



TRABAJO FIN DE GRADO  
INGENIERÍA INFORMÁTICA

# Implementación de IMAP sobre una pasarela de datos

---

para la recepción de correos electrónicos en una red aislada  
mediante AIRGAP

**Autor**

Francisco José Moreno Vílchez

**Directores**

Rafael Alejandro Rodríguez Gómez



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍAS INFORMÁTICA Y DE  
TELECOMUNICACIÓN

Granada, Septiembre de 2023







# Implementación del protocolo IMAP para la recepción de correos en una red aislada mediante Air Gap

Francisco José Moreno Vílchez

**Palabras clave:** airgap, pasarela, IMAP, email, seguridad

## Resumen

En el entorno actual de ciberseguridad, donde la protección de datos y la integridad de la información son esenciales, surge la necesidad de establecer medidas sólidas para salvaguardar la infraestructura y la confidencialidad de las organizaciones.

Una de las medidas es el más comunes para mantener los datos de una organización seguros es utilizar *airgap*. Hacer uso de una red aislada, mantiene los datos y servicios inaccesibles desde otras redes o desde Internet, ya sea física o lógicamente.

En este trabajo se aborda el desafío de permitir un acceso seguro a uno los recursos externos, más utilizados en el mundo, el correo electrónico, habilitando el acceso y su gestión desde una red interna aislada de Internet, sin comprometer la seguridad de la red ni la confidencialidad de los datos de residen en esta. Para ello se proponer implementar una pasarela de de correos electrónicos que permita esta comunicación sin perder el enfoque en la seguridad de los datos internos.

Se diseñará una pasarela de correo electrónico a partir de la pasarela de ficheros de *jtsec*, una empresa que hace uso de este tipo de sistemas, sacando de esta los requisitos que la hacen funcionar como pasarela y pasar las auditorías necesarias para ser utilizada en un entorno real.

La pasarela de correos electrónicos actúa como un filtro inteligente que regula el flujo de correos, autenticando a los usuarios autorizados y analizando exhaustivamente el contenido y los adjuntos de los correos entrantes. De esta manera se consigue habilitar a los usuarios de la red aislada separada del Internet mediante airgap acceder a sus correos electrónicos.

Se implementa una API para permitir a otras aplicaciones hacer uso de la pasarela, por ejemplo, un cliente web para facilitar el uso de la pasarela a los usuarios finales. También se desarrolla una interfaz de usuario para permitir a los usuarios hacer uso de sus correos electrónicos de manera sencilla.

Es repositorio con el código de la pasarela de correos puede ser encontrado en github en la siguiente dirección: <https://github.com/morevi/pasarela-imap>.



# Implementation of IMAP protocol for mail reception in an isolated network using Air Gap

Francisco José Moreno Vílchez

**Keywords:** airgap, pasarela, IMAP, email, security

## Abstract

In today's cybersecurity environment, where data protection and information integrity are essential, there is a need to establish robust measures to safeguard the infrastructure and confidentiality of organizations.

One of the most common measures to keep an organization's data secure is the use of *airgap*. Making use of an isolated network keeps data and services inaccessible from other networks or from the Internet, either physically or logically.

This paper addresses the challenge of enabling secure access to one of the most widely used external resources in the world, e-mail, by enabling access and management from an internal network isolated from the Internet, without compromising the security of the network or the confidentiality of the data residing on it. To this end, it is proposed to implement an e-mail gateway that allows this communication without losing focus on the security of internal data.

An e-mail gateway will be designed based on the file gateway of *jtsec*, a company that makes use of this type of systems, taking from it the requirements that make it work as a gateway and pass the necessary audits to be used in a real environment.

The e-mail gateway acts as an intelligent filter that regulates the flow of e-mails, authenticating authorized users and thoroughly analyzing the content and attachments of incoming e-mails. This enables users of the isolated network separated from the Internet via airgap to access their e-mails.

An API is implemented to allow other applications to make use of the gateway, e.g. a web client to facilitate the use of the gateway by end users. A user interface is also developed to allow users to easily make use of their e-mails.

The repository with the mail gateway code can be found in github at <https://github.com/morevi/pasarela-imap>.





---

Yo, Francisco José Moreno Vílchez, alumno del Grado en Ingeniería Informática de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI 76588371G, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Francisco José Moreno Vílchez

Granada a 4 de Septiembre de 2023



---

D. **Rafael Alejandro Rodríguez Gomez**, Profesor del Área de Telemática del Departamento de Teoría de la Señal, Telemática y Comunicaciones de la Universidad de Granada.

**Informa:**

Que el presente trabajo, titulado ***Implementación del protocolo IMAP para la recepción de correos en una red aislada mediante Air Gap***, ha sido realizado bajo su supervisión por **Francisco José Moreno Vílchez**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expide y firma el presente informe en Granada a 4 de Septiembre de 2023.

**El director:**

**Rafael Alejandro Rodríguez Gómez**



# Agradecimientos

Quiero expresar mi profundo agradecimiento a todas las personas que han contribuido de alguna manera en la realización de mi trabajo de fin de grado.

- En primer lugar, quiero agradecer a *jtsec*, la empresa donde realicé mi práctica profesional y que me brindó la oportunidad de aplicar los conocimientos adquiridos durante mi formación académica. Su apoyo y orientación fueron fundamentales para el éxito de este proyecto.
- También quiero agradecer a mis compañeros de piso y amigos por su comprensión y apoyo durante esta etapa de mi vida. Gracias por aguantar mis momentos de estrés y por ser una fuente constante de motivación.
- Por último, pero no menos importante, quiero expresar mi profundo agradecimiento a mi familia. Gracias por estar siempre a mi lado, por apoyarme en cada paso de mi camino académico y por ser mi mayor fuente de inspiración. Sé que no ha sido fácil aguantarme durante estos años de estudio, pero su amor incondicional me ha impulsado a seguir adelante y a alcanzar mis metas.



# Índice general

<b>1. Introducción</b>	<b>1</b>
1.1. Motivación . . . . .	1
1.2. Solución propuesta . . . . .	2
1.3. Estructura de este documento . . . . .	2
<b>2. Contexto</b>	<b>5</b>
2.1. Pasarelas de datos . . . . .	5
2.2. Correo electrónico . . . . .	9
2.3. Estado del arte . . . . .	12
<b>3. Objetivos y planificación</b>	<b>15</b>
3.1. Objetivo general . . . . .	15
3.2. Objetivos específicos . . . . .	15
3.2.1. Estudio del EMAIL . . . . .	16
3.2.2. Análisis de la pasarela de <i>jtsec</i> . . . . .	16
3.2.3. Seguridad, monitorización y administración . . . . .	16
3.2.4. Diseño . . . . .	16
3.2.5. Desarrollo de nuestra pasarela . . . . .	17
3.3. Planificación . . . . .	17
<b>4. Análisis</b>	<b>21</b>
4.1. Pasarela de <i>jtsec</i> . . . . .	21
4.2. Actores . . . . .	26
4.3. Requisitos funcionales . . . . .	26
4.4. Requisitos no funcionales . . . . .	27
<b>5. Diseño</b>	<b>29</b>
5.1. Casos de uso . . . . .	29
5.1.1. Actores . . . . .	29
5.1.2. Casos de uso . . . . .	29
5.2. Materiales . . . . .	37

<b>6. Implementación</b>	<b>41</b>
6.1. Sprint 4. Montaje de la infraestructura . . . . .	41
6.2. Sprint 5. Configuración por defecto y administración . . . . .	44
6.3. Sprint 6. Login, carpetas y listado de carpetas . . . . .	49
6.4. Sprint 7. Listado y descarga de correos . . . . .	52
6.5. Sprint 8. Operaciones de gestión sobre correos . . . . .	58
6.6. Sprint 9. MUA web. . . . .	59
6.6.1. Descripción de la interfaz . . . . .	61
6.6.2. Review . . . . .	64
<b>7. Conclusiones y trabajos futuros</b>	<b>65</b>
7.1. Conclusiones . . . . .	65
7.2. Trabajos futuros . . . . .	66
<b>Bibliografía</b>	<b>70</b>
<b>Glosario</b>	<b>71</b>
<b>A. Guía de instalación de la pasarela de correos</b>	<b>73</b>
A.1. Instalación requerimientos de sistema . . . . .	73
A.2. Instalación <i>backend</i> . . . . .	74
A.3. Instalación <i>frontend</i> . . . . .	75
A.4. Instalación de los dispositivos en sus respectivas redes . . . . .	75
<b>B. Guía de uso de la pasarela de correos</b>	<b>77</b>



# Índice de figuras

3.1. Planificación. . . . .	19
4.1. Estructura de la pasarela de <i>jtsec</i> . . . . .	22
4.2. Uso de la API para subir y descargar un fichero en la pasarela de <i>jtsec</i> . . . . .	23
5.1. Esquema de casos de uso . . . . .	30
6.1. Raspberry PI Imager . . . . .	42
6.2. Diagrama de los dispositivos y protocolos usados en la pasa- rela de correos . . . . .	42
6.3. Disposición cruzada de los cables para el montaje de la pasa- rela de correos . . . . .	43
6.4. Estructura objetivo de la pasarela de correos . . . . .	45
6.5. Uso de CURL para obtener el listado de carpetas . . . . .	53
6.6. Uso de CURL para pedir los correos . . . . .	55
6.7. Uso de CURL para la descarga de un correo . . . . .	58
6.8. Uso de CURL para marcar como no leído . . . . .	60
6.9. Uso de curl CURL para borrar un correo . . . . .	61
6.10. Formulario de inicio de sesión . . . . .	62
6.11. Listado de carpetas del usuario . . . . .	62
6.12. Listado de correos del usuario . . . . .	63
6.13. Sección con el contenido del correo . . . . .	64
6.14. Interfaz de usuario completa . . . . .	64
B.1. Configuración de ejemplo . . . . .	77
B.2. Interfaz sin iniciar sesión . . . . .	78
B.3. Inicio de sesión fallido . . . . .	78
B.4. Interfaz tras realizar inicio de sesión correcto . . . . .	78
B.5. Uso del botón de borrado en la lista de correos . . . . .	79
B.6. Uso del botón para marcar como no leído en la lista de correos	80
B.7. Interfaz de usuario completa con carpetas, correos y vista de correo . . . . .	80



# Índice de tablas

2.1. Comparación de métodos tradicionales de transferencia de datos a una red aislada . . . . .	13
5.1. Caso de uso: inicio de sesión . . . . .	31
5.2. Caso de uso: selección de carpeta . . . . .	32
5.3. Caso de uso: selección de correo . . . . .	33
5.4. Caso de uso: descarga de adjunto . . . . .	34
5.5. Caso de uso: borrado de correo . . . . .	35
5.6. Caso de uso: marcado como “no leído” . . . . .	36
5.7. Caso de uso: configuración de usuarios . . . . .	36



# Índice de Códigos

4.1. Código de jtsec para realizar una petición de sha256 a un fichero en la red aislada . . . . .	25
6.1. Configuración de usuario para permitir uso del cable . . . . .	43
6.2. Instalación de requisitos del sistema . . . . .	43
6.3. Inicialización del proyecto y requisitos de la pasarela . . . . .	44
6.4. Script de punto de entrada del servidor del cable en la red externa . . . . .	46
6.5. Primera implementación de la ruta / para la autenticación frente a la configuración . . . . .	46
6.6. Ejemplo de configuración . . . . .	48
6.7. Script de configuración y valores por defecto . . . . .	48
6.8. Enumerador de tipos de paquetes . . . . .	49
6.9. Clase RequestController con métodos de recepción y de listado de carpetas . . . . .	50
6.10. Clases Request y JsonRequest que encapsulan el uso de Re- questController . . . . .	51
6.11. Ruta / para pedir las carpetas . . . . .	52
6.12. Método de RequestController para pedir el listado de correos	53
6.13. Ruta para pedir los correos en una carpeta . . . . .	54
6.14. Función para hacer uso del antivirus . . . . .	55
6.15. Método de RequestController que descarga y analiza los ad- juntos de un correo . . . . .	56
6.16. Definición de la ruta para descargar un correo electrónico . .	57
6.17. Definición de una ruta para descargar los adjuntos de un correo	57
6.18. Uso de imap-tools para modificar los flags y borrar un correo	59
A.1. Instalación de requisitos del sistema . . . . .	73
A.2. Configuración de usuario para permitir uso del cable . . . . .	74
A.3. Uso de pip para instalar las dependencias . . . . .	74
A.4. Configuración de un servicio de systemd para poder ejecutar la pasarela en segundo plano . . . . .	74
A.5. Configuración de NGINX para servir archivos estáticos . . . .	75
A.6. Lanzado de los servicios . . . . .	76



# Capítulo 1

## Introducción

### 1.1. Motivación

En la actual era digital, las conexiones entre los dispositivos electrónicos que utilizamos en el día a día crean el ciberespacio, un lugar donde la gestión segura y eficiente de información es una prioridad fundamental para organizaciones de todo tipo. Y aunque todo aquello conectado a Internet forma parte del ciberespacio, existen redes independientes que no forman parte de Internet. Estos pueden ser sistemas como las instalaciones de enriquecimiento nuclear de Irán, o también procesadores y controladores integrados en diferentes plataformas, desde vehículos hasta satélites. Esa desconexión del sistema de Internet es nombrada como “airgap” [1].

A menudo, los sistemas clasificados están aislados y se construyen como una red independiente. No siempre tiene por qué ser así, pueden estar físicamente aislados, pero también pueden tener una separación lógica.

Es en este punto donde las pasarelas de datos desempeñan un papel fundamental. Una pasarela de datos actúa como un intermediario que filtra y gestiona el tráfico entre la red aislada y los recursos externos. Esto permite inspeccionar y validar la información antes de que ingrese a la red aislada de Internet, previniendo la propagación de posibles amenazas. Por otro lado, la implementación de airgap implica crear un espacio físico, o lógico, completamente aislado entre la red interna y externa, reduciendo drásticamente las posibilidades de que los ataques cibernéticos crucen esta barrera.

La motivación detrás de este trabajo radica en abordar el desafío de permitir un acceso seguro a uno los recursos externos, más utilizados en el mundo, el correo electrónico, desde una red interna aislada de Internet, sin comprometer la seguridad de la red ni la confidencialidad de los datos de residen en esta.

Estamos hablando del correo electrónico. En la actualidad, este medio se ha convertido en un pilar fundamental de las comunicaciones empresariales y personales [2]. Su importancia radica en su capacidad para trascender

barreras geográficas y temporales, permitiendo la transferencia instantánea de información y facilitando la colaboración a nivel global. Además de su eficacia en la comunicación, el correo electrónico es una herramienta de gran utilidad para el intercambio de documentos, la coordinación de proyectos y la toma de decisiones. Sin embargo, su valor inherente también lo convierte en un objetivo atractivo para la emisión de amenazas cibernéticas. Por lo tanto, garantizar un acceso seguro y controlado a los correos electrónicos se vuelve esencial para mantener la confidencialidad, integridad y disponibilidad de la información en un entorno cada vez más digital y conectado.

La importancia del correo electrónico radica en su capacidad para ser el conducto principal de comunicación en el mundo moderno. Sin embargo, acceder de manera segura a correos electrónicos desde la red aislada de una organización presenta un desafío significativo. Esta dificultad surge de la necesidad de equilibrar la conveniencia de acceso a la información con la necesidad crítica de proteger los activos digitales. La red aislada, que a menudo alberga datos sensibles y confidenciales, requiere una salvaguardia robusta contra las amenazas cibernéticas que pueden infiltrarse a través de los correos electrónicos. Por lo tanto, encontrar soluciones que permitan un acceso fluido y seguro a los correos electrónicos desde el entorno interno se convierte en un imperativo para garantizar la continuidad de los negocios y la protección de la información estratégica.

## 1.2. Solución propuesta

La solución que se propone aborda los desafíos planteados anteriormente al implementar una pasarela de datos, cuyo enfoque se centre en permitir el acceso seguro a los correos electrónicos desde la red aislada de la organización. Esta pasarela de datos representa un paso crucial hacia la mejora de la calidad de trabajo manteniendo la seguridad y la protección de la información confidencial, a la vez que permite el flujo de correos electrónicos.

La pasarela de correos electrónicos se concibe como una barrera de seguridad inteligente que regula el flujo de mensajes electrónicos hacia y desde la red. Diseñada meticulosamente, su función principal radica en autenticar y filtrar a los usuarios permitidos, evitando así el acceso no autorizado a la red. Al recibir correos electrónicos entrantes, la pasarela realiza un análisis anti-virus al contenido del mensaje y sus adjuntos. La pasarela de datos examina cada adjunto en busca de contenido potencialmente dañino, garantizando así que solo los archivos seguros sean entregados a los destinatarios finales.

## 1.3. Estructura de este documento

Este apartado presenta los capítulos de este documento, ofreciendo para cada uno de ellos, un breve resumen para permitir al lector moverse por



el documento con facilidad.

1. **Capítulo 1. Introducción.** Este capítulo, inicia presentando la motivación detrás de este trabajo, se plantea el problema a abordar y se examinan las soluciones actuales. Después de identificar el problema, se propone nuestra propia solución. El capítulo concluye proporcionando esta estructura general de los capítulos.
2. **Capítulo 2. Contexto y estado del arte.** En este capítulo, se proporciona al lector una comprensión profunda de qué es una pasarela, cómo se utilizan y cuáles son sus requisitos típicos. También se explora el origen de los correos electrónicos y los protocolos comunes asociados. En la sección final de estado del arte, se realiza una comparativa de las soluciones convencionales en este para este problema.
3. **Capítulo 3. Objetivos y planificación.** Este capítulo se dedica a exponer los objetivos fundamentales de nuestro proyecto. Además, se presenta una planificación del trabajo detallada, que abarcará las etapas clave del trabajo que se pretende realizar.
4. **Capítulo 4. Análisis.** Durante este capítulo del proyecto, se llevará a cabo un análisis exhaustivo del software de una pasarela de datos en la vida real. Se identificarán los componentes clave que sustentan su funcionamiento. Como resultado de este análisis, se extraen los requisitos funcionales y no funcionales que orientarán la construcción de nuestra propia pasarela.
5. **Capítulo 5. Diseño.** Se amplían los requisitos que se definen en el capítulo anterior, son traducidos en casos de uso específicos. Detallamos cómo el sistema se comportará durante cada uno de estos casos. También se discuten los materiales y recursos necesarios para llevar a cabo el desarrollo.
6. **Capítulo 6. Implementación.** En este capítulo, se inicia el proceso de desarrollo de la pasarela de correos. El capítulo queda dividido en *sprints*, cada uno de los cuales abarca una porción del trabajo realizado. Al final de cada sprint, se presentarán los resultados del sprint para exhibir el estado alcanzado.
7. **Capítulo 7. Conclusión y trabajos futuros.** El capítulo final, donde se resumirá el trabajo realizado y se comparan los logros obtenidos y el grado de cumplimiento alcanzado. También se señalarán posibles líneas de trabajo futuro, así como alternativas a considerar en relación con este proyecto.



