UNIVERSITAT POLITÈCNICA DE VALÈNCIA

_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería de Telecomunicación

Desarrollo de un servicio de intercambio de archivos con cifrado extremo a extremo (E2E)

Trabajo Fin de Grado

Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación

AUTOR/A: Martín Vargas, Pablo

Tutor/a: López Patiño, José Enrique

CURSO ACADÉMICO: 2025/2026

Resumen

El presente Trabajo de Fin de Grado propone el diseño e implementación de SecureShare, una aplicación web orientada al intercambio seguro de archivos mediante enlaces compartibles, incorporando mecanismos de cifrado extremo a extremo (End-to-End Encryption, E2E) y control de acceso mediante enlaces de un solo uso. La motivación del proyecto surge de la necesidad de disponer de herramientas de compartición de archivos que garanticen la confidencialidad de la información incluso frente al propio servidor que aloja los ficheros. En muchas soluciones convencionales, el archivo se almacena o procesa en el servidor en texto claro, lo que implica un riesgo potencial de exposición de datos. Frente a ello, la propuesta plantea una arquitectura en la que el archivo es cifrado localmente en el navegador del emisor antes de ser enviado, de forma que el servidor únicamente almacena una versión cifrada del contenido y nunca accede al fichero original en claro. El sistema permitirá al usuario subir distintos tipos de archivos, configurar opciones de seguridad como la caducidad del enlace, el uso único o la protección mediante contraseña, y generar un enlace seguro para compartir con el destinatario. Posteriormente, el receptor podrá acceder al enlace, descargar el contenido cifrado y realizar el proceso de descifrado en su propio navegador para recuperar el archivo original. Desde el punto de vista técnico, el proyecto se estructura en tres componentes principales. En primer lugar, un frontend web encargado de la selección del archivo, el cifrado local —incluyendo tratamiento por bloques para soportar archivos de gran tamaño—, la subida del contenido cifrado y su posterior descarga y descifrado. En segundo lugar, un backend responsable del almacenamiento del archivo cifrado y de la gestión de reglas de negocio, como la expiración de enlaces, la revocación o el marcado de enlaces consumidos. Por último, una base de datos almacenará únicamente los metadatos mínimos necesarios, como el identificador, la fecha de creación, la expiración, el estado de consumo y la ruta del archivo cifrado.

Resum

El present Treball de Fi de Grau proposa el disseny i implementació de *SecureShare, una aplicació web orientada a l'intercanvi segur d'arxius mitjançant enllaços compartibles, incorporant mecanismes de xifratge extrem a extrem (*End-*to-*End *Encryption, E2E) i control d'accés mitjançant enllaços d'un sol ús. La motivació del projecte sorgix de la necessitat de disposar de ferramentes de compartició d'arxius que garantisquen la confidencialitat de la informació fins i tot enfront del propi servidor que allotja els fitxers. En moltes solucions convencionals, l'arxiu s'emmagatzema o processa en el servidor en text clar, la qual cosa implica un risc potencial d'exposició de dades. Enfront d'això, la proposta planteja una arquitectura en la qual l'arxiu és xifrat localment en el navegador de l'emissor abans de ser enviat, de manera que el servidor únicament emmagatzema una versió xifrada del contingut i mai accedix al fitxer original en clar. El sistema permetrà a l'usuari pujar diferents tipus d'arxius, configurar opcions de seguretat com la caducitat de l'enllaç, l'ús únic o la protecció mitjançant contrasenya, i generar un enllaç segur per a compartir amb el destinatari. Posteriorment, el receptor podrà accedir a l'enllaç, descarregar el contingut xifrat i realitzar el procés de desxifrat en el seu propi navegador per a recuperar l'arxiu original. Des del punt de vista tècnic, el projecte s'estructura en tres components principals. En primer lloc, un *frontend web encarregat de la selecció de l'arxiu, el xifratge local —incloent tractament per blocs per a suportar arxius de gran grandària—, la pujada del contingut xifrat i la seua posterior descàrrega i desxifrat. En segon lloc, un *backend responsable de l'emmagatzematge de l'arxiu xifrat i de la gestió de regles de negoci, com l'expiració d'enllaços, la revocació o el marcat d'enllaços consumits. Finalment, una base de dades emmagatzemarà únicament les metadades mínimes necessàries, com l'identificador, la data de creació, l'expiració, l'estat de consum i la ruta de l'arxiu xifrat.

