



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.

<b>I Trabajo Terminal I</b>	<b>8</b>
<b>1. Introducción</b>	<b>9</b>
1.1. Objetivos. . . . .	12
1.1.1. Objetivo general. . . . .	12
1.1.2. Objetivos particulares. . . . .	12
1.2. Justificación. . . . .	12
1.3. Metodología. . . . .	13
1.4. Organización del documento . . . . .	15
1.4.1. Capítulo 2. Marco teórico. . . . .	15
1.4.2. Capítulo 3. Análisis. . . . .	15
1.4.3. Capítulo 4. Diseño. . . . .	16
1.4.4. Capítulo 5. Desarrollo. . . . .	16
1.4.5. Capítulo 6. Avances y trabajo por hacer. . . . .	16
<b>2. Marco teórico</b>	<b>17</b>
2.1. Redes neuronales . . . . .	17
2.2. Base de datos . . . . .	20
2.2.1. Base de datos NoSQL o no relacionales . . . . .	20
2.3. ¿Qué es una canción? . . . . .	20
2.3.1. Elementos de una canción . . . . .	20
2.3.2. Introducción . . . . .	20
2.3.3. Verso . . . . .	21
2.3.4. Pre-estribillo . . . . .	21
2.3.5. Estribillo . . . . .	21
2.3.6. Puente . . . . .	21
2.3.7. Cierre . . . . .	21
2.3.8. Estructura de una canción . . . . .	21
2.4. Cadenas de Markov . . . . .	22
2.5. Flask . . . . .	23
2.6. Apache . . . . .	25
2.7. Servidor Web . . . . .	26

2.8. Certificado SSL . . . . .	27
--------------------------------	----

## **Sprint 1. 28**

### **3. Análisis. 29**

3.1. Estudio de Factibilidad. . . . .	29
3.1.1. Factibilidad Técnica . . . . .	29
3.1.2. Factibilidad Operativa . . . . .	31
3.1.3. Factibilidad Económica . . . . .	32
3.2. Herramientas a usar. . . . .	33
3.2.1. Software. . . . .	33
3.2.2. Hardware. . . . .	43
3.3. Arquitectura del sistema. . . . .	44
3.3.1. Descripción de la arquitectura del sistema. . . . .	44

### **4. Diseño. 46**

4.1. Componente I: Extensión. . . . .	46
4.1.1. Diagrama de casos de uso . . . . .	46
4.1.2. Diagrama de flujo. . . . .	59
4.1.3. Diagrama de flujo de datos. . . . .	61
4.1.4. Diagrama de clases. . . . .	62
4.1.5. Diagramas de secuencia. . . . .	65
4.1.6. Diagrama de actividades . . . . .	72
4.1.7. Interfaz de usuario. . . . .	73
4.1.8. Requisitos de diseño. . . . .	83
4.1.9. Algoritmos. . . . .	84
4.2. Componente II: Servidor Autentificador. . . . .	87
4.2.1. Diagrama de casos de uso. . . . .	87
4.2.2. Diagrama de flujo. . . . .	95
4.2.3. Diagrama de flujo de datos. . . . .	96
4.2.4. Diagrama de clases. . . . .	97
4.2.5. Diagramas de secuencias. . . . .	100
4.2.6. Diagrama de actividades. . . . .	104
4.3. Componente III: API. . . . .	105
4.3.1. Diagrama de casos de uso. . . . .	105
4.3.2. Diagrama de flujo. . . . .	110
4.3.3. Diagrama de flujo de datos. . . . .	111
4.3.4. Diagrama de clases. . . . .	112
4.3.5. Diagramas de secuencias. . . . .	115
4.3.6. Diagrama de actividades . . . . .	118

4.3.7.	Interfaz de usuario. . . . .	118
4.3.8.	Algoritmos. . . . .	122
<b>5.</b>	<b>Desarrollo.</b>	<b>126</b>
5.1.	Componente I. Extensión. . . . .	126
5.1.1.	Archivo popup.html . . . . .	126
5.1.2.	Archivo background.js . . . . .	127
5.2.	Componente II. Servidor Autentificador. . . . .	128
5.2.1.	Manejador de paquetes de Node (npm). . . . .	128
5.2.2.	Componentes. . . . .	129
5.3.	Componente III. API. . . . .	130
5.3.1.	API . . . . .	131
5.3.2.	Servicio Web. . . . .	133
<b>6.</b>	<b>Pruebas.</b>	<b>135</b>
6.1.	Pruebas de integración. . . . .	135
6.1.1.	Extensión y Servidor autentificador. . . . .	135
6.1.2.	Extensión y Servicio Web. . . . .	136
6.1.3.	Servidor Autentificador y API. . . . .	136
6.2.	Tiempos de ejecución. . . . .	136
6.2.1.	Algoritmo de Chaffing. . . . .	137
6.2.2.	Algoritmo de Winnowing. . . . .	137
6.2.3.	Inicio de sesión. . . . .	137

# Índice de figuras.

1.1. Fases de la metodología por prototipos . . . . .	15
2.1. Diagrama Red Neuronal Recurrente . . . . .	19
2.2. Diagrama De Estados finitos De Una Cadena De Markov . . .	23
3.1. Arquitectura General del Sistema . . . . .	44
4.1. Diagrama de casos de uso del Componente I. . . . .	46
4.2. Diagrama de flujo del Componente I. . . . .	59
4.3. Diagrama de flujo de datos del Componente I. . . . .	61
4.4. Diagrama de clases de Componente I. . . . .	62
4.5. Diagrama de secuencia 1 del Componente I . . . . .	65
4.6. Diagrama de secuencia 2 del Componente I . . . . .	66
4.7. Diagrama de secuencia 3 del Componente I . . . . .	67
4.8. Diagrama de secuencia 4 del Componente I . . . . .	68
4.9. Diagrama de secuencia 5 del Componente I . . . . .	69
4.10. Diagrama de secuencia 6 del Componente I . . . . .	70
4.11. Diagrama de secuencia 7 del Componente I . . . . .	71
4.12. Diagrama de actividades del Prototipo 2. . . . .	72
4.13. Pantalla de interfaz de la extensión. . . . .	73
4.14. Pantalla inicial. . . . .	74
4.15. Pantalla de aviso. . . . .	75
4.16. Tab de la extensión. Permite inicio de sesión . . . . .	76
4.17. Mensaje de éxito . . . . .	76
4.18. Mensaje de error . . . . .	77
4.19. Tab de la extensión. Permite registrarse . . . . .	78
4.20. Mensaje de éxito . . . . .	79
4.21. Mensaje de error . . . . .	79
4.22. Mensaje de error . . . . .	80
4.23. Mensaje de error . . . . .	80
4.24. Mensaje de error . . . . .	81
4.25. Interfaz . . . . .	81

4.26. Mensaje de éxito . . . . .	82
4.27. Mensaje de error . . . . .	82
4.28. Arreglo del patrón de chaffing final . . . . .	86
4.29. Diagrama de casos de uso del Componente II. . . . .	87
4.30. Diagrama de flujo de Componente II. . . . .	95
4.31. Diagrama de flujo de datos del Componente II. . . . .	96
4.32. Diagrama de clases de Componente II. . . . .	97
4.33. Diagrama de secuencia 1 del Componente II . . . . .	100
4.34. Diagrama de secuencia 2 del Componente II . . . . .	101
4.35. Diagrama de secuencia 3 del Componente II . . . . .	102
4.36. Diagrama de secuencia 4 del Componente II . . . . .	103
4.37. Diagrama de actividades de Componente II. . . . .	104
4.38. Diagrama de caso de uso del componente III . . . . .	105
4.39. Diagrama de flujo de Componente III. . . . .	110
4.40. Diagrama de flujo de datos de Componente III. . . . .	111
4.41. Diagrama de Clases de componente 3. . . . .	112
4.42. Diagrama de Secuencia de Caso de Uso 1. Comprobar certificado. . . . .	115
4.43. Diagrama de Secuencia de Caso de Uso 2. Enviar petición. . . . .	116
4.44. Diagrama de Secuencia de Caso de Uso 3. Redirigir página. . . . .	117
4.45. Diagrama de Actividades de componente 3. . . . .	118
4.46. Mensaje de éxito . . . . .	119
4.47. Interfaz . . . . .	120
4.48. Interfaz . . . . .	120
4.49. Interfaz . . . . .	121
4.50. Interfaz . . . . .	121
4.51. Mensaje de error . . . . .	122
5.1. Estructura del desarrollo de la Autoridad Certificadora. . . . .	129
5.2. Estructura del desarrollo de la API. . . . .	132
6.1. Sesión iniciada . . . . .	136
6.2. Resultados de la API al recibir petición de la extensión. . . . .	136
6.3. Resultados de la comunicación entre la API y el Servidor Autenticador. . . . .	136

# Índice de cuadros.

1.1. Resumen de productos similares comparados con nuestra propuesta . . . . .	11
2.1. Tabla comparativa de los distintos servidores web contemplados	26
3.1. Herramientas de Software . . . . .	30
3.2. Equipo de Hardware 1 . . . . .	30
3.3. Equipo de Hardware 2 . . . . .	30
3.4. Equipo de Hardware 3 . . . . .	31
3.5. Horas de trabajo . . . . .	32
4.1. DCU: CL_CU1 . . . . .	47
4.2. DCU: CL_CU2 . . . . .	48
4.3. DCU: CL_CU3 . . . . .	49
4.4. DCU: CL_CU4 . . . . .	50
4.5. DCU: CL_CU5 . . . . .	53
4.6. DCU: CL_CU6 . . . . .	54
4.7. DCU: CL_CU7 . . . . .	57
4.8. DCU: CH_CU1 . . . . .	88
4.9. DCU: CH_CU2 . . . . .	90
4.10. DCU: CH_CU3 . . . . .	92
4.11. DCU: CH_CU4 . . . . .	94
4.12. DCU: CHH_CU1 . . . . .	106
4.13. DCU: CHH_CU2 . . . . .	108
4.14. DCU: CHH_CU3 . . . . .	109





# 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 [3]), 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 [4], Convolutional Neural Networks for Sentence Classification[5]) 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 [6]), o solo generar un tipo específico de letra (Conditional Rap Lyrics Generation with Denoising Autoencoders [7]). 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 irá dirigido.	Gratuito (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.	Gratuito
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.	Gratuito
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 nal con coros y versos distintos cada vez que se utilice.	Gratuito

## **1.1. Objetivos.**

### **1.1.1. Objetivo general.**

Crear una herramienta de apoyo para estudiantes o aficionados interesados en este rubro que se les dificulte componer nuevas letras musicales de un solo género musical debido a la carencia de creatividad, la falta de conocimientos en la estructura del género o que no tengan inspiración suficiente para poder crear nuevas canciones, esto 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. 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. Se hará uso de un conjunto de datos (dataset) y herramientas alojadas en la nube (Google Cloud Platform o Amazon Web Services) para procesar estos datos; se pretende utilizar 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 [10], spaCy[11]) 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.

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.

### **1.3. 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 com-

plejidad 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.

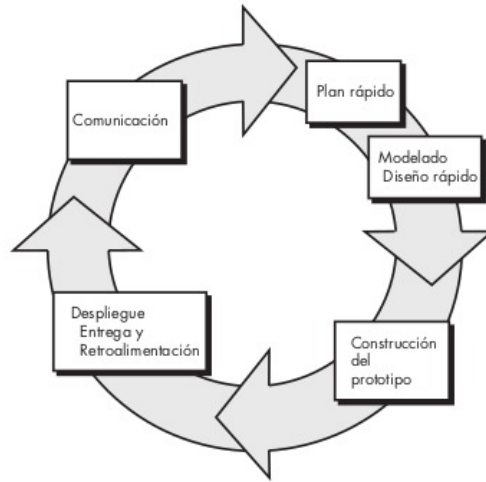


Figura 1.1: Diagrama de las fases de la metodología de prototipos evolutivos.

A continuación se indica cada una de las etapas de la metodología con lo que se realizará en cada una de estas:[48].

## 1.4. 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.

### 1.4.1. 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.

### 1.4.2. 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



te 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 componente que tenemos en el sistema.

### **1.4.3. Capítulo 4. Diseño.**

En el tercer capítulo, nos adentraremos en el desarrollo del 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.

### **1.4.4. 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.

### **1.4.5. 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

### 2.1. 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).[14]

#### 2.1.0.1. 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.[15]

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

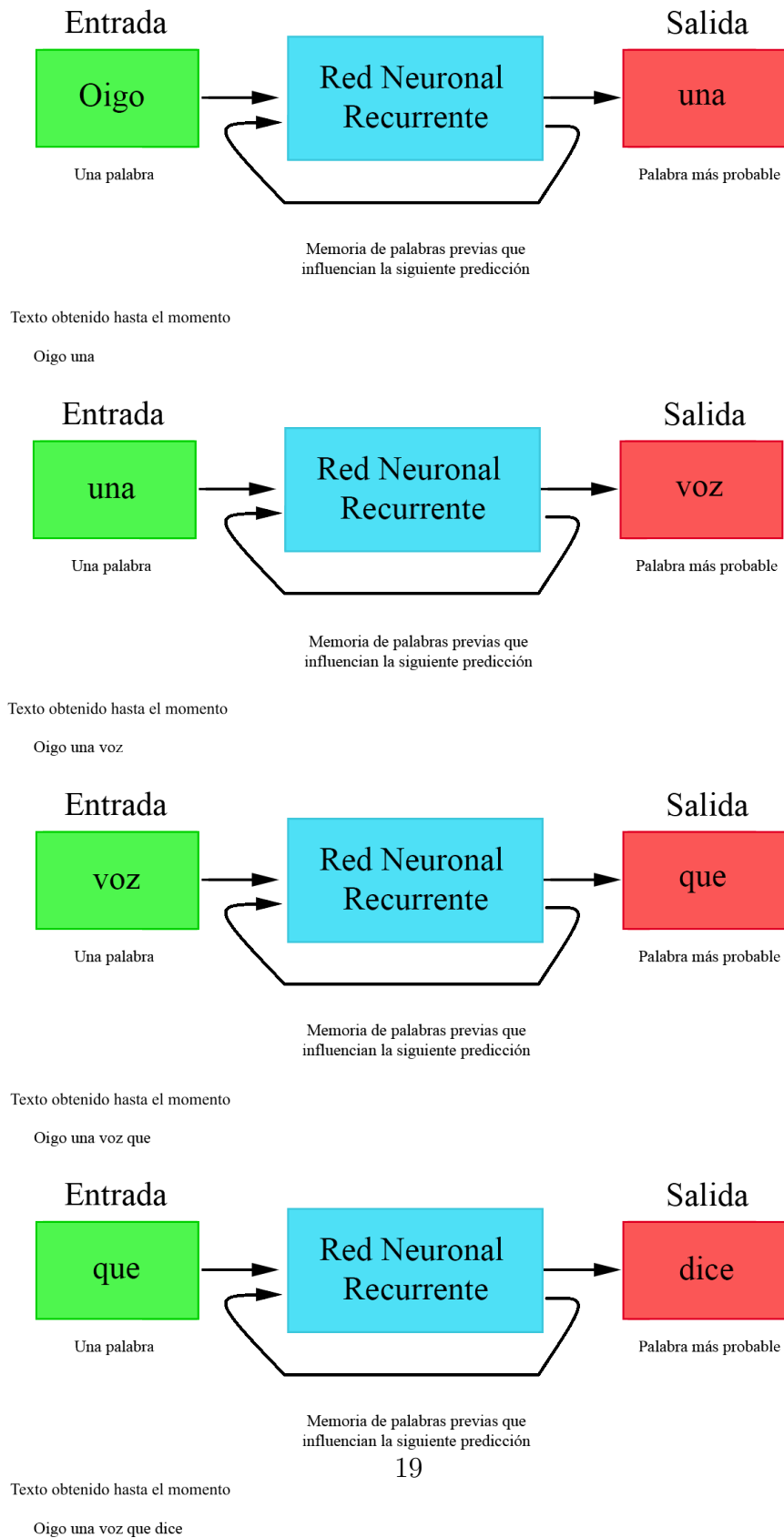


Figura 2.1: Diagrama Red Neuronal Recurrente

## **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.[18]

### **2.2.1. 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 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.

