a los usuarios que hagan uso de la extensión (Componente I).

7.3.2. Cambios Planteados.

- **Control de accesos por FTP:** En este prototipo se creara un servidor FTP la cual solamente la autoridad certificadora tendrá acceso, debido a que en este servidor la autoridad almacenará los certificados de los usuarios. Donde, en la base de datos la autoridad certificadora solamente tendrá almacenados los datos del usuario como lo son:

- Correo electrónico (email)
- Contraseña

7.3.3. Cambios al estudio de requerimientos y reglas de negocio.

Los requerimientos funcionales de este componente para este prototipo deberán cambiar en el aspecto de la creación de certificados. En el prototipo 1 los requerimientos funcionales hacían alusión en crear los certificados por cada usuario dentro del servidor (Autoridad certificadora).

Puesto que esto deja una vulnerabilidad en la autoridad certificadora, todos los requerimientos funcionales deberán:

- Generar
- Guardar
- Revocar
- Actualizar
- Comprobar

Todo lo anterior, desde los certificados almacenados en el servidor FTP y ya no desde los certificados almacenados en la autoridad certificadora. Como requerimiento no funcional se agrega:

CII_RNF5. Servidor de acceso. la autoridad certificadora deberá tener acceso y privilegios en el servidor FTP donde se almacenarán los certificados.

7.4. Componente III: API.

7.4.1. Descripción.

Para este prototipo, hemos planteado la modificación del proceso de *Winnowing* que realiza la API, debido a que el patrón y por tanto el proceso de *Chaffing* en el Componente I han cambiado. Además, como una mejora en la seguridad, nos hemos planteado que el Servicio Web no tenga almacenado en su base de datos el certificado del usuario. De esta manera la única entidad que almacenará los certificados de los usuarios será el Componente II. El resto de la funcionalidad se mantendrá tal y como estaba en el Prototipo 1.

7.4.2. Cambios planteados.

- **Nuevo proceso de Winnowing:** debido a los cambios realizados para el Prototipo 2 al Componente I, es necesario crear otro algoritmo capaz de obtener el certificado del header de la petición HTTP que el usuario mando. Este algoritmo se adaptará al nuevo patrón teniendo en cuenta que su tamaño ahora es menor y buscando el óptimo desempeño. En la sección de diseño se puede apreciar las cambios al algoritmo.
- **Eliminación del descifrado AES:** el patrón ya no se descifrará con AES puesto que en el Componente I se eliminó ese cifrado para mejorar la velocidad del algoritmo, quedando sólo el cifrado RSA.
- **Guardado del certificado:** buscando evitar que el servicio web guarde el certificado en su base de datos, hemos planteado que la API, una vez realizado el proceso de winnowing y la validación del certificado regrese un código SHA256 del mismo. De esta manera, el servicio web sólo guardará en su base de datos un código el cual, gracias a la función SHA, no puede ser revertido en caso de que un atacante quisiera obtener el certificado. Una vez realizado los cambios, el certificado sólo se almacenará en el Componente II.

7.4.3. Cambios al estudio de requerimientos y reglas de negocio.

El requerimiento funcional **CIII_RF10. Retorno del certificado** de la sección 3.8.2.1, explica que la API deberá retornar el certificado y su estatus al servicio web. Este requerimiento corregido queda de la siguiente manera.

CIII_RF10. Retorno de certificado. La API deberá retornar el código SHA256 del certificado y su estatus al servicio para que este pueda realizar la lógica del negocio necesaria para el inicio de sesión.

Finalmente, el requerimiento funcional **CIII_RF11. Descifrado AES al patrón** de la sección 3.8.2.1 se eliminará como consecuencia del cambios planteado en la sección anterior.

El resto de los requerimientos y reglas del negocio quedan de la misma manera.

Capítulo 8

Diseño.

8.1. Componente I: Extensión.

8.1.1. Cambios en el algoritmo.

Los cambios planteados son muy parecidos entre si ya que uno es consecuencia del otro por lo que la parte que se modificó recae completamente en el algoritmo de este componente, resultando como se muestra a continuación:
Algoritmo utilizado para la generación del patrón de chaffing:

```
Data: low, high  
Result: pattern,ones  
1 lenPattern  $\leftarrow$  150 * 8;  
2 ones  $\leftarrow$  rand(low, high);  
3 pattern[lenPattern]  $\leftarrow$  0;  
4 i  $\leftarrow$  0;  
5 while i < ones do  
6   | x  $\leftarrow$  securerand(size);  
7   | if pattern[x] != 1 then  
8   |   | pattern[x]  $\leftarrow$  1;  
9   |   | i  $\leftarrow$  i + 1;  
10  | end  
11 end
```

Algoritmo 5: *getPattern*: Generación de patrón de chaffing del Prototipo 2

Consideramos este algoritmo muy simple ya que es bastante intuitivo analizar su comportamiento, el algoritmo inicia definiendo la longitud del patrón para luego calcular de manera aleatoria la cantidad de unos que contendrá, esto en un rango definido por nosotros que más adelante se explicará, se ini-

cializa el patrón con ceros para mas tarde iniciar con un ciclo que con base en posiciones aleatorias obtendrá una posición el la cual se colocará un 1 terminando en el momento que todos los unos calculados se encuentren en el patrón.