Abstract

This Bachelor's Thesis proposes the design and implementation of SecureShare, a web application focused on secure file sharing through shareable links, incorporating end-to-end encryption (E2E) and access control through one-time-use links. The motivation behind the project comes from the need for file-sharing tools that ensure data confidentiality even against the very server hosting the files. In many conventional solutions, files are stored or processed on the server in plaintext, which creates a potential risk of data exposure. In contrast, this proposal defines an architecture in which the file is encrypted locally in the sender's browser before being uploaded, so that the server only stores an encrypted version of the content and never has access to the original file in plaintext. The system will allow users to upload different types of files, configure security options such as link expiration, single-use access, and optional password protection, and generate a secure link to share with the recipient. The recipient will then be able to open the link, download the encrypted content, and decrypt it directly in their own browser in order to recover the original file. From a technical perspective, the project is structured into three main components. First, a web frontend responsible for file selection, local encryption —including chunk-based processing to support large files—, encrypted upload, and later download and decryption. Second, a backend server in charge of storing the encrypted file and enforcing business rules such as link expiration, revocation, or marking a link as consumed after first use. Finally, a database will store only the minimum required metadata, such as the identifier, creation date, expiration date, consumption status, and encrypted file path.

RESUMEN EJECUTIVO

La memoria del TFG del Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación debe desarrollar en el texto los siguientes conceptos, debidamente justificados y discutidos, centrados en el ámbito de la ingeniería de telecomunicación

CONCEPT (ABET)	**CONCEPTO (traducción)**	**¿Cumple? (S/N)**	**¿Dónde? (páginas)**
1. IDENTIFY:	1. IDENTIFICAR:		
1.1. Problem statement and opportunity	1.1. Planteamiento del problema y oportunidad	S	5, 6, 8
1.2. Constraints (standards, codes, needs, requirements & specifications)	1.2. Toma en consideración de los condicionantes (normas técnicas y regulación, necesidades, requisitos y especificaciones)	S	10, 11, 17, 18, 19, 42, 43
1.3. Setting of goals	1.3. Establecimiento de objetivos	S	5
2. FORMULATE:	2. FORMULAR:		
2.1. Creative solution generation (analysis)	2.1. Generación de soluciones creativas (análisis)	S	7, 8, 11, 17, 18
2.2. Evaluation of multiple solutions and decision-making (synthesis)	2.2. Evaluación de múltiples soluciones y toma de decisiones (síntesis)	S	8, 17, 19, 21
3. SOLVE:	3. RESOLVER:	S	
3.1. Fulfilment of goals	3.1. Evaluación del cumplimiento de objetivos	S	35, 36, 37, 38, 39, 45
3.2. Overall impact and significance (contributions and practical recommendations)	3.2. Evaluación del impacto global y alcance (contribuciones y recomendaciones prácticas)	S	42, 43, 44, 45

Índice

Índice de Figuras

Índice de Tablas

Capítulo 1. Introducción

1.1 Contexto y Motivación

En la actualidad, el intercambio de archivos a través de internet es una práctica habitual tanto en entornos personales como profesionales. Existen múltiples plataformas que permiten compartir información de manera rápida y sencilla, como servicios de almacenamiento en la nube o sistemas de transferencia de archivos mediante enlaces.

Sin embargo, muchas de estas soluciones presentan limitaciones desde el punto de vista de la seguridad y la privacidad. En muchos casos, los archivos son almacenados en servidores donde el proveedor mantiene acceso potencial a las claves o a mecanismos que permiten recuperar la información original. Esto supone un riesgo potencial de exposición de datos sensibles ante accesos no autorizados, filtraciones o ataques dirigidos contra la infraestructura del servicio.

Ante este escenario, surge la necesidad de desarrollar herramientas que permitan garantizar la confidencialidad de la información durante todo su ciclo de vida, incluyendo el almacenamiento y la transferencia. En este contexto, el cifrado extremo a extremo (*End-to-End Encryption*, E2E) se presenta como una solución eficaz, ya que asegura que únicamente los extremos de la comunicación (emisor y receptor) pueden acceder al contenido.