## **2.3. ¿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.[19]

### **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.[20]

### **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. [21]

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

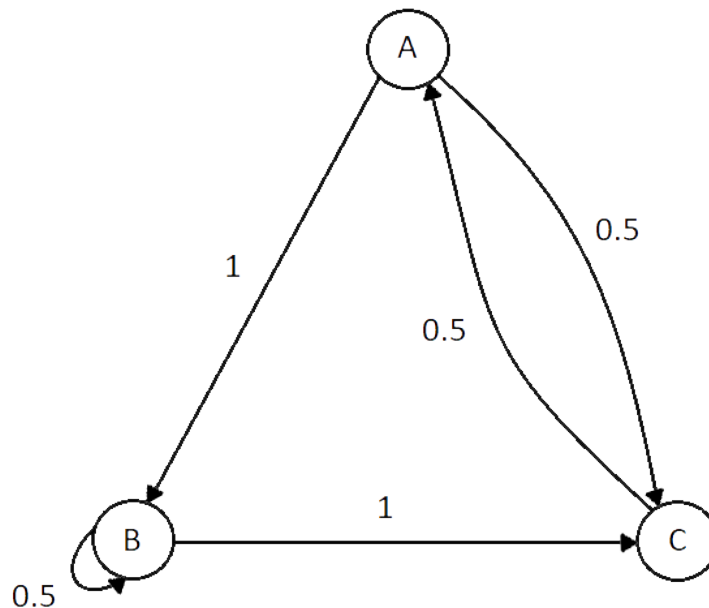


Figura 2.2: Diagrama De Estados finitos De Una Cadena De Markov

## 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 Relacionales 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.[24]

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.[23]