## Capítulo 2

# Contexto

### 2.1. Pasarelas de datos

Las pasarelas de intercambio seguro de información se pueden definir como aquellos dispositivos de protección de perímetro que se utilicen para controlar la información que se transfiere entre diferentes redes o sistemas, dentro de una organización o entre varias. Estas pasarelas son capaces de actuar como filtros de seguridad que permiten únicamente el flujo de información autorizado, mientras bloquean cualquier intento de acceso o transferencia no autorizado o malicioso [3].

Dentro del contexto de la seguridad de la información, perímetro se refiere al límite, ya sea físico o digital, entre una red y, el mundo exterior, u otra red, incluyendo las diferentes medidas de seguridad tomadas para su protección.

Como se ha mencionado, esas medidas pueden ser tanto físicas, tratándose por ejemplo de paredes, puertas bajo llave, código, personal de seguridad, etc, como virtuales, así como, portales de acceso con credenciales, el uso de *proxies*, y cualquier tipo de medida digital. La existencia de un perímetro, da a entender que dentro de este se encuentran los llamados activos, sean estos datos confidenciales, o servicios con cierta privacidad, a los que se quieren proteger de los accesos y manipulación no deseados.

Para lograr esa protección, dentro del perímetro se establecen políticas de seguridad, y con estas se establecen cuáles son los preciados activos que se tendrán que proteger, para más tarde decidir qué medidas y dispositivos serán necesario implementar e instalar para mantener esa seguridad.

Las políticas de seguridad pueden definir reglas y procedimientos que se aplican sobre los empleados, usuarios y sistemas involucrados, con el fin de garantizar la confidencialidad, integridad y disponibilidad de los datos y servicios de una organización. Cualquier organización o entidad que trabaje con información sensible necesitará aplicar políticas de seguridad, esto incluye empresas, instituciones educativas o gubernamentales o cualquier otro tipo de organización. Además de para proteger la información, las políti-

cas de seguridad ayudan a cumplimentar las regulaciones y normas legales vigentes en cuanto a la seguridad de la información.

Algunos ejemplos de políticas de seguridad pueden ser:

- Acceso mediante código o tarjetas en lugar de llaves.
- Concienciación para evitar dar información extra o tener cuidado con la información recibida, para evitar filtrar información o *phishing*.
- Longitud y tipo de caracteres en una contraseña. Favoreciendo la dificultad.
- Distintos niveles de acceso a lugares y servicios.
- Monitorización y registro de información de sistemas.
- Políticas de actualizaciones de sistemas. El servicio y el servidor deben estar actualizados a la última versión, para que estén presentes los últimos parches de seguridad.
- Obligatoriedad de usar HTTPS en los sistemas habilitados.

Las guías de Seguridad de Tecnologías de la Información y la Comunicaciones (STIC) [4], desarrolladas por el Centro Criptológico Nacional (CCN) de España, proporciona recomendaciones, directrices, y políticas en materia de seguridad de la información para la administración pública y otros organismos que manejen información sensible. Además, la guía se alinea con el Esquema Nacional de Seguridad (ENS) para establecer las políticas necesarias para las interconexiones entre organismos internos o externos con los sistemas de la administración pública.

Entre estas guías de estas guías se pueden encontrar recomendaciones de diferentes dispositivos o métodos que se pueden aplicar para cumplir cierto nivel de seguridad según ENS. Niveles que las organizaciones externas a la administración pública deben respetar para poder cooperar con esta. Son el caso, por ejemplo, de proveedores que quieran que sus productos puedan ser usados dentro de la administración pública, o de laboratorios de ciberseguridad que intervengan en el proceso de certificación de productos en metodologías como pueden ser Lince o *Common Criteria* en su esquema español.

Algunos de los dispositivos de protección de perímetro recomendados son:

- Cortafuegos, o *firewall*. Están diseñados para proteger una red de accesos no autorizados. Este, controla el flujo de datos, filtrando los paquetes que entran y salen de la red, permitiendo o denegando el acceso a servicios en función de la política de seguridad. En esencia, el cortafuegos analiza el paquete IP, validando el origen, destino y el puerto, permitiendo bloquear emisor, receptor y aplicación.

- *Proxies*. Suele ser un servidor que actúa intermediario entre los clientes y los servidores destino, recibiendo todas las peticiones del cliente, enviadas a través de determinado protocolo, analizando el contenido de los paquetes TCP/IP y redirigiendo, o bloqueando, la petición al destino original. El *proxy* también recogerá la respuesta, para luego devolverá al cliente original.
- Pasarelas de datos. actúan como filtros para bloquear accesos o transferencias no autorizados o maliciosos, realizando inspecciones posiblemente más elaboradas de la información que fluye a través de ella.

La principal característica de una pasarela, es que establece una separación de redes. La pasarela consigue romper la continuidad de los protocolos de comunicaciones tradicionales del modelo OSI. Los dos dispositivos que suelen formar la pasarela, uno con mayor nivel de seguridad dentro del perímetro, y otro con menor nivel de seguridad fuera de este, no deben ser capaces de interconectarse utilizando conexiones TCP/IP. Si no que lo harán implementando un protocolo específico, desarrollado precisamente (*ad hoc*) para la pasarela en cuestión. De manera que los dispositivos de la red interna, no son direccionables o accesibles mediante TCP/IP desde la red externa.

En segundo lugar, la pasarela establece el filtrado de los contenidos. Al analizarse la información que un cliente quiere transmitir se pueden aplicar diferentes tipos de filtrado, aparte de los vistos en un cortafuegos o un proxy. Dependiendo de la pasarela concreta, y de las políticas de seguridad configuradas sobre misma se aplicaran esos filtros. Uno de los filtros más habituales es que el contenido que se quiera mover desde la red interna a la red externa deberá estar cifrado usando algún sistema de claves pública y privada, para así asegurar que el mensaje llegue de manera confidencial e ilegible al destinatario, y este sea capaz de confirmar la autenticidad del emisor mensaje. También se puede dar el caso de que en el sentido opuesto, desde la red externa hacia la red interna, los datos hayan de pasar por un test de antivirus, para evitar comprometer la red aislada con virus, *malware*, troyanos, gusanos y otros tipos de software malicioso que puedan comprometer a la organización o a los datos y servicios de la red interna.

Por último la configuración de la pasarela debe estar en un lugar seguro, accesible únicamente por el personal autorizado, habitualmente el personal de administración de seguridad o protección. La pasarelas han de ser únicamente configurables desde la red interna, para dificultar en medida el acceso de un agente malicioso a esta ya que desde el exterior ningún dispositivo de la red interna es direccionable. Además, por defecto, los canales de comunicación de la pasarela deben estar deshabilitados, ninguna información debe poder ser transmitida, ni con

el dispositivo apagado, ni durante su instalación ni actualización, de hardware o de software [5].

Las descripciones de otros muchos dispositivos de seguridad pueden ser encontradas en la Guía CCN-STIC-140 [?].

La guía CCN-STIC-811 Interconexión en el ENS [6] recomienda el uso de pasarelas de datos, como medida recomendada, en para aquellas empresas o administraciones que colaboren con la administración pública. En concreto, aquellas que necesiten el cumplimentar el certificado de ENS-Alta para llevar a cabo sus operaciones. No tiene por que limitarse a este tipo de organizaciones, estos dispositivos pueden implantarse sobre cualquier red cuyos activos se benefician de las mencionadas medidas de seguridad.

Se puede dar como ejemplo de organización que hace uso de una pasarela de datos a, *jtsec: beyond it security*. Esta empresa, con sede en Granada, ejerce principalmente como laboratorio de ciberseguridad, e interactúa constantemente con el Organismo de Certificación (OC) en el proceso de certificación de productos en Lince [7] y *Common Criteria*.

Durante el proceso evaluación de los productos, se trata con información de los clientes, de los mismos productos y de la extensa documentación que se elabora sobre las pruebas y vulnerabilidades encontradas. Es por esto que *jtsec* establece sus políticas conforme al ENS para facilitar esa colaboración, confianza con los clientes y los certificadores y validez de los certificados.

Entre los datos que se tratan en su modelo de negocio, encontramos información de los clientes y de sus productos, especificaciones, código, el objeto de evaluación (*TOE*) y los informes de los análisis de vulnerabilidades que se le aplican, cuya confidencialidad requieren altos niveles de seguridad. Para protegerlos, *jtsec* aplica políticas de defensa en profundidad y mantiene los datos más importantes sobre las evaluaciones de ciberseguridad de los productos dentro de una red interna, desconectada de internet.

Si queremos hablar de la pasarela de *jtsec*, se puede mencionar que implementación propia se realizó con el apoyo de algunos estudiantes de másters y de grados en la universidad Granada. Con el objetivo de lograr el intercambio controlado de archivos entre la red aislada y el exterior.

Esta pasarela comienza su desarrollo con el prototipo de Pablo Rey Pedrosa. En su trabajo de fin de máster [8] se realiza una primera implementación de la funcionalidad de transmisión de archivos entre dos carpetas situadas en dos Raspberries Pi, cada una en una de las redes, realizando la conexión a través de un cable USB a USB [9]. Se basa en el uso de la librería de c++ *libusb* [10] mediante el *wrapper Pyusb* [11], para realizar el manejo de los eventos del cable para realizar las transferencias.

En su trabajo se explica como se realiza una primera implementación del protocolo de transmisión de ficheros que será usado en la pasarela. Las transferencias que se realizan aún eran unidireccionales, siendo necesario en desarrollos posteriores duplicar los dispositivos para habilitar el doble

sentido. Además, las comprobaciones de seguridad: antivirus o de cifrado y firma quedaban a cargo de los usuarios de la pasarela.

Posteriormente, *jtsec* continúa con un desarrollo propio, para añadir las funcionalidades de:

- Interfaz web para el manejo de los archivos. Los usuarios pueden hacer uso de la interfaz web para soltar y recoger los archivos, de manera que el servidor puede hacer las comprobaciones necesarias para cumplir con las políticas de manera transparente. La pasarela ofrece la misma interfaz desde ambas redes para facilitar su uso.
- Test antivirus con ClamAV de los archivos entrantes desde la red externa. Evitando así la entrada de contenido malicioso, que pudiese provocar graves daños como pudiese ser, un ransomware.
- Comprobación de cifrado y firma GPG, con la que se inhabilita a que los usuarios no autorizados, aquellos que no tienen clave GPG autorizada no puedan ser sacados fuera de la red interna. Estas claves GPG son, expedidas por el administrador de sistemas de la empresa y con plazo de validez controlado.
- Acceso a algunos sitios web. Habilitadas por configuración, se permite acceso a ciertos sitios. Entre los que están los repositorios de paquetes necesarios para poder descargar las actualizaciones de los paquetes, programas, parches de seguridad a los sistemas de la red interna.

Es importante destacar que esta segmentación de redes dificulta en gran medida el uso de los servicios y protocolos de comunicación tradicionales, y que cada nuevo servicio o funcionalidad que se pretenda habilitar requiere adecuarse al protocolo o software ya implementado. Como es el caso para las transferencias HTTP, para el implementan de proxy a través de la pasarela. Lo mismo ocurre con los protocolos de correo electrónico.

Siendo el envío de correos lo que motiva a Jose Luis López Sánchez a realizar en su TFM [12] una expansión sobre esta pasarela. En su trabajo, implementa la funcionalidad necesaria para utilizar SMTP a través de la pasarela. El dispositivo recoge las peticiones envío de correo de la red aislada para analizarlas, filtrarlas y reenviarlas a través del protocolo al extremo de la red externa. Además implementa las configuraciones y comprobaciones necesarias para asegurarse de que los mensajes salientes han sido cifrados correctamente y así evitando filtraciones.

## 2.2. Correo electrónico

Es fundamental adquirir un conocimiento profundo sobre el funcionamiento, la estructura y la infraestructura subyacente del correo electrónico,

así como comprender el flujo de información que sigue el correo electrónico y los protocolos que se emplean. En este proyecto, nos centraremos en un aspecto específico: el protocolo IMAP (*Internet Message Access Protocol*), el cual desempeña un papel crucial en la recepción de correos electrónicos. Abordaremos en detalle el funcionamiento de IMAP, explorando cómo facilita el acceso a los mensajes de correo electrónico almacenados en servidores remotos.

El servicio de correo electrónico es una herramienta de acceso a la información que posibilita el envío de mensajes a uno o varios destinatarios, independientemente de si están en línea en ese momento o no. Este servicio opera de manera similar al correo postal tradicional, donde los mensajes son depositados en una bandeja de entrada virtual o servidor de correo, desde donde los destinatarios pueden recuperarlos a su conveniencia.

El RFC5321 [13], es una revisión del RFC821 de *Simple Mail Transfer Protocol* (SMTP), ambos definen dirección de correo electrónico como una cadena de caracteres, que identifica a quien va dirigido un correo o donde este correo será almacenado. El termino "buzón" se refiere a ese almacenamiento.

El sistema funciona gracias a la actuación conjunta de dos procedimientos bien diferenciados tal y como se indica en [14]:

- El envío de correos electrónicos, implica la transferencia de mensajes hacia el buzón de correo correspondiente. Para llevar a cabo esta acción, se emplea el protocolo de aplicación SMTP, el cual será explorado con mayor detalle en las próximas secciones.

En el TFM de Jose Luis [12], se implementa SMTP para ser utilizado sobre una pasarela de datos, habilitando el envío de correos electrónicos usando Thunderbird desde la red aislada.

- La recepción de correos electrónicos se efectúa en la máquina de destino una vez que se accede al buzón de correo, utilizando dos protocolos: POP3 (Post Office Protocol) e IMAP4 (Internet Message Access Protocol). Estos dos protocolos serán analizados en detalle en secciones posteriores.

El sistema de correo electrónico se compone de una serie de componentes interconectados que para lograr esa comunicación entre usuarios. El Agente de Usuario o MUA actúa como la interfaz entre el usuario y el sistema, permitiendo la composición, lectura y gestión de mensajes. A su vez, el Agente de Entrega de Mensajes o MSA recibe los mensajes del MUA y los prepara para su envío, aplicando políticas de seguridad y autenticación. Los mensajes se almacenan temporalmente en la Cola de Salida, junto con la información del destinatario. El Agente de Transferencia de Mensajes o MTA se encarga de transmitir los mensajes utilizando el protocolo SMTP y decide si el mensaje es para un buzón local o debe enviarse a otro servidor. El Agente



de Reparto de Mensajes o MDA acepta mensajes destinados al buzón local y realiza operaciones como ordenarlos y filtrar spam. Los buzones de correo almacenan los mensajes entrantes.

Concretamente interesa centrarse en el flujo de recepción de correos electrónicos por lo que se entrará en detalle mientras comparamos POP3 e IMAP4.

Según David Wood en *Programming Internet Email* [15], la mayor diferencia entre IMAP y POP se reduce a donde queda almacenado, es decir, donde se sitúa el buzón de correos electrónicos:

Con POP, el correo se transporta desde el buzón en servidor a la máquina local, donde quedan almacenados. Esto hace POP más eficiente en términos de ancho de banda, si el usuario quiere volver a acceder al correo lo tendrá almacenado en su máquina.

Por otro lado, con IMAP, el correo queda almacenado en el buzón en el servidor. Esto permite que los usuarios puedan acceder desde diferentes máquinas, mientras se haya autenticado correctamente, pero, si se pretende visualizar el mismo mensaje en dos situaciones distintas, el mensaje tendrá que volver a ser descargado.

Acceder al mismo buzón desde diferentes máquinas puede permitir a los usuarios de la red aislada, acceder desde la red aislada mediante la pasarela al buzón del servidor la red externa, y de manera normal desde la red externa.

Ambos protocolos hacen uso de conexiones TCP y de diferentes comandos para realizar las operaciones permitidas. Como pueden ser, en caso de IMAP, la descarga y gestión de correos mediante la modificación de sus etiquetas.

En el RFC9051 [16], revisión del RFC3501 IMAP, se pueden encontrar los atributos de un mensaje de correo electrónico:

- *Unique identifier* (UID), es un identificador incremental que hace referencia de manera única a un correo en un buzón, además este no se debe de modificar entre sesiones.
- *Flags*, es una lista de cero o más etiquetas asociadas a un mensaje:
  - `\Seen`, indica si el mensaje ha sido leído,
  - `\Answered`, si el mensaje ha sido respondido,
  - `\Flagged`, el mensaje está marcado como importante/urgente,
  - `\Deleted`, el mensaje ha sido marcado como borrado, y será eliminado posteriormente,
  - `\Draft`, el mensaje no ha sido enviado, es marcado como borrador,
  - `\Recent`, si ha llegado recientemente al buzón, es decir si la sesión actual es la primera en encontrar el mensaje.
- *Size*, el tamaño del mensaje,

- *Envelope*, las cabeceras del mensaje,
- *Body*, el cuerpo del mensaje

Ese mismo RFC también define como "sesión" la secuencia de interacciones cliente/servidor en un mismo buzón.

Las sesiones de correo electrónico se realizan mediante el uso de clientes de correo electrónico, también conocidos como MUA (Mail User Agent). Estos clientes proporcionan a los usuarios o scripts una interfaz que les permite manipular diversos aspectos del correo, incluyendo la recepción de correos, el envío de mensajes y la respuesta a los mismos.

Podemos encontrar MUA de instalación local, como sería el caso de *Thunderbird*, MUA web, como serían *Gmail* o *Outlook*, o también clientes más simples, como serían las librerías de python *smtplib* y *imaplib*, para enviar y gestionar el buzón de correos, respectivamente.

## 2.3. Estado del arte

En esta sección, se presentarán las soluciones actuales para abordar el desafío de transferir datos de una red aislada a otra. Cada solución será explicada en detalle y luego se realizará una comparación en una tabla resumida.

En primer lugar, se describe el método tradicional que implica el uso de dispositivos USB para la transferencia de datos entre redes aisladas. Aunque este enfoque es simple en su implementación, también presenta una serie de desafíos significativos. Los usuarios son responsables de cargar los datos en el dispositivo USB y luego descargarlos en el dispositivo de la red interna. Además, recae sobre los usuarios la responsabilidad de verificar que el dispositivo USB no esté infectado con malware y cumplir con las políticas de seguridad establecidas para la transferencia de datos entre las redes.

En segundo lugar, se examina el uso de pasarelas de transmisión de archivos, como se menciona en el catálogo CPSTIC [17]. Una de las pasarelas de archivos identificadas en este catálogo es la pasarela de archivos PSTFile de Autek [18]. Esta pasarela facilita la transferencia de archivos mediante protocolos comunes como SFTP, FTP o SMB y lleva a cabo las verificaciones de seguridad necesarias para cumplir con los requisitos del Centro Criptológico Nacional (CCN) y así ser incluida en el catálogo mencionado.

Por otro lado, también se destaca la pasarela de archivos desarrollada por *jtsec*. Se ha obtenido el permiso necesario para ramificar o utilizar partes de su código fuente. Esto brinda la posibilidad de implementar una pasarela de correo electrónico de manera eficiente y rentable en un período de tiempo razonable, lo que contribuiría a la reducción de costos.

Otra solución de Autek que se debe considerar es PSTMail [19]. Esta pasarela está diseñada específicamente para facilitar el envío y recepción de

correos electrónicos y cumple con los rigurosos requisitos de seguridad necesarios para ser incluida en la lista de CPSTIC. PSTMail es capaz de habilitar los protocolos de correo electrónico previamente mencionados, como SMTP, IMAP y POP3, lo que la convierte en una opción viable para gestionar la transferencia de mensajes de correo entre redes aisladas de manera segura.