Algoritmo utilizado para la generación del Chaffing:

```

Data: Cert
Result: Chaffing,pattern
1 lenCert  $\leftarrow$  length(Cert);
2 pattern,ones  $\leftarrow$  getPattern(550,650);
3 Chaffing  $\leftarrow$  " ";
4 rep  $\leftarrow$  roof(lenCert/ones);
5 i  $\leftarrow$  0;
6 k  $\leftarrow$  0;
7 while i < rep do
8   foreach j in pattern do
9     if j ==' 1' then
10      Chaffing  $\leftarrow$  Chaffing + Cert[k];
11      k  $\leftarrow$  k + 1;
12      ones  $\leftarrow$  ones - 1;
13    else
14      Chaffing  $\leftarrow$  Chaffing + fake();
15    end
16    if ones == 0 then
17      break;
18    end
19  end
20  i  $\leftarrow$  i + 1;
21 end
```

Algoritmo 6: getChaff: Generación de chaffing del prototipo 2

En este caso el algoritmo que genera gran parte de los datos que se incluirán en la cabecera *HTTP* inicia calculando la longitud del certificado original para luego mandar a llamar a la función *getPattern(int, int)* que calcula el patrón como se explicó previamente; esta función recibe dos enteros como entrada los cuales estáticamente definimos como 150 bytes necesarios para determinar el números de unos con los que contará el patrón, esto debido a que la longitud máxima de RSA es de 254 bytes, sin embargo requiere de ciertos bytes extras que utiliza para poder realizar el cifrado y descifrado de manera correcta, por lo que a nuestro sistema se ajusto a esta cantidad de 150 bytes.

8.1.1.1. Complejidad computacional.

Haciendo un análisis de los algoritmos anteriormente mostrados, deducimos que la complejidad del algoritmo de *chaffing* del prototipo 2 es: $O(n)$ donde n es el

8.2. Componente II: Servidor Autentificador.

8.2.1. Cambios realizados.

Debido a las reglas del negocio que se tuvieron que ajustar, también se realizaron unos cambios en alguno de los diagramas, a continuación se mostrarán los cambios realizados.

8.2.2. Diagrama de Actividades.

Para el Diagrama de Actividades, no se realizaron cambios drásticos en la lógica de su diseño, sin embargo se agregó el servidor FTP y con esto tener un control de acceso para que solo puedan acceder usuarios permitidos, y con ello este diagrama se ajustó a como sigue:

Figura 8.1: Ajustes al Diagrama de Actividades del Componente II.

8.2.3. Diagrama de Secuencia.

8.2.3.1. Diagrama de secuencia: Crear nuevo usuario.

Se ajustó para que los certificados se almacenaran en el servidor de acceso FTP.

Figura 8.2: Ajustes al Diagrama de Secuencia crear nuevo usuario del componente II.

8.2.3.2. Diagrama de secuencia: Revocar certificado.

Se realizaron cambios para que cuando se revoque el certificado, todas las validaciones ahora se contemplen para acceder en el servidor de acceso FTP.

Figura 8.3: Ajustes al Diagrama de Secuencia revocar certificado del componente II.

8.2.3.3. Diagrama de secuencia: Obtener certificado.

Al momento de que la extensión requiera del certificado, este componente tendrá que acceder al servidor de acceso FTP para obtener y devolverle el certificado.

Figura 8.4: Ajustes al Diagrama de Secuencia obtener certificado del componente II.

8.2.3.4. Diagrama de secuencia: Verificar certificado.

Nuevamente se realizaron cambios para ajustar la procedencia del certificado para que se obtenga del servidor de acceso FTP al momento en que la API requiera buscar a un usuario en el Servidor Autentificador. Autentificador.

Figura 8.5: Ajustes al Diagrama de Secuencia verificar certificado del componente II.

8.3. Componente III: API.

8.3.1. Cambios realizados.

Para estos cambios se eliminó la lógica en la que el mismo Servicio Web guarda los certificados con los que le da acceso a los usuarios en una base de datos, en cambio se sustituyó con una comunicación con el componente dos, Servidor Autentificador, para que le pregunte únicamente si el usuario se encuentra registrado, y le devuelva un hash del certificado para compararlo con el que recibió por parte de la Extensión, esto con el fin de saber si existe el usuario y su certificado está actualizado o si requiere de actualizarlo por alguna revocación.

8.3.2. Cambio en el algoritmo.

Debido a que se realizaron cambios en la etapa de *Chaffing* del componente 1, es necesario ajustar igualmente la forma en la que se realiza el Winnowing para este componente, ahora el algoritmo que realiza la obtención del patrón

y la etapa de *Winnowing* se explica a continuación.