Para poder empezar a trabaja 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 creará 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.<sup>[22]</sup>

```
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 lo 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 requerimos 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...

## 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 2.1: 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:

- Autenticar 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.

**Sprint 1.**

# Capítulo 3

## Análisis.

### 3.1. Estudio de Factibilidad.

El estudio de factibilidad sirve como instrumento 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 cual 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 como 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 [34].

### **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 [59]:

- 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 normas 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[59]:

- 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. [60]

### **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 Netscape, 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 [46].

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 [34].

### **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.[36]

### **CryptoJs.**

CryptoJs es una colección estándar y segura de algoritmos criptográficos implementados para JavaScript usando las mejores practicas 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.[37]

### **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. [61]

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 mas 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). [62]

### **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 [63].

### **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) [64]. 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 [65].

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 [53].

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 [52].

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 [66].

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 [67].

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. [69] 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 reemplaza alguno, éste no podrá recuperarse [71].
- **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 [72]. Las tres operaciones más usadas que permiten a un cliente dialogar con el servidor son [71]:

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) [73].
  - **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 [74].
  - **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 [75]:
    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 analizador de paquetes de red. Un analizador captura paquetes de red y muestra los datos del paquete con el mayor detalle posible.

Esta herramienta es software libre y es el mejor analizador de paquetes hoy en día [54]. 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 ocultos 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.

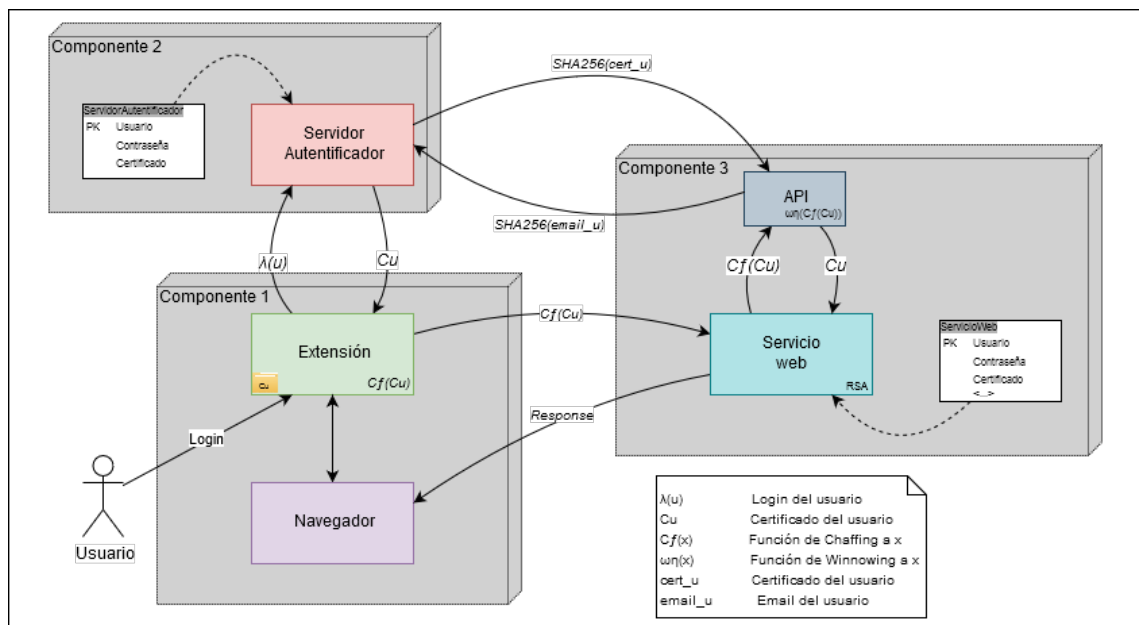


Figura 3.1: Arquitectura General del Sistema

#### 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 autenticador:** 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 autenticador y la Extensión se encontrará oculto mediante un socket seguro.

# Capítulo 4

## Diseño.

### 4.1. Componente I: Extensión.

#### 4.1.1. Diagrama de casos de uso

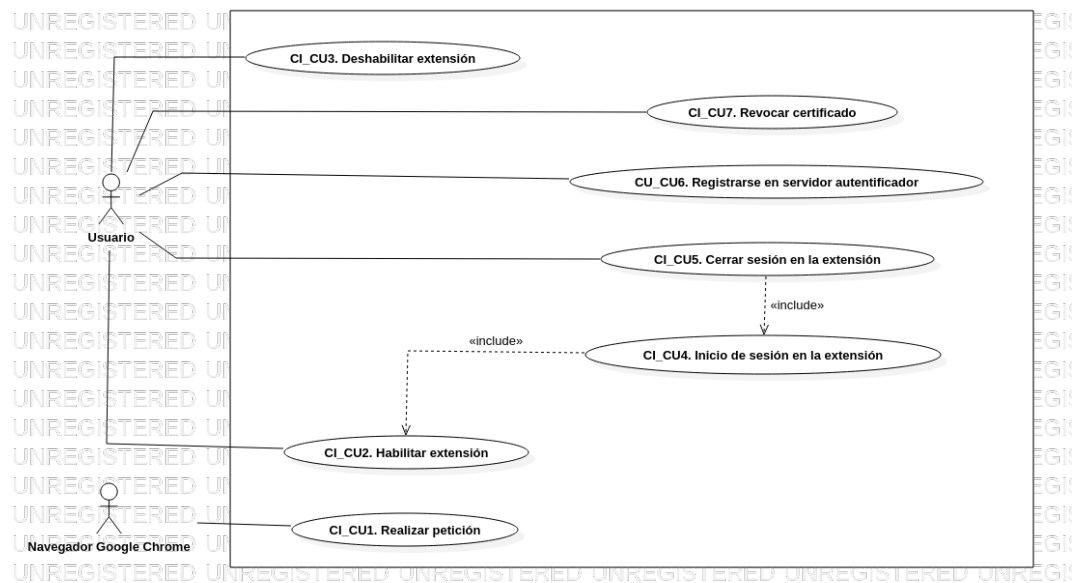


Figura 4.1: Diagrama de casos de uso del Componente I.

#### 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 ordenado 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	<b>CI_CU2.</b>
Post-condiciones	-