1.2 Planteamiento del Problema

El problema principal que aborda este proyecto es la falta de mecanismos de compartición de archivos que garanticen la privacidad completa de la información frente a terceros, incluyendo el propio servidor que actúa como intermediario.

Las soluciones tradicionales permiten compartir archivos mediante enlaces, pero en muchos casos el proveedor del servicio mantiene la capacidad de acceder al contenido almacenado. Esto implica que la confidencialidad de la información depende no solo del usuario, sino también de la confianza depositada en la plataforma y en la protección de las claves de descifrado almacenadas o gestionadas por el proveedor.

Por tanto, se plantea la necesidad de diseñar un sistema que permita compartir archivos de forma sencilla mediante enlaces, pero que al mismo tiempo garantice que el contenido permanece cifrado en todo momento y que el servidor no pueda acceder a la información en claro.

1.3 Objetivos del proyecto

El objetivo principal de este Trabajo de Fin de Grado es diseñar e implementar una aplicación web que permita el intercambio seguro de archivos mediante el uso de cifrado extremo a extremo.

De forma más concreta, se establecen los siguientes objetivos específicos:

- Desarrollar un sistema que permita cifrar los archivos localmente en el navegador del usuario antes de su envío al servidor.
- Garantizar que el servidor únicamente almacena versiones cifradas de los archivos, sin acceso al contenido original.

- Implementar un sistema de generación de enlaces únicos para la compartición de archivos.
- Incorporar mecanismos de control de acceso mediante enlaces de un solo uso y caducidad configurable.
- Permitir la compartición de archivos entre usuarios registrados mediante un sistema de solicitudes controladas.
- Implementar un mecanismo seguro de intercambio de claves criptográficas mediante criptografía asimétrica, evitando que el servidor tenga acceso a las claves de descifrado en texto claro.
- Diseñar una arquitectura sencilla y eficiente basada en tecnologías web accesibles.

1.4 Alcances y Limitaciones

El sistema desarrollado en este proyecto se centra en la implementación de un prototipo funcional que demuestra la viabilidad del intercambio seguro de archivos mediante cifrado extremo a extremo en un entorno web.

El alcance del proyecto incluye el desarrollo de una aplicación web que permite la subida, cifrado, almacenamiento y descarga de archivos, así como la gestión de enlaces de compartición y solicitudes entre usuarios.

Asimismo, el sistema almacena las claves privadas de los usuarios localmente en el navegador mediante *localStorage*, por lo que la recuperación de dichas claves depende del dispositivo y navegador utilizados por el usuario.

No obstante, el sistema presenta ciertas limitaciones propias de un entorno académico, como la ausencia de despliegue en infraestructura de producción, la falta de mecanismos avanzados de escalabilidad o la implementación de medidas de seguridad adicionales propias de sistemas comerciales a gran escala.

Capítulo 2. Análisis de soluciones existentes

2.1 Plataformas actuales de compartición de archivos

En la actualidad existen numerosas plataformas que permiten compartir archivos de forma sencilla a través de internet. Entre las más utilizadas se encuentran servicios de almacenamiento en la nube como Google Drive, Dropbox o OneDrive, así como herramientas específicas de transferencia de archivos mediante enlaces, como WeTransfer.

Estas soluciones ofrecen funcionalidades como la subida de archivos, la generación de enlaces compartibles y, en algunos casos, la configuración de caducidad o restricciones de acceso. Gracias a estas características, se han convertido en herramientas ampliamente adoptadas tanto en entornos personales como profesionales.

No obstante, la mayoría de estas plataformas se basan en un modelo en el que el proveedor del servicio actúa como intermediario de confianza, siendo responsable tanto del almacenamiento de los archivos como de la gestión de los mecanismos de acceso y recuperación de la información.

2.2 Limitaciones de seguridad en soluciones tradicionales

A pesar de su popularidad, muchas de estas soluciones presentan limitaciones importantes en términos de seguridad y privacidad. En muchos casos, aunque los archivos puedan almacenarse cifrados, el proveedor mantiene acceso a las claves criptográficas o a mecanismos que permiten recuperar la información original.