A continuación la tabla comparativa.

	<b>Pincho USB</b>	<b>PSTFile</b>	<b>PSTMail</b>	<b>Pasarela propia</b>
<b>Sencillo de usar</b>	Si	Si	Si	Si
<b>Correos</b>	No	No	Si	Si
<b>Seguridad</b>	No	Si	Si	Si
<b>Precio</b>	Bajo	Alto	Alto	Medio

Tabla 2.1: Comparación de métodos tradicionales de transferencia de datos a una red aislada

De esta comparación se extrae que se podría intentar desarrollar una pasarela de correos, aún manteniendo un costo menor al de una pasarela de correos real, haciéndola sencilla de utilizar y realizando las comprobaciones de seguridad pertinentes para evitar abrir una brecha en la red aislada.



## Capítulo 3

# Objetivos y planificación

En este capítulo, se indica el objetivo principal de trabajo con el fin de adquirir una visión general de su alcance. Adicionalmente, se detallará una lista de objetivos específicos, fundamentales para la formación del Producto Mínimo Viable (PMV). De la misma manera, se explora la posibilidad de incluir objetivos opcionales para tener una visión general del proyecto. De esta manera se establece una base sólida para el desarrollo y ejecución de este proyecto. Al concluir el capítulo, se incluye una planificación inicial estimada sobre los tiempos de análisis y desarrollo del proyecto, además de un diagrama de Gantt para proporcionar una perspectiva clara de la secuencia de tareas en el tiempo.

### 3.1. Objetivo general

Considerando la restricción actual que impide a los empleados de las redes internas aisladas por *airgap* acceder al correo electrónico corporativo. El principal objetivo consistirá en habilitar el acceso al correo electrónico, cuerpo, algunos metadatos y archivos adjuntos directamente desde dicha red interna. Para ello, se propone superar esta limitación construyendo una pasarela de datos que permita la recepción de correos de manera fluida y eficiente, mejorando así la conectividad y la eficacia operativa de los empleados en el entorno interno, salvaguardando la seguridad de la red aislada, utilizando:

- Configuración manual de los usuarios y el servidor al que tengan acceso.
- Análisis de los datos entrantes a la red aislada, mediante el uso de antivirus sobre los adjuntos.

### 3.2. Objetivos específicos

A partir del objetivo general se consiguen extraer los objetivos específicos que deberán de ser completados para obtener el PMV, una pasarela de

correos y un MUA web para poder consumir los correos de manera sencilla.

### 3.2.1. Estudio del EMAIL

Realizar un estudio del funcionamiento de los correos electrónicos es esencial ya que forma parte de la funcionalidad básica que se pretende ofrecer. Este análisis es esencial para comprender en profundidad las etapas que comprenden el proceso de envío y especialmente en la recepción (IMAP) de correos.

Al acabar, se conocerán los detalles sobre de qué es necesario para recibir correos y qué se pueden hacer con ellos.

### 3.2.2. Análisis de la pasarela de *jtsec*

Se tendrá que analizar exhaustivamente la pasarela actual de *jtsec*. Este análisis se erige como el paso esencial para la posterior implementación de una pasarela de datos diseñada para la recepción de correos electrónicos. Al entender en profundidad la estructura, funcionalidades y limitaciones de la pasarela existente, se podrá trazar una estrategia sólida y adaptada, garantizando así la eficaz transición hacia una solución más correcta y real.

En este punto se tendrá que descubrir los detalles de seguridad primordiales que hacen que una pasarela pueda ser instalada en un entorno real. Este proceso proporcionará las bases prácticas necesarias para aprovechar parte de el marco de trabajo ya establecido. Pudiendo así partir desde una base sólida para construir la pasarela de correos electrónicos.

### 3.2.3. Seguridad, monitorización y administración

Se deberán tomar nota de los requisitos de seguridad, monitorización y administración para que la solución propuesta pueda ser desplegada correctamente en un entorno de producción.

Al ser una pasarela con diferente funcionalidad la de *jtsec*, se tendrá que analizar qué mismas situaciones problemáticas para la seguridad del sistema de pasarela podrían llegar a aplicarse y qué otras nuevas situaciones pudiesen presentarse, al tratar con correos electrónicos en lugar de transferencias de archivos. Posteriormente se podrán tomar las medidas que puedan ser necesarias para solventarlas.

### 3.2.4. Diseño

Será necesario traducir lo aprendido hasta este momento, en requisitos funcionales y no funcionales. Se diseñará de manera teórica el sistema final, incluyendo no solo las descripciones de la funcionalidad deseada, sino también los aspectos de seguridad, configuración, registros de *logs*, para garantizar un entorno confiable y corporativo. Asimismo se delinearán los casos

de uso detallados y diagramas que representen la arquitectura propuesta y los flujos de trabajo.

Durante esta fase también se definirán los materiales que van a ser utilizados y se establece una metodología para gestionar de manera eficiente el trabajo de desarrollo.

Todo este trabajo hará más sencilla la comprensión al lector de esta memoria y permite al desarrollador entender en detalle el sistema a implementar.

### 3.2.5. Desarrollo de nuestra pasarela

En este tercer objetivo, se pondrán en práctica los conocimientos adquiridos a través de los objetivos anteriores y sintetizados durante la fase de diseño, para llevar a cabo el desarrollo de nuestra propia pasarela de correos electrónicos.

El desarrollo de podrá dividir en dos grandes etapas bien diferenciadas:

- Una *API* HTTP, con la que que pueda hacer uso de la pasarela, ya sea con clientes como CURL o WGET, o mediante el uso de alguna interfaz.
- Un *MUA* WEB, la interfaz con la que los usuarios puedan trabajar con su correo de manera interactiva y simple.

El producto mínimo viable, resultado de completar el objetivo, consistirá de una pasarela de correos que permita a los usuarios finales recibir su correspondencia desde esa red interna aislada, sin comprometer, utilizando las medidas oportunas, la seguridad de los datos de la red aislada.

## 3.3. Planificación

En esta sección se presenta la metodología usada durante el proceso y se organizan los objetivos específicos anteriores en el tiempo y. Finalmente, usando un diagrama de Gantt se ilustrará la planificación.

Los objetivos previos están planificados para ser completados siguiendo una metodología basada en *sprints*, donde cada uno de estos representa un ciclo de trabajo y revisión, focalizado en alcanzar un objetivo específico. En este contexto, un *sprint* se define como un período de tiempo limitado y bien definido durante el cual se trabaja en una serie de tareas predefinidas para lograr un objetivo determinado. Al final de cada *sprint* se realiza una revisión junto al equipo, que en este caso consiste en alumno y profesor, donde se muestra el trabajo realizado, y se recibe retroalimentación, permitiendo evaluar si se ha cumplido la meta del *sprint* o si se debe continuar este.

En el capítulo 14 de [20], habla de como cada *sprint* requiere producir *working software*, o un producto mínimo viable. Esto es, código que funcione y sea desplegable. También se comentan tres principales razones para usarlos:

- Un código que funciona favorece la retroalimentación. Al poder ver y probar el producto, los clientes/usuarios pueden describir mejor sus valoraciones en cuanto al trabajo realizado. Para este proyecto, permite ir evaluando si se van alcanzando los objetivos.
- Un código que funciona ayuda al equipo a evaluar el progreso. Las entregas frecuentes hacen más sencillo ver cuanto proyecto queda por delante.
- Un código que funciona permite desplegar frecuentemente.

Así, el trabajo quedará planificado de la siguiente manera:

- **Sprint 1.** Estudio detallado del funcionamiento de correos electrónicos, flujo, protocolo, y prueba de uso de clientes de IMAP.
- **Sprint 2.** Análisis y extracción de requisitos de la pasarela de *jtsec*. Se determinará que parte del código se puede extraer como librería y se tomará nota de qué requisitos se aplican a la pasarela de correos.
- **Sprint 3.** Diseño una pasarela de correo, usando casos de uso.

Tras finalizar todo el trabajo de análisis y estudio, es cuando se definen los siguientes, esta vez, mayoritariamente orientados al desarrollo práctico del sistema.

- **Sprint 4.** Montaje de la infraestructura básica del dispositivo.
- **Sprint 5.** Configuración por defecto, administración, primera autenticación contra la configuración. Se deberán implementar los mecanismos necesarios para el administrador pueda habilitar las cuentas de usuario de los servidores de correos externos y la filtrado de a los usuarios
- **Sprint 6.** Segunda autenticación y seleccionado de carpetas.
- **Sprint 7.** Listado y descarga de correos.
- **Sprint 8.** Borrado y marcado como no leído, para gestionar los correos electrónicos.
- **Sprint 9.** Implementación de un MUA WEB.

Cada *sprint* podrá tomar entre una y dos semanas de tiempo, aunque si uno de estos se acabase antes del tiempo estimado, simplemente se continuaría con el siguiente. En el siguiente diagrama de Gantt se aprecia la planificación del proyecto de manera más concisa.



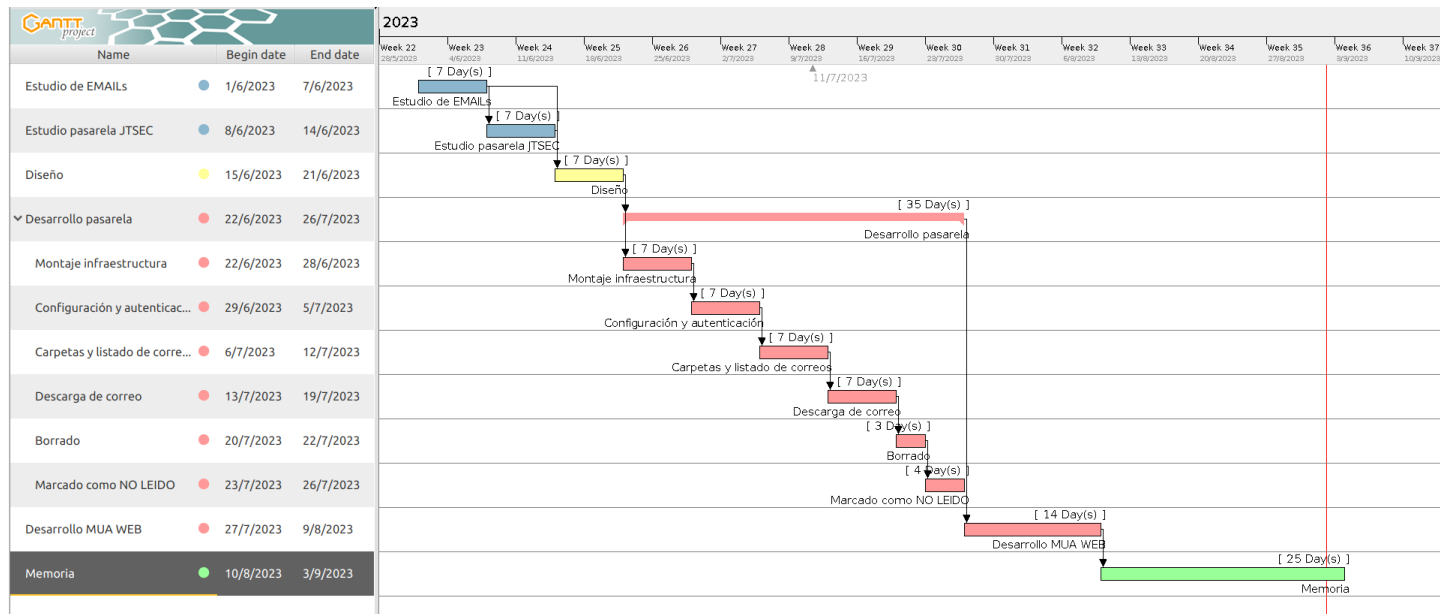


Figura 3.1: Planificación.



## Capítulo 4

# Análisis

En este cuarto capítulo, se aborda el análisis del sistema de pasarela de archivos de *jtsec*. Se explorará su funcionamiento para identificar elementos que puedan ser útiles en el desarrollo de nuestra propia pasarela de correos. Además, se examinan detenidamente los aspectos relacionados con la seguridad que deben considerarse durante la fase de implementación. Finalmente se señalan los actores y se extraen los requisitos funcionales, no funcionales y casos de uso.

### 4.1. Pasarela de *jtsec*

Inicialmente se exponen los requisitos que debe cumplir como pasarela, que en principal medida son aquellos definidos por el CCN en su guía de taxonomía de pasarelas [6].

- La configuración por defecto del flujo de datos es cerrado. Permaneciendo así hasta el administrador habilite los canales, usuarios y claves.
- En el caso de la red interna, se establece una política en la cual ningún documento podrá salir sin previo cifrado. Este cifrado puede llevarse a cabo a través de Gnu Privacy Guard (GPG), una herramienta ampliamente reconocida para el cifrado de datos. Alternativamente, se podrá recurrir al cifrado mediante el formato zip. Además, se requerirá que algunos documentos sean cifrados utilizando claves específicas obligatoriamente.
- En lo concerniente a la red interna, únicamente se permitirá la entrada de archivos que hayan superado un análisis antivirus. Para llevar a cabo este proceso, se implementa el uso de ClamAV, un software antivirus de código abierto que se encarga de detectar y prevenir posibles amenazas en los archivos.

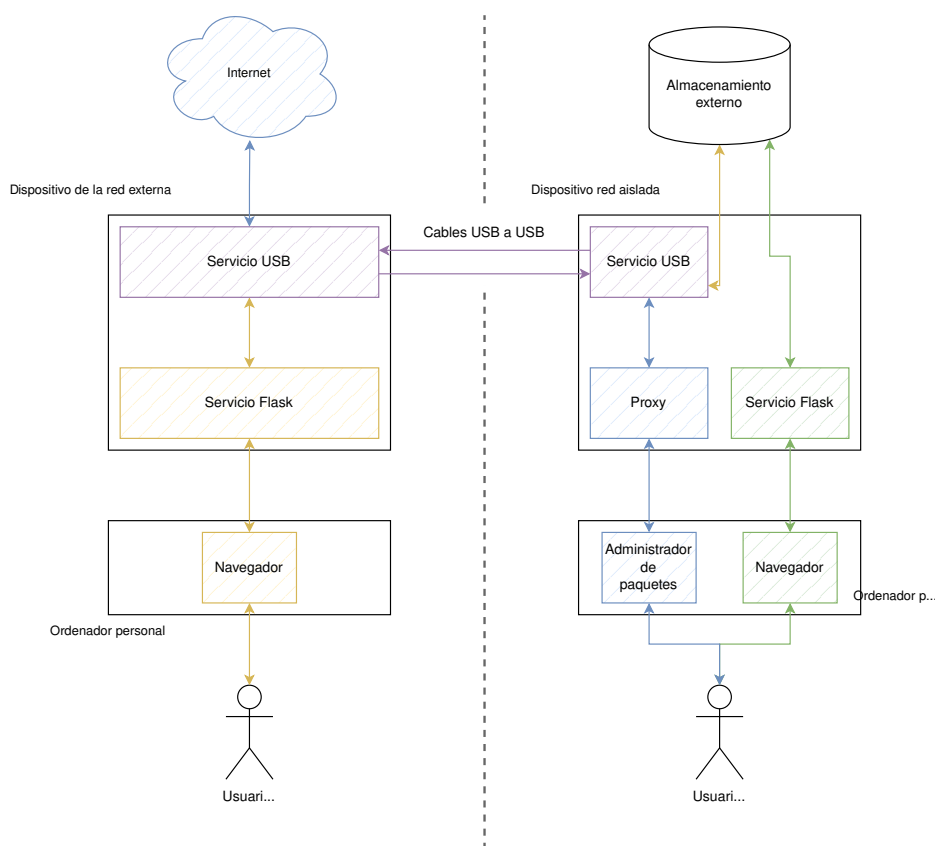


Figura 4.1: Estructura de la pasarela de *jtsec*

- Entre los dos extremos de la pasarela, se garantiza la ausencia total de conexión a través de los protocolos convencionales. Sino que se establece un protocolo ad hoc diseñado exclusivamente para las necesidades de comunicación entre los extremos de la pasarela, asegurando un entorno controlado y protegido para el intercambio de información.

En segundo lugar describiremos el hardware que la compone. Formada a partir de dos dispositivos Raspberry Pi 4 Modelo B[21]. Esos son ampliamente conocidos como ordenadores de reducido tamaño, bajo consumo y alto rendimiento (en relación a su tamaño).

Cada dispositivos se sitúan en redes diferentes, una en la red interna aislada de Internet y la otra en la red con Internet. Ambos dispositivos quedan conectados entre sí mediante un par de cables USB a USB[9]. Utilizando un total de 4 puertos USB entre las dos raspberries, puesto que se usan de manera unidireccional.

Ambos dispositivos sirven una versión diferente de la misma API HTTP, desde ambas se pueden utilizar las mismas rutas, pero dependiendo de en qué red esté situada se lanzan unas operaciones u otras.

Esta API, puede ser consumida con cualquier cliente HTTP, por tanto puede ser usada desde el navegador o desde la consola utilizando curl. Las rutas más importantes que encontramos para hacer uso de la funcionalidad de transferencia de archivos son:

- GET `/download/{path}`, se usa para descargar el archivo indicado en la ruta.
- POST `/upload`, para subir archivos a una ruta dada, realizando las comprobaciones exigidas por las políticas de seguridad establecidas (GPG).

```
~/tmp
> curl -ks -F file=@upload.txt -F path='' -X POST http://localhost:8000/api/upload | jq
{
  "msg": "Not all files were uploaded",
  "report": [
    {
      "filename": "upload.txt",
      "msg": "bad filename or not allowed extension (zip, gpg, asc)",
      "uploaded": false
    }
  ]
}

~/tmp
> curl -ks -F file=@not_encrypted.gpg -F path='' -X POST http://localhost:8000/api/upload | jq
{
  "msg": "Not all files were uploaded",
  "report": [
    {
      "filename": "not_encrypted.gpg",
      "msg": "not gpg encrypted",
      "uploaded": false
    }
  ]
}

~/tmp
> curl -ks -F file=@upload.txt.gpg -F path='' -X POST http://localhost:8000/api/upload | jq
{
  "msg": "All files were uploaded",
  "report": [
    {
      "filename": "upload.txt.gpg",
      "msg": "encrypted correctly",
      "uploaded": true
    }
  ]
}
```

Figura 4.2: Uso de la API para subir y descargar un fichero en la pasarela de *jtsec*

La API con la que se interactúa desde ambos lados es mayoritariamente la mismos, pero internamente cada dispositivo trabajará de manera diferente. El software diferenciar en que red se sitúa para realizar comprobaciones antivirus o GPG. Además, la configuración ha de estar exclusivamente en la red interna, para reducir la superficie de ataque y evitar que se puedan habilitar maliciosamente canales o claves desde el exterior.

Además, con el fin de evitar redundancias, implementa un enfoque donde los archivos son almacenados exclusivamente en la red interna. Esto conlleva

a que el dispositivo ubicado en la red interna cumpla el rol de servidor desde la perspectiva del dispositivo en la red externa. Entre los dos dispositivos, para la comunicación que se realiza a través de los cables USB a USB, funciones, se emplea un protocolo privado, diseñado específicamente para esta pasarela.