Reglas del negocio	<b>CI_RN2.</b>
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	Este caso de uso permite al usuario poder iniciar sesión en la extensión para posteriormente obtener un código autenticador, es decir, el certificado del usuario, al cual se le aplicará <i>Chaffing</i> . Sólo se requerirá iniciar sesión una sola vez ya que después de ello el certificado se guardará en la extensión.
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 desee identificarse en un servicio web.
Reglas del negocio	<b>CI_RN1.</b> <b>CI_RN4.</b> <b>CI_RN7.</b> <b>CI_RN8.</b> <b>CI_RN9.</b> <b>CI_RN10.</b> <b>CI_RN11.</b>
Errores	No se encuentra usuario registrado. No se puede obtener el código autenticador. No se pudo guardar el código autenticador. 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	<b>CI_CU4.</b>
Post-condiciones	La extensión debe mantenerse habilitada para su correcto funcionamiento.
Reglas del negocio	<b>CI_RN1.</b> <b>CI_RN4.</b> <b>CI_RN5.</b>
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 autenticador.	
Concepto	Descripción
Actor	Usuario
Propósito	Este caso de uso permitirá al usuario poder registrarse en el servidor autenticador 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	<b>CI_RN5.</b> <b>CI_RN7.</b> <b>CI_RN8.</b> <b>CI_RN9.</b> <b>CI_RN10.</b> <b>CI_RN11.</b> <b>CI_RN12.</b> <b>CI_RN13.</b>
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 autenticador.
Salidas	Certificado autenticador.
Pre-condiciones	CI_CU4, CI_CU6.
Post-condiciones	CI_CU4
Reglas del negocio	CI_RN4. CI_RN5. CI_RN6.
Errores	No se puede almacenar el certificado autenticador.

Cuadro 4.7: Descripción CU: CI\_CU7

... Trayectoria Principal ...

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

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

... Trayectoria Alternativa 2 ...

1. **La extensión** ha creado el certificado autenticador.
2. **La extensión** no puede guardar el certificado autenticador 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 autenticador.
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.

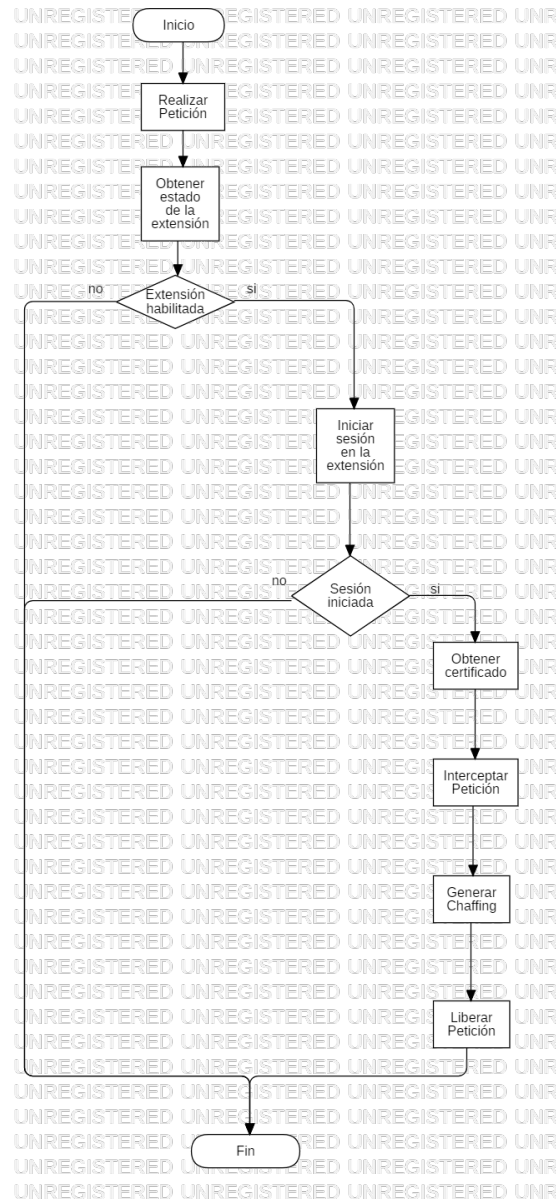


Figura 4.2: Diagrama de flujo del Componente I.

##### 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.

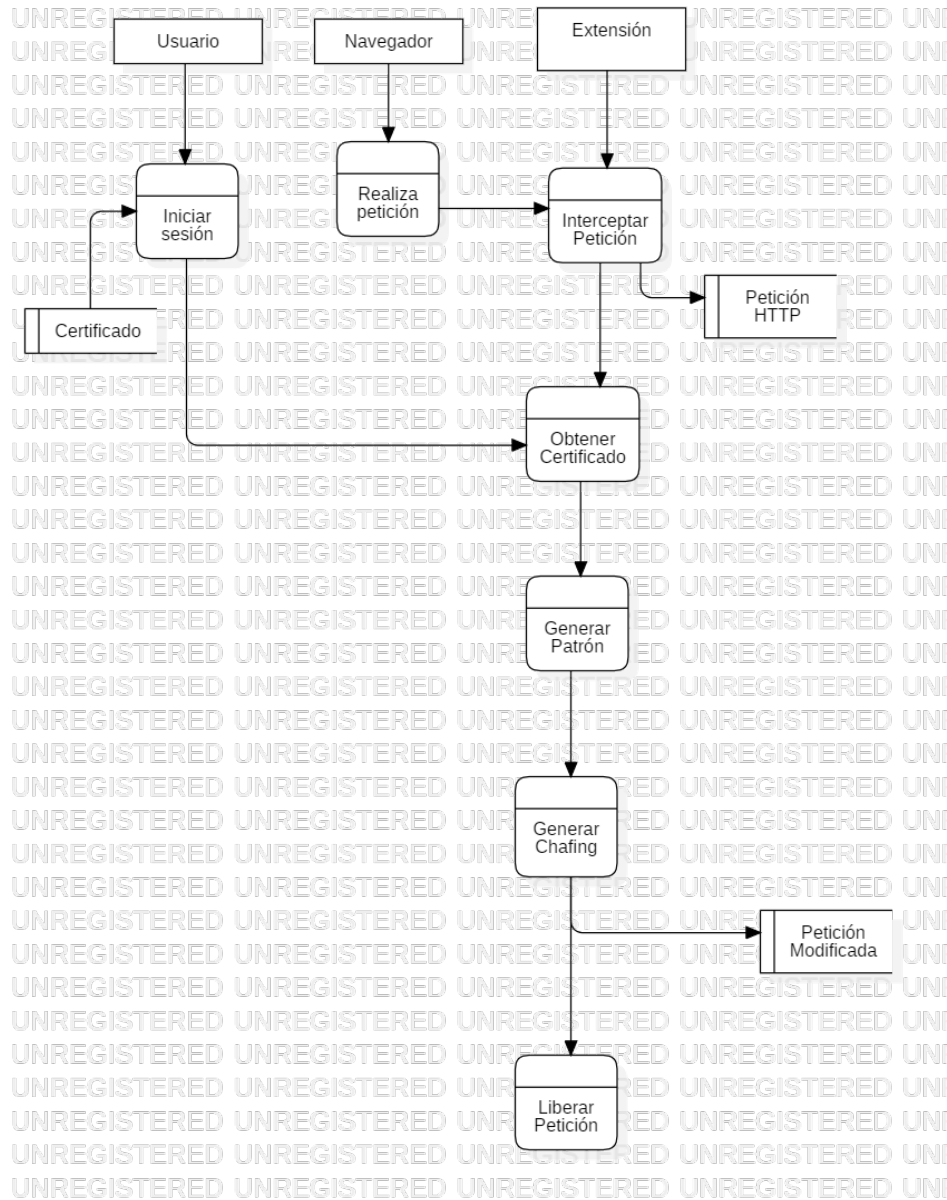


Figura 4.3: Diagrama de flujo de datos del Componente I.

#### 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.

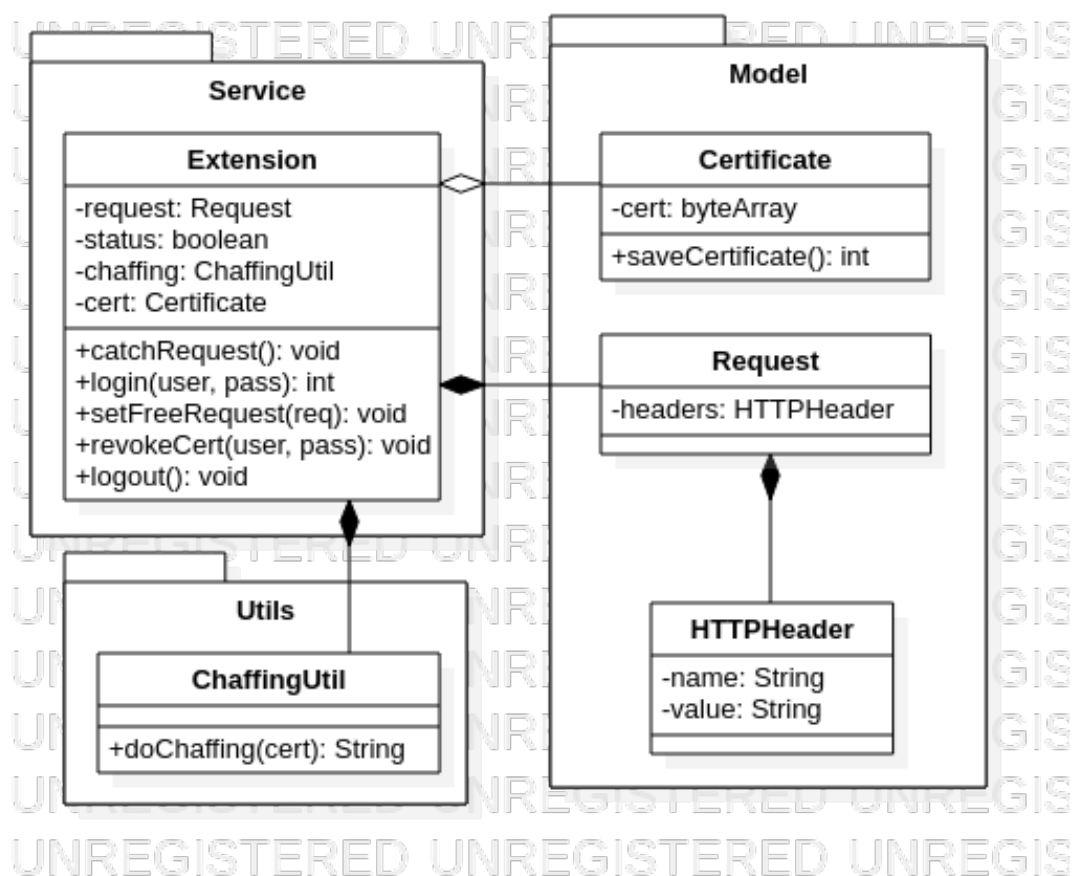


Figura 4.4: Diagrama de clases de Componente I.

##### 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.

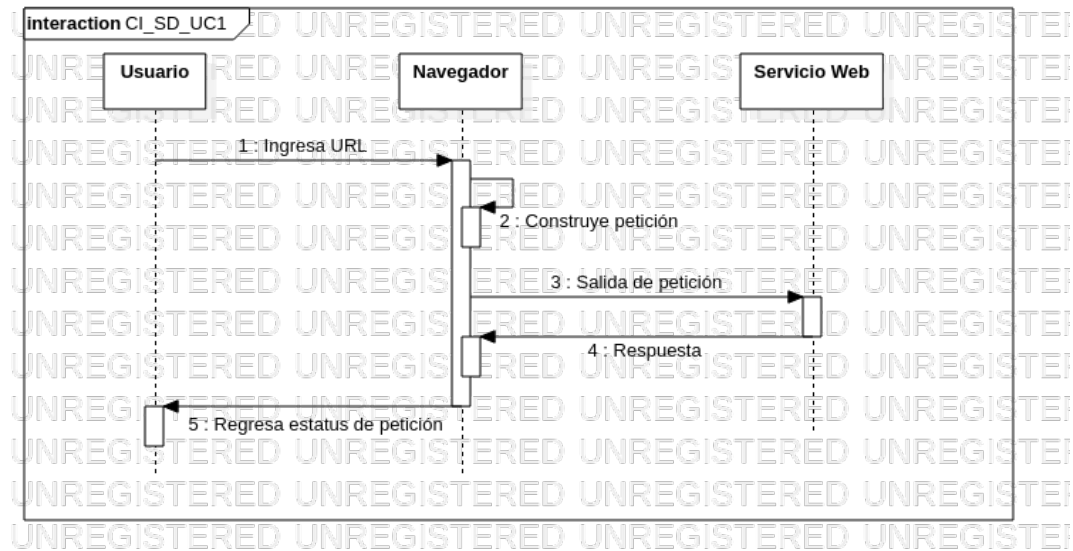


Figura 4.5: Diagrama de secuencia del CI\_CU1. Realizar petición.

**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.

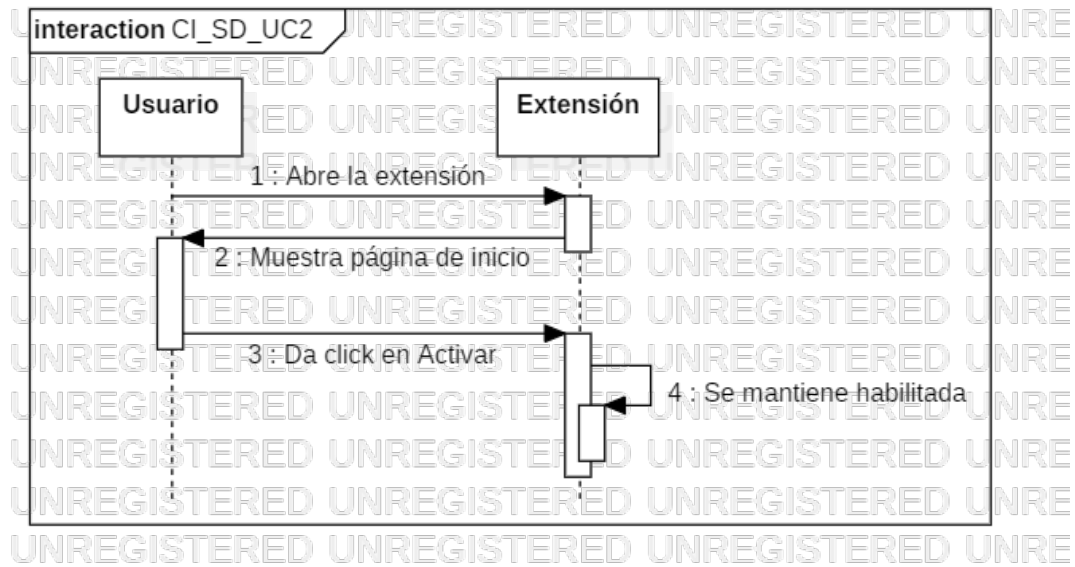


Figura 4.6: Diagrama de secuencia del CLCU2. Habilitar extensión.

**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.

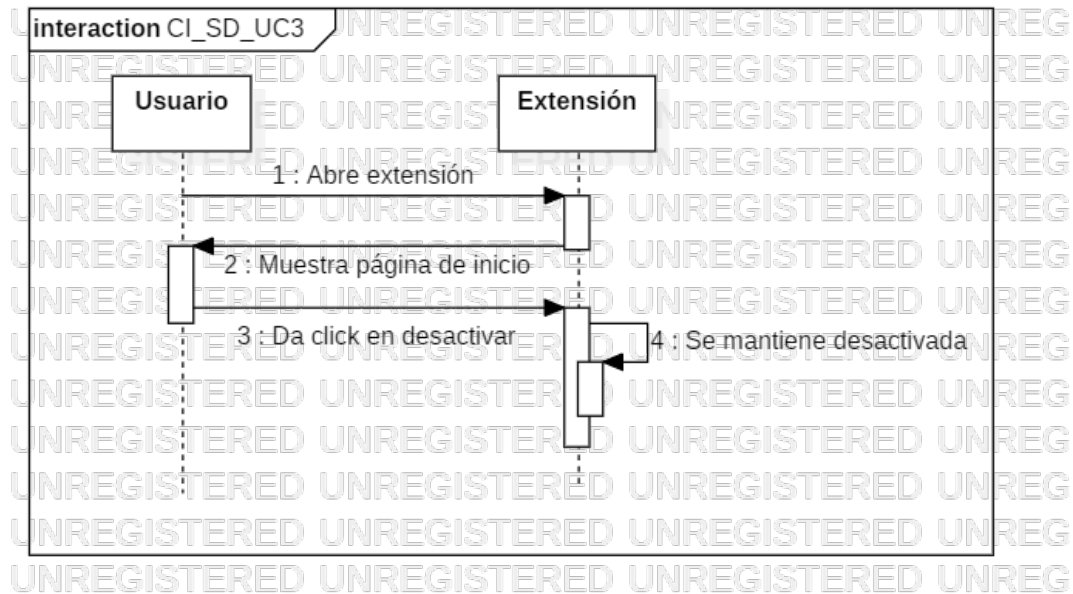


Figura 4.7: Diagrama de secuencia del CI.CU3. Deshabilitar extensión.

**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

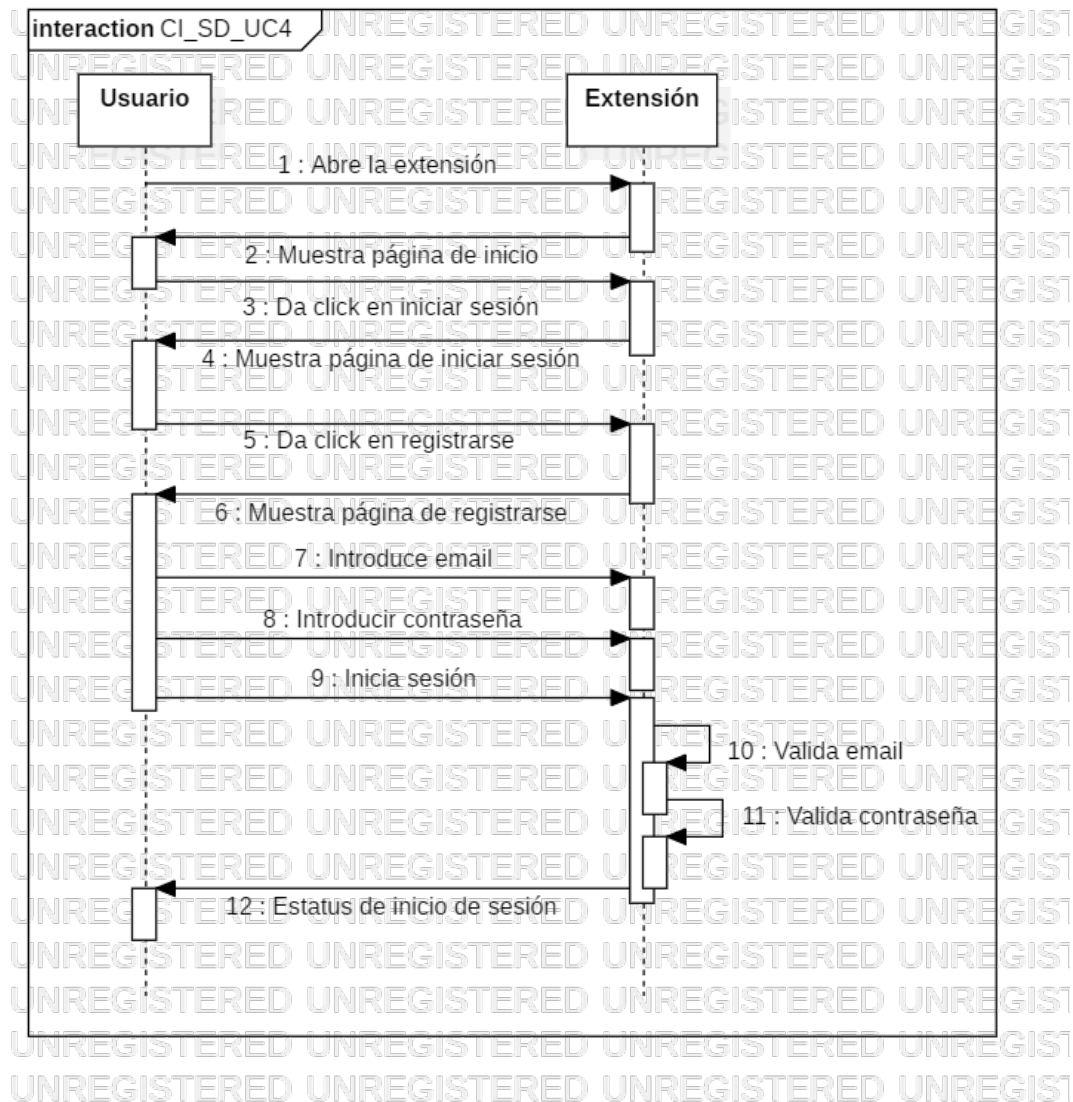


Figura 4.8: Diagrama de secuencia del CI\_CU4. Inicio de sesión en la extensión.

**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.

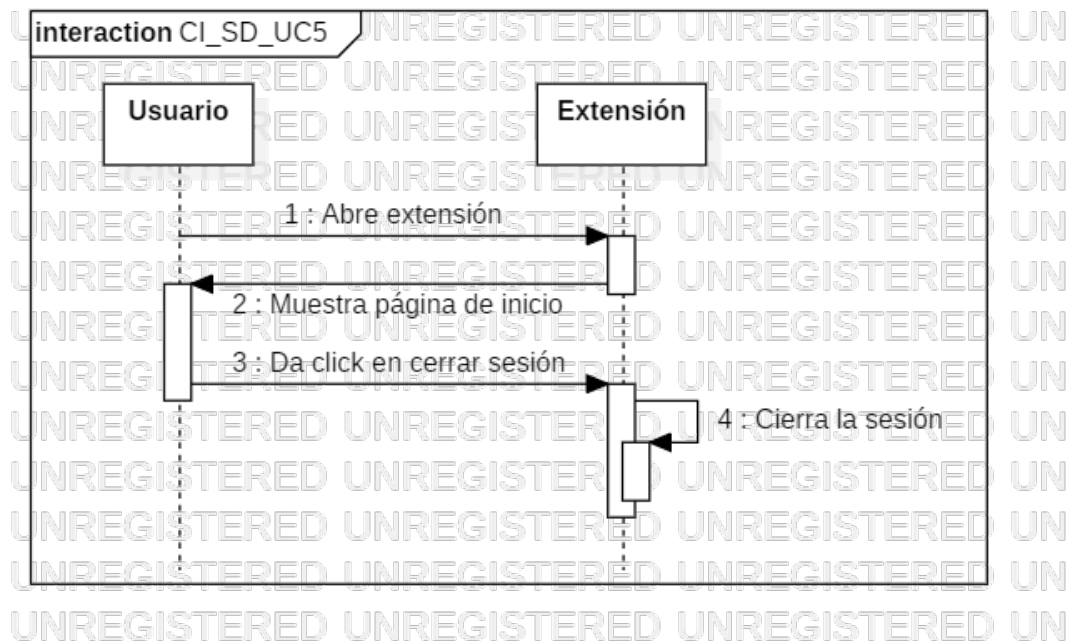


Figura 4.9: Diagrama de secuencia del CI-CU5. Cerrar sesión en la extensión.

**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.

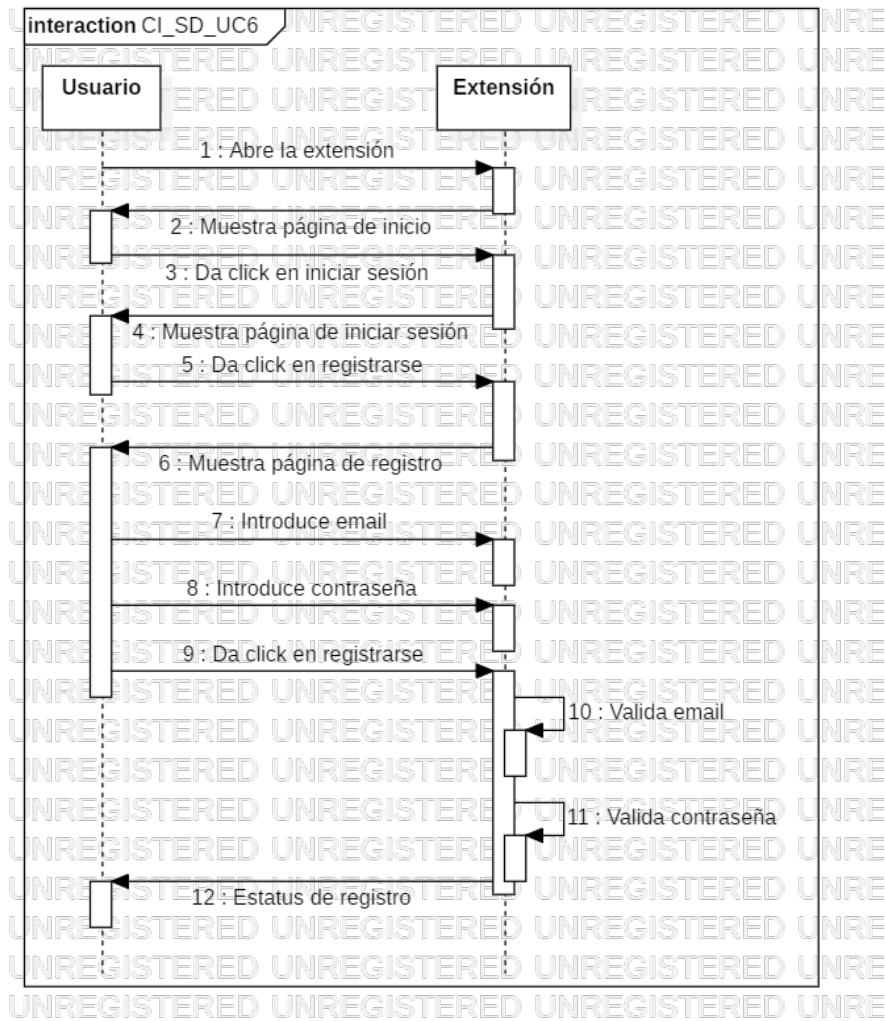


Figura 4.10: Diagrama de secuencia del CI.CU6. Registrarse en servidor autenticador.

**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.

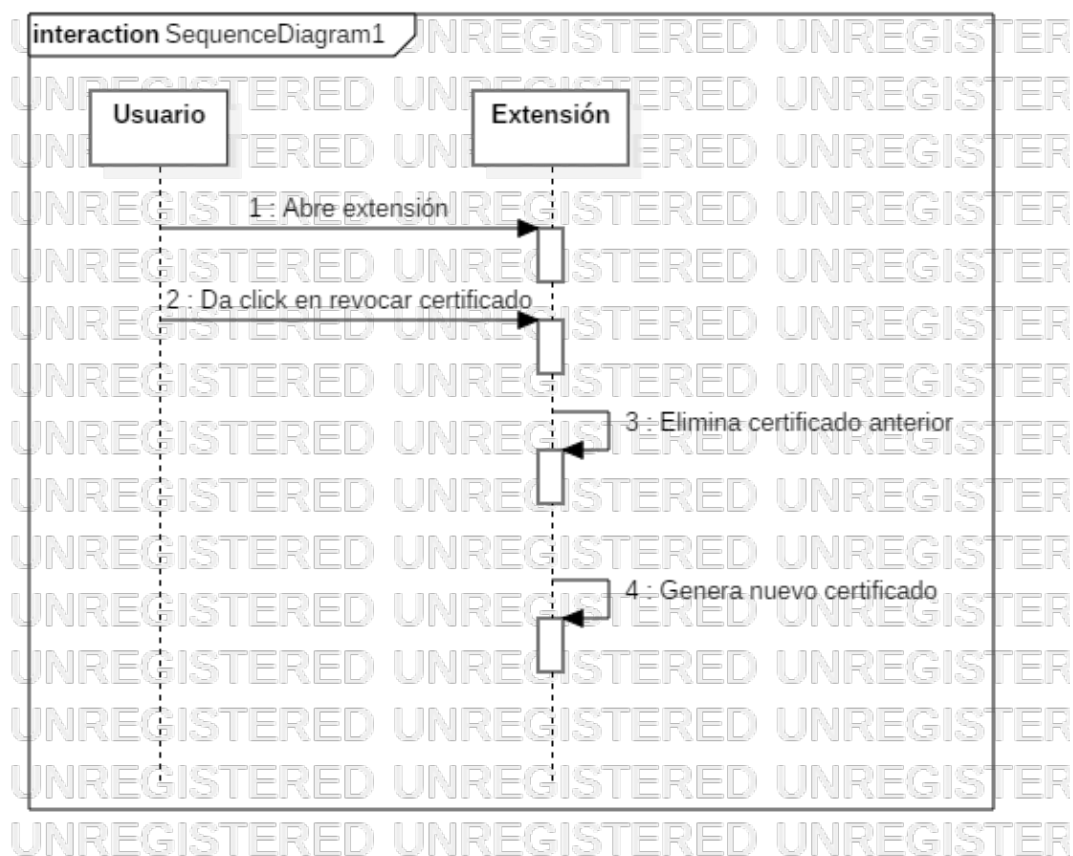


Figura 4.11: Diagrama de secuencia del CL\_CU7. Revocar certificado.