Esto implica que:

- el proveedor del servicio puede potencialmente acceder al contenido de los archivos,
- la información puede verse comprometida en caso de brechas de seguridad,
- la confidencialidad depende en gran medida de la confianza depositada en terceros.

Esto implica que la seguridad del sistema depende no solo de la fortaleza del cifrado utilizado, sino también de cómo se gestionan y almacenan las claves criptográficas.

Además, aunque muchas plataformas utilizan cifrado durante la transmisión de datos (por ejemplo, mediante protocolos seguros como HTTPS), esto no garantiza la protección de los datos una vez almacenados en el servidor.

Por tanto, las soluciones tradicionales no aseguran una protección completa de la información, especialmente en contextos donde la confidencialidad es un requisito crítico.

Muchas de las vulnerabilidades más comunes en aplicaciones web están recogidas en OWASP Top Ten [3].

2.3 Cifrado extremo a extremo (E2E)

El cifrado extremo a extremo (*End-to-End Encryption*, E2E) se presenta como una alternativa eficaz para garantizar la confidencialidad de la información durante todo su ciclo de vida. En este modelo, los datos son cifrados en el dispositivo del emisor antes de ser enviados. Para ello,

los sistemas E2E suelen utilizar mecanismos de criptografía híbrida, combinando algoritmos simétricos para el cifrado de los datos y algoritmos asimétricos para el intercambio seguro de claves criptográficas.

De este modo:

- el servidor actúa únicamente como intermediario de almacenamiento y transmisión,
- no tiene acceso a la información en claro, ni dispone de las claves privadas necesarias para recuperar el contenido original.
- la seguridad depende exclusivamente de los extremos de la comunicación.

Este enfoque es ampliamente utilizado en aplicaciones de mensajería segura, como Signal o WhatsApp, donde la privacidad de las comunicaciones es un requisito fundamental.

Sin embargo, su aplicación en sistemas de compartición de archivos mediante enlaces es todavía limitada, especialmente en soluciones que combinan simplicidad de uso con altos niveles de seguridad.

2.4 Enfoques actuales de protección de datos

En el ámbito de la protección de la información, existen diferentes enfoques orientados a garantizar la seguridad de los datos durante su transmisión y almacenamiento.

Uno de los más comunes es el cifrado en tránsito, que protege la comunicación entre el cliente y el servidor mediante protocolos seguros. Este enfoque evita que terceros puedan interceptar la información durante su envío, pero no impide que el servidor tenga acceso al contenido una vez almacenado.

Por otro lado, el cifrado en el lado del cliente permite que los datos sean protegidos antes de ser enviados, reduciendo la dependencia de la seguridad del servidor y evitando que el proveedor tenga acceso directo al contenido original o a las claves necesarias para descifrarlo. Este enfoque se ha visto reforzado en los últimos años gracias al desarrollo de tecnologías web que permiten realizar operaciones criptográficas directamente en el navegador.

A pesar de estos avances, muchas plataformas siguen adoptando modelos en los que el control de la información recae parcialmente en el proveedor del servicio, lo que pone de manifiesto la necesidad de soluciones que prioricen la privacidad desde el diseño.

2.5 Justificación de la propuesta

A partir del análisis de las soluciones existentes, se observa que, aunque ofrecen facilidad de uso y funcionalidades avanzadas, no garantizan en todos los casos la privacidad completa de la información.

La propuesta desarrollada en este proyecto surge como respuesta a esta necesidad, planteando un sistema que combina la simplicidad de los enlaces compartibles con las garantías de seguridad proporcionadas por el cifrado extremo a extremo.

De este modo, se busca ofrecer una alternativa que permita a los usuarios compartir archivos de forma sencilla, manteniendo al mismo tiempo el control sobre la confidencialidad de la información, sin depender de la confianza en terceros.

Asimismo, el sistema incorpora un mecanismo de intercambio seguro de claves basado en criptografía asimétrica, permitiendo que únicamente los usuarios autorizados puedan recuperar las claves de descifrado de los archivos compartidos.

Capítulo 3. Análisis y diseño del sistema

3.1 Requisitos funcionales

A partir de los objetivos planteados en el proyecto, se definieron una serie de requisitos funcionales que describen el comportamiento esperado de la aplicación y las funcionalidades que debe ofrecer al usuario.