Data:	chaffing, patternCipher, privateKey, sizeCert
Result:	cert,header

```

1 pattern[] ← rsa.decrypt(patternCipher, privateKey);
2 cert ← "";
3 rep ← sizeCert/numOnes(pattern);
4 contRep ← 0;
5 while i < rep do
6   while j < pattern.length do
7     if cert.length == sizeCert then
8       | break;
9     end
10    if patt[j] == 1 then
11      | cert+ = chaffing[i + (pattern.length * contRep)];
12    end
13    | j ← j + 1;
14  end
15  | i ← i + 1;
16 end

```

Algoritmo 7: Cambios en el algoritmo de Winnowing para obtener el certificado.

Para este prototipo, ahora vamos a tener un el patrón de chaffing cifrado únicamente mediante RSA, y este patrón será de un tamaño aproximado de 150 bytes, para realizar este algoritmo se requiere del chaffing con el que se va a trabajar, el patrón cifrado con RSA, la llave privada para descifrar el patrón y el tamaño en bytes del certificado.

Primero se realiza el descifrado de este patrón mediante RSA, una vez que tenemos descifrado el patrón, vamos a proceder a realizar la etapa de Winnowing. Necesitamos saber cuántas veces necesitamos recorrer este patrón, por lo cual se obtiene en una variable dividiendo el tamaño del certificado, mencionado anteriormente, y el número de unos que contiene el patrón, ahora recorremos el patrón en busca de un uno, cuando lo encontramos quiere decir que este es un carácter válido para tomar como carácter del certificado, por lo que lo agregamos a nuestra variable del certificado, y repetiremos esto tantas veces como el resultado de la división anteriormente mencionada.

Al finalizar esta etapa de *Winnowing*, nos quedamos con el certificado que fue enviado en la petición, así podremos proseguir con la lógica de nuestro sistema, que es comunicarnos con el Servidor Autentificador, para verificar

con este mismo si el certificado es uno válido para nuestro inicio de sesión.

8.3.2.1. Complejidad computacional.

Haciendo un análisis del algoritmo anteriormente mostrado, deducimos que la complejidad del algoritmo de *winnowing* del prototipo 2 es: $O(n)$

8.3.3. Diagrama de Flujo.

Figura 8.6: Ajustes al Diagrama de Flujo Componente III.

Se realizaron unos cambios para este diagrama, se agregó un flujo en el cuál se muestra la forma en el que la API se comunica con el Servidor Autentificador para saber la validez del certificado, primero le envía el Hash del usuario que se puede obtener con el certificado, el Servidor Autentificador lo busca y le envía el hash de este certificado, después la API transforma el primer certificado que recibió mediante la petición y lo compara con el Hash que recibe del Servidor Autentificador, si son iguales entonces podemos dar acceso al usuario que envío esta petición, sin embargo si la comparación resulta diferente, entonces quiere decir que el usuario no tiene un certificado válido o actualizado, por lo que debe de volver a iniciar sesión en el Servicio Web para que pueda reasignarle correctamente su certificado válido a esta cuenta.

Capítulo 9

Desarrollo.

9.1. Componente I: Extensión.

Los cambios planteados y diseñados en las secciones Análisis y Diseño del Prototipo 2 tuvieron consecuencias en el desarrollo del Componente I. En esta sección se mostrarán los cambios hechos para la implementación del Prototipo 2.

9.1.1. Cambios en el manifest.

Para este segundo prototipo hemos hecho cambios en el archivo manifest de la extensión. Como se puede notar en el código ?? la versión de la extensión cambió de 1.0 a 2.0, con el objetivo de tener un control de versiones y se removió el uso de la librería *CryptoJS* que usabamos para cifrar con AES, ya que en este prototipo removimos ese método de cifrado.

9.1.2. Implementación del Chaffing.

Los cambios planteados en este prototipo, repercuten por sobre todas las cosas en la implementación del algoritmo de *Chaffing*. El código ?? muestra el proceso de chaffing, invocando a las funciones *getPattern* del código ?? para obtener el patrón del chaffing y *makeChaffing* el código ?? para realizar el chaffing. Además en la línea 11 del código ?? se muestra que ahora sólo se cifra con RSA el patrón de chaffing.

Con el código mostrado anteriormente, ahora el Componente I es capaz de cumplir con los cambios propuestos y las ventajas que ellos lograron.

9.2. Componente II: Servidor Autentificador.

Los cambios planteados en Análisis y Diseño del prototipo 2, tuvieron como consecuencia modificaciones en las diferentes funciones del componente 2. En esta sección se mostrarán los cambios realizados para tener la funcionalidad requerida del prototipo 2.

9.2.1. Servidor de accesos FTP.

Debido a que en el prototipo 1, la autoridad certificadora (Componente 2) queda vulnerable a posibles ataques para la obtención de certificados guardados de los usuarios. Este prototipo proporciona a la autoridad un servidor de accesos por FTP, en el que éste le dará únicamente acceso a la autoridad certificadora, para que así ningún otro usuario pueda obtener los certificados guardados. La autoridad certificadora, sólo necesitará tener almacenado el correo email y contraseña del usuario en su base de datos, por lo que la misma no necesita estar protegida ante posibles ataques. El paquete **ftp** para NodeJs nos ayudará a llevar a cabo esta conexión.