Desde la red externa, se accede a los servicios del dispositivo de la red interna a través de dicho protocolo. En esta configuración, el uso del cable se convierte en imprescindible. Por otro lado, desde la red interna, no es necesario depender del cable ya que el servidor responde directamente a las solicitudes realizadas.

Al profundizar en el código, podemos observar que para establecer la interfaz de programación de aplicaciones (API) HTTP se hace uso de Flask [22], un framework web minimalista y flexible para Python que facilita la creación de aplicaciones web.

En ambos extremos, tanto en el dispositivo de la red interna como en el de la red externa, se ejecuta una hebra dedicada exclusivamente a escuchar las comunicaciones a través de un cable USB específico, cuyo puerto está definido en el archivo de configuración en formato JSON como *USB\_READ\_PORT*.

El código necesario para realizar las comunicaciones a través del cable, se encuentra contenido dentro de una carpeta fácilmente identificable llamada “cable”. En esta encontramos los siguientes archivos:

- En **USBWriter.py** se encuentra la definición de los paquetes utilizados en el protocolo. Y de funciones necesarias para enviar ese paquete.
- En **USBReader.py** se lanza la hebra que controla el servidor que tratará de responder a las peticiones enviadas desde el otro extremo de la pasarela.
- **RequestController.py** define las operaciones disponibles y que se deben de ejecutar tanto en un lado como en otro del dispositivo, por ejemplo, un lado deberá esperar a recibir información mientras el otro le envía el listado de archivos. También contiene un *Actions*, un enumerador que mapea el nombre de una acción con un entero, que es el que se utiliza en los paquetes que constituyan una nueva petición.

Para explicar como funciona de manera práctica el proceso de hacer peticiones al otro extremo de la pasarela, llamaremos cliente y servidor a las *Raspberries* externa e interna aislada, respectivamente:

1. El cliente lanza un *RequestController* con el *Action* de debe ejecutarse localmente, para recibir los datos.
2. A continuación el cliente usa *USBWriter* para crear un paquete que contiene:

- el identificador de *request\_controller* que acaba de ser creado,
  - el valor de un *Action* que se necesite ser ejecutado en la otra máquina,
  - un entero para el tipo de paquete,
  - los argumentos codificados en binario.
3. La instancia de *USBReader* del servidor recibirá un paquete, leerá el identificador y lo buscará entre las hebras que tiene ya lanzadas para decidir si lanzar una nueva hebra o entregar el paquete a la identificada.
  4. La hebra lanzada en el servidor realizará las operaciones necesarias y pasará la respuesta al cliente usando *USBWriter*. El paquete estará formado de la misma manera.
  5. En el cliente, otra instancia de *USBReader*, recogerá el paquete y lo entregará al *request\_controller*.

Inicialmente, los paquetes constan de 2MB de espacio total, incluyendo en ese tamaño el espacio reservado para los dos identificadores (hebras de origen y destino), el numero entero del *Action*, el numero entero que representa el estado (0, 1, 2 para continuar, final y error, respectivamente).

En situaciones en las cuales el cliente necesite transmitir una cantidad de datos que excede la capacidad de un solo paquete, quedará en espera hasta que la hebra generada por el servidor reciba otro paquete con su propio identificador. Permitiendo de esta manera que el servidor pueda direccionar todos los paquetes con destino a un mismo identificador a la hebra correspondientes.

Es resto de espacio disponible en el paquete está a disposición del programador, pudiendo mandarse cualquier tipo o estructura de datos siempre que se codifique y decodifique al entrar y salir del cable.

El uso de un *Action* sirve para decirle al servidor, qué método debe utilizar, que comprenda, que comprende cómo recibir los paquetes y el formato que su contenido posee.

En el siguiente fragmento de código se puede ver el código del cliente, del servidor y del *RequestController* usados para obtener el SHA256 de un archivo.

Código 4.1: Código de jtsec para realizar una petición de sha256 a un fichero en la red aislada

```
1 controller = RequestController(thread_controller)
2 controller.action = Action.HASH
3 controller.start()
4 thread_controller.add_thread(controller)
5 packet = writer.build_packet(controller.ident, 0, 0, self.HASH_ANSWER.
6     value, bytes(path))
```

```
7 writer.usb_write_package(packet)
8 status, bytemessage = controller.join()
```

Dentro de las directrices o políticas definidas en esta pasarela, se destaca la necesidad de registrar las acciones realizadas por los usuarios, en concreto aquellas que potencialmente puedan generar inconvenientes. Estas acciones incluyen tanto la carga como la descarga de archivos, incluso aquellos que no hayan superado los controles previos a su carga. Esto es además de las comprobaciones de GPG y antivirus previamente mencionadas.

Para los datos que salen de la red interna y que están sujetos a las verificaciones de GPG, es esencial habilitar las claves correspondientes en el archivo de configuración JSON. Y la configuración por defecto no debe dejar salir ningún archivo.

Con todo esto, se procede a definir los requisitos de la nueva pasarela de correos.

## 4.2. Actores

- **Usuario.** El trabajador que tratará de leer su correo electrónico desde su puesto de trabajo en la red interna.
- **Administrador.** El empleado que se encarga de mantener los sistemas a punto para su uso por los usuarios. También es costumbre que se encargue de dar de alta o baja a los usuarios en los diferentes sistemas.

## 4.3. Requisitos funcionales

- **RF1. Inicio de sesión.** Los usuarios utilizarán sus contraseñas habituales de sus cuentas de correo del servidor en internet oportuno para acceder. El administrador no debe tener acceso a la contraseña del usuario, por lo que la aplicación deberá hacer el puente e iniciar sesión directamente al servidor de correo.
- **RF2. Visualización de correos.** Los usuarios tendrán que poder, de alguna manera, navegar sus carpetas en el servidor para llegar hasta el correo deseado. Posteriormente el correo podrá ser seleccionado para leerlo con mayor detalle.
- **RF3. Descarga de adjuntos.** Dado a que los correos electrónicos pueden tener adjuntos. El usuario podrá descargar los adjuntos de un correo, permitiéndole trabajar con los nuevos archivos en la red aislada. Para mantener la seguridad, estos adjuntos deben pasar un análisis antivirus, aunque esto último se considera un requisito no funcional.



- **RF4. Borrado de correos.** El usuario podrá realizar acciones básicas de gestión de correo. Los correos deberán poder ser borrados, favoreciendo al usuario cuando pretenda la limpieza de su bandeja.
- **RF5. Marcado como no leído.** El usuario debe poder realizar esta operación si quiere que la aplicación vuelva a señalar el correo la próxima vez que la utilice. Permite gestionar la importancia de un correo.
- **RF6. Correos leídos.** Automáticamente, cuando el usuario realice la visualización de uno de sus correos. El correo deberá de marcarse como leído en el servidor.
- **RF7. Configuración.** Deberá consistir en una lista blanca de usuarios. Inicialmente, y por defecto el canal estará bloqueado, la pasarela debe evitar transmitir los datos. Cuando el administrador añada las cuentas de correo, servidor y puerto, para poder hacer la conexión con el servidor de correo. Estas cuentas de correos serán en mayor frecuencia corporativa, de los usuarios que tengan permitido utilizar la pasarela desde la red interna.

#### 4.4. Requisitos no funcionales

- **RNF1. Antivirus.** Los adjuntos deberán de pasar un antivirus antes de entrar a la red interna. El cuerpo de los correos no necesitará de pasar este, ya que no podrán ser ejecutados, y si este tuviese enlaces a sitios externos podrán ser visitados desde la red aislada.
- **RNF2. Registro de acciones.** El administrador tendrá que poder saber si se han intentado accesos a cuentas de correos no habilitadas. Esto se puede lograr registrando en un archivo cada intento de comprobación fallido. También
- **RNF3. HTTPS.** Los métodos habituales de autenticación usan las cabeceras de paquetes HTTP para enviar algún tipo de token. Dado que los usuarios tendrán que iniciar sesión en su servicio de correo para acceder a estos, usar HTTPS nos permitirá dificultar la lectura maliciosa de este, ya que mantiene las cabeceras cifradas hasta llegar al servidor de nuestra pasarela (aunque la autenticación se haga desde el lado interno aislado de la pasarela).

Con esta definición de requisitos habremos delineado qué va a hacer el sistema de pasarela de correos y qué se tendrá que tener en cuenta durante el diseño y la implementación.



## Capítulo 5

# Diseño

En este quinto capítulo, se tratará de convertir esos requisitos delineados en el capítulo anterior a casos de uso. Finalmente se describen los materiales que va a ser utilizar durante la implementación.

### 5.1. Casos de uso

#### 5.1.1. Actores

Antes de comenzar a ver los casos de uso, se explicará de manera un poco mas detallada los dos actores principales que actuarán sobre la pasarela.

- **Administrador del sistema.** El Administrador del Sistema desempeña un papel central en la gestión y el control de la pasarela.
  - es el único con acceso a los servidores de la pasarela,
  - se encarga de añadir o quitar usuarios de la pasarela,
  - puede revisar el fichero de *logs*
- **Usuario.** Pretenden acceder a su correo electrónico a través de la pasarela desde la red aislada. Solo pueden acceder con usando las cuentas habilitadas. Realizarán las tareas básicas de gestión de correos, lectura, borrado y edición (marcado como no leído).

#### 5.1.2. Casos de uso

Cada caso de uso constará de varias partes: nombre, descripción, actores, precondiciones, postcondiciones y el flujo que debe llevar la operación.

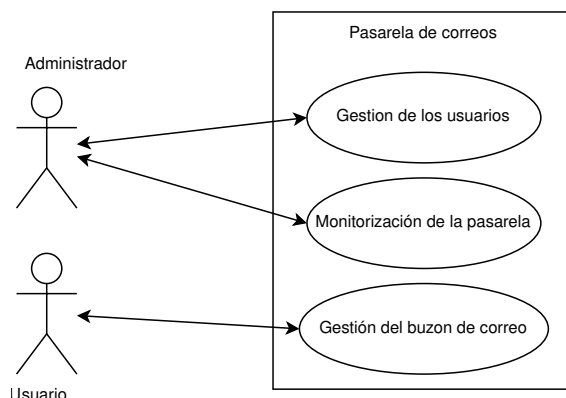


Figura 5.1: Esquema de casos de uso

### Inicio de sesión 5.1

Un usuario, tiene una cuenta de correo electrónico en un servidor con acceso desde Internet. Esta cuenta tendrá que haber sido habilitada por el administrador en la pasarela antes de ser utilizada. En caso de utilizar una cuenta no autorizada el sistema dejará registro y se retornará el error al usuario. En caso correcto se pasará a contrastar sus credenciales contra el servidor real. En caso correcto se piden las carpetas y se le devuelven al cliente.

### Selección de carpeta 5.2

Este caso de uso se lleva a cabo cuando el usuario desea acceder a una carpeta específica en su buzón de correo electrónico antes de seleccionar un correo para su visualización o descarga.

Como resultado de esta acción, el usuario obtiene acceso a un índice de correos electrónicos en la carpeta seleccionada, que incluye información básica como el asunto, la fecha y el remitente de los correos.

El flujo de este caso de uso consta de dos partes: primero, el usuario solicita una carpeta; luego, el sistema verifica las credenciales del usuario. En caso de credenciales válidas, el sistema comprueba la existencia de la carpeta deseada y proporciona al usuario el listado de correos en esa carpeta. Si las credenciales no son válidas, el sistema devuelve el error al usuario impidiendo el acceso a la carpeta.

### Selección de correo 5.3

El actor principal es el usuario, quien tiene la capacidad de seleccionar un correo de la lista, lo que le permite visualizar el cuerpo del mensaje y acceder a los enlaces a los archivos adjuntos. Para que este caso de uso se lleve a cabo,

Nombre	Inicio de sesión
<b>Descripción</b>	El usuario envía sus credenciales, provocando fallo de autenticación si se ha intentado con una cuenta no habilitada o realmente fallaron las credenciales. Termina retornando las carpetas disponibles.
<b>Actores</b>	Usuario
<b>Precondiciones</b>	El usuario está logueado, o no
<b>Postcondiciones</b>	
<b>Flujo</b>	<ol style="list-style-type: none"><li>1. El usuario usa sus credenciales para iniciar sesión</li><li>2. El sistema comprueba si el correo está aceptado en la configuración del administrador<ol style="list-style-type: none"><li>a) Cuenta de correo válido, los datos se mueven a la otra raspberry para comprobación contra el servidor real usando la contraseña. Devuelve las carpetas, o error.</li><li>b) Cuenta de correo no habilitada, devuelve error</li></ol></li></ol>

Tabla 5.1: Caso de uso: inicio de sesión

Nombre	Selección de carpeta
<b>Descripción</b>	El usuario, previamente a seleccionar un correo para leer o descargar, seleccionará la carpeta de su buzón que desea inspeccionar.
<b>Actores</b>	Usuario
<b>Precondiciones</b>	Puede acceder a un índice de emails con los datos básicos de un correo: asunto, fecha, emisor
<b>Postcondiciones</b>	
<b>Flujo</b>	<ol style="list-style-type: none"><li>1. El usuario pide una carpeta</li><li>2. El sistema comprueba las credenciales del usuario<ol style="list-style-type: none"><li>a) Credenciales correctas. Se comprueba que la carpeta existe, devuelve error en caso contrario. Se devuelve el listado de correos.</li><li>b) Devuelve no autorizado.</li></ol></li></ol>

Tabla 5.2: Caso de uso: selección de carpeta

Nombre	Selección de correo
<b>Descripción</b>	El usuario, podrá seleccionar un correo de la lista, permitiendo ver el cuerpo, y los enlaces a los adjuntos.
<b>Actores</b>	Usuario
<b>Precondiciones</b>	Ha elegido un correo de una carpeta
<b>Postcondiciones</b>	Recibe el cuerpo, y los enlaces a los adjuntos, el adjunto queda en el servidor interno esperando ser descargado.
<b>Flujo</b>	<ol style="list-style-type: none"> <li>1. El usuario pide una correo</li> <li>2. El sistema comprueba las credenciales del usuario, en caso negativo devuelve no autorizado.</li> <li>3. Se pide el correo, en caso de no existir se devuelve, error en la petición.</li> <li>4. El usuario recibe el cuerpo del correo y los adjuntos se guardan en la red interna. En caso de dar fallo de antivirus el archivo no se copiará pero se notificará al usuario.</li> </ol>

Tabla 5.3: Caso de uso: selección de correo

es necesario que el usuario haya previamente elegido un correo de una carpeta existente en su buzón. Después de seleccionar el correo, el sistema verifica las credenciales del usuario, y en caso de autenticación negativa, devuelve el error, impidiendo el acceso al contenido del correo. Si las credenciales son válidas, el sistema procede a recuperar el correo deseado; en caso de que no exista, se devuelve un error en la petición. Una vez que el sistema ha obtenido el correo, el sistema realiza un análisis antivirus de los adjuntos si existiesen. El usuario recibe los datos del mensaje y los archivos adjuntos se almacenan en la red interna del sistema. Si se detecta un problema con un archivo adjunto, como un fallo en el antivirus, el archivo no se copia, pero se notifica al usuario.

#### Descarga de adjunto 5.4

El usuario tiene la capacidad de descargar archivos adjuntos de correos electrónicos. Para realizar esta acción, el usuario debe haber seleccionado previamente un correo y utilizado el enlace al adjunto deseado. El sistema

Nombre	Descarga de adjunto
<b>Descripción</b>	Los correos pueden tener adjuntos y el usuario debe poder descargarlos
<b>Actores</b>	Usuario
<b>Precondiciones</b>	Ha pedido un correo, y usado el enlace al adjunto
<b>Postcondiciones</b>	Adjunto descargado
<b>Flujo</b>	<ol style="list-style-type: none"> <li>1. El usuario pide adjunto</li> <li>2. El sistema comprueba las credenciales del usuario, en caso negativo devuelve no autorizado.</li> <li>3. Se comprueba que el adjunto exista, en caso contrario devuelve error.</li> <li>4. El adjunto es descargado.</li> </ol>

Tabla 5.4: Caso de uso: descarga de adjunto

verifica las credenciales del usuario y, en caso de autenticación negativa, devuelve un mensaje de error. Si las credenciales son válidas, el sistema verifica la existencia del adjunto y, si este existe, procede a descargarlo.

### Borrado de correos 5.5

El usuario tiene la capacidad de eliminar correos electrónicos de su buzón para gestionar su contenido de manera más efectiva. Para llevar a cabo esta acción, el usuario debe seleccionar previamente un correo existente en su buzón para su borrado. El sistema, después de verificar las credenciales del usuario, comprueba la existencia del correo. En caso de credenciales no válidas, se devuelve un mensaje de error y si el correo no existe, se retorna un mensaje de error. Si todas las verificaciones son exitosas, el sistema procede a eliminar el correo seleccionado en el servidor. Este caso de uso permite al usuario administrar su buzón de correo electrónico eliminando mensajes no deseados o innecesarios, contribuyendo así a una mejor gestión de su bandeja de entrada.

### Marcado como no leído 5.6

El usuario tiene la capacidad de cambiar el estado de un correo electrónico a “no leído”, permitiendo al usuario gestionar su bandeja de entrada y mantener un control preciso sobre los correos que ha leído o no, contribu-



Nombre	Borrado de correo
<b>Descripción</b>	El usuario debe poder borrar correos para poder gestionar su buzón
<b>Actores</b>	Usuario
<b>Precondiciones</b>	El correo existe y ha sido seleccionado
<b>Postcondiciones</b>	Correo eliminado en el servidor
<b>Flujo</b>	<ol style="list-style-type: none"><li>1. El usuario pide borrar un correo</li><li>2. El sistema comprueba las credenciales del usuario, en caso negativo devuelve no autorizado.</li><li>3. Se borra el correo, en caso de que no exista se retorna error.</li></ol>

Tabla 5.5: Caso de uso: borrado de correo

yendo a una mejor organización de su correo electrónico. Para realizar esta acción, el usuario debe haber seleccionado previamente un correo existente en su bandeja de entrada. El sistema, tras verificar las credenciales del usuario, comprueba la existencia del correo. En caso de autenticación negativa, o de que el correo no exista se devuelve un mensaje de error indicándolo. Si todas las comprobaciones son exitosas, el sistema procede a eliminar la marca de “leído” en el servidor, marcando así el correo como “no leído”.

### Configuración de usuarios 5.7

En este caso de uso el actor pasa a ser el administrador. Este caso de uso se realiza cuando el sistema, el programa de la pasarela, se encuentra apagado y es necesario habilitar o configurar a los usuarios. El administrador lleva a cabo los siguientes pasos:

- El administrador edita un archivo JSON en la red aislada, con la configuración deseada, que incluye la habilitación de usuarios.
- El administrador coloca el archivo de configuración JSON en una ubicación accesible para la aplicación, donde pueda ser leído.
- El administrador reinicia el sistema con el objetivo de cargar la nueva configuración.
- Una vez reiniciado, el sistema queda en funcionamiento con la nueva configuración, lo que implica que los usuarios han sido habilitados de acuerdo con la información proporcionada en el archivo JSON.