**Descripción:**



#### 4.1.6. Diagrama de actividades

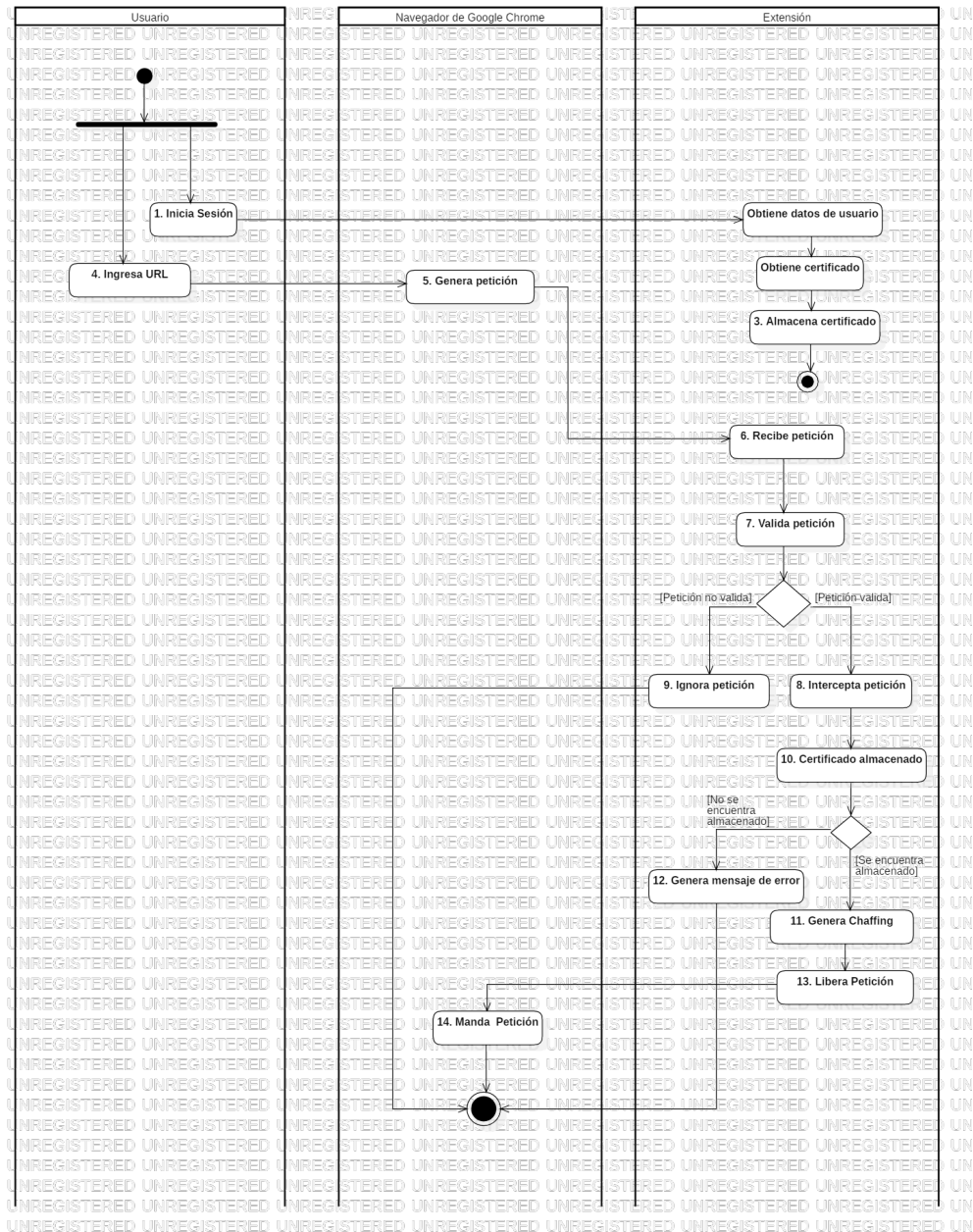


Figura 4.12: Diagrama de actividades del Prototipo 2.

#### 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 icono 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, además 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.

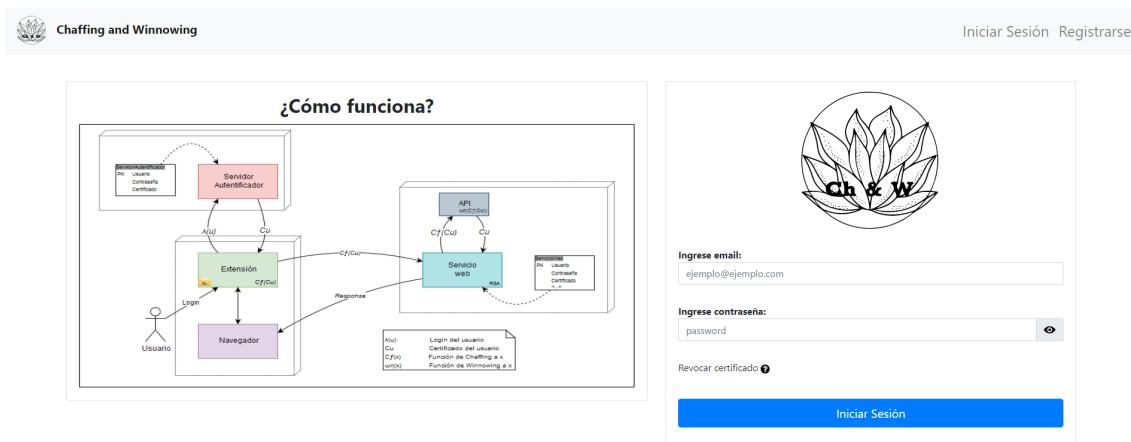


Figura 4.14: Pantalla inicial.

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 ('Ingrese email' e 'Ingrese contraseña'), y un botón 'Iniciar Sesión'.



**Ingrese email:**

**Ingrese contraseña:**

**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.

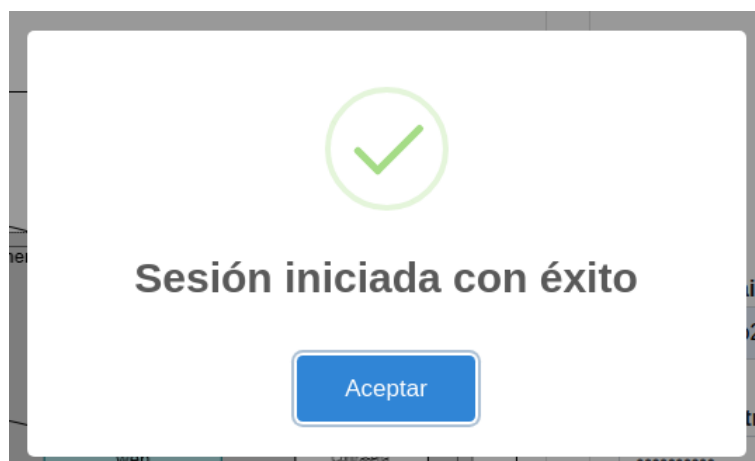


Figura 4.17: Mensaje de éxito tras obtener el certificado de la autoridad y guardarlo en el storage de Google Chrome.

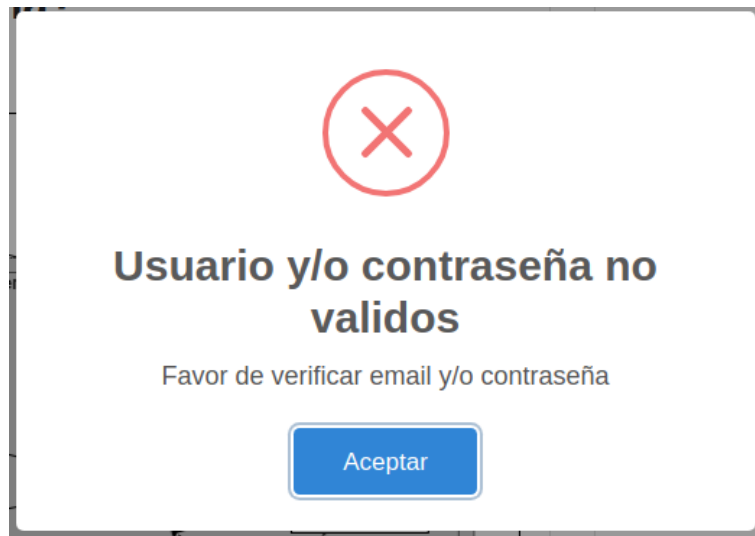




Figura 4.18: Mensaje de error al tratar de obtener certificado de la autoridad (Usuario y/o contraseña incorrectos).

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:**

**Ingrese contraseña: ?**

**Repetir contraseña:**

**Crear Usuario**

Figura 4.19: Tab de la extensión. Permite registrarse

Al llenar los campos y enviar la petición a la autoridad certificadora de registrase, 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.

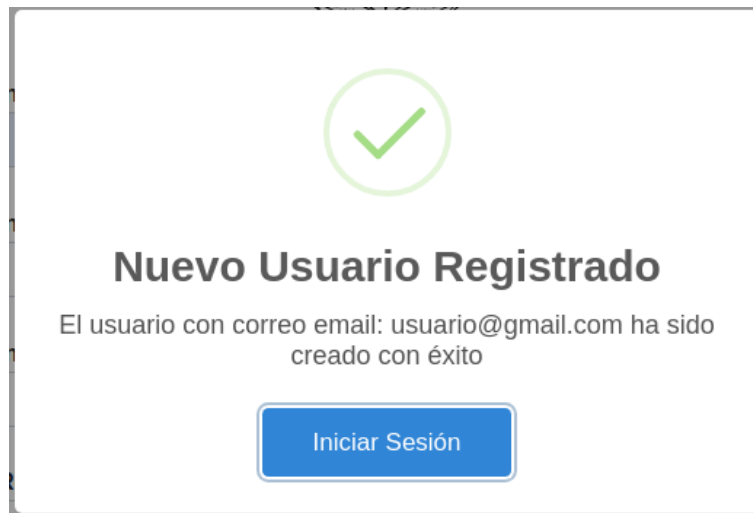


Figura 4.20: Mensaje de éxito tras obtener el certificado de la autoridad y guardarlo en el storage de Google Chrome.

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).

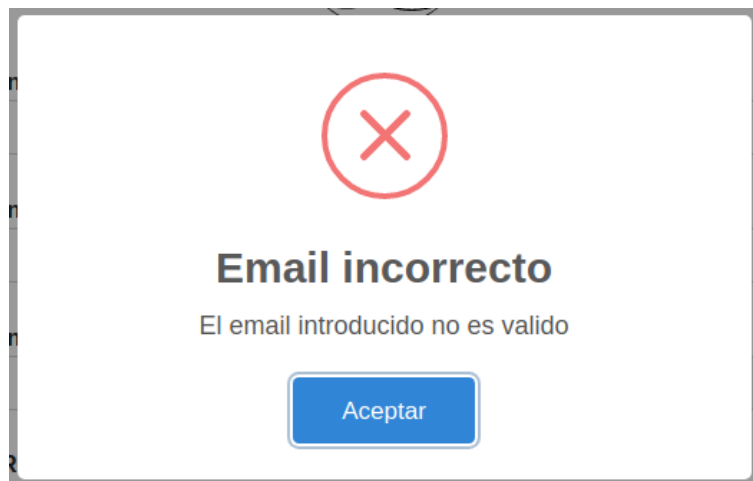


Figura 4.21: Mensaje de error tras detectar que no se cumplen los requisitos para el correo ingresado.

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



contraseña introducida.

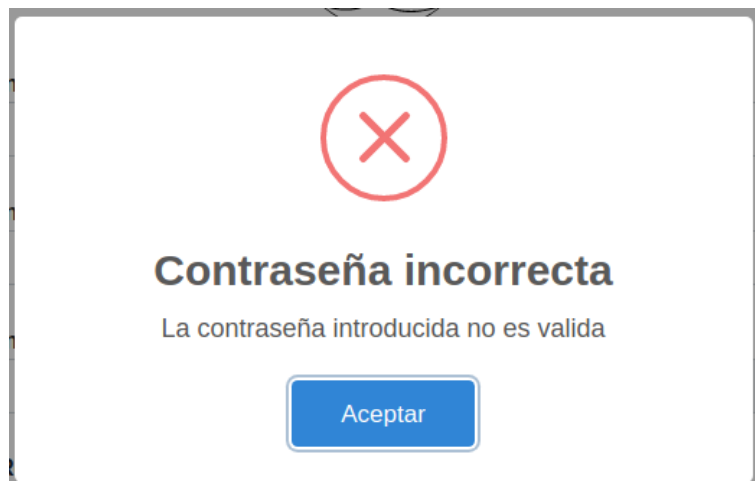


Figura 4.22: Mensaje de error tras detectar que no se cumplen los requisitos para la contraseña ingresada.

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

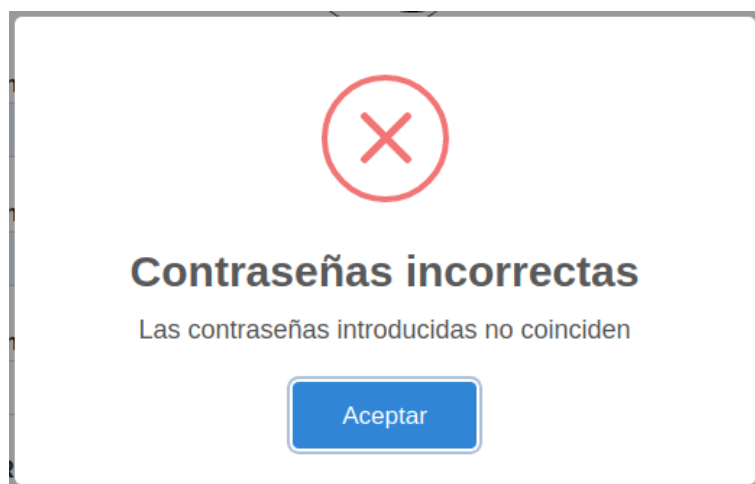


Figura 4.23: Mensaje de error tras detectar que no coinciden el par de contraseñas ingresadas.

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.

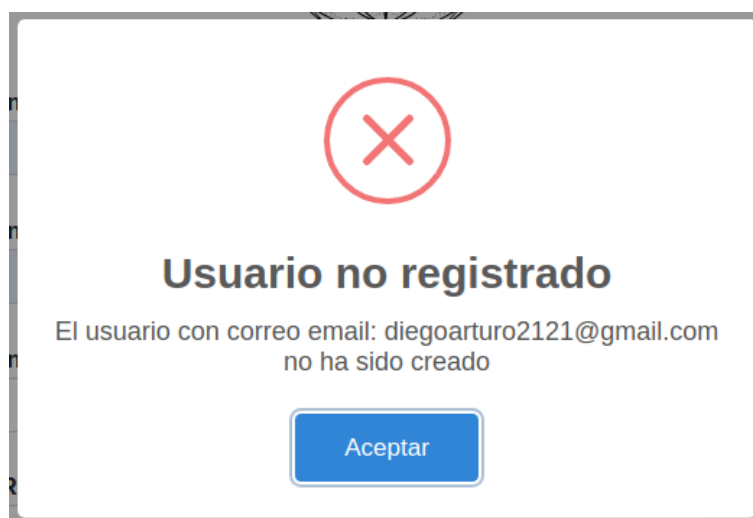


Figura 4.24: Mensaje de error tras detectar que el usuario ya se encuentra registrado.

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 white rectangular form with a thin grey border. At the top center is a circular logo featuring a stylized plant with leaves and the text 'Ch & W' in the center. Below the logo, the text 'Ingrese email:' is followed by a text input field containing 'ejemplo@ejemplo.com'. Below that, the text 'Ingrese contraseña:' is followed by a password input field containing 'password'. To the right of the password field is a small grey button with an eye icon. At the bottom center is a wide blue button with rounded corners and a thin white border, containing 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.

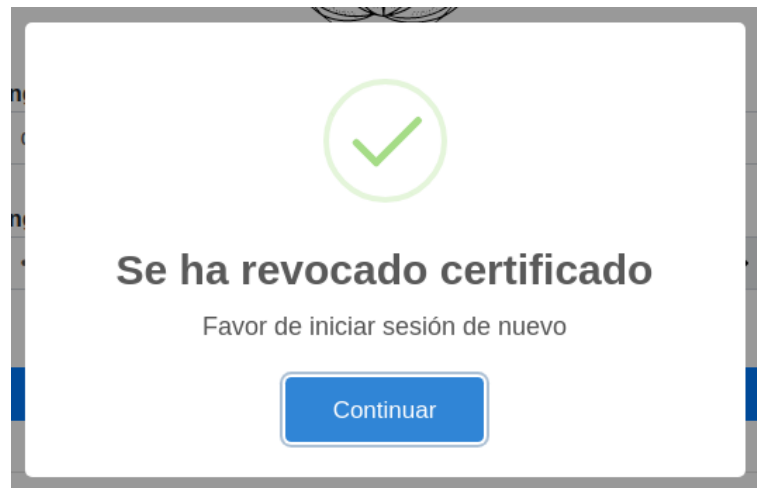


Figura 4.26: Mensaje de éxito al revocar el certificado de un usuario.

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

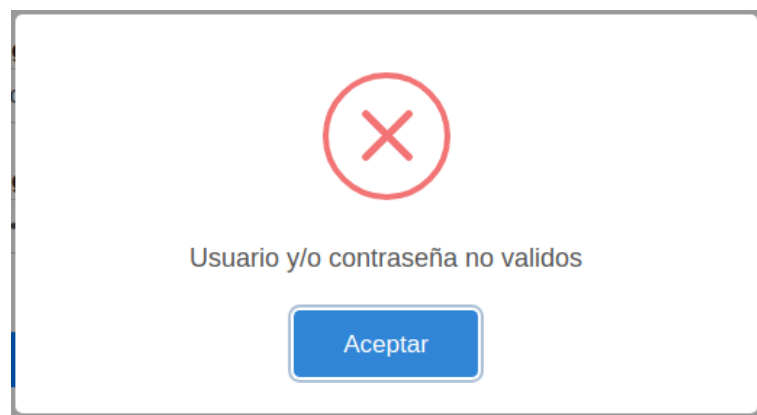


Figura 4.27: Mensaje de error al tratar de revocar el certificado de un usuario con credenciales incorrectas.

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 cachea la

petición e inyecta el resultado del Chaffing en el header de la petición hecha previamente. Una vez inyectado 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 igual manera 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 autenticador

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

- **Código Autenticador** 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  $Pattern[random] \neq 0$ ;  
8 |    $Pattern[random] \leftarrow '1'$ ;  
9 end
```