- **ftp** (versión 0.3.10): Este paquete nos ayuda a establecer conexión a un servidor ftp con funciones preestablecidas de la librería, por lo que en NodeJs solo bastará con llamar a las funciones put o get para obtener el archivo o guardar un archivo según sea el caso.

El código ?? muestra la implementación necesaria para poder conectar al servidor FTP y obtener certificados del mismo. En este caso el usuario es **diegoarturomg** y contraseña **211096** para acceder al servidor FTP.

Esta implementación se llevará a cabo tanto en obtener certificado como en verificar certificado. Debido a que son funciones get de FTP.

El código ?? muestra la implementación necesaria para poder conectar al servidor FTP y guardar certificados en el mismo. El usuario y contraseña son los mismos.

Esta implementación se llevará a cabo tanto en guardar usuario como en revocar certificado. Debido a que son funciones get de FTP.

9.3. Componente III: API.

9.3.1. Nuevo Proceso de Winnowing y eliminación del descifrado AES.

Debido a los cambios realizados con el patrón en el Componente I es necesario realizar la adaptación pertinente a este proceso considerando que la parte encargada de este proceso involucra completamente a la API, por lo tanto el código del proceso de winnowing quedó como se muestra en el siguiente fragmento de código:

En donde nos podemos percatar de la eliminación por un lado del decode en base 64 y del descifrado del patrón mediante AES y por otro de la nueva implementación del patrón y el ajuste para obtener el mismo resultado eliminando algunos pasos.

9.3.2. Guardado del certificado

Algo muy importante a considerar en este prototipo es que de ahora en adelante el único que almacena el certificado del usuario es el mismo usuario en su extensión y el servidor autenticador, por lo que el servicio web solo pasa a recibir un código SHA256 del mismo certificado, cuidando siempre la confidencialidad de este dato. Esta implementación de manera similar a la anterior se realiza en la API en estas simples lineas de código:

```
1 ...
2 String shaCert = cryptoService.doSHA(certificate);
3 ...
4 return shaCert + " 1";
5
```

Código 9.1: Salida de la función makeWinnowing del Prototipo 2

Es importante aclarar que esta salida solo se obtiene mediante el mismo proceso de validación que se mencionó en el Prototipo 1 de este Componente, de la misma manera se sigue regresando el status como se había analizado desde un inicio.

Capítulo 10

Pruebas.

Las pruebas que aquí se presentan fueron realizadas con los equipos de la sección de Hardware en Herramientas a Usar, sobre una red local creada con un celular Huawei P20, cuya velocidad de conexión es de 76Mb/s, con una intensidad de señal catalogada como 'Excelente', es decir, los equipos estaban dentro de un radio de 3m del punto de conexión (Huawei P20).

10.1. Pruebas de integración.

En esta sección se mostrará las pruebas de comunicación entre los distintos componentes, comprobando así que los datos enviados entre uno y otro sean los indicados para llevar a cabo un proceso de *Chaffing* y *Winnowing* correctamente, y por ende un inicio de sesión correcto.

10.1.1. Extensión y Servidor autentificador

La comunicación entre el Componente I y el Componente II se establece por medio de una conexión SSL, por que los tanto los datos de inicio de sesión como el certificado viajan de forma segura. La figura 10.1 muestra los datos recibidos por el componente 1, el cual es el certificado. Cabe destacar que los datos que se muestran están planos ya que los datos se descifran y se muestran en el apartado *Network* de Google Chrome.

Figura 10.1: Respuesta de la Autoridad Certificadora hacia la Extensión

Por otra parte al ver el mensaje de la figura 10.2 se puede decir que la conexión se estableció satisfactoriamente y se obtuvo el certificado.

Figura 10.2: Sesión iniciada con éxito (Obtención de certificado de la Autoridad certificadora.)

10.1.2. Extensión y Servicio Web

La comunicación entre el Componente I y Componente III es donde viajará el *Chaffing*, esta parte es importante debido a que el Chaffing viaja en el encabezado de la petición, y para saber que esta viajando en red hicimos uso del analizador de protocolos **Wireshark**. La figura 10.3 muestra la petición en wireshark donde se puede ver el apartado de **Chaffing** y el apartado de **Pattern**.

Figura 10.3: Analizador de protocolos Wireshark de la petición dada por el usuario hacia servicio web.)

10.1.3. Servidor Autentificador y API

La comunicación entre el Componente II y Componente III sirve para verificar el certificado, debido a que la API (después de obtener el Winnowing) hace una petición a la autoridad certificadora para verificar si el certificado que obtuvo aún no ha sido revocado. En la figura 10.4 muestra desde el analizador de protocolos *Wireshark* la respuesta que le manda la autoridad certificadora a la API.

Figura 10.4: Analizador de protocolos Wireshark de la petición dada por la API hacia la autoridad certificadora.)

10.2. Pruebas funcionales.

Realizamos pruebas del funcionamiento de nuestro prototipo 2, para asegurarnos que sean correctas las entradas y salidas de nuestro sistema para este prototipo. Empezaremos con la parte del usuario, donde comprobaremos que la extensión está recibiendo el certificado del usuario de manera correcta al iniciar sesión.