Nombre	Marcado como no leído
<b>Descripción</b>	El usuario marca como “no leído” un correo
<b>Actores</b>	Usuario
<b>Precondiciones</b>	El correo existe y ha sido seleccionado para esta operación
<b>Postcondiciones</b>	El <i>flag</i> de leído ha sido eliminado en el servidor
<b>Flujo</b>	<ol style="list-style-type: none"> <li>1. El usuario pide marcar un correo</li> <li>2. El sistema comprueba las credenciales y existencia del correo, devuelve error si procede.</li> <li>3. Se marca el correo.</li> </ol>

Tabla 5.6: Caso de uso: marcado como “no leído”

Nombre	Configuración de usuarios
<b>Descripción</b>	El administrador confecciona un archivo JSON con la configuración
<b>Actores</b>	Administrador
<b>Precondiciones</b>	El sistema está apagado
<b>Postcondiciones</b>	El sistema está encendido, la configuración leída es la del JSON
<b>Flujo</b>	<ol style="list-style-type: none"> <li>1. El administrador habilita a los usuarios en el archivo de configuración JSON.</li> <li>2. El administrador deposita el fichero bajo el alcance de la aplicación, donde sepa leerlo.</li> <li>3. El administrador reinicia el sistema para cargar la nueva configuración.</li> <li>4. El sistema queda lanzado con la nueva configuración.</li> </ol>

Tabla 5.7: Caso de uso: configuración de usuarios

## 5.2. Materiales

Esta sección describirá los materiales y metodologías con las que se contarán para realizar el desarrollo.

Se ha decidido emplear como lenguaje de programación en nuestro proyecto, *Python*, con el objetivo de garantizar la compatibilidad con el código utilizado para las transferencias USB en la pasarela de ficheros analizada. Esta elección se debe a que se ha recibido la autorización de *jtsec* para reutilizar una parte esencial del núcleo de la pasarela de archivos existente, *USBReader.py*, *RequestController.py* y *USBWriter.py*, que será aprovechado como si de un *framework* se tratase, para poder realizar las comunicaciones.

Existen varias razones que respaldan esta elección y por las que probablemente se eligiesen por *jtsec* en primer lugar:

- Simplicidad y legibilidad. La sintaxis es clara, lo que facilita la comprensión para el desarrollador.
- Amplia biblioteca. Tener una gran cantidad de módulos y paquetes, debido a la amplia comunidad, permite aprovechar funcionalidades pre-existentes y acelerar el desarrollo al evitar tener que escribir código desde cero para tareas comunes.
- Versatilidad. Aunque sea un lenguaje que suele ser más usado en materia de *scripting*, análisis de datos o incluso IA, también puede ser usado para conseguir desarrollar un prototipo de una idea rápidamente.

Se empleará *Python* en su versión 3.10, la más reciente en ser oficialmente lanzada, y con la ventaja de contar con un período de tres años antes de alcanzar su final de vida (end-of-life)[23]. Esta elección estratégica asegura que nuestro proyecto se beneficie de las últimas características, mejoras y correcciones de errores proporcionadas por *Python* 3.10.

Aunque la versión utilizada para programar en la pasarela de *jtsec* aún sea la 3.8, la versión 3.10 es compatible con versiones anteriores.

Entre las librerías de Python que se incluirán se encuentran:

- **libusb1** Es un *wrapper* de la librería de C++ *libusb* que permite utilizar sus funcionalidades directamente desde *Python*. Esta biblioteca que proporciona una interfaz para comunicarse con dispositivos USB desde programas en espacio de usuario. Permite a los desarrolladores interactuar con dispositivos USB de manera más directa y flexible. Permite enumerar los dispositivos conectados, enviar y recibir datos a través de ellos, y gestionar eventos relacionados con USB. Esta biblioteca será la pieza central de las comunicaciones de la pasarela de correo, y, aunque no se tratará directamente con ella, debe ser incluida como dependencia.

- **flask** *Flask* es un *framework* web ligero y flexible para construir aplicaciones web en *Python*. Está diseñado para ser simple y modular. Proporciona las herramientas necesarias para crear rutas, gestionar solicitudes HTTP, manejar formularios, implementar autenticación y sesiones, y mucho más. Permitirá levantar un servidor HTTP para gestionar las peticiones de los usuarios.
- **flask-httpauth** [24]. Esta librería facilitará el proceso de autenticación. Permite varios tipos de autenticación: Basic, Bearer, etc. Esta librería ha sido escogida por la simplicidad de establecer una autenticación. Se usará para comprobar que las cuentas utilizadas estén en el fichero de configuración.
- **imap-tools** [25] Esta librería desempeñará un papel crucial en la gestión de la comunicación mediante el protocolo IMAP. Aunque se podría utilizar el modulo *imaplib* de *Python* directamente para realizar las peticiones IMAP al servidor de correos, esto causaría que también se tuviese que analizar el correo directamente desde su formato básico RFC822. *imap-tools* se encarga de hacer las peticiones y obtener una las partes del correo en un objeto desde el que es sencillo acceder a sus atributos.

Para desarrollar la interfaz web que consuma la API al servicio de pasarela, se utilizará *React*, debido a la facilidad de construir paginas web separando cada sección en componentes. Además, mediante el uso de estados se podrán cargar las diferentes partes del sitio de manera independiente reaccionando a los cambios.

En cuanto al estilo, *Bootstrap* se encargará de eliminar el peso de crear un CSS personalizado, consiguiendo acelerar el desarrollo una página con un diseño profesional.

*React* y *Bootstrap* también han sido elegidos por la experiencia del desarrollados en su uso, frente a otras como serían: *Vue*, *Angular*, HTML y *Javascript* directamente, en lugar de *React*, y otras como *Tailwind* o directamente CSS.

Para compilar la aplicación en HTML y *Javascript* para poder ser desplegada, en la red aislada, usaremos *Vite*. *Vite* también da apoyo durante el desarrollo permitiéndo recargar la página con cada cambio en el código.

En cuanto a la infraestructura del sistema de pasarela se utilizarán dos sistemas *Raspberry Pi 4 Model B*. Estos dispositivos destacan por su rendimiento y versatilidad. La primera simulará estar en la red interna y la segunda en la red externa. Ambas han sido prestadas para este proyecto por *jtsec*, igual que dos cables USB a USB de startech.

En ambos computadores se instalará, usando el software *Pi Imager*, una imagen de *Raspberry Pi OS*. Este sistema operativo, basado en *Debian*, está optimizado para ser utilizado sobre estos dispositivos. Se usará una versión

Lite, para que solo se incluyan el mínimo número de paquetes necesario para funcionar como servidores.

Finalmente, el entorno de desarrollo. Se utilizará GIT como software de control de versiones. Usarlo permitirá experimentar mediante el uso de ramas, también simplificará la subida a un repositorio en Github, donde será alojado, para posteriormente clonarlo en las *Raspberries* y así probar nuestro código.

Además se utilizará *autopep8*, siendo esta una herramienta que automatiza la corrección de estilo y formato en código *Python*. Su principal beneficio es que ayuda a mantener el código limpio y consistente, lo que facilita la lectura, colaboración y mantenimiento del software, mejorando así la calidad del código y la productividad del equipo de desarrollo.



## Capítulo 6

# Implementación

Este capítulo se ponen en práctica los conocimientos adquiridos hasta ahora para desarrollar la pasarela de correos. Se avanzará sobre cada uno de los *sprints* mencionados en la planificación del capítulo 3. Pasando por el montaje de la infraestructura, el sistema de respuestas de IMAP mediante el uso del servidor de USB, se implementará una API HTTP para permitir el acceso a la pasarela y se creará finalmente una interfaz web para facilitar al usuario su uso.

Al final de cada *sprint* se observará con una o más capturas el resultado del trabajo realizado durante ese tiempo.

### 6.1. Sprint 4. Montaje de la infraestructura

Se comienza realizando el montaje, instalando en las tarjetas SD de las *Raspberries* el sistema operativo *Raspberry PI OS Lite*. Para ellos se utiliza el programa *Raspberry PI Imager*. Este programa brinda una interfaz sencilla e intuitiva desde la que seleccionar el sistema operativo, además de crear usuarios, *hostnames*, configurar a la red, y habilitar SSH, lo que nos permitira utilizar el dispositivo directamente en cuanto conectemos las SD y encendamos los dispositivos.

Usaremos en ambas un usuario llamado “pasarela”, al que tendremos que dar permisos de uso del cable añadiéndolo al grupo *plugdev*.

A continuación pasamos a montar los dispositivos e interconectarlos usando los cables USB a USB. En la pasarela de datos antigua, su configuración se realizaba manualmente por del administrador, quien debe especificar en el archivo de configuración los puertos que se utilizarán tanto para el envío como para la recepción. El dispositivo en el extremo opuesto deberá emplear puertos correspondientes y opuestos a los seleccionados en la primera. Todo esto con el fin de lograr que cada cable funcione de manera unidireccional y ambos programas sepan que puerto deben usar para el envío y para la recepción.



Figura 6.1: Raspberry PI Imager

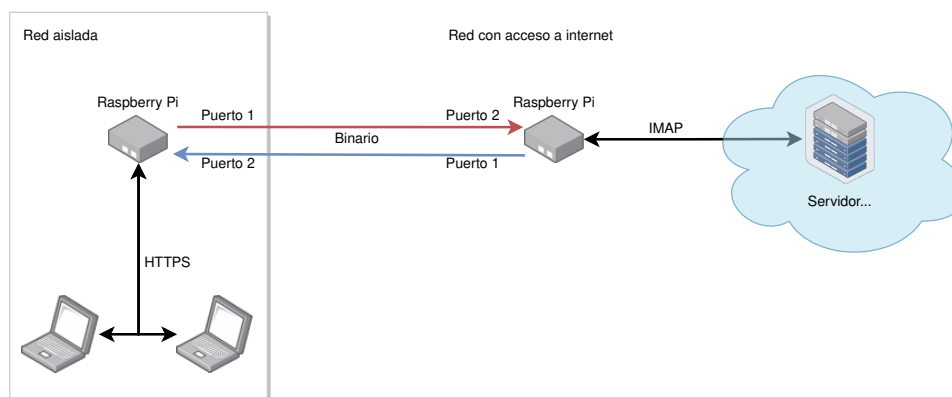


Figura 6.2: Diagrama de los dispositivos y protocolos usados en la pasarela de correos

Para simplificar este proceso, se ha decidido eliminar esta responsabilidad del administrador, esta añade complejidad sin aportar beneficios significativos. En lugar de eso, ambos cables se conectarán siguiendo una estructura predeterminada y documentada. Tomando en cuenta que una RPi 4B tiene dos puertos USB 2.1 y otros dos puertos USB 3.0, los extremos de los cables se conectarán a los puertos USB 3.0 en ambos lados. Además, los cables se cruzarán entre sí, no usarán el mismo puerto de ambos dispositivos.

De esta manera, ambas *Raspberry Pi* siempre utilizarán el primer puerto USB 3.0 para enviar y el segundo puerto USB 3.0 para recibir. Esta configuración eliminará la necesidad de que el administrador realice esta tarea, que solo deberá seguir las instrucciones indicadas durante la fase de instalación.

Posteriormente al montaje físico, tras encender ambos dispositivos y pa-



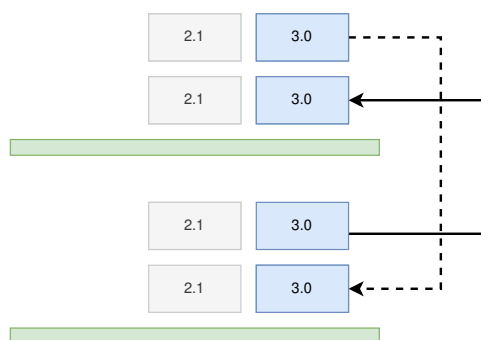


Figura 6.3: Disposición cruzada de los cables para el montaje de la pasarela de correos

saremos a instalar los requerimientos básicos del sistema. También vamos a habilitar al usuario “pasarela” utilizar los puertos USB para el dispositivo conectado (el cable) con identificador de producto “27a1” y con identificador de proveedor “067b”, siendo estos los identificadores del cable USB a USB. Esa información es normalmente visible con un dispositivo conectado y el comando *lsusb*. Los siguientes comandos serán ejecutados como administrador del sistema o root.

Código 6.1: Configuración de usuario para permitir uso del cable

```
1 usermod -a -G plugdev pasarela;
2 echo 'SUBSYSTEM=="usb", ATTRS{idVendor}=="067b", ATTRS{idProduct}=="27
3   a1" MODE="0666"' >> /etc/udev/rules.d/99-pasarela.rules;
4 udevadm control --reload-rules;
```

Posteriormente se instalan los requisitos de sistema necesarios para ejecutar crear la aplicación web, para que esta pueda utilizar el cable y el antivirus, ClamAV, aunque este último solo es en el dispositivo de la red externa.

Código 6.2: Instalación de requisitos del sistema

```
1 apt-get install
2   libusb-1.0-0-dev
3   python3.8
4   python3.8-venv
5   python3.8-pip
6   clamav
7   clamav-daemon;
```

Al mismo tiempo, en local, se lleva a cabo la inicialización del repositorio del proyecto y configurar sus dependencias. Para establecer el proyecto en el entorno de desarrollo local, procederemos a crear un repositorio utilizando Git, al que añadiremos un archivo *.gitignore*. En este archivo incluiremos los patrones típicos que deben ser excluidos cuando se trabaja con Python.

Tras crear el repositorio y clonarlo, se utiliza *python-venv* para crear un entorno de desarrollo específico para el proyecto en *Python*. Esta medida garantiza que todas las dependencias están encapsuladas adecuadamente y que las instalaciones se pueden reproducir de manera consistente.

Utilizando *pip*, se instalarán los paquetes de *Python* previamente mencionados. Finalmente, se vuelcan todos los requisitos de la aplicación en un fichero de dependencias, para tener un registro claro de las dependencias del proyecto.

Código 6.3: Inicialización del proyecto y requisitos de la pasarela

```
1 mkdir pasarela-imap; cd pasarela-imap
2
3 git init
4 curl https://raw.githubusercontent.com/gitignore/main/Python.gitignore > .gitignore
5
6 python3.8 -m venv env
7 source env/bin/activate
8 pip install imap-tools flask flask-httpauth gunicorn libusb1
9 pip freeze > requirements.txt
```

## 6.2. Sprint 5. Configuración por defecto y administración

Durante este sprint, se desarrollará la estructura fundamental del proyecto, que consiste en un servicio web con funciones de autenticación. Este servicio estará diseñado para leer información desde un archivo JSON, almacenado en el mismo dispositivo, que contendrá una lista de usuarios autorizados, cada uno con su correspondiente dirección de servidor y número de puerto. Esta configuración permitirá que el sistema identifique con precisión las conexiones que debe establecer.

Con el propósito de llevar a cabo esta tarea, se propone implementar el código necesario para la administración de conexiones, además de crear la pasarela con la funcionalidad básica requerida para establecer conexión con el servidor IMAP. Será esencial considerar la configuración predeterminada y también asegurarnos de que los cambios realizados por los administradores sean tomados en cuenta antes de verificar las contraseñas de los usuarios en el servidor externo. Cabe destacar que los datos contenidos en el archivo JSON se leerán cada vez que la aplicación se inicie.

La estrategia consistirá en lanzar servidores de cable en cada red, utilizando los archivos de manejo de cable. Junto a esto, se establece una API HTTP que permitirá introducir gradualmente las distintas funcionalidades que vayan desarrollando con el tiempo.

La figura 6.4 ofrece una visualización aproximada del flujo que seguirá la información. Un usuario, o cliente HTTP, como por ejemplo curl, o in-

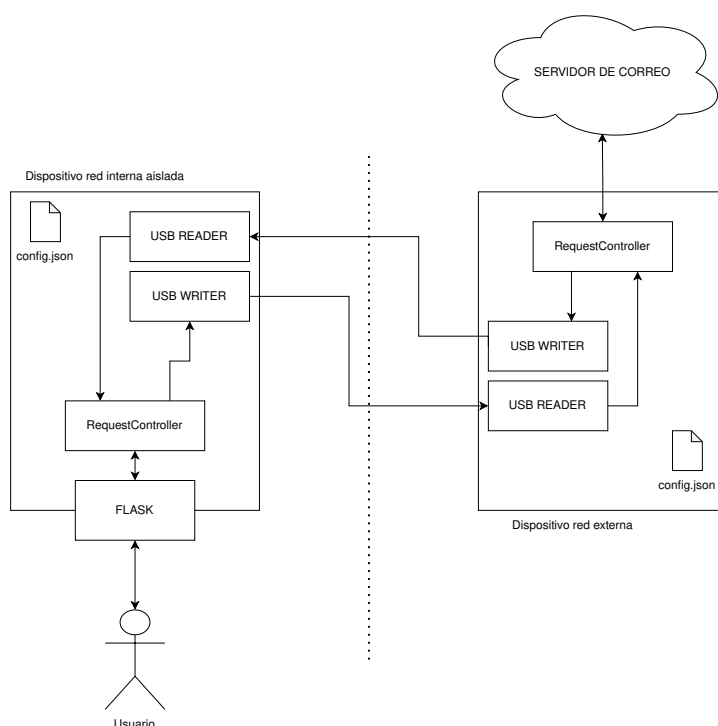


Figura 6.4: Estructura objetivo de la pasarela de correos

cluso una interfaz web, utilizarán los *endpoints* habilitados, tales como `/` o `/ {dirname} / {uid}`, para acceder a los recursos de su correo electrónico. Estos *endpoints* activarán métodos en Flask encargados de manejar esas rutas específicas.

Posteriormente, esos métodos se encargaran de crear el **RequestController** con la **Action** requerida a ser ejecutada localmente. También se encargarán de enviar los datos necesarios al otro dispositivo. Una vez que estos datos lleguen al otro dispositivo, el servidor del cable entrará en acción para lanzar otro **RequestController**. Este último ejecutará las operaciones pertinentes haciendo uso de su conexión a Internet, y finalmente, devolverá los datos de nuevo a la red interna.

El controlador de la ruta quedará bloqueado mientras espera la llegada de una respuesta. Una vez que los reciba, realizará las operaciones pertinentes y terminará devolviendo al cliente su respuesta.

Para llevar a cabo esta implementación, se comienza copiando el directorio `cable` de la pasarela de archivos para añadirlo al repositorio de la carpeta de correos. En este proceso, nuestro enfoque será depurar la clase **RequestController** para crear una versión personalizada que incluya las funcionalidades IMAP requeridas. El código que sea de *jtsec* y no haya sido modificado será añadido al `.gitignore` y no será incluido en el repositorio.

El extremo exterior de la pasarela de correos solo va a responder a las peticiones del extremo de la red aislada, por lo que se creará el fichero `cable_server.py` como punto de entrada del programa en el extremo externo. En él solo se creará y lanzará una instancia de `USBReader`.

Código 6.4: Script de punto de entrada del servidor del cable en la red externa

```

1
2 from cable.thread_controller import ThreadController
3 from cable.usb_reader import USBReader
4
5 thread_controller = ThreadController()
6 reader = USBReader(thread_controller)
7 reader.start()
```

Adicionalmente, se realizarán modificaciones en las clases `USBReader` y `USBWriter`, para que, tal como se detalló en el sprint anterior permitan el uso consistente de los cables sin hacer necesario su configuración manual. Es importante mencionar que estas modificaciones se consideran válidas sólo si la instalación se ha efectuado de acuerdo con la configuración descrita en la figura presentada durante el sprint de instalación. Con esto, se liberará al administrador de la responsabilidad de llevar a cabo esta tarea específica.