**Algoritmo 1:** getPattern: Generación de patrón de chaffing

```

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

$$C_k = \text{MITZ057abZ251}$$

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

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

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 caracter 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 colocar el byte siguiente correspondiente al arreglo del certificado. En caso análogo, si se encuentra con un 0, entonces se colocará el siguiente caracter 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 caracteres 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.

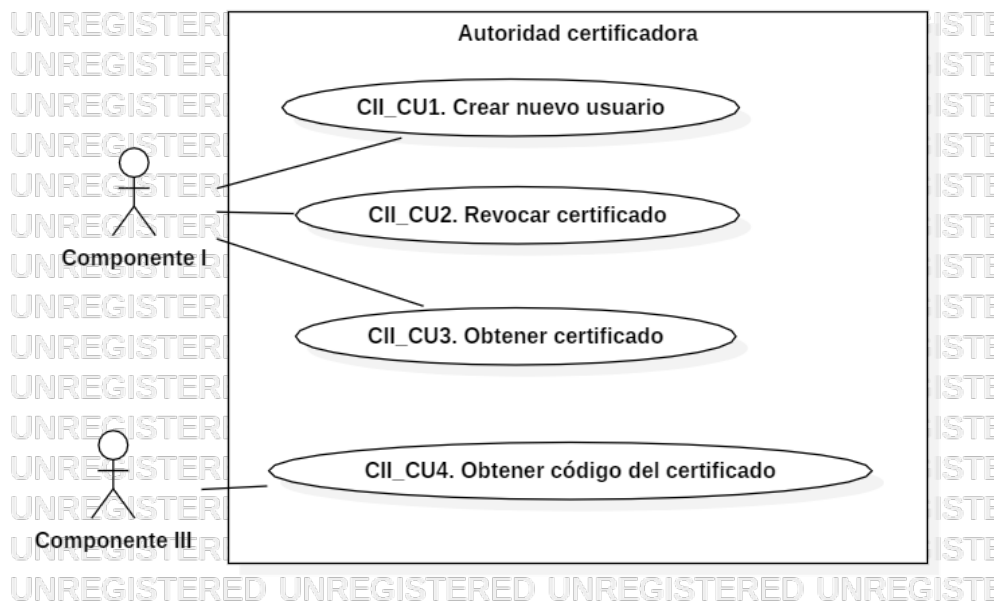


Figura 4.29: Diagrama de casos de uso del Componente II.



#### 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	<b>CII_RN1</b> <b>CII_RN2</b>
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 autentificador.
Post-condiciones	-
Reglas del negocio	<b>CII_RN1</b> <b>CII_RN2</b> <b>CII_RN4</b> <b>CII_RN6</b>
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 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 nuevo certificado al usuario.
6. **El servidor autentificador** 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 Trayectorai 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	<b>CII_CU3.</b>
Post-condiciones	-
Reglas del negocio	<b>CII_RN1</b> <b>CII_RN2</b> <b>CII_RN3</b>
Errores	El servidor autenticador 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 autenticador** valida al usuario registrado en su base de datos.
3. **El servidor autenticador** 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 autenticador** no encuentra el usuario registrado en su base de datos.
3. **El servidor autenticador** 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 autenticador.
Post-condiciones	-
Reglas del negocio	<b>CII_RN1</b> <b>CII_RN2</b> <b>CII_RN6</b>
Errores	El servidor autenticador no puede responder.

Cuadro 4.11: Descripción CU: CII\_CU4

... Trayectoria principal ...

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

... Fin Trayectoria principal ...

... Trayectoria alternativa 1 ...

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

... Fin Trayectoria alternativa 1 ...

### 4.2.2. Diagrama de flujo.

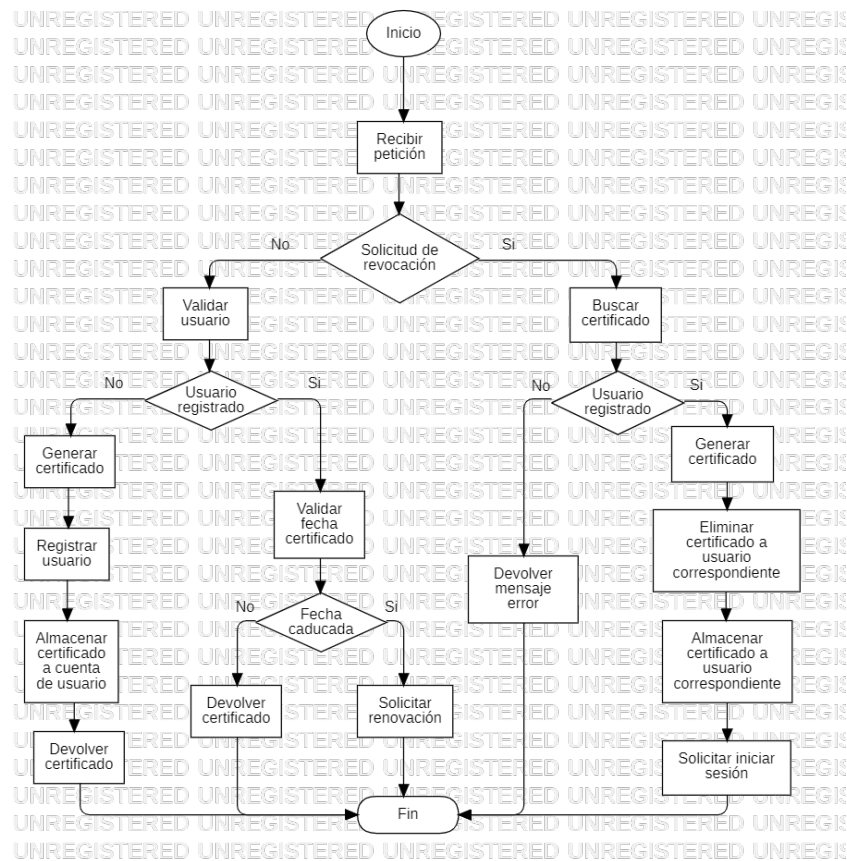


Figura 4.30: Diagrama de flujo de Componente II.

#### 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.

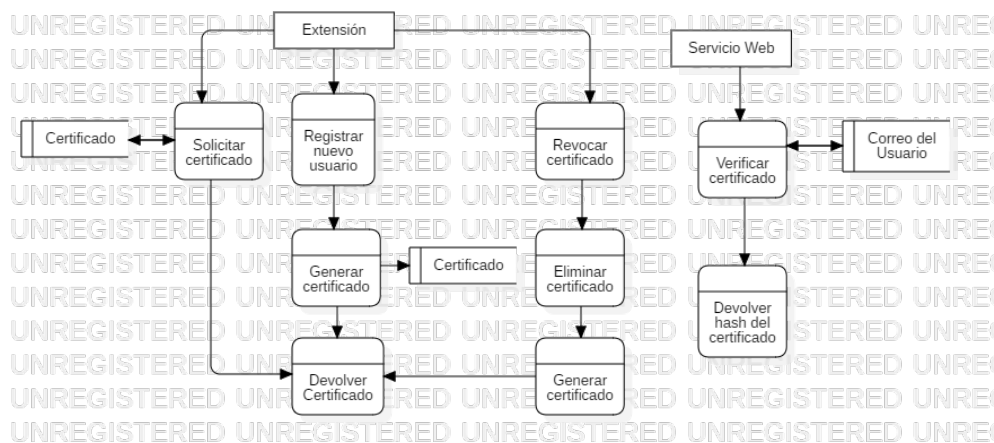


Figura 4.31: Diagrama de flujo de datos del Componente II.

#### 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.

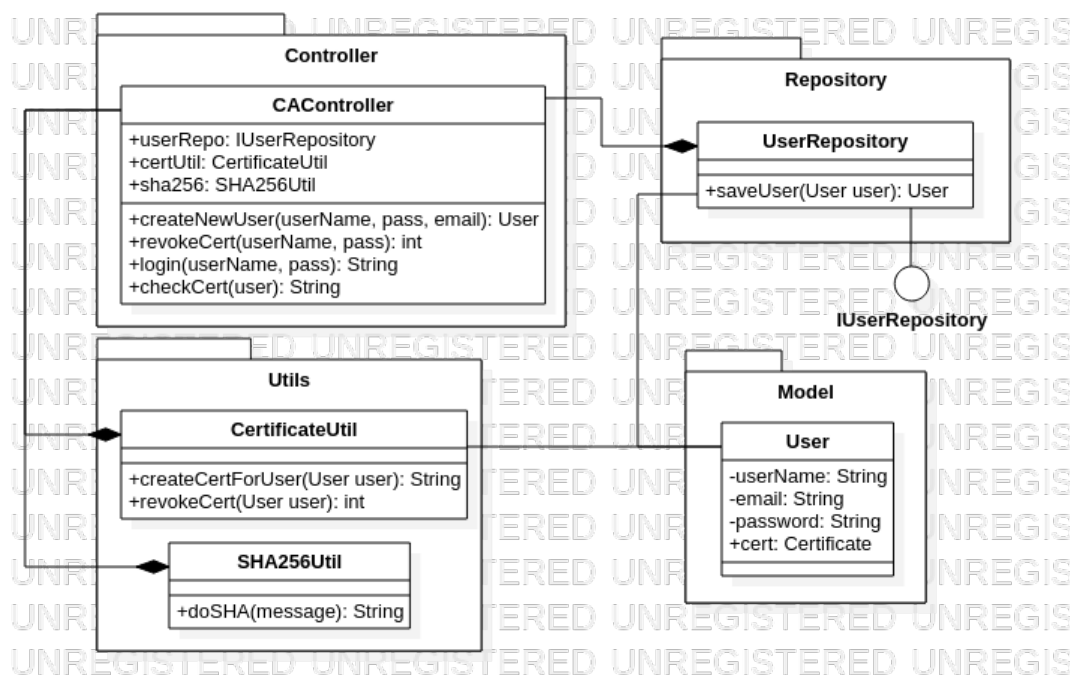


Figura 4.32: Diagrama de clases de Componente II.

##### 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

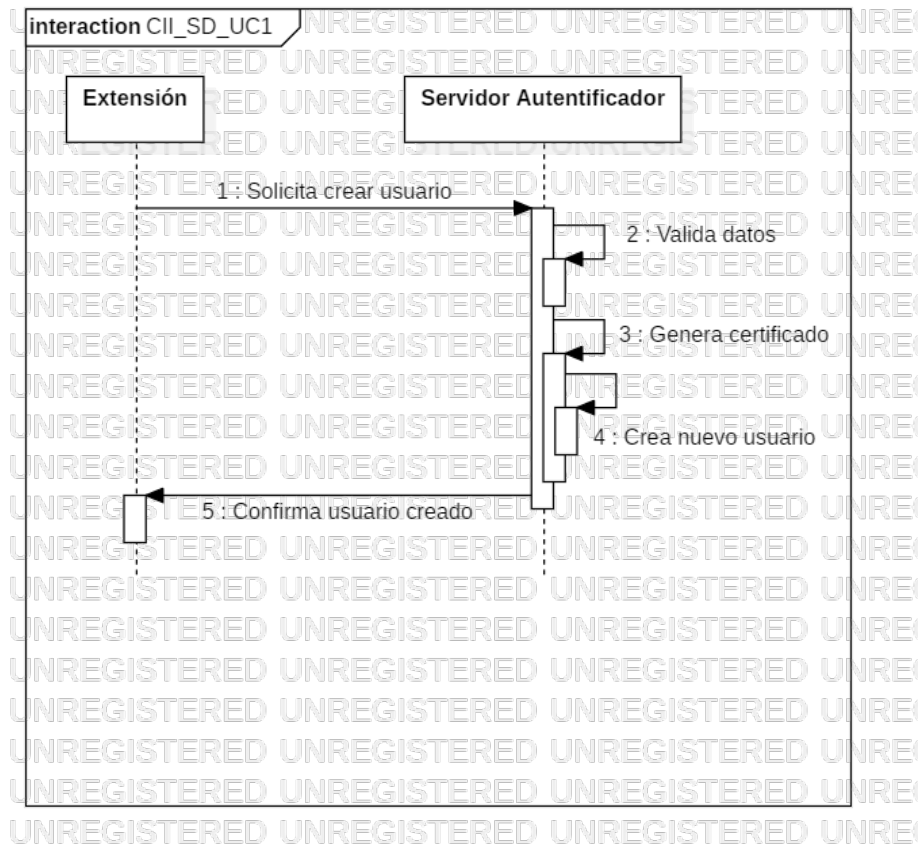


Figura 4.33: Diagrama de secuencia del CII\_CU1. Crear Nuevo usuario.

**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

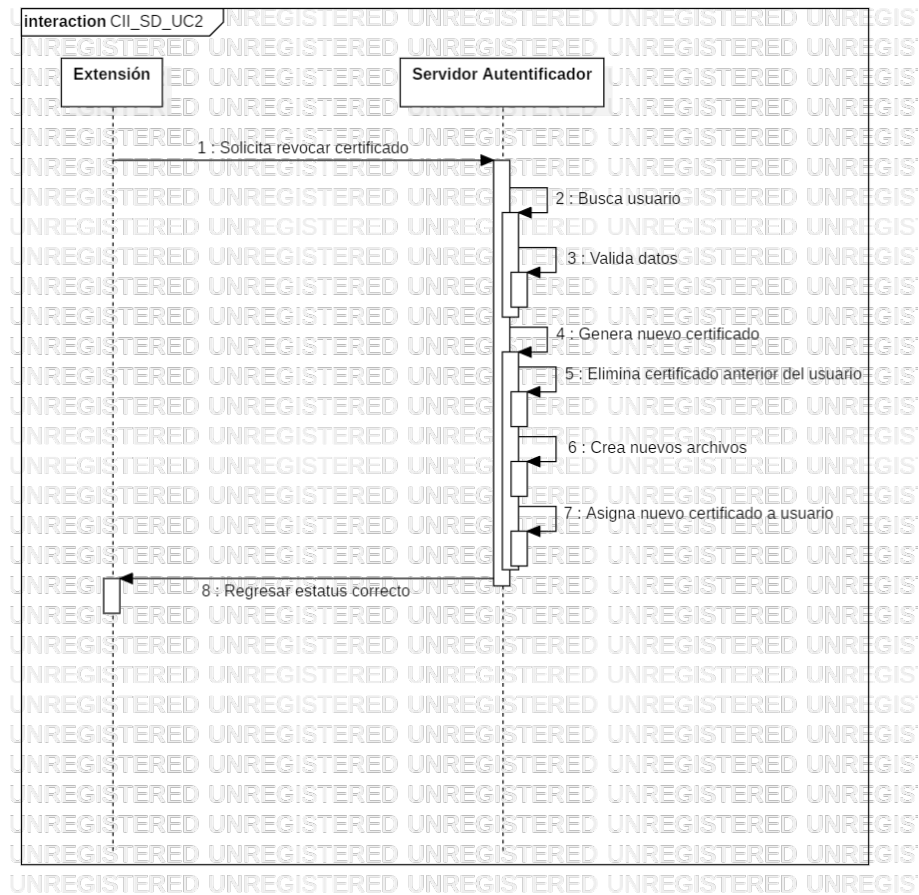


Figura 4.34: Diagrama de secuencia del CII.CU2. Revocar certificado

**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

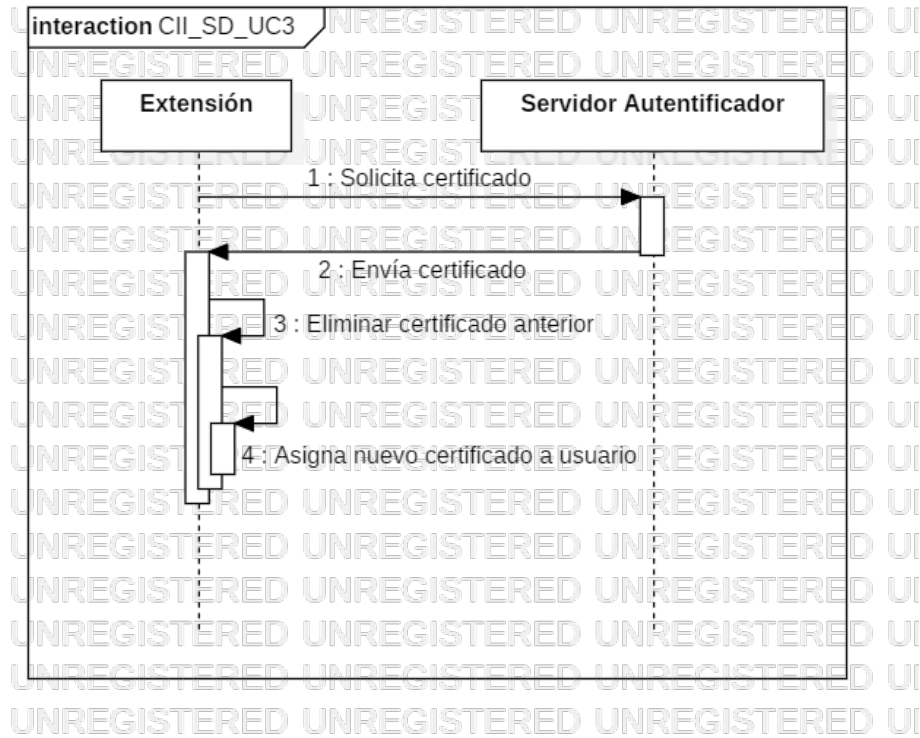


Figura 4.35: Diagrama de secuencia del CII\_CU3. Obtener certificado.

**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

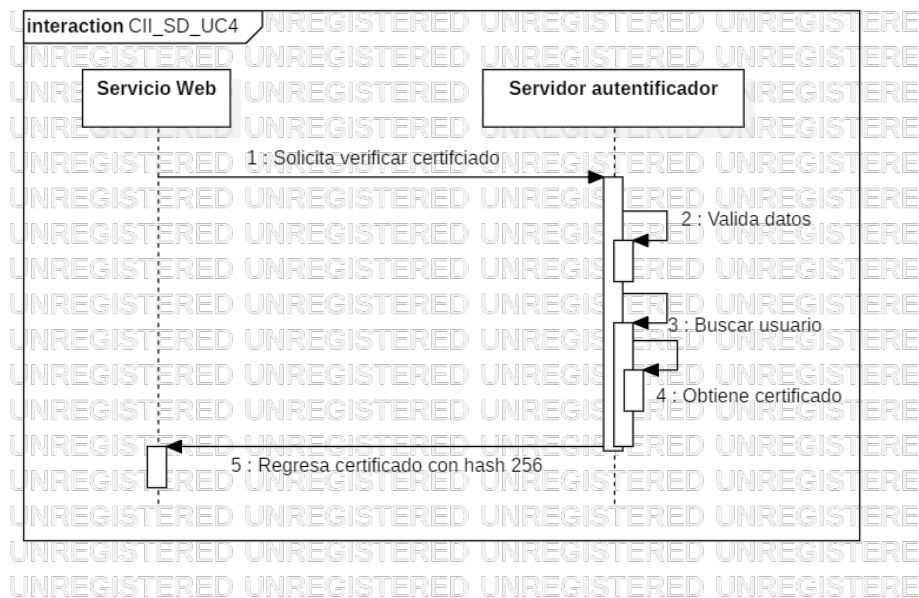


Figura 4.36: Diagrama de secuencia del CII-CU4. Verificar certificado.

**Descripción:** Este diagrama servirá 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.

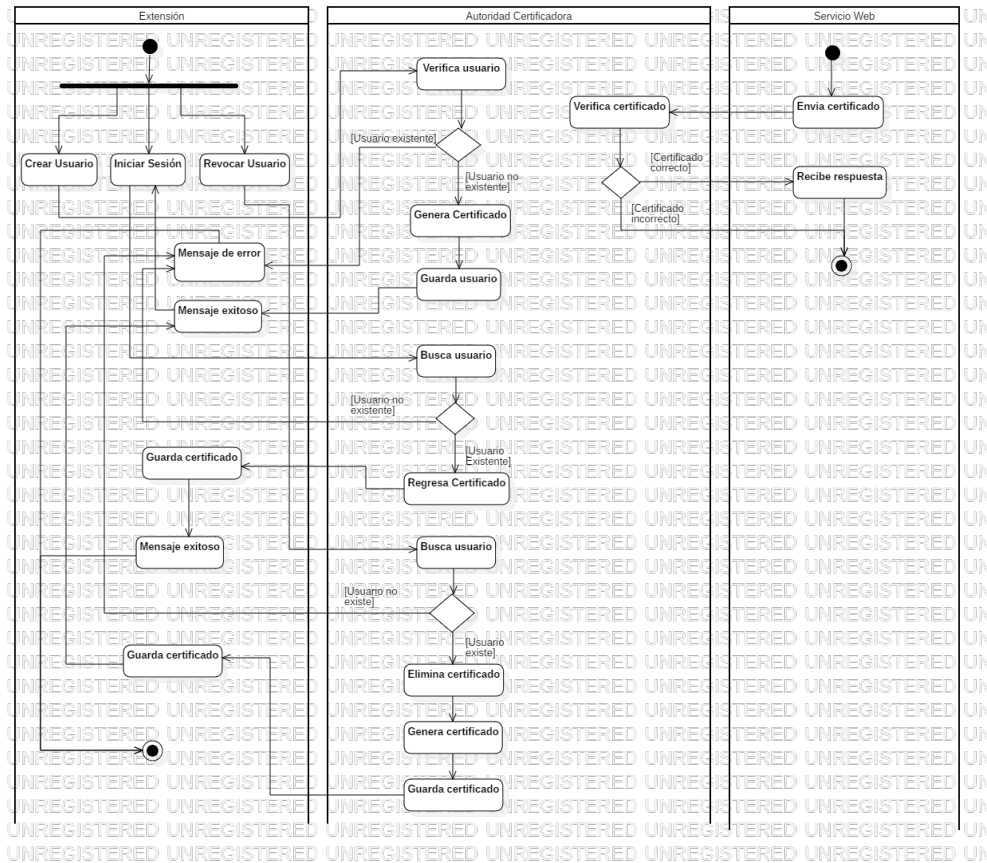


Figura 4.37: Diagrama de actividades de Componente II.

##### 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.

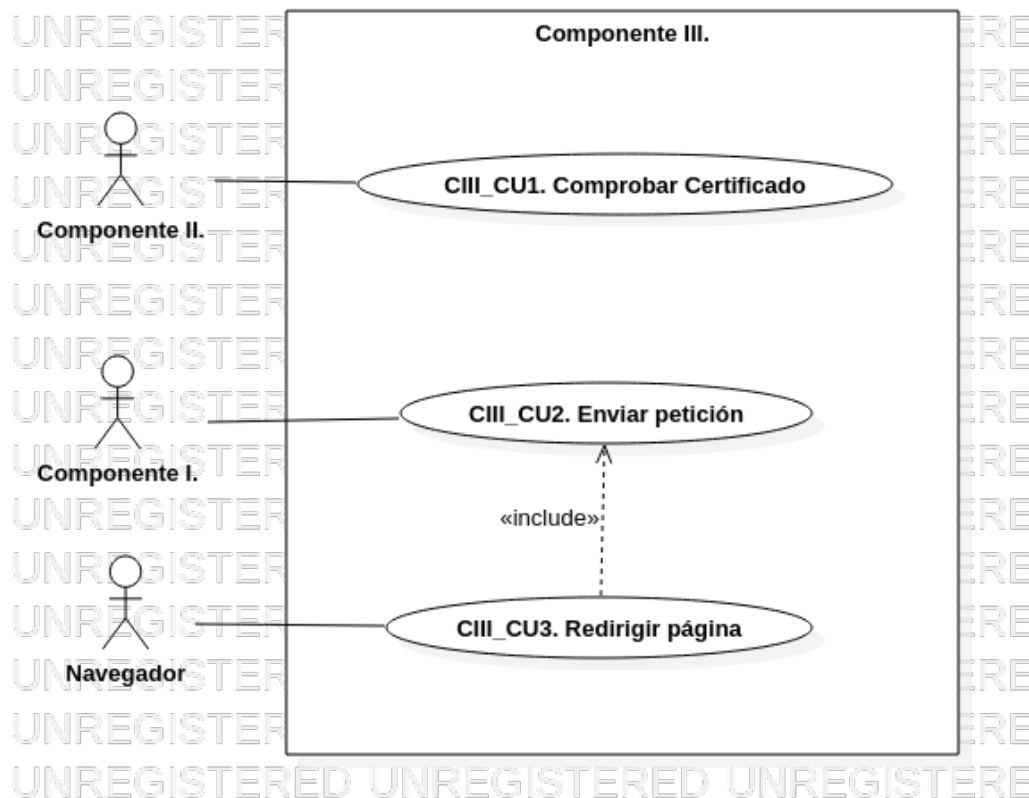


Figura 4.38: Diagrama de caso de uso del componente III

#### 4.3.1.1. Descripción de diagrama de casos de uso.

Caso de uso: CIII_CU1. Comprobar certificado.	
Concepto	Descripción
Actor	Componente II. Servidor autenticador
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 autenticador del usuario.
Pre-condiciones	-
Post-condiciones	-
Reglas del negocio	<b>CIII_RN3</b>
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 autenticador**.
2. **El servidor autenticador** 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 autenticador**
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 autenticador**.
2. **El servidor autenticador** 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 autentificador***
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 autentificador***.
2. ***El servidor autentificador*** no encuentra 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	<b>CIII_CU4</b>
Reglas del negocio	<b>CIII_RN4</b>
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	<b>CIII_CU3.</b>
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.

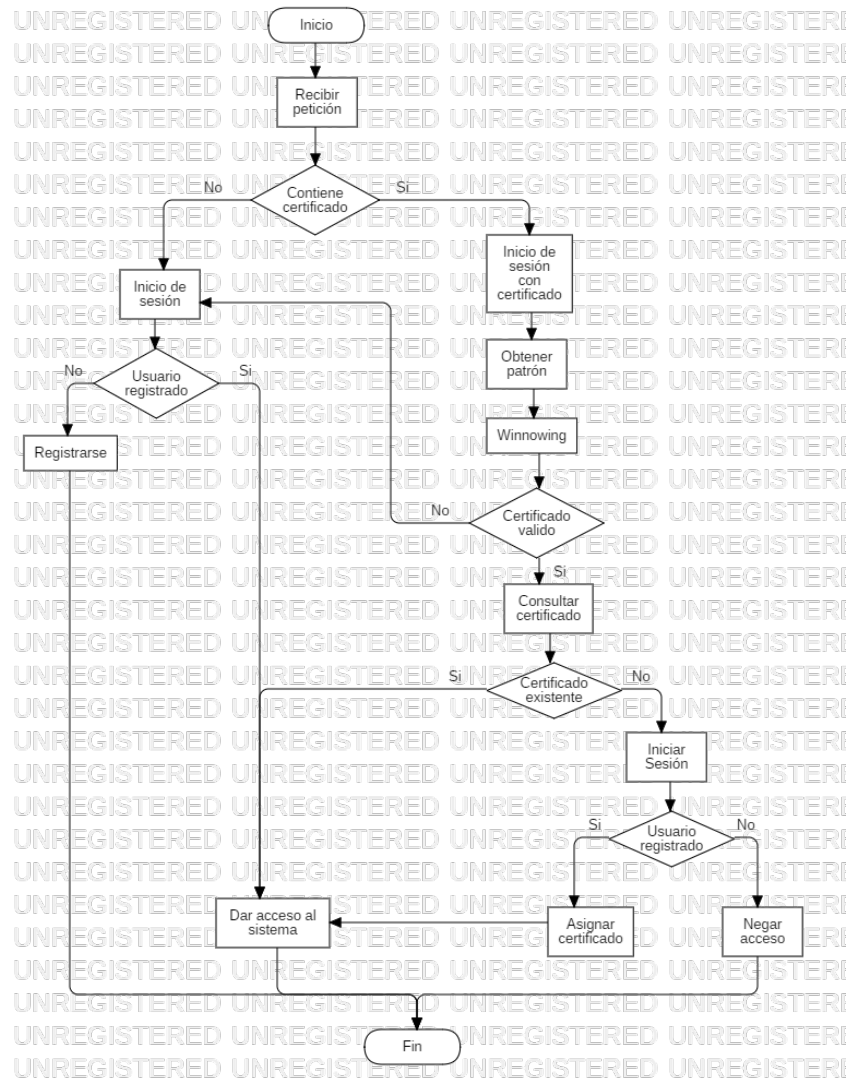


Figura 4.39: Diagrama de flujo de Componente III.

#### 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.

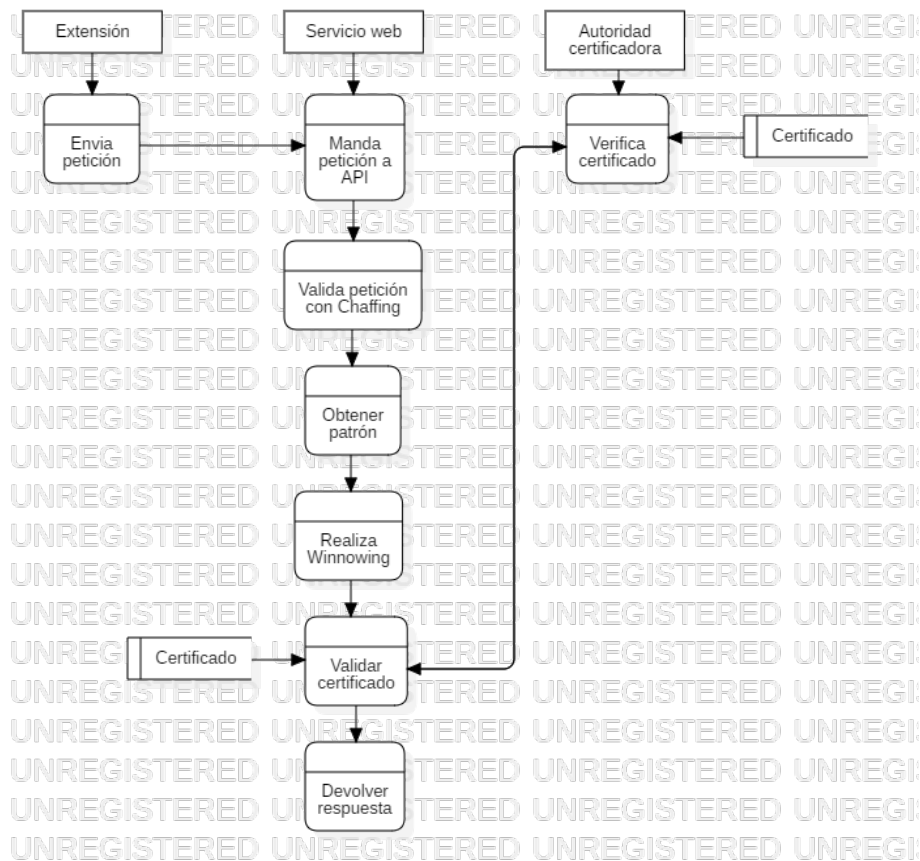


Figura 4.40: Diagrama de flujo de datos de Componente III.

#### 4.3.3.1. Descripción diagrama de flujo de datos.

A continuación tenemos el diagrama de flujo de datos, donde podemos ver claramente como viaja la información principal a través 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.

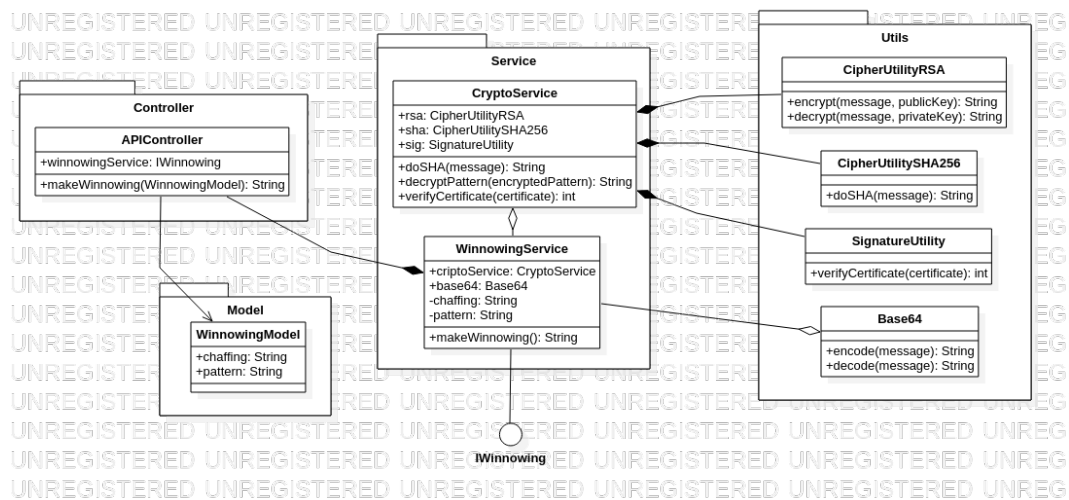


Figura 4.41: Diagrama de Clases de componente 3.

##### 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: *SignatureUtility*

#### 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.

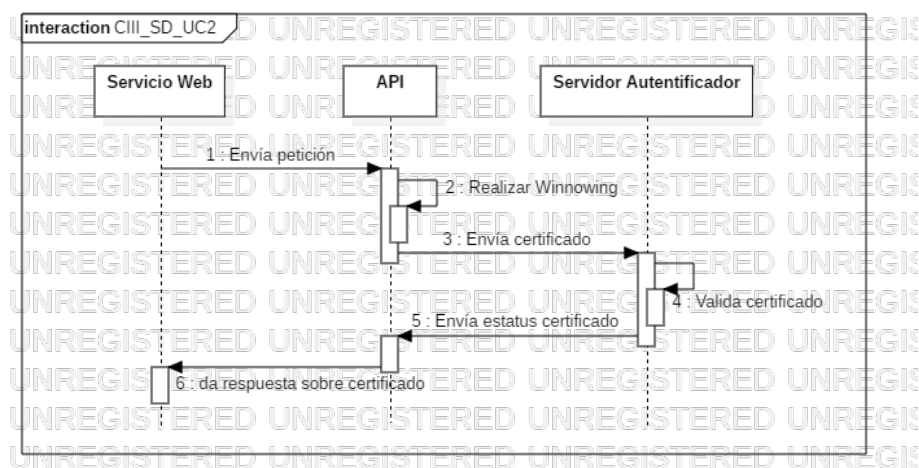


Figura 4.42: Diagrama de Secuencia de Caso de Uso 1. Comprobar certificado.

#### 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 Autenticador 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 Autenticador.

#### Diagrama de Secuencia 2. Enviar Petición.

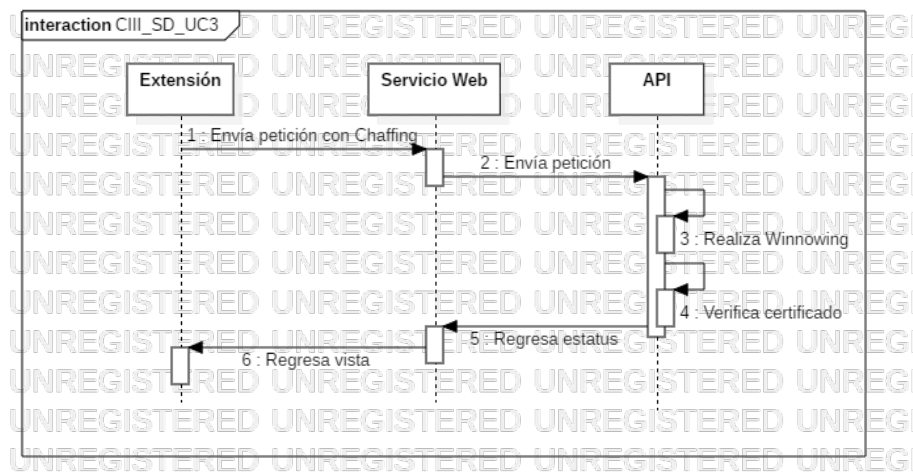


Figura 4.43: Diagrama de Secuencia de Caso de Uso 2. Enviar petición.

#### 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.

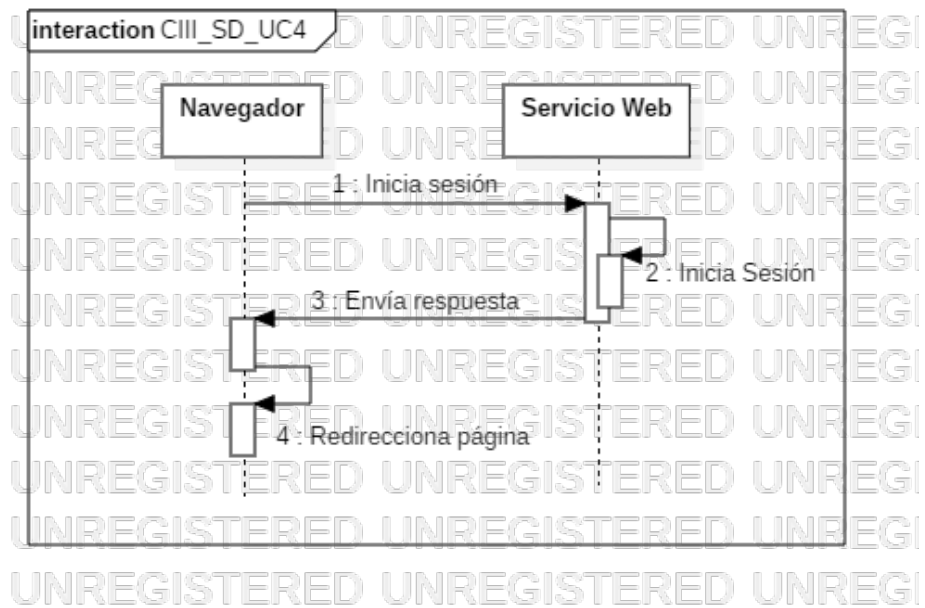


Figura 4.44: Diagrama de Secuencia de Caso de Uso 3. Redirigir página.

#### 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

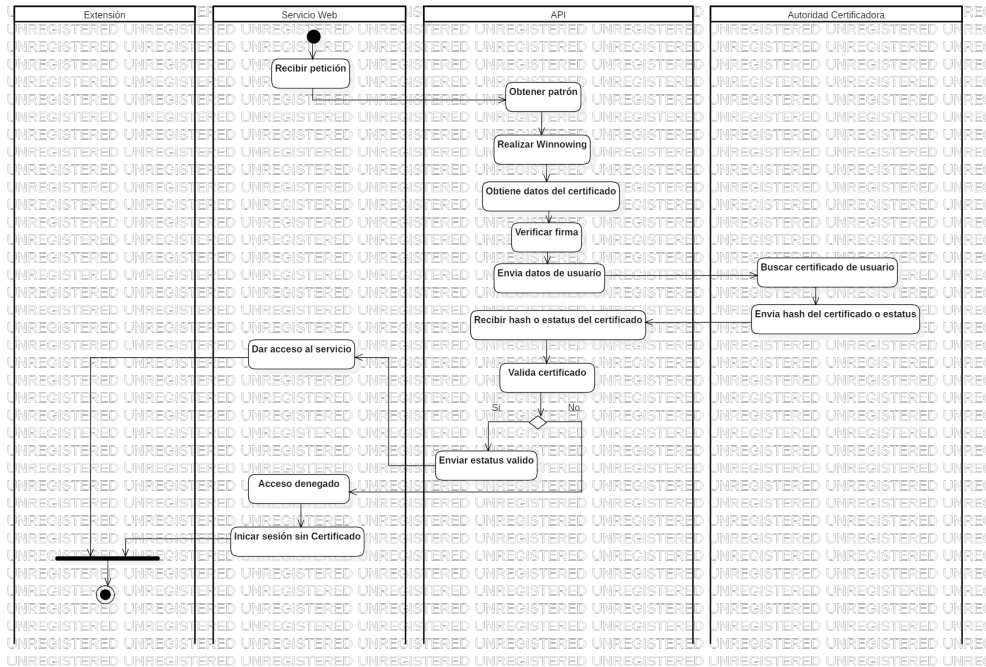


Figura 4.45: Diagrama de Actividades de componente 3.

#### 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.



The image shows a login form titled "Inicia sesión" in teal. Below the title is a subtitle "Inicia sesión para vincular las credenciales" in blue. There are two input fields: the first is labeled "Nombre de Usuario" with a person icon, and the second is labeled "Contraseña de Usuario" with a key icon. At the bottom is a teal button with the text "INICIAR SESIÓN" and a right-pointing arrow.