Figura 10.5: Inicio de sesión en la extensión

Figura 10.6: Certificado del cliente devuelto por el Servidor Autentificador

Si el usuario logra obtener su certificado, entonces puede proceder a utilizar la extensión, vamos ahora a ingresar a nuestro Servicio Web de prueba, la página de la veterinaria. Si es la primera vez que accedemos, entonces debemos de registrarnos para crear nuestra cuenta de usuario en este servicio, y después iniciar sesión de manera rutinaria con nuestras credenciales, esto para que la extensión pueda asociar el certificado de la extensión de este usuario a esta cuenta, y así cuando vuelva a acceder a este mismo Servicio Web, nuestro sistema le dará acceso.

Figura 10.7: Página de inicio del Servicio Web

Figura 10.8: Inicio de sesión con la extensión habilitada.

Como podemos observar, nuestro inicio de sesión en la extensión nos pide que iniciemos sesión para vincular las credenciales, si este mensaje aparece quiere decir que todo va bien, porque la extensión detectó que es la primera vez que se inicia sesión en el Servicio Web, por lo que requiere de tu inicio de sesión para poder asignarle este usuario al certificado que va a ir inyectado con *Chaffing* en la petición al servidor.

Cuando ingresemos nuestras credenciales, vamos a probar cerrando sesión en el Servicio Web y volviendo a entrar a la misma:

Figura 10.9: Inicio de sesión exitoso a la cuenta con el certificado asignado.

Como podemos darnos cuenta, al querer volver a iniciar sesión nos da acceso de inmediato a nuestra cuenta para este Servicio Web, debido a que anteriormente se había asignado el mismo certificado a la cuenta que se le dio acceso.

10.3. Tiempos de ejecución.

En esta sección mostraremos el tiempo de ejecución de los algoritmos desarrollados para el Prototipo 2, así como la comparación con el Prototipo 1

para demostrar la mejoría que hubo. Todas las pruebas fueron realizadas en igualdad de condiciones, es decir, la computadora recién prendida (sin ningún programa abierto) y sólo ejecutando el algoritmo.

10.3.1. Algoritmo de Chaffing.

Se realizó una prueba al algoritmo de *Chaffing*, el cual incluye: creación de patrón, proceso de chaffing y cifrado de patrón, todo del prototipo 2. Para la prueba se utilizó una función de JavaScript llamada *performance.now()*, la cual mide el tiempo con una precisión de milisegundos.

Tiempo de ejecución: 17.43 ms.

Si comparamos el tiempo de ejecución del algoritmo de *Chaffing* del Prototipo 1, el cual era de 434.2925 ms, podemos ver que la mejoría fue de 416.8625ms gracias, principalmente, a la disminución del tamaño del patrón (que afecta directamente a la manera en que se crea y en que se recorre para hacer el Chaffing) y a la eliminación del cifrado por bloques AES.

10.3.2. Algoritmo de Winnowing.

Se realizó una prueba al algoritmo de *Winnowing*, el cual incluye: descifrado del patrón y proceso de winnowing. Para la prueba se utilizó una función de Java llamada *System.currentTimeMillis()*, la cual mide el tiempo con una precisión de milisegundos.

Tiempo de ejecución: 12.46 ms.

Si comparamos el tiempo de ejecución del algoritmo de *Winnowing* del Prototipo 1, el cual era de 53.99ms, podemos ver que la mejoría fue de 41.53ms gracias a la disminución del tamaño del patrón y a la eliminación del descifrado por bloques AES.

10.3.3. Inicio de sesión.

Se realizaron un total de 100 pruebas del inicio de sesión por medio de este Trabajo Terminal. Este proceso incluye el tiempo que transcurre desde que

el usuario da click en el botón de iniciar sesión en el Servicio Web, es decir, la interceptación de la petición por parte del Componente I) hasta la impresión de la respuesta del Servicio Web, es decir, el inicio de sesión completado exitosamente mostrando la pantalla de inicio del Servicio Web. Para medir el tiempo, se utilizó la función de JavaScript llamada *performance.now()*, al igual que en la medición del algoritmo de Chaffing, puesto que esta medición se realiza desde el Componente I. Extensión.

El tiempo que se muestra a continuación es el promedio de 100 inicios de sesión.

Tiempo promedio de ejecución: 335.28 ms.

Si comparamos el tiempo de inicio de sesión del Prototipo 1, el cual era de 1101.68ms, podemos ver que la mejoría fue de 766.4ms con lo cual se comprueba que el Prototipo 2 es una mejora sustancial al sistema desarrollado en este trabajo sin la pérdida de seguridad.

Capítulo 11

Conclusiones

Capítulo 12

Trabajo a futuro.