Las modificaciones en sí son simples, en lugar de buscar por el puerto indicado en el fichero de configuración, `USBWriter` siempre usará el puerto 1, y `USBReader` el puerto 2. Esto se ha realizado usando una constante (en Python no hay constantes, pero es convención utilizar para ello una variable declarada en mayúsculas) al inicio de los ficheros donde se declaran esas clases.

El siguiente paso consiste en crear `app.py`, el fichero de rutas y sus funciones. En él se utilizará `Flask`, y `flask_httpauth` para crear una ruta `/`. En esta, y en el resto de rutas, se utilizará `Basic Auth` para recoger las credenciales del usuario y se contrastan con las cuentas configuradas en el fichero de configuración. Posteriormente esta misma ruta será utilizada para recibir el índice de carpetas en el servidor de correos.

`flask_httpauth`, concretamente, ofrece una clase `HTTPBasicAuth`, con la que se crea un método personalizado para realizar la primera comprobación en la red aislada, además se introduce aquí un registro de *log* para registrar qué cuentas de correo no permitidas han resultado en un error de autenticación.

Código 6.5: Primera implementación de la ruta `/` para la autenticación frente a la configuración

```

1 from flask import Flask
2 from flask_httpauth import HTTPBasicAuth
3 from config import ALLOWED_ACCOUNTS, STORAGE, PAGE_SIZE
4
5 def abort(msg, code):
6     return
```

```
7
8 @auth.verify_password
9 def check_allowed(username: str, password: str) -> dict:
10     account = ALLOWED_ACCOUNTS.get(username)
11     if account is None:
12         logging.info(f"Attempt to use {username} FAILED. not allowed")
13         return False
14
15     account['email'] = username
16     account['pwd'] = password
17     return account
18
19
20 @auth.error_handler
21 def unauthorized():
22     return jsonify(error='Unauthorized'), 401
23
24
25 def get_account() -> dict:
26     return dict(auth.current_user())
27
28
29 @app.route('/')
30 @auth.login_required
31 def get_folders():
32     account: dict = get_account()
33     # continue if account is allowed
34
35
36 if __name__ == "__main__":
37     app.run(host="0.0.0.0", port=5000,
38           ssl_context=('cert/cert.pem', 'cert/key.pem'))
```

Basic Auth es un método simple de autenticación para obtener acceso a los recursos de un servicio en el que las credenciales son enviadas en la cabecera "*Authorization*" de la petición HTTP. Las credenciales, en formato usuario:contraseña son codificadas en Base64. La razón por la que se puede utilizar este método, al igual que otros más seguros como *Digest* o JSON web token (JWT) es que las peticiones solo se podrán iniciar desde la red interna. Los usuarios de esta, en ocasiones llamados personal con habilitación de seguridad, van a ser habitualmente, un número reducido de personas que han pasado los requerimientos necesarios para operar en la red interna. Además, al ser una red interna aislada solo se puede acceder a esta desde los equipos con acceso a esta red, en sitio de la organización. Por todo esto, usar Basic Auth y HTTPS presenta un nivel de seguridad razonable.

Para HTTPS se utilizarán unos certificados generados por nosotros con *openssl*. En un entorno de producción el administrador del sistema de pasarela se encargará de mover a la red interna el certificado y la clave de este, firmados por el CA Autorizado, para que puedan ser usados.

Al llegar a este punto también se ha usado un archivo `config.py` para centralizar la lectura de la configuración desde un fichero JSON situado en la raíz del proyecto. En concreto, el administrado podrá configurar:

- `imap_allowed_accounts`, contendrá un diccionario en el que cada par

clave-valor están formados por, la cuenta de correo y un segundo diccionario donde se puede indicar el dominio del servidor externo de correo y el puerto al que la pasarela ha de conectarse, necesarios para realizar la conexión IMAP con el servidor.

- *imap\_storage*, contiene la ruta absoluta hasta la carpeta que se utilizará más adelante para guardar los adjuntos temporalmente.
- *imap\_log*, contiene la ruta absoluta hasta la carpeta donde se escribe el fichero de logs.

Código 6.6: Ejemplo de configuración

```

1 {
2   "imap_storage": "/tmp/pasarela",
3   "imap_log": "/var/log/pasarela",
4   "imap_default_page_size": 10,
5   "imap_allowed_accounts": {
6     "1998morevi@gmail.com": {"server": "imap.gmail.com", "port":
7       993},
8     "morevi@correo.ugr.es": {"server": "correo.ugr.es", "port":
9       143}
10  }
11 }
```

El fichero `config.py` aprovecha para dar los valores por defecto a la pasarela, obligatoriamente sin cuentas permitidas. También se crean los directorios y se configura el registro de *logs*. Para este último se ha configurado utilizando la librería estándar *logging*, y se ha utilizado un *RotatingFileHandler*, para rotar el fichero y limitar el tamaño máximo de *logs*. Hasta 10 ficheros de 32MB cada uno. Si el administrador necesitase de más espacio solo tendría que modificarlo aquí.

Código 6.7: Script de configuración y valores por defecto

```

1 import json
2 import logging
3 from os import makedirs
4 from os.path import join
5 from logging.handlers import RotatingFileHandler
6 from logging import StreamHandler
7
8 # read config file
9 with open('config.json') as f:
10     config = json.load(f)
11
12 # initialize constants
13 ALLOWED_ACCOUNTS: dict = config.get('imap_allowed_accounts', {})
14 LOG: str = config.get('log', '/var/log/pasarela')
15 STORAGE: str = config.get('imap_storage', '/tmp/pasarela')
16 PAGE_SIZE: int = config.get('imap_default_page_size')
17
18 makedirs(LOG, exist_ok=True)
19 makedirs(STORAGE, exist_ok=True)
```

```
20
21 # logging
22 log_path = join(LOG, 'imap.log')
23 log_format = '%(asctime)s %(levelname) %(message)s'
24
25 logging.basicConfig(
26     level=logging.INFO,
27     format="%(asctime)s %(threadName)s %(levelname)s: %(message)s",
28     handlers=[
29         StreamHandler(),
30         RotatingFileHandler(log_path, maxBytes=32*1024*1024,
31                             backupCount=10),Endpoint
32     ])
33
```

Inicialmente, *imap\_allowed\_accounts* se declaraba como una lista, donde se buscaba por el correo. Posteriormente se eliminó esta búsqueda declarándose en forma de diccionario y accediendo directamente a los datos de determinado correo electrónico.

Tras este *sprint*, se permite la configuración de la pasarela, y se bloquea a los usuarios de usar cuentas de correo no permitidas.

### 6.3. Sprint 6. Login, carpetas y listado de carpetas

Durante este sprint, se aprovecha la oportunidad para introducir ciertas modificaciones que simplificarán significativamente el código, ayudando en el proceso de desarrollo. También se incorporarán las operaciones requeridas en la clase `RequestController` para permitir la recepción de los nombres de las carpetas disponibles en el servidor de correos.

La primera de estas mejoras consiste en hacer uso de un `enum` para los tipos de un paquete, son:

- 0, para indicar que aún no ha terminado la comunicación,
- 1, para indicar que ha terminado,
- 2, para indicar que ha habido algún error, siendo acompañado del mensaje en el cuerpo del paquete.

Estos valores eran introducidos como literales a la hora de enviar un paquete por el cable. Se van a terminar de fijar estos valores, siendo declarados con `enum`, para eliminar por completo el número de literales en el código, además de clarificar la intención de cada tipo de paquete.

Código 6.8: Enumerador de tipos de paquetes

```
1 class PacketType(Enum):
2     KEEP_ON = 0
3     END = 1
4     ERR = 2
```

Una vez hecho eso, se procede a crear las Action FOLDERS y RECV en la clase modificada RequestController. La primera de estas, será ejecutada en la red externa, recibirá como entrada en el primer y único paquete las credenciales del correo, junto al servidor y puerto obtenidos desde el fichero de configuración. Para enviar los datos, se utiliza un solo paquete (menos de 2MB), y se envía, en formato JSON codificado, los datos previamente mencionados.

Una vez se tienen los datos ya decodificados en la red externa, se iniciará sesión en el servicio de correo externo utilizando *imap-tools* como cliente de IMAP. En caso de dar error, pudiese ser por correo, contraseña, servidor o puerto incorrecto, se responde a la red interna con un código 401 (*Unauthorized*). En caso de autenticación correcta, se obtienen los nombres de las carpetas y se envían de vuelta en otro JSON codificado.

En la red interna, se ejecuta RECV. Esta define una operación genérica de recepción de uno o más paquetes, acumulando paquetes hasta recibir uno que tenga como tipo PacketType.END.value o PacketType.ERR.value.

Código 6.9: Clase RequestController con métodos de recepción y de listado de carpetas

```

1 class Action(Enum):
2     RECV = 1
3     FOLDERS = 2
4
5 class RequestController(Thread):
6     ...
7
8     def recv(self) -> tuple[dict, bytes]:
9         packet = self.get_packet()
10        data = packet['data']
11
12        while packet['last'] != PacketType.END.value:
13            packet = self.get_packet()
14            data += packet['data']
15
16        # return last packet metadata and all data
17
18        del packet['data']
19        del packet['util_data']
20
21        return packet, data
22
23    def folders(self) -> None:
24        _, data = self.recv()
25        data = json.loads(data.decode())
26
27        try:
28            mailbox = self.login(data['server'], data['port'], data['
29                                email'], data['pwd'])
30        except MailboxLoginError as e:
31            self.answer_error({"error": "Unauthorized", "status": 401})
32            return
33        except KeyError as e:
34            self.answer_error({"error": str(e), "status": 400})
35            return

```



```
35
36     # get folders
37     folders = [x.name for x in mailbox.folder.list()]
38     self.answer(json.dumps(folders).encode())
39     mailbox.logout()
```

Ahora, se hablará sobre la segunda mejora introducida. Para facilitar en gran medida el uso de la escritura y lectura del cable, se ha implementado, dentro de la carpeta `cable`, una nueva clase `Request`. Esta clase encapsula la creación y uso del `RequestController` ???. Permite una más sencilla comunicación y uso, recibiendo como entrada solo los datos codificados y la operación a realizar.

A partir de esta nueva clase se ha creado una segunda clase, `JsonRequest`. Esta se crea a partir de la primera, implementando además la codificación de diccionario a binario y viceversa.

Código 6.10: Clases `Request` y `JsonRequest` que encapsulan el uso de `RequestController`

```
1 class Request:
2     def __init__(self, msg: bytes, action: Action):
3         self.msg = msg
4         self.action = action.value
5         self.controller = self._create_controller()
6
7     def _create_controller(self):
8         controller = RequestController(thread_controller)
9         controller.action = Action.RECV.value
10        controller.start()
11        thread_controller.add_thread(controller)
12        return controller
13
14    def _send(self) -> int:
15        packet = writer.build_packet(
16            self.controller.ident, 0, PacketType.END.value, self.action
17            , self.msg)
18        status = writer.usb_write_package(packet)
19        return status
20
21    def _join(self) -> tuple[bytes, int]:
22        return self.controller.join()
23
24    def send(self) -> tuple:
25        self._send()
26        return self._join()
27
28 class JsonRequest(Request):
29     def __init__(self, msg: dict, action: Action):
30         bmsg = json.dumps(msg).encode()
31         super().__init__(bmsg, action)
32
33     def _join(self) -> tuple[bytes, dict]:
34         status, data = self.controller.join()
35         info = json.loads(data.decode())
36
37         return status, info
```

Finamente se aplica esta nueva clase para realizar ampliar el *sprint* anterior. Se envían los datos, y se recogen las carpetas, estas son enviadas al usuario en la respuesta JSON.

Código 6.11: Ruta / para pedir las carpetas

```
1 from cable.request_controller import Action
2 from cable.client import JsonRequest as Req
3
4 app.route('/')
5 @auth.login_required
6 def get_folders():
7     account: dict = get_account()
8
9     data, status = Req(account, Action.FOLDERS).send()
10    if status == 401:
11        logging.info(f"Attempt to use {account['email']} FAILED. {data['error']}")
12
13    return jsonify(data), status
```

Repasando el flujo de esta operación tendremos que:

1. El usuario o cliente utiliza "https://ip:5000/"
2. El sistema comprueba contra la configuración del administrador.
3. Se envían los datos con `Action.FOLDERS.value`.
4. La red interna inicia sesión y devuelve la lista de carpetas.
5. Se devuelven las carpetas al usuario si no hubo error.

Y en la siguiente figura un ejemplo de uso con CURL. La opción `-k` indica que el certificado no sea comprobado, ya que dará error al estar firmado por nosotros. La opción `-s` y `jq` es opcional, permite visualizar la respuesta con formato y color.

## 6.4. Sprint 7. Listado y descarga de correos

Siguiendo con la base que se establece en el sprint anterior, se procede a incorporar las funcionalidades de visualización de un listado de correos en una carpeta específica, así como la capacidad de descargar uno de estos correos basándonos en sus identificadores.

Para obtener el listado de correos, crearemos la `Action MAILS`, Esta se encargará de, habiendo recibido las credenciales y ahora también la carpeta que se pretende inspeccionar, realizar una petición al servidor y traer la información de los correos de esa carpeta.

```
~  
> curl -ks https://192.168.1.137:5000 -u 'morevi@correo.ugr.es:' | jq  
[  
  "INBOX.Trash",  
  "INBOX.BUZONInfoUGR",  
  "INBOX.BUZONdeEntradaUGR",  
  "INBOX.Drafts",  
  "INBOX.SPAM",  
  "INBOX.Sent",  
  "INBOX"  
]  
~  
> curl -ks https://192.168.1.137:5000 -u 'not_allowed@correo.ugr.es:bad_pass' | jq  
{  
  "error": "Unauthorized"  
}  
~  
> curl -ks https://192.168.1.137:5000 -u 'morevi@correo.ugr.es:bad_pass' | jq  
{  
  "error": "Unauthorized"  
}
```

Figura 6.5: Uso de CURL para obtener el listado de carpetas

Para evitar descargar el cuerpo y adjuntos de todos los correos en la carpeta, se ha utilizado la opción `headers_only=True` de *imap-tools*, que sirve con este objetivo en específico. De esta manera se reducirá en notable medida el tiempo de descarga y la cantidad de información que pide al ver la lista de correos.

También introduciremos como argumento el tamaño de página, configurable por el administrador del sistema. Con este argumento podremos reducir aún más la cantidad de información descargada, en caso de que el usuario tenga muchos correos.

Otra opción útil que se ha utilizado es `mark_seen=False`, esto permitirá obtener la lista de correo sin modificar si el correo ha sido leído.

En la red externa, una vez recibidos los correos, estos son transformados en JSON antes de ser enviados a la red aislada por el cable.

Código 6.12: Método de RequestController para pedir el listado de correos

```
1 def msg_to_json(self, msg) -> dict:  
2     return {  
3         "uid": msg.uid,  
4         "date": msg.date.strftime("%Y/%m/%d, %H:%M:%S"),  
5         "subject": msg.subject,  
6         "from": msg.from_,  
7         "cc": msg.cc,  
8         "bcc": msg.bcc,  
9         "to": msg.to,  
10        "flags": msg.flags  
11    }  
12  
13 def mails(self):  
14     _, data = self.recv()
```

```

15     data = json.loads(data.decode())
16
17     # split data a
18     page = data.get('page', 0)
19     page_size = data.get('page_size', PAGE_SIZE)
20     criteria = data.get('criteria', 'ALL')
21
22     try:
23         mailbox = self.login(data['server'], data['port'], data['email'],
24                               data['pwd'])
25     except MailboxLoginError as e:
26         self.answer_error({"error": "Unauthorized", "status": 401})
27         return
28     except KeyError as e:
29         self.answer_error({"error": str(e), "status": 400})
30         return
31
32     try:
33         mailbox.folder.set(data['folder'])
34     except:
35         self.answer_error({"error": "Folder not found", "status": 404})
36         return
37
38     # pagination
39     page_limit = slice(page * page_size, page * page_size + page_size)
40     mails = []
41     for msg in mailbox.fetch(criteria, bulk=True, limit=page_limit,
42                             mark_seen=False, headers_only=True):
43         mails.append(self.msg_to_json(msg))
44
45     self.answer(json.dumps({"mails": mails}).encode())
46
47     # release
48     mailbox.logout()

```

Al llegar la lista de correos a la red aislada, los datos no necesitan ser transformados por lo que se retornan al cliente directamente desde la función llamada con el *endpoint* `/carpeta`.

Código 6.13: Ruta para pedir los correos en una carpeta

```

1 @app.route('/<folder>')
2 @auth.login_required
3 def get_headers(folder):
4     data: dict = get_account()
5     data['folder'] = folder
6
7     try:
8         # must be integers
9         data['page_size'] = int(request.args.get('page_size', PAGE_SIZE))
10        data['page'] = int(request.args.get('page', 0))
11    except:
12        abort(400)
13
14    data, status = Req(data, Action.MAILS).send()
15    if status == 401:
16        logging.info(f"Attempt to use {data['email']} FAILED. {data['error']}")

```

```
~  
> curl -ks https://192.168.1.137:5000/INBOX.PASARELA -u 'morevi@correo.ugr.es:' | jq  
{  
  "mails": [  
    {  
      "bcc": [],  
      "cc": [],  
      "date": "2023/07/09, 12:32:39",  
      "flags": [  
        "\\Seen"  
      ],  
      "from": "morevi@correo.ugr.es",  
      "subject": "TEST",  
      "to": [  
        "morevi@correo.ugr.es"  
      ],  
      "uid": "1"  
    }  
  ]  
}
```

Figura 6.6: Uso de CURL para pedir los correos

```
17  
18     return jsonify(data), status
```

A a continuación se muestra como se solicitaría la lista de correos utilizando CURL.

Ahora que se tiene la capacidad de visualizar la lista de correos, se avanzará para permitir la selección de un correo específico utilizando su `uid` (el identificador) y la carpeta correspondiente. El objetivo será descargar tanto el cuerpo del correo como sus archivos adjuntos. Inicialmente, esta nueva funcionalidad seguirá una estructura similar, pero el JSON devuelto por la red externa contendrá información adicional sobre los adjuntos y su contenido, en formato Base64.

En caso de que el JSON resultante supere los 2MB, será dividido en trozos más pequeños para su transmisión a través del cable. Dado que los servidores de correo electrónico suelen aplicar límites de tamaño para los correos electrónicos, no esperamos tener problemas de memoria.

Tras intentar el acceso a la cuenta de correo y descargar el correo, se itera sobre sus adjuntos y se escriben en el almacenamiento temporal del dispositivo de la red externa, para realizar su análisis.

Los adjuntos deberán someterse a un análisis antivirus, para ello ClamAV. Si un adjunto no pasa el análisis, esta información se indicará en el JSON resultante. Se ha creado una función `avscan` que recibe como entrada la ruta absoluta hacia el archivo. Esta función lanzara un proceso para analizar el archivo. En caso de encontrar algo también devolverá un mensaje para notificar al usuario de qué se ha encontrado.