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.

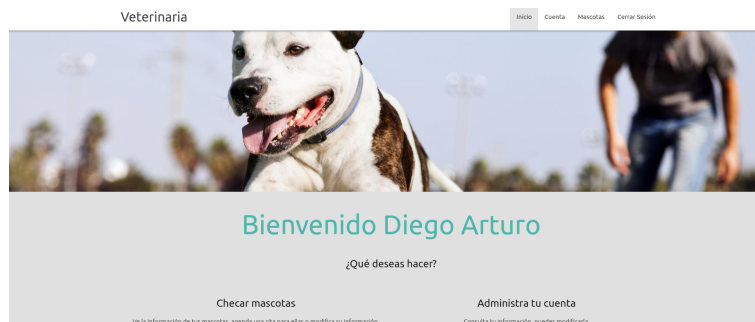


Figura 4.48: Interfaz del servicio web con acceso de una cuenta.

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 dejó su sesión iniciada en otra máquina pública. El tener la capacidad de generar otro certificado, le permite al servicio web no dar acceso si el certificado recibido se encuentra revocado, pidiendo así 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.

La imagen muestra una interfaz de inicio de sesión con un fondo gris. En la parte superior, el título "Inicia sesión" está en un color verde azulado. Debajo, hay dos campos de entrada: el primero está precedido por un ícono de persona y etiquetado como "Nombre de Usuario"; el segundo está precedido por un ícono de llave y etiquetado como "Contraseña de Usuario". Ambos campos tienen líneas horizontales para escribir. Justo debajo de estos campos, el mensaje "Credenciales Inválidas" aparece en un color rojo. En la parte inferior, hay un botón rectangular de color verde azulado con el texto "INICIAR SESIÓN" y un símbolo de flecha hacia la derecha.

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:

<p><b>Data:</b> chaffing, patternCipher, aesCipher, privateKey</p> <p><b>Result:</b> cert,header</p> <pre> 1 <i>chaffingDecode</i>[] <math>\leftarrow</math> <i>base64.decode(chaffing)</i>; 2 <i>aesKey</i> <math>\leftarrow</math> <i>rsa.decipher(aesCipher, privateKey)</i>; 3 <i>pattern</i>[] <math>\leftarrow</math> <i>aes.decipher(patternCipher, aesKey)</i>; 4 <i>cert</i>[]; 5 <i>header</i>[]; 6 <i>i</i> <math>\leftarrow</math> 0; 7 <b>while</b> <i>i</i> &lt; <i>pattern.length</i> <b>do</b> 8     <b>if</b> <i>pattern</i>[<i>i</i>] == 1 <b>then</b> 9         <i>header.add(chaffingDecode[i])</i>; 10    <b>else</b> 11        <i>cert.add(chaffingDecode[i])</i>; 12    <b>end</b> 13    <i>i</i> <math>\leftarrow</math> <i>i</i> + 1; 14 <b>end</b> </pre>
---

**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 registrarte 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 autenticación.

### **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 llevo a cabo para la creación del componente II. La implementación de este componente se desarrollo 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 mas 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 código

#### 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 Componente 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.

- Project: **Maven Project**
- Language: **Java**
- Spring Boot: **2.2.0**
- Project 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 inicializ. 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 [68].

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-*

*winnowing*. Dicha clase hace uso a su vez de la clase *CryptoService.java* encargada de realizar el descifrado 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 autenticación 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 autenticación 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 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 autenticador, 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 interactúa 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 Autenticador y API.

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

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

## 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 prueba al algoritmo de *Winnowing*, el cual incluye: descifrado 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.

# 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] Chaney, D. (2012). The Music Industry in the Digital Age: Consumer Participation in Value Creation. *International Journal of Arts Management*, (1), pp. 15.
- [4] Sutskever, I., Martens, J., Hinton, G. E. (2011, January). Generating text with recurrent neural networks. In *ICML*.
- [5] Kim, Y. (2014). Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.
- [6] Monteith, K., Martinez, T. R., & Ventura, D. (2012, May). Automatic Generation of Melodic Accompaniments for Lyrics. In *ICCC*, pp. 87-94.
- [7] Nikolov, N. I., Malmi, E., Northcutt, C. G., Parisi, L. (2020). Conditional Rap Lyrics Generation with Denoising Autoencoders. *arXiv preprint arXiv:2004.03965*.
- [8] Baker, F. A. (2015). What about the music? Music therapists' perspectives on the role of music in the therapeutic songwriting process. *Psychology of Music*, 43(1), pp. 122-139.
- [9] Guerrero, J. (2012). El género musical en la música popular: algunos problemas para su caracterización. *Trans. Revista transcultural de música*, (16), pp. 1-22.

- [10] Wang, A., & Cho, K. (2019). Bert has a mouth, and it must speak: Bert as a markov random field language model. arXiv preprint arXiv:1902.04094.
- [11] Honnibal, M., & Montani, I. (2017). spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing.
- [12] 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].
- [13] 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](http://delta.cs.cinvestav.mx/~mcintosh/cellularautomata/Summer_Research_files/maquinasef.pdf) [Último acceso: 15 de diciembre del 2020].
- [14] 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].
- [15] S. Leijnen, F. van Veen (2020, mayo), The Neural Network Zoo, [En línea]. Disponible: [https://www.researchgate.net/publication/341373030\\_The\\_Neural\\_Network\\_Zoo](https://www.researchgate.net/publication/341373030_The_Neural_Network_Zoo) [Último acceso: 20 de noviembre del 2020].
- [16] 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].
- [17] 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].
- [18] 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].

- [19] Escribir Canciones (2008), Estructura y elementos de una canción, [En línea]. Disponible: <https://dle.rae.es> [Último acceso: 2 de diciembre del 2020].
- [20] 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].
- [21] 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].
- [22] 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].
- [23] 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].
- [24] 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].
- [25] 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].
- [26] J. R. Aguirre, *Seguridad Informática y Criptografía*, 2da edición, Madrid, España: Universidad Politécnica de Madrid, 2006.
- [27] 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].
- [28] A. Beutelspacher, *Cryptology*, edición 1994, Washington, Estados Unidos de América: Mathematical Association of America, 1994.