Bibliografía

- [1] Real Academia Española (2020, noviembre), Diccionario de la lengua española, [En línea]. Disponible: <https://dle.rae.es> [Último acceso: 10 de diciembre del 2020].
- [2] Oxford Lexico (2020, noviembre), Definitions, Meanings, Synonyms, and Grammar by Oxford, [En línea]. Disponible: <https://www.lexico.com> [Último acceso: 2 de diciembre del 2020].
- [3] O. Jaramillo, Universidad Nacional Autonoma de México (2007, mayo 03), El concepto de Sistema, [En línea]. Disponible: <https://www.ier.unam.mx/ojs/pub/Termodinamica/node9.html> [Último acceso: 2 de diciembre del 2020].
- [4] J. A. Gutiérrez Orozco, Escuela Superior de Cómputo (2008, septiembre 15), Máquinas de Estados Finitos, [En línea]. Disponible: http://delta.cs.cinvestav.mx/mcintosh/cellularautomata/Summer_Research_files/maquinasef.pdf [Último acceso: 15 de diciembre del 2020].
- [5] L. Hardesty (2017, abril), Explained: Neural networks, [En línea]. Disponible: <https://news.mit.edu/2017/explained-neural-networks-deep-learning-0414> [Último acceso: 15 de noviembre del 2020].
- [6] S. Leijnen, F. van Veen (2020, mayo), The Neural Network Zoo, [En línea]. Disponible: https://www.researchgate.net/publication/341373030_The_Neural_Network_Zoo [Último acceso: 20 de noviembre del 2020].
- [7] A. Mehta (2019, enero 25), A Comprehensive Guide to Types of Neural Networks, [En línea]. Disponible: <https://www.digitalvidya.com/blog/types-of-neural-networks/> [Último acceso: 15 de noviembre del 2020].

- [8] P. Shukla, R. Iriondo (2020, agosto 11), Main Types of Neural Networks and its Applications, [En línea]. Disponible: <https://medium.com/towards-artificial-intelligence/main-types-of-neural-networks-and-its-applications-tutorial-734480d7ec8e> [Último acceso: 20 de noviembre del 2020].
- [9] Oracle México (2020, noviembre), ¿Qué es una base de datos?, [En línea]. Disponible: <https://www.oracle.com/mx/database/what-is-database/> [Último acceso: 20 de noviembre del 2020].
- [10] Escribir Canciones (2008), Estructura y elementos de una canción, [En línea]. Disponible: <https://dle.rae.es> [Último acceso: 2 de diciembre del 2020].
- [11] Swing this Music (2008), ¿QUÉ SECCIONES PUEDE TENER UNA CANCIÓN?española, [En línea]. Disponible: <https://sites.google.com/site/swingthismusiccast/interpretacio/estructura-cancion/secciones-de-una-cancion> [Último acceso: 2 de diciembre del 2020].
- [12] J. R. Norris (1997), Markov Chains, [En línea]. Disponible: <https://cape.fcfm.buap.mx/jdzf/cursos/procesos/libros/norris.pdf> [Último acceso: 15 de diciembre del 2020].
- [13] Flask (2019, julio 04), Flask's Documentation, [En línea]. Disponible: <https://flask.palletsprojects.com/en/1.0.x/> [Último acceso: 16 de diciembre del 2020].
- [14] Gajesh (2019, julio 17), The complete Flask beginner tutorial, [En línea]. Disponible: <https://dev.to/gajesh/the-complete-flask-beginner-tutorial-124i> [Último acceso: 16 de diciembre del 2020].
- [15] A. Abdelaal (2019), Deploying a Flask Application to Heroku, [En línea]. Disponible: <https://stackabuse.com/deploying-a-flask-application-to-heroku/> [Último acceso: 16 de diciembre del 2020].
- [16] A. Aguilera Hernández (2014, abril 25), Sugerencias de Seguridad para Sitios Web, [En línea]. Disponible: <https://www.seguridad.unam.mx/historico/documento/index.html?id=1143>. [Último acceso: 15 de febrero del 2019].
- [17] J. R. Aguirre, *Seguridad Informática y Criptografía*, 2da edición, Madrid, España: Universidad Politécnica de Madrid, 2006.