---

Código 6.14: Función para hacer uso del antivirus

```
1 def avscan(path: str) -> tuple[bool, str]:
2     if not os.path.exists(path):
3         raise FileNotFoundError(path + "was not found")
4
5     proc = subprocess.Popen(['clamdscan', '-m', '--fdpass', path],
6                             stdout=subprocess.PIPE, stderr=subprocess.PIPE)
7     out, _ = proc.communicate()
8
9     # passed
10    if str(out).find('FOUND') == -1:
11        return True, ""
12
13    # failed
14    return False, out
```

Código 6.15: Método de RequestController que descarga y analiza los adjuntos de un correo

```
1 def download(self):
2     ...
3
4     msg_data = self.msg_to_json(msg)
5     msg_data['text'] = msg.text
6     msg_data['html'] = msg.html
7     msg_data['attachments'] = []
8
9     for attachment in msg.attachments:
10        attachment_data = {
11            'filename': attachment.filename,
12            'content_type': attachment.content_type,
13            'size': attachment.size,
14        }
15
16        # write to tmp
17        tmp_path = f"/tmp/{email}_{uid}_{attachment.filename}"
18        with open(tmp_path, 'wb') as f:
19            f.write(attachment.payload)
20
21        # av
22        ok, msg = avscan(tmp_path)
23        if not ok:
24            attachment_data['av'] = msg
25        else:
26            attachment_data['payload'] = base64.b64encode(attachment.
27                payload).decode()
28
29        msg_data['attachments'].append(attachment_data)
30        os.remove(tmp_path)
31
32        self.answer(json.dumps(msg_data).encode())
```

Al llegar a la red interna, los archivos adjuntos se almacenarán en el directorio marcado en la configuración como temporal para que los usuarios puedan acceder a ellos y descargarlos. Una vez almacenados, los enlaces originales en el diccionario se reemplazarán por enlaces que permitan el acceso a estos archivos almacenados. Antes de terminar se escribirá al fichero de *logs* los resultados de los análisis realizados.

Al intentar utilizar un enlace de descarga, se verificarán nuevamente las credenciales del usuario, asegurando así un proceso de descarga seguro y controlado.

Código 6.16: Definición de la ruta para descargar un correo electrónico

```
1
2 @app.route('/<folder>/<uid>', methods=['GET'])
3 @auth.login_required
4 def get_mail(folder, uid):
5     data: dict = get_account()
6     data['folder'] = folder
7     data['uid'] = uid
8     email = data['email']
9
10    response, status = Req(data, Action.DOWNLOAD).send()
11    if status == 401:
12        logging.info(f"Attempt to use {email} FAILED. {response['error']}")
13        abort(401)
14
15    def store_att(email: str, uid: str, att: dict) -> str:
16        filename = att['filename']
17        path = join(STORAGE, email, uid, filename)
18
19        makedirs(dirname(path), exist_ok=True)
20        with open(path, 'wb') as f:
21            payload = att['payload']
22            content = base64.b64decode(payload.encode())
23            f.write(content)
24
25        return join('/att', uid, filename)
26
27    if not 'attachments' in response:
28        return jsonify(response), status
29
30    for i, att in enumerate(response['attachments']):
31        if 'av' in att:
32            logging.critical(f"AV scan for {att['filename']} FAILED. {att['av']}")
33            continue
34        else:
35            logging.info(f"AV scan for {att['filename']} PASSED")
36
37        url = store_att(email, uid, att)
38        del att['payload']
39        att['link'] = url
40        response['attachments'][i] = att
41
42    return jsonify(response), status
```

Código 6.17: Definición de una ruta para descargar los adjuntos de un correo

```
1 @app.route('/att/<uid>/<filename>', methods=['GET'])
2 @auth.login_required()
3 def download_att(uid, filename):
4     account: dict = get_account()
5
6     # do a folder request to force an authentication
```

```

7  # data retrieved is useless, we only have to check for 200
8  # to know if the user can access the <email> folder
9  # inside the storage (or happened some error)
10 #
11 response, status = Req(account, Action.FOLDERS).send()
12 if status == 401:
13     logging.info(f"Attempt to use {account['email']} FAILED. {
14         response['error']}")
15 if status != 200:
16     return jsonify(response), status
17 return send_from_directory(join(STORAGE, account['email'], uid),
18     filename)

```

A continuación podremos probar estas nuevas rutas usando de nuevo CURL. En la figura podremos como, a parte de devolver los datos del correo también hemos logrado traer el cuerpo del mensaje junto a los enlaces para acceder a los adjuntos descargados.

```

~
> curl -ks https://192.168.1.137:5000/INBOX.PASARELA/1 -u 'morevi@correo.ugr.es: ' | jq
{
  "attachments": [
    {
      "content_type": "application/zip",
      "filename": "test.zip",
      "link": "/att/1/test.zip",
      "size": 1596
    }
  ],
  "bcc": [],
  "cc": [],
  "date": "2023/07/09, 12:32:39",
  "flags": [
    "\\Seen"
  ],
  "from": "morevi@correo.ugr.es",
  "html": "",
  "subject": "TEST",
  "text": "Hello here is your attachment.",
  "to": [
    "morevi@correo.ugr.es"
  ],
  "uid": "1"
}

```

Figura 6.7: Uso de CURL para la descarga de un correo

Al llegar a este punto, se puede elegir una carpeta, dentro de esta un correo y descargarlo. Suponiendo un gran avance para los usuarios, o para el desarrollador para continuar expandiendo.

## 6.5. Sprint 8. Operaciones de gestión sobre correos

Es por eso que ahora se tratará de introducir algunas de las operaciones más habituales que se realizan sobre correos electrónicos. Permitiremos a un usuario marcar un correo como NO LEÍDO, o, lo que es lo mismo, le permitiremos borrar de un correo el *flag* de LEÍDO, que es añadido interna



y automáticamente al descargar el correo. También daremos la posibilidad de eliminar el correo completamente.

El código de ambas operaciones funciona de manera similar así que se comentará solo una de estas tareas haciendo hincapié en las diferencias. Las rutas de ambas van a necesitar de la carpeta y del `uid` del correo sobre el que actuaremos.

En el caso del borrado se aprovecha la misma ruta que la usada para descargar el correo, pero usando en lugar de utilizar el método HTTP GET, usaremos DELETE.

Para el caso del marcado como no leído, usaremos PUT, y obligaremos a usar un argumento `unseen` para indicar qué es este el *flag* que se pretende modificar. De esta manera, en caso de querer añadir más operaciones se pueden seguir añadiendo como argumentos.

Una vez iniciada la operación se enviarán a la red externa las credenciales, el servidor y puerto, tal y como hemos estado haciendo hasta ahora. Una vez allí se iniciará sesión, devolviendo error si fuese necesario. En caso afirmativo se dará paso al uso de *imap-tools*, de manera similar que en resto de operaciones, para llevar a cabo las tareas de marcado y de borrado.

Código 6.18: Uso de *imap-tools* para modificar los flags y borrar un correo

```
1
2 # remove seen flag
3 mailbox.flag([uid], (MailMessageFlags.SEEN), False)
4
5 # delete
6 mailbox.delete([data['uid']])
```

Para probar el marcado como NO LEÍDO, se muestra un correo que tiene el *flag* activo, se modifica y vuelve a enseñar desde el listado de correos.

Finalmente, para el borrado se muestra de manera de manera similar, enseñando la lista de correos de la carpeta antes y después de borrar el correo electrónico.

## 6.6. Sprint 9. MUA web.

Una vez finalizados los *sprints* previos, se habría completado la infraestructura necesaria para aprovechar las funcionalidades implementadas en la pasarela de correos. Al menos desde un cliente HTTP, como se viene mostrando hasta ahora.

Con el fin de simplificar su utilización, se plantea implementar una interfaz web utilizando *React*. Esta interfaz permitirá a los usuarios acceder a los casos de uso que hemos implementado en etapas anteriores.

Para agilizar el proceso de desarrollo y lograr una interfaz visualmente atractiva, se utiliza Bootstrap 5. Esto permitirá incorporar fácilmente elementos visuales básicos que contribuirán a la usabilidad y la experiencia del

```
double-client on git main [!>] via nodejs v18.14.0 via py v3.10.12
> curl -sk https://192.168.1.137:5000/INBOX.PASARELA/1 -u 'morevi@correo.ugr.es:' | jq
{
  "attachments": [
    {
      "content_type": "application/zip",
      "filename": "test.zip",
      "link": "/att/1/test.zip",
      "size": 1596
    }
  ],
  "bcc": [],
  "cc": [],
  "date": "2023/07/09, 12:32:39",
  "flags": [
    "\\Seen"
  ],
  "from": "morevi@correo.ugr.es",
  "html": "",
  "subject": "TEST",
  "text": "Hello here is your attachment.",
  "to": [
    "morevi@correo.ugr.es"
  ],
  "uid": "1"
}

double-client on git main [!>] via nodejs v18.14.0 via py v3.10.12
> curl -sk -X PUT 'https://192.168.1.137:5000/INBOX.PASARELA/1?unseen=1' -u 'morevi@correo.ugr.es:' | jq
{
  "msg": "OK"
}

double-client on git main [!>] via nodejs v18.14.0 via py v3.10.12
> curl -sk 'https://192.168.1.137:5000/INBOX.PASARELA' -u 'morevi@correo.ugr.es:' | jq
{
  "mails": [
    {
      "bcc": [],
      "cc": [],
      "date": "2023/07/09, 12:32:39",
      "flags": [],
      "from": "morevi@correo.ugr.es",
      "subject": "TEST",
      "to": [
        "morevi@correo.ugr.es"
      ],
      "uid": "1"
    }
  ]
}
```

Figura 6.8: Uso de CURL para marcar como no leído

usuario en la web. De esta manera, se podrá centrar en la funcionalidad esencial de la interfaz sin enfocarse demasiado por los detalles de diseño y estilo.

También se ha manejado la posibilidad de usar otras librerías similares como MaterialUI o Tailwind. MaterialUI ofrece, además de los estilos componentes ya creados para ser usados inmediatamente, como pueden ser calendarios. Tailwind trae cantidad estilos para personalizar el sitio web un en detalle sin llegar a escribir CSS.

Bootstrap al contrario, pretende el minimalismo, ofrece menos estilos y componentes, pero usando esta librería se contará con los recursos necesarios para escribir los estilos de un MUA rápidamente.

Por otro lado, en cuanto a la decisión de utilizar React para desarrollar esta interfaz, esta fundamentada en la experiencia laboral del desarrollador

```
double-client on git main [!>] via nodejs v18.14.0 via py v3.10.12
> curl -sk https://192.168.1.137:5000/INBOX.PASARELA/1 -u 'morevi@correo.ugr.es:' | jq
{
  "attachments": [
    {
      "content_type": "application/zip",
      "filename": "test.zip",
      "link": "/att/1/test.zip",
      "size": 1596
    }
  ],
  "bcc": [],
  "cc": [],
  "date": "2023/07/09, 12:32:39",
  "flags": [
    "\\Seen"
  ],
  "from": "morevi@correo.ugr.es",
  "html": "",
  "subject": "TEST",
  "text": "Hello here is your attachment.",
  "to": [
    "morevi@correo.ugr.es"
  ],
  "uid": "1"
}

double-client on git main [!>] via nodejs v18.14.0 via py v3.10.12
> curl -sk -X PUT 'https://192.168.1.137:5000/INBOX.PASARELA/1?unsee=1' -u 'morevi@correo.ugr.es:' | jq
{
  "msg": "OK"
}

double-client on git main [!>] via nodejs v18.14.0 via py v3.10.12
> curl -sk 'https://192.168.1.137:5000/INBOX.PASARELA' -u 'morevi@correo.ugr.es:' | jq
{
  "mails": [
    {
      "bcc": [],
      "cc": [],
      "date": "2023/07/09, 12:32:39",
      "flags": [],
      "from": "morevi@correo.ugr.es",
      "subject": "TEST",
      "to": [
        "morevi@correo.ugr.es"
      ],
      "uid": "1"
    }
  ]
}
```

Figura 6.9: Uso de curl CURL para borrar un correo

con este framework. Aprovechando las capacidades de React, se consigue actualizar dinámicamente tanto la lista de correos como la sección de visualización de mensajes al modificar el estado de la aplicación conforme el usuario interactúa con ella.

La interfaz se compilará usando Vite, y podrá ser servida con cualquier servidor web, en este caso se ha utilizado Nginx, debido a lo sencillo de configurar que resulta desde una instalación reciente. Simplemente se modifica el directorio a servir desde la configuración por defecto tendremos el MUA accesible.

### 6.6.1. Descripción de la interfaz

La interfaz en sí, consiste de cuatro partes diferenciadas, de arriba a abajo y de izquierda a derecha tendremos:

### Cabecera

Aquí se encuentra el nombre de la aplicación junto con el formulario de inicio de sesión. Si ocurre algún error, el botón se resaltará en rojo, para indicarlo.

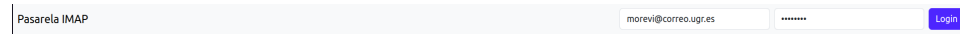


Figura 6.10: Formulario de inicio de sesión

### Listado de carpetas

Una vez que el usuario inicie sesión de manera exitosa, se inicializará el listado de carpetas. En él aparecen todas las carpetas disponible en el servidor de correos. Seleccionar una carpeta provoca que se cargue la lista de correos en el siguiente componente de la interfaz, la lista de correos.

Tras seleccionar una carpeta se mantendrá iluminada de azul para indicar el usuario cuál carpeta esta visualizando en cualquier momento.

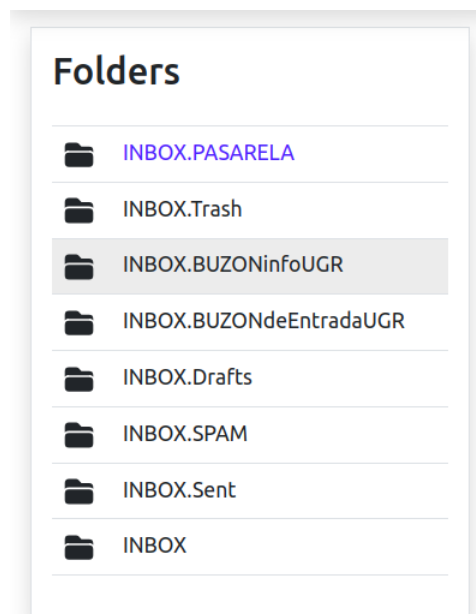


Figura 6.11: Listado de carpetas del usuario

### Listado de correos electrónicos

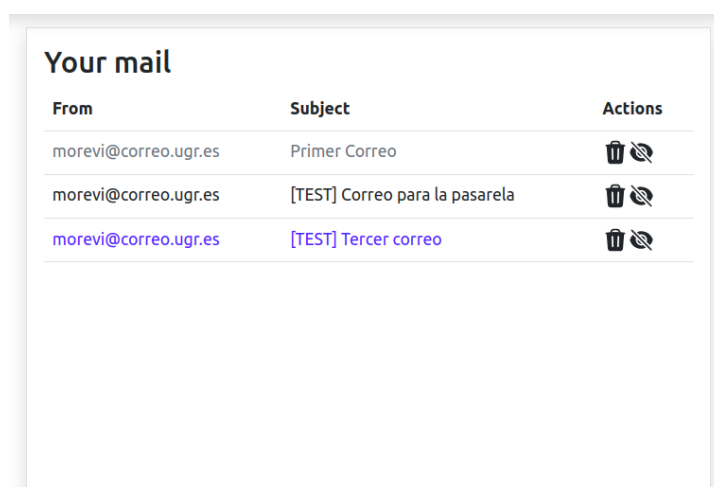
Tras seleccionar una de las carpetas disponibles en la lista de carpetas, se realizará una nueva petición al servidor, que cargará este nuevo componente, el listado de correos electrónicos.

En esta lista se muestran los datos necesarios para seleccionar un correo, el emisor y el asunto, además a la derecha de cada uno se incluyen dos botones, para poder modificar el correo. El primero, una papelera, permitirá borrar el correo, mientras el segundo se puede usar para marcar el correo como no leído. Ambos botones envían la petición HTTP de la que se habló en su sprint.

Los correos, aparecen de tres maneras diferentes:

- En gris, si ya han sido leídos.
- en negro, si es un nuevo correo que aún no ha sido leído. Marcar el correo como no leído hace que vuelva a aparecer en este color.
- En azul, si es el correo seleccionado por el usuario.

Al seleccionar un correo, haciendo clic sobre él, se realizará la petición para descargar el correo y sus adjuntos, rellenando el siguiente componente, la vista de correo.









From	Subject	Actions
morevi@correo.ugr.es	Primer Correo	 
morevi@correo.ugr.es	[TEST] Correo para la pasarela	 
morevi@correo.ugr.es	[TEST] Tercer correo	 

Figura 6.12: Listado de correos del usuario

### Vista de correo

Inicialmente en blanco hasta que se seleccione un correo, esta sección permitirá a los usuarios visualizar el emisor, al receptor, el asunto, a la fecha de envío, el cuerpo del mensaje, así como el nombre de los adjuntos en un enlace para poder descargarlos.

Al seleccionar el adjunto este es descargado a la máquina del usuario para que este pueda trabajar con él.



Figura 6.13: Sección con el contenido del correo

### 6.6.2. Review

Tras completar este sprint, software está listo para ser desplegado y usado por los usuarios. El administrador del sistema con acceso a la red interna tendrá que instalar los dispositivos y conectarlos a las redes reales. Es necesario tener en cuenta que tanto el código del *backend* como sus dependencias, o el código compilado del *frontend*, deberán moverse a la red interna de manera manual.

Finalmente, la interfaz completa, los usuarios de la red aislada ahora son capaces de leer y borrar sus correos electrónicos en los servicios de correo habilitados por el administrador, desde una interfaz sencilla e intuitiva.

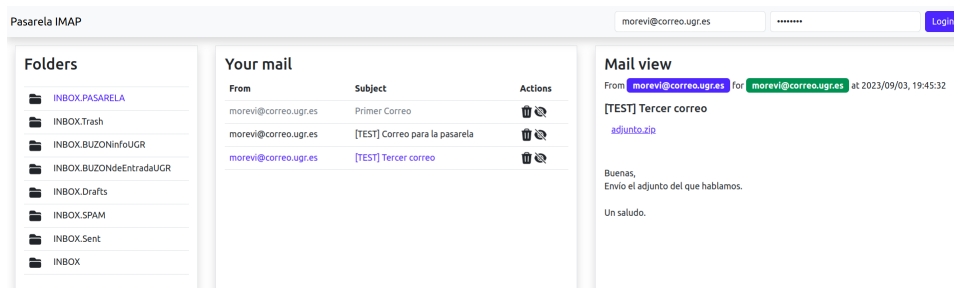


Figura 6.14: Interfaz de usuario completa

El repositorio completo con el código escrito durante este proyecto está alojado en Github en la dirección <https://github.com/morevi/pasarela-imap>.

## Capítulo 7

# Conclusiones y trabajos futuros

En este último capítulo se valorarán de nuevo los objetivos iniciales tras la finalización del proyecto, además se ofrecerán posibles líneas para otros trabajos en la misma materia.

### 7.1. Conclusiones

El objetivo principal de este proyecto era implementar una solución que permitiera a los usuarios de una red aislada de Internet acceder al correo electrónico de manera segura y controlada, con el propósito de mitigar posibles brechas de seguridad en dicha red.