- [29] BBVA de México (2018), ¿Es seguro guardar las contraseñas en el navegador? [En línea] Disponible: <https://www.bbva.com/es/seguo-guardar-contrasenas-navegador/> [Último acceso: 15 de noviembre de 2019]
- [30] CCM (2017), El protocolo HTTP, [En línea]. Disponible: <https://es.ccm.net/contents/264-el-protocolo-http>. [Último acceso: 6 Mayo 2019].
- [31] S. Díaz Santiago, "Generacion De Sucesiones Criptográficamente Fuertes", tesis de Maestría, División de ciencias básicas e ingeniería, UAM, D.F., México, 2005.
- [32] 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].
- [33] 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].
- [34] J. D. Gauchat, *El gran libro de HTML5, CSS3 y Javascript*, 1ra edición, Barcelona, España: MARCOMBO S.A., 2012.
- [35] S. Goldwasser y S. Micali, "Probabilistic encryption", *Journal of Computer and System Science*, vol. 28, no. 4, pp. 270–299, 1984.
- [36] desarrolloweb.com (2019), Manual de JQuery, [En línea]. Disponible: [http://dmaspv.com/files/page/07042011180222\\_manual%20de%20jquery%20en%20pdf%20desarrollowebcom.pdf](http://dmaspv.com/files/page/07042011180222_manual%20de%20jquery%20en%20pdf%20desarrollowebcom.pdf). [Último acceso: 2 de noviembre del 2019].
- [37] Google code (2019), crypto-js, [En línea]. Disponible: <https://code.google.com/archive/p/crypto-js/>. [Último acceso: 5 de mayo del 2019].
- [38] 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].

- [39] 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](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].
- [40] 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].
- [41] 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.
- [42] 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].
- [43] A. Maiorano, *Criptografía: Tecnicas De Desarrollo Para Profesionales*, 1ra edición, D.F., México: Alfaomega, 2009.
- [44] A. Maurya<sup>1</sup>, 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
- [45] A. J. Menezes, P. v. Oorschot y S. Vanstone, *Handbook of Applied Cryptography*, 1ra edición, Florida, Estados Unidos de América: CRC Press, 1996.
- [46] Mozilla (2019, mayo 2), JavaScript. [En línea]. Disponible: <https://developer.mozilla.org/en-US/docs/Web/JavaScript> [Último acceso: 5 de mayo del 2019].
- [47] Mozilla, (2019, 19 marzo), FileSystem, [En línea]. Disponible: <https://developer.mozilla.org/es/docs/Web/API/FileSystem>. [Último acceso: 5 de mayo del 2019].

- [48] R. S. Pressman, *Ingeniería del Software. Un enfoque práctico*, 7ma edición, D.F, México: McGraw-Hill, 2010.
- [49] R. L. Rivest, "Chaffing and Winnowing: Confidentiality without Encryption", Laboratorio de ciencias de la computación del MIT, Massachusetts, Estados Unidos de América, 1998.
- [50] A. Romero Mier y Terán y M. A. Vasquéz Martínez (2016, abril 19), Aspectos Básicos de la Seguridad en Aplicaciones Web. [En línea]. Disponible: <https://www.seguridad.unam.mx/historico/documento/index.html-id=17>. [Último acceso: 5 de mayo del 2019].
- [51] Universidad Politécnica de Madrid (2004), Criptografía, [En línea]. Disponible: [http://www.dma.fi.upm.es/recursos/aplicaciones/matematica\\_discreta/web/aritmetica\\_modular/criptografia.html](http://www.dma.fi.upm.es/recursos/aplicaciones/matematica_discreta/web/aritmetica_modular/criptografia.html). [Último acceso: 5 de mayo del 2019].
- [52] MongoDB. (2019), ¿Qué es MongoDB?, [En línea]. Disponible: <https://www.mongodb.com/es>. [Último acceso: 2 de noviembre del 2019].
- [53] 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](https://www.verisign.com/es_LA/website-presence/online/ssl-certificates/index.xhtml). [Último acceso: 5 de mayo del 2019].
- [54] Wireshark (2019), What is Wireshark?, [En línea]. Disponible: [https://www.wireshark.org/docs/wsug\\_html\\_chunked/ChapterIntroduction.html#ChIntroWhatIs](https://www.wireshark.org/docs/wsug_html_chunked/ChapterIntroduction.html#ChIntroWhatIs). [Último acceso: 20 abril 2019].
- [55] 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](https://www.cisco.com/c/es_mx/support/docs/security-vpn/kerberos/16087-1.html#whatisit)
- [56] 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.
- [57] 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]



- [58] Universitat de València (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]
- [59] J. C. Teague, *Speaking in Styles: Fundamentals of CSS for Web Designers*. 1ra edición. California EEUU: New Riders, 2009.
- [60] Bootstrap (2019), About Bootstrap [En línea]. Disponible: <https://getbootstrap.com/docs/4.3/about/overview/> [Último acceso: 6 noviembre 2019]
- [61] Node.js Foundation (2019), Acerca de Node.js [En línea]. Disponible: <https://nodejs.org/es/about/> [Último acceso: 6 de noviembre 2019]
- [62] Noteworthy Programming Masterpiece (2019), openssl-nodejs [En línea]. Disponible: <https://www.npmjs.com/package/openssl-nodejs> [Último acceso: 6 noviembre 2019]
- [63] D. Bell, M. Parr, *Java para estudiantes*. 3ra edición. México: Pearson Education, 2003.
- [64] Walls, Craig, *Spring in Action*. 5ta edición. EEUU: Manning, 2018.
- [65] The Apache Software Foundation (2019), Apache Tomcat [En línea]. Disponible: <http://tomcat.apache.org/index.html> [Último acceso: 6 noviembre 2019]
- [66] Oracle (2018), MySQL [En línea]. Disponible: <https://www.oracle.com/mysql/> [Último acceso: 6 noviembre 2019]
- [67] 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]
- [68] Oracle (2010), Package java.security [En línea]. Disponible: [https://docs.oracle.com/javase/7/docs/api/java/security/package-summary.html#package\\_description](https://docs.oracle.com/javase/7/docs/api/java/security/package-summary.html#package_description) [Último acceso: 6 noviembre 2019]
- [69] Security Appspot (2012), About vsftpd [En línea]. Disponible: <https://security.appspot.com/vsftpd.html#about> [Último acceso: 10 de noviembre 2019]

- [70] Johannes A. Buchmann, Evangelos Karatsiolis, A.W.: Introduction to Public Key Infrastructures. Springer (2013)
- [71] 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]
- [72] E-Reding (2001), Protocolo HTTP [En línea]. Disponible: <http://bibing.us.es/proyectos/abreproy/11372/fichero/Memoria%252F05+-+Protocolo+HTTP.pdf> [Último acceso: 21 de octubre de 2019]
- [73] 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](https://www.ibm.com/support/knowledgecenter/es/ssw_ibm_i_73/rzal5/rzal5overview.htm) [Último acceso: 28 de octubre de 2019]
- [74] EcuRed (2011), Protocolo SMTP [En línea]. Disponible: [https://www.ecured.cu/Protocolo\\_SMTP](https://www.ecured.cu/Protocolo_SMTP) [Último acceso: 05 de noviembre de 2019]
- [75] 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]