- [18] M. Bellare y A. Boldyreva, "The Security of Chaffing and Winnowing", California, San Diego, 2015. Disponible: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.160.4853&rep=rep1&type=pdf>. [Último acceso: 5 de mayo del 2019].
- [19] A. Beutelspacher, *Cryptology*, edición 1994, Washington, Estados Unidos de América: Mathematical Association of America, 1994.
- [20] BBVA de México (2018), ¿Es seguro guardar las contraseñas en el navegador? [En línea] Disponible: <https://www.bbva.com/es/seguro-guardar-contrasenas-navegador/> [Último acceso: 15 de noviembre de 2019]
- [21] CCM (2017), El protocolo HTTP, [En línea]. Disponible: <https://es.ccm.net/contents/264-el-protocolo-http>. [Último acceso: 6 Mayo 2019].
- [22] S. Díaz Santiago, "Generacion De Sucesiones Criptográficamente Fuer tes", tesis de Maestría, División de ciencias básicas e ingeniería, UAM, D.F., México, 2005.
- [23] C. C. Espinosa Madrigal (2011, junio 16), Robo de Identidad y Consecuencias Sociales, [En línea]. Disponible: <https://www.seguridad.unam.mx/historico/documento/index.html-id=16?fbclid=IwAR0u8WAXORvBxZ3H-aMzlBhd-6o7g8ycS88eRu7nY1t1XVtCufhEcQ7hWDs>. [Último acceso: 15 de febrero del 2019].
- [24] T. Fisher (2019), What Is SHA-1 and How Is It Used for Data Verification?, [En línea]. Disponible: <https://www.lifewire.com/what-is-sha-1-2626011>. [Último acceso: 5 mayo 2019].
- [25] J. D. Gauchat, *El gran libro de HTML5, CSS3 y Javascript*, 1ra edición, Barcelona, España: MARCOMBO S.A., 2012.
- [26] S. Goldwasser y S. Micali, "Probabilistic encryption", *Journal of Computer and System Science*, vol. 28, no. 4, pp. 270–299, 1984.
- [27] desarrolloweb.com (2019), Manual de JQuery, [En línea]. Disponible: http://dmaspv.com/files/page/07042011180222_manual%20de%20jquery%20en%20pdf%20desarrollowebcom.pdf. [Último acceso: 2 de noviembre del 2019].

- [28] Google code (2019), crypto-js, [En línea]. Disponible: <https://code.google.com/archive/p/crypto-js/>. [Último acceso: 5 de mayo del 2019].
- [29] IBM Community (2019), TRIRIGA Wiki Home, [En línea]. Disponible: <https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/IBM%20TRIRIGA1>. [Último acceso: 5 mayo 2019].
- [30] IBM (2019), IBM TRIRIGA Application Platform, [En línea]. Disponible: https://www.ibm.com/support/knowledgecenter/es/SSHEB3_3.6.0/com.ibm.tap.doc/sso_topics/t_ctr_authenticate.html. [Último acceso: 5 mayo 2019].
- [31] J. Pacheco (2018, julio 18), Entre cookies y privacidad, Oficina de Seguridad del Internauta, [En línea]. Disponible: <https://www.osi.es/es/actualidad/blog/2018/07/18/entre-cookies-y-privacidad>. [Último acceso: 5 de mayo del 2019].
- [32] A. Komarova y A. Menshchikov, "Comparison of Authentication Methods on Web Resources" en Proceedings of the Second International Scientific Conference "Intelligent Information Technologies for Industry", Varna, Bulgaria, 14–16 Septiembre, 2017, pp 106-120.
- [33] J. Larkin, "Implementation of Chaffing and Winnowing: providing confidentiality without encryption", tesis de Licenciatura en Ciencias de la Computación, University of Bath Bath, Inglaterra, 2006. [En línea]. Disponible: <https://pdfs.semanticscholar.org/5520/1a43a747f4a3fb554f241c7b7bc7636b4a07.pdf>. [Último acceso: 5 de mayo del 2019].
- [34] A. Maiorano, *Criptografia: Tecnicas De Desarrollo Para Profesionales*, 1ra edición, D.F., México: Alfaomega, 2009.
- [35] A. Maurya1, P. Kumar Saini y N. Goel, "Chaffing and Winnowing without using steganography and encryption technique", *International Journal of Information Technology and Knowledge Management*, vol. 4, no. 4, pp 515-517, Diciembre, 2011
- [36] A. J. Menezes, P. v. Oorschot y S. Vanstone, *Handbook of Applied Cryptography*, 1ra edición, Florida, Estados Unidos de América: CRC Press, 1996.

- [37] Mozilla (2019, mayo 2), JavaScript. [En línea]. Disponible: <https://developer.mozilla.org/en-US/docs/Web/JavaScript> [Último acceso: 5 de mayo del 2019].
- [38] Mozilla, (2019, 19 marzo), FileSystem, [En línea]. Disponible: <https://developer.mozilla.org/es/docs/Web/API/FileSystem>. [Último acceso: 5 de mayo del 2019].
- [39] R. S. Pressman, *Ingeniería del Software. Un enfoque práctico*, 7ma edición, D.F, México: McGraw-Hill, 2010.
- [40] R. L. Rivest, ”Chaffing and Winnowing: Confidentiality without Encryption”, Laboratorio de ciencias de la computación del MIT, Massachusetts, Estados Unidos de América, 1998.
- [41] A. Romero Mier y Terán y M. A. Vasquéz Martínez (2016, abril 19), Aspectos Básicos de la Seguridad en Aplicaciones Web. [En línea]. Disponble: <https://www.seguridad.unam.mx/historico/documento/index.html-id=17>. [Último acceso: 5 de mayo del 2019].
- [42] Universidad Politécnica de Madrid (2004), Criptografía, [En línea]. Disponible: http://www.dma.fi.upm.es/recursos/aplicaciones/mathematica_discreta/web/aritmetica_modular/criptografia.html. [Último acceso: 5 de mayo del 2019].
- [43] MongoDB. (2019), ¿Qué es MongoDB?, [En línea]. Disponible: <https://www.mongodb.com/es>. [Último acceso: 2 de noviembre del 2019].
- [44] VeriSign Inc. (2019), Todo lo que debe saber sobre certificados SSL, [En línea]. Disponible: https://www.verisign.com/es_LA/website-presence/online/ssl-certificates/index.xhtml. [Último acceso: 5 de mayo del 2019].
- [45] Wireshark (2019), What is Wireshark?, [En línea]. Disponible: https://www.wireshark.org/docs/wsug_html_chunked/ChapterIntroduction.html#ChIntroWhatIs. [Último acceso: 20 abril 2019].
- [46] Kerberos (2019), Descripción del Kerberos un servicio de autenticación para los sistemas de red abierta, [En línea]. Disponible: https://www.cisco.com/c/es_mx/support/docs/security-vpn/kerberos/16087-1.html#whatisit