Los principales requisitos funcionales identificados son los siguientes:

- Permitir la subida de archivos desde el navegador del usuario.
- Realizar el cifrado del archivo localmente antes de enviarlo al servidor.
- Generar enlaces únicos asociados a cada archivo compartido.
- Permitir configurar la caducidad del enlace generado.
- Implementar enlaces de un solo uso para limitar el acceso al archivo.
- Permitir el registro e inicio de sesión de usuarios.
- Posibilitar el envío de solicitudes de compartición entre usuarios registrados.
- Implementar un mecanismo seguro de intercambio de claves criptográficas entre usuarios registrados mediante criptografía asimétrica.
- Permitir aceptar o rechazar solicitudes de acceso a archivos compartidos.
- Descargar y descifrar archivos directamente desde el navegador del receptor.

Además, el sistema diferencia dos modos principales de utilización:

- un modo invitado, orientado a comparticiones rápidas sin necesidad de registro,
- y un modo autenticado, que incorpora funcionalidades adicionales de gestión y compartición entre usuarios.

3.2 Requisitos no funcionales

Además de las funcionalidades principales, el sistema debe cumplir una serie de requisitos no funcionales relacionados con la seguridad, el rendimiento y la experiencia de usuario.

Entre ellos destacan:

- Garantizar que el servidor nunca tenga acceso ni al archivo en texto claro ni a las claves privadas necesarias para su descifrado.
- Mantener una arquitectura sencilla y fácilmente desplegable en entornos locales.
- Permitir el manejo de distintos tipos de archivos.
- Mantener tiempos de respuesta adecuados durante el cifrado y la subida de archivos.
- Proporcionar una interfaz web simple e intuitiva.
- Utilizar tecnologías ampliamente compatibles con navegadores modernos.
- Mantener las claves privadas de los usuarios exclusivamente en el lado cliente, evitando su almacenamiento o transmisión al servidor.
- Asimismo, se buscó que la aplicación pudiera ser utilizada sin necesidad de instalar software adicional, ejecutándose completamente desde el navegador mediante tecnologías web estándar.

3.3 Arquitectura general del sistema

La aplicación desarrollada sigue una arquitectura cliente-servidor dividida en tres componentes principales: frontend, backend y base de datos.

El frontend es el encargado de la interacción con el usuario y de las operaciones criptográficas realizadas en el navegador. Entre sus funciones se encuentran la selección de archivos, la generación local de claves criptográficas, el cifrado y descifrado de archivos, así como la recuperación segura de claves mediante criptografía asimétrica.

El backend actúa como intermediario encargado de gestionar las peticiones realizadas por los usuarios, almacenar los archivos cifrados y aplicar las reglas de negocio relacionadas con la expiración de enlaces, el control de enlaces consumidos o la gestión de solicitudes entre usuarios.

Por último, la base de datos almacena los metadatos necesarios para el funcionamiento del sistema, evitando almacenar archivos en texto claro o claves criptográficas de descifrado sin protección.

La Figura 1 muestra un esquema simplificado de la arquitectura general de la aplicación.

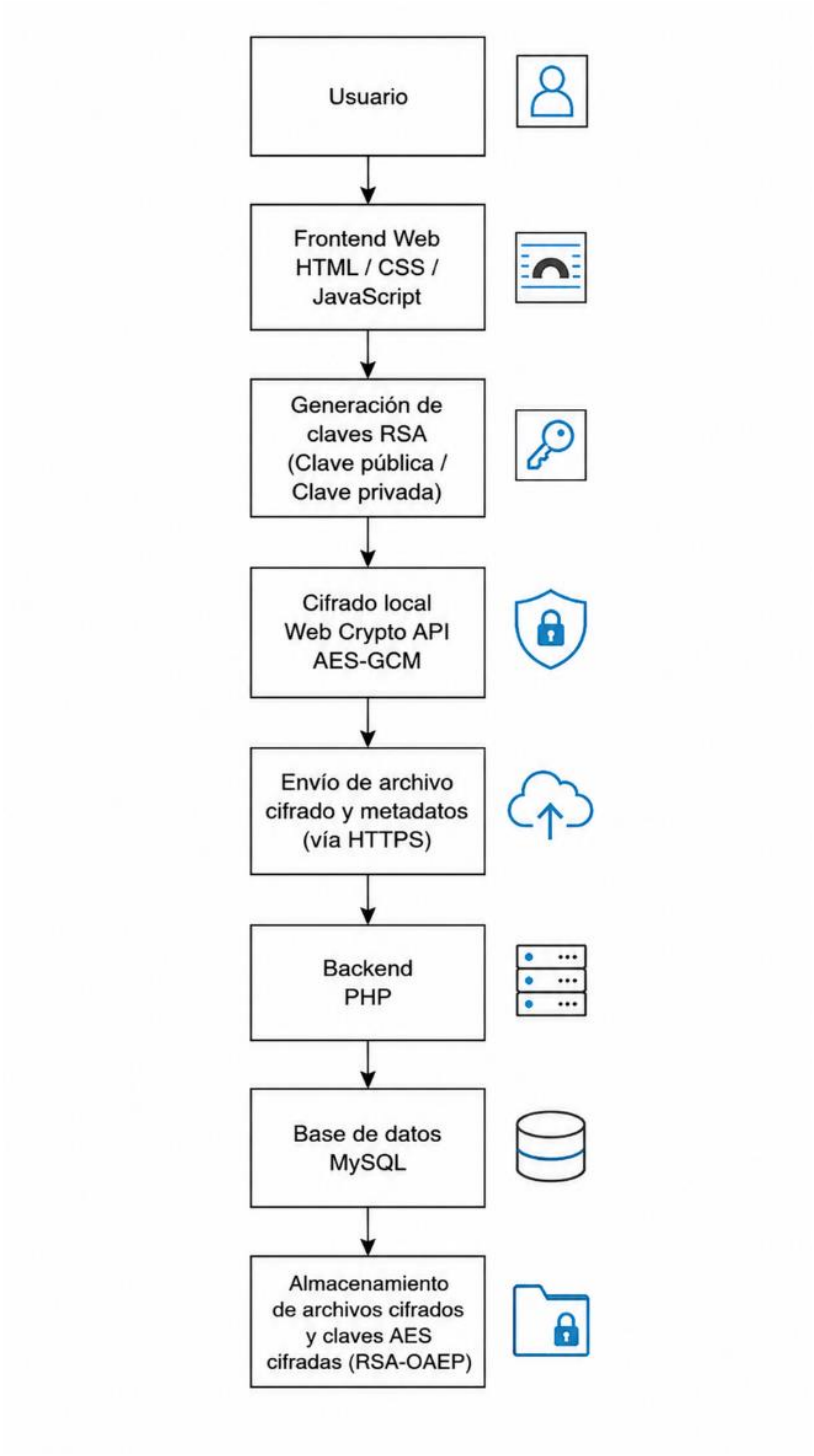

Figura 1. Arquitectura general de la aplicación SecureShare.

3.4 Diseño de la base de datos

La base de datos utilizada en el proyecto se implementó mediante MySQL y está compuesta por varias tablas principales relacionadas entre sí.

La tabla users almacena la información de los usuarios registrados, incluyendo el correo electrónico, la contraseña protegida mediante funciones hash seguras y la clave pública asociada a cada usuario.

La tabla shares contiene la información asociada a los archivos compartidos, como:

- el identificador único,
- el token del enlace,
- la ruta del archivo cifrado,
- la fecha de expiración,
- el estado del enlace,
- y los metadatos necesarios para recuperar y validar el archivo cifrado.

Además, la tabla share_requests permite gestionar el envío de solicitudes entre usuarios registrados, almacenando el emisor, el receptor y el estado de aceptación de cada solicitud.

Asimismo, la tabla share_requests almacena las claves AES cifradas mediante la clave pública del receptor, permitiendo implementar un mecanismo de intercambio seguro de claves sin que el servidor tenga acceso a las claves originales en texto claro.

Este diseño permite separar correctamente la información de autenticación, los archivos compartidos y la gestión de permisos entre usuarios.

La Figura 2 es un esquema representativo de la base de datos creada con todos sus parámetros.