Para lograr este objetivo, llevamos a cabo un estudio exhaustivo sobre el concepto de *airgap* y las pasarelas de datos, identificando sus aplicaciones típicas y los requisitos esenciales para su implementación efectiva. De donde se pudo extraer que una configuración adecuada debe residir en la red interna y cerrar el canal de comunicación de forma predeterminada.

En particular, se realizó un análisis detallado de la pasarela de archivos de la empresa *jtsec*, la cual se encuentra actualmente en uso. De esta pasarela, pudimos extraer conocimientos valiosos que sirvieron como base para el desarrollo de nuestra propia pasarela de correo electrónico.

Asimismo, se llevó a cabo un análisis teórico exhaustivo de los correos electrónicos, comprendiendo su funcionamiento y las diversas partes que interactúan durante el proceso de envío y recepción. Este estudio nos proporcionó una comprensión profunda de cómo operan los clientes de IMAP y los MUA, cuyo conocimiento fue esencial para el desarrollo de nuestro proyecto.

En lo que respecta a la seguridad, tras analizar las notas recopiladas durante el estudio y consultar las guías proporcionadas por el CCN (Centro Criptológico Nacional), llegamos a la conclusión de que la pasarela que se iba a implementar debía llevar a cabo un escaneo antivirus de los archivos adjuntos entrantes. Además, se determinó que era fundamental mantener un registro de *logs* que donde tanto los intentos de acceso como los nombres de

los archivos adjuntos que no pasaran el análisis quedasen registrados.

Posteriormente, siguiendo con el objetivo de diseño previo al desarrollo, se decidió la arquitectura de la nueva pasarela de correos, detallando sus requisitos funcionales y no funcionales, y se ampliándolos para incluir casos de uso específicos.

Estos pasos sirvieron como base fundamental para la posterior fase de desarrollo de la pasarela. Se implementaron mejoras con el fin de optimizar el desarrollo, y se procedió a construir una API utilizando Flask. Finalmente, se desarrolló una interfaz simple que permitiera la utilización de esta API.

De esta manera, logramos satisfactoriamente el objetivo principal de habilitar la lectura y gestión de correos electrónicos desde la red aislada, y al mismo tiempo, marcamos el cumplimiento de los demás objetivos que se habían establecido en el transcurso del proyecto.

## 7.2. Trabajos futuros

Tras la realización del proyecto, existen ciertas líneas de trabajo que pueden partir desde este, o de manera alternativa, repetirse con otro enfoque.

- Añadir el resto de funcionalidades habituales que suele implementar un MUA en la pasarela de correos. Estas pueden ser, por mencionar algunas:
  - Selección múltiple, para mejorar la usabilidad para el usuario,
  - Etiquetas, o filtrado de correos según su contenido, asunto, emisor.

Estas son solo algunas, los MUA están llenos de funcionalidades.

- Una línea, que no ramificada de la realizada, sino alternativa, podría tratar de implementar algún tipo de *proxy* IMAP, de manera que cualquier cliente pueda directamente utilizar directamente el protocolo para comunicarse con el servidor. De esta manera utilizando, por ejemplo *Thunderbird* podría hacerse uso de gran parte de las funcionalidades. Esto podría abrir incluso la idea de unir el trabajo del SMTP con el de IMAP bajo la misma pasarela, con acceso a ambos desde el mismo agente de correos.
- En caso de poder implementarse un proxy del protocolo directamente sobre la pasarela, podría ser interesante explorar diferentes protocolos, de uso frecuente. Como SFTP, para depositar y recoger archivos directamente en un servidor externo, o, para seguir trabajando con los correos electrónicos, POP3.

En definitiva, el ámbito de las pasarelas de datos puede dar varias vías de investigación. Todas ellas requieren la exploración del protocolo que se



quiera implementar y la búsqueda de los puntos clave que pueden afectar a la seguridad de los datos en la red interna, que son los que se pretende proteger desde un principio.



# Bibliografía

- [1] W. D. Bryant, *International Conflict and Cyberspace Superiority: Theory and Practice*. July 2015.
- [2] R. Fernández, “Emails: correos electrónicos recibidos diariamente en el mundo hasta 2025,” Feb 2022.
- [3] CCN, “Pasarelas de intercambio seguro de información,” *Píldoras tecnológicas PYTEC*, June 2018.
- [4] CCN-CERT, “Guías ccn-stic de seguridad,” 2023.
- [5] CCN-CERT, “Taxonomía de referencia para productos de seguridad tic - anexo d.6: Pasarelas para intercambio de datos,” *Guías CCN-STIC de Seguridad*, July 2023.
- [6] CCN-CERT, “Interconexión en el ens,” *Guías CCN-STIC de Seguridad*, Oct. 2017.
- [7] CCN-CERT, “Metodología de evaluación para la certificación nacional esencial de seguridad (lince),” *Guías CCN-STIC de Seguridad*, Mar. 2022.
- [8] P. R. Pedrosa, “Air gap para sistemas clasificados,” *Universidad de Granada*, 2020.
- [9] StarTech.com, “USB 3.0 Data Transfer Cable for Mac & PC - USB & PS/2 Devices | StarTech.com Europe.”
- [10] libusb, “libusb: libusb-1.0 API Reference,” June 2022.
- [11] J. Malaco, “pyusb: Python USB access module.”
- [12] J. L. L. Gómez, “Email airgap,” *Universidad de Granada*, Sept. 2022.
- [13] J. C. Klensin, “Simple Mail Transfer Protocol,” RFC 5321, Internet Engineering Task Force, Oct. 2008.
- [14] G. M. Fernández, R. M. Carrión, and R. A. R. Gómez, *SISTEMAS Y SERVICIOS TELEMÁTICOS*. 2018.

- 
- [15] D. Wood, “Programming internet email: Mastering internet messaging systems,” Aug. 1999.
  - [16] A. Melnikov and B. Leiba, “Internet Message Access Protocol (IMAP) - Version 4rev2,” RFC 9051, Internet Engineering Task Force, Aug. 2021.
  - [17] CPSTIC, “Products in cpstic.”
  - [18] CCN-CERT, “Configuración segura de pasarelas de AUTEK INGENIERÍA,” *Guías CCN-STIC de Seguridad*, June 2018.
  - [19] Autek Ingenierías, *PSTmail Version 3.0*, May 2011. Revision 5.
  - [20] M. Cohn, *Succeeding With Agile*. Nov. 2009.
  - [21] Raspberry Pi, *Raspberry Pi 4 Model B*, June 2019.
  - [22] Flask, *Flask Documentation (2.3.x)*.
  - [23] “Status of Python Versions,” *Python Developer’s Guide*, 2023.
  - [24] Flask-HTTPAuth, *Flask-HTTPAuth’s documentation*.
  - [25] V. Kaukin, “imap-tools: Work with email by IMAP,” Aug. 2023.

# Glosario

**Airgap** Medida de seguridad implementada para asegurar que una red está aislada de otras redes menos seguras.

**CCN** Centro Criptológico Nacional

**CURL** Herramienta de línea de comandos con la que se pueden realizar peticiones HTTP

**Flask** *Web framework* escrito en Python

**HTTP** *Hypertext Transfer Protocol*.

**IMAP** *Internet Message Access Protocol*

**jtsec** Laboratorio de ciberseguridad en Granada

**Pasarela de datos** Dispositivo que permite algún tipo de comunicación controlada entre la red aislada y la red externa

**POP3** *Post Office Protocol version 3*

**Red aislada** Red sin acceso a Internet

**Red externa** Red con acceso a Internet

**SMTP** *Simple Mail Transfer Protocol*

**USB** Universal Serial Bus



## Apéndice A

# Guía de instalación de la pasarela de correos

En este primer anexo está destinado al administrador del sistema que quiera hacer uso esta pasarela de correos. Esta instalación se tendrá que realizar en primera instancia, con ambos dispositivos conectados a la red externa, para poder hacer la instalación del sistema, paquetes, software y dependencias. Finalmente se conectarán ambos dispositivos en sus respectivas redes y se encenderá el software.

### A.1. Instalación requerimientos de sistema

1. En primer lugar se instala sobre las SD (puede usar RPI Imager), el sistema operativo Raspberry PI OS Lite, formateando completamente el almacenamiento. Durante este paso se configurará el usuario y SSH para poder acceder a las raspberries. También se configurará el acceso a internet en este paso.
2. Posteriormente se montarán las Raspberry PI sobre su funda, ventilador, disipadores, cables de corriente. En este paso se han de seguir las instrucciones suministradas junto a los dispositivos.
3. Tras el montaje, se conectarán ambos dispositivos a la corriente eléctrica y serán encendidos. Si la configuración del router hace la asignación de IP dinámicamente se puede hacer uso de otras herramientas, como Fing una aplicación de android, para encontrar las IP de los dispositivos.
4. Haciendo uso de SSH para acceder a los sistemas, se instalan los requerimientos del sistema de pasarela

---

Código A.1: Instalación de requisitos del sistema

```

1 apt-get update
2 apt-get install libusb-1.0-0-dev python3.8 python3.8-venv
  python3.8-pip clamav clamav-daemon git nginx

```

5. A continuación se al usuario a hacer uso de los dispositivos USB

Código A.2: Configuración de usuario para permitir uso del cable

```

1 usermod -a -G plugdev pasarela
2 echo 'SUBSYSTEM=="usb", ATTRS{idVendor}=="067b", ATTRS{
3   idProduct}=="27a1" MODE="0666"' >> /etc/udev/rules.d/99-
  pasarela.rules
4 udevadm control --reload-rules

```

## A.2. Instalación *backend*

Ahora se instalará el software de la pasarela y se creará un demonio para que se mantenga lanzada en segundo plano en ambos dispositivos.

1. Se clona el repositorio `morevi/pasarela-imap` en ambas máquinas. Este repositorio contiene tanto el backend como el frontend.
2. Una vez desplazado a la nueva carpeta con el código, para instalar las dependencias del software se utiliza:

Código A.3: Uso de pip para instalar las dependencias

```

1 python -m pip install -r requirements.txt

```

3. A continuación se creará un servicio de systemd. Para ello, se crea en cada dispositivo un fichero `/etc/systemd/system/pasarela.service` que posea el siguiente contenido:

Código A.4: Configuración de un servicio de systemd para poder ejecutar la pasarela en segundo plano

```

1 [Unit]
2 After=network.target
3 [Service]
4 Restart=always
5 RestartSec=1
6 User=pasarela
7 ExecStart=/usr/bin/python /home/pasarela/pasarela-imap/
  app.py
8
9 [Install]
10 WantedBy=multi-user.target

```



En el dispositivo que finalmente quede en la red aislada. En el dispositivo de la red externa, el punto de entrada del programa es `cable_server.py`. Como ambos dispositivos empiezan a diferenciarse en su contenido, se recomienda el uso de algún tipo de pegatina o método para identificarlas de manera externa.

### A.3. Instalación *frontend*

A continuación se muestra como se compila y deja funcionando el *frontend* de la pasarela, haciendo uso de *npm*, *vite* y *nginx* en el proceso. Esta sección solo ha de realizarse en el dispositivo marcado para la red aislada.

1. Se desplaza hacia la subcarpeta `web` dentro del repositorio clonado, y se realiza `npm install` para traer los requisitos para la compilación.
2. Posteriormente se hace uso de `npm run build` para compilar la aplicación web.
3. Una vez se dispone de la carpeta `build` con la interfaz compilada en HTML y Javascript, se habilita en `\etc\nginx\sites-enabled` un fichero de configuración para indicar a `nginx` que archivos estáticos debe servir.

Código A.5: Configuración de NGINX para servir archivos estáticos

```
1  server {  
2      listen 8000;  
3      root /home/pasarela/pasarela-imap/web/build  
4  }
```

En el ejemplo se usa el puerto 8000, se puede configurar tanto el puerto como el uso de un certificado para habilitar el uso de HTTPS. Visite la documentación de NGINX para esto.

### A.4. Instalación de los dispositivos en sus respectivas redes

1. Se conecta cada dispositivo a la red correspondiente, mediante uso de Ethernet. La interfaz WIFI debe ser apagada, para evitar conexiones externas, con `sudo ifconfig wlan0 down`.
2. A continuación puede conectar los cables USB a USB de la manera indicada en la imagen 6.3.
3. Una vez realizado esto y comprobado que sea correcto con `lsusb`. Se procede a habilitar y encender los servicios configurados:

Código A.6: Lanzado de los servicios

```
1       systemctl enable nginx
2       systemctl start nginx
3       systemctl enable pasarela
4       systemctl start pasarela
```

Realizado este paso se puede comprobar, con `systemctl status {service}` si ha ocurrido algún. Si todo ha ido correctamente, tendremos en el puerto indicado al interfaz web funcionando.

## Apéndice B

# Guía de uso de la pasarela de correos

Tras instalar la aplicación tal y como fue explicado en el Anexo I. Guía de instalación de la pasarela de correos, se procede a mostrar como se usa para recibir correos en la red aislada.

En un primer momento el administrador del sistema debe entrar con ssh al servidor, y en la raíz del código de la pasarela escribir un `config.json` B.1 con un contenido similar al siguiente. En esta configuración se habilitan dos cuentas de correo electrónico, indicándose junto a su puerto y servidor IMAP.



```
"imap_storage": "/tmp/pasarela",
"imap_log": "/var/log/pasarela",
"imap_default_page_size": 10,
"imap_allowed_accounts": [{
  "1998morevi@gmail.com": {"server": "imap.gmail.com", "port": 993},
  "morevi@correo.ugr.es": {"server": "correo.ugr.es", "port": 143}
}]
```

Figura B.1: Configuración de ejemplo

Una vez modificado el fichero de configuración el servicio debe ser reiniciado.

Posteriormente, como usuario, lugar se accede con el navegador desde la red aislada a la IP (o dominio) que se ha habilitado para la pasarela, para encontrar la interfaz base, sin datos cargados. Además se utiliza HTTPS aunque ponga que el certificado no es seguro, al ser auto-firmado y no estar importado como autoridad certificadora en el navegador B.2.

A continuación, se intenta hacer inicio de sesión a una cuenta no habilitada para demostrar el filtrado de acceso y las entradas generadas en el fichero de logs B.3.

Ahora se introducen credenciales correctas para realizar el acceso al bu-

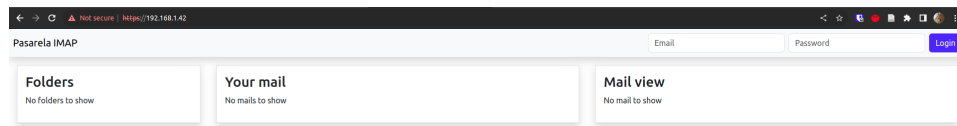


Figura B.2: Interfaz sin iniciar sesión

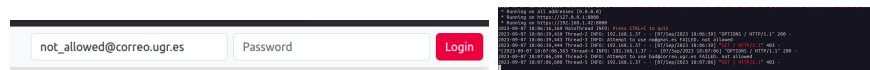


Figura B.3: Inicio de sesión fallido

zón de correos desde la red aislada. Al hacer eso se cargarán las carpetas disponibles B.4.

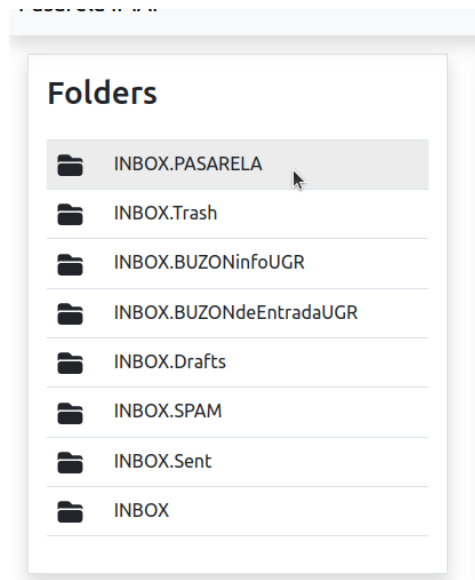


Figura B.4: Interfaz tras realizar inicio de sesión correcto

Una vez tiene las carpetas cargadas puede hacer clic en cualquiera de ellas para cargar la lista de correos disponibles. En esta nueva lista, se pueden realizar diferentes acciones. Utilizando el icono de la "papelera" se podrá borrar ese correo B.5.

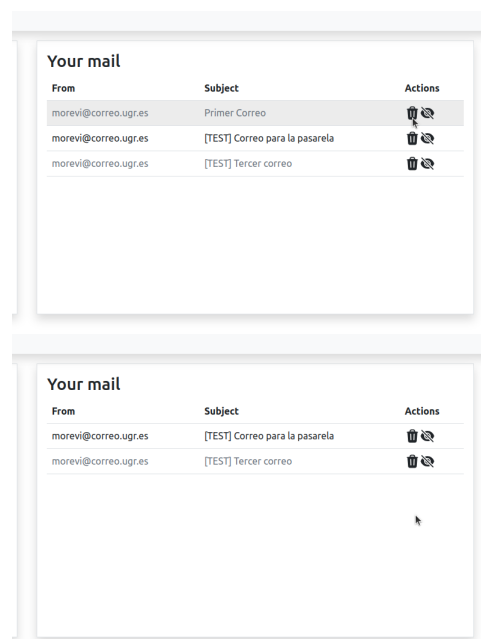


Figura B.5: Uso del botón de borrado en la lista de correos

En esta vista también se encuentra el botón de "no leído", señalado con el icono de un ojo. Usando este botón, podremos marcar los correos que hayan sido previamente leídos (los correos en gris) B.6.

Haciendo clic en un correo, se consigue que este sea completamente descargado y mostrado en la sección de la derecha de la interfaz B.7. En esta sección se encuentran otros datos útiles como la fecha de envío, el destinatario (sería diferente en caso de ser una lista de difusión). Haciendo clic en el adjunto conseguimos descargar este.

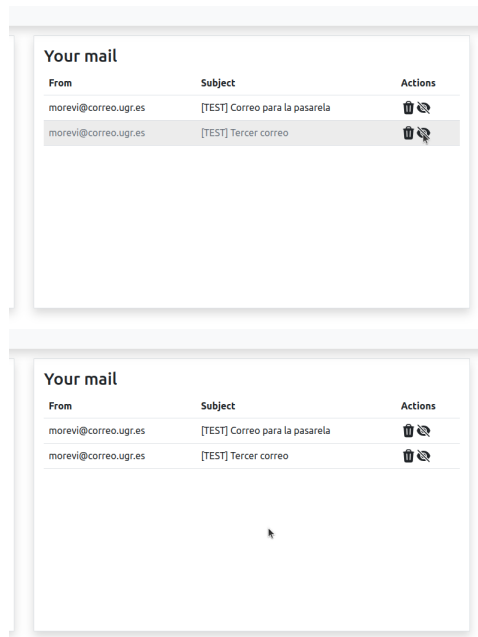


Figura B.6: Uso del botón para marcar como no leído en la lista de correos

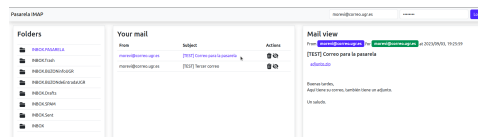


Figura B.7: Interfaz de usuario completa con carpetas, correos y vista de correo