- [47] Van Tilborg, Henk C.A, "Encyclopedia of Cryptography and Security". *Eindhoven University of Technology The Netherlands*, United States of America, Editorial Springer, 2005, 671 pag.
- [48] L. Hernández Encinas (2015), Las Firmas y los Certificados electrónicos (de la Administración Pública del Estado-CSIC) [En línea]. Disponible: <http://www.itefi.csic.es/sites/default/files/hernandez-encinas/firma-y-certificado-electronico.pdf> [Último acceso: 6 noviembre 2019]
- [49] Universitat de Valéncia (2016), ¿Qué son las cookies? [En línea]. Disponible: <https://www.uv.es/uvweb/universidad/es/politica-privacidad/politica-cookies/son-cookies-1285919089226.html> [Último acceso: 6 noviembre 2019]
- [50] J. C. Teague, *Speaking in Styles: Fundamentals of CSS for Web Designers*. 1ra edición. California EEUU: New Riders, 2009.
- [51] Bootstrap (2019), About Bootstrap [En línea]. Disponible: <https://getbootstrap.com/docs/4.3/about/overview/> [Último acceso: 6 noviembre 2019]
- [52] Node.js Foundation (2019), Acerca de Node.js [En línea]. Disponible: <https://nodejs.org/es/about/> [Último acceso: 6 de noviembre 2019]
- [53] Noteworthy Programming Masterpiece (2019), openssl-nodejs [En línea]. Disponible: <https://www.npmjs.com/package/openssl-nodejs> [Último acceso: 6 noviembre 2019]
- [54] D. Bell, M. Parr, *Java para estudiantes*. 3ra edición. México: Pearson Education, 2003.
- [55] Walls, Craig, *Spring in Action*. 5ta edición. EEUU: Manning, 2018.
- [56] The Apache Software Foundation (2019), Apache Tomcat [En línea]. Disponible: <http://tomcat.apache.org/index.html> [Último acceso: 6 noviembre 2019]
- [57] Oracle (2018), MySQL [En línea]. Disponible: <https://www.oracle.com/mysql/> [Último acceso: 6 noviembre 2019]
- [58] Tecnicatura de gestión universitaria (2008), Modelo relacional. Conceptos básicos y fundamentos [En línea]. Disponible: <http://oftgu.eco.catedras.unc.edu.ar/unidad-3/sistemas-de-gestion-de-base-de-datos/modelo-relacional-conceptos-basicos-y-fundamentos/> [Último acceso: 6 de noviembre 2019]

- [59] Oracle (2010), Package java.security [En línea]. Disponible: https://docs.oracle.com/javase/7/docs/api/java/security/package-summary.html#package_description [Último acceso: 6 noviembre 2019]
- [60] Security Appstop (2012), About vsftpd [En línea]. Disponible: <https://security.appspot.com/vsftpd.html#about> [Último acceso: 10 de noviembre 2019]
- [61] Johannes A. Buchmann, Evangelos Karatsiolis, A.W.: Introduction to Public Key Infrastructures. Springer (2013)
- [62] Hostalia whitepapers (2010), ¿Sabes cómo utilizar el protocolo FTP? [En línea]. Disponible: <https://www.hostalia.com/news/noviembre10/sabes-como-utilizar-el-protocolo-FTP.pdf> [Último acceso: 19 de octubre de 2019]
- [63] E-Reding (2001), Protocolo HTTP [En línea]. Disponible: <http://bibing.us.es/proyectos/abreproj/11372/fichero/Memoria%252F05+-+Protocolo+HTTP.pdf> [Último acceso: 21 de octubre de 2019]
- [64] IBM (2018), Protocolo trivial de transferencia de archivos [En línea]. Disponible: https://www.ibm.com/support/knowledgecenter/es/ssw_ibm_i_73/rzal5/rzal5overview.htm [Último acceso: 28 de octubre de 2019]
- [65] EcuRed (2011), Protocolo SMTP [En línea]. Disponible: https://www.ecured.cu/Protocolo_SMTMP [Último acceso: 05 de noviembre de 2019]
- [66] Ayuda de Search de Console (2019), Proteger sitios web con el protocolo HTTPS [En Línea]. Disponible: <https://support.google.com/webmasters/answer/6073543?hl=es-419> [Último acceso: 06 de noviembre de 2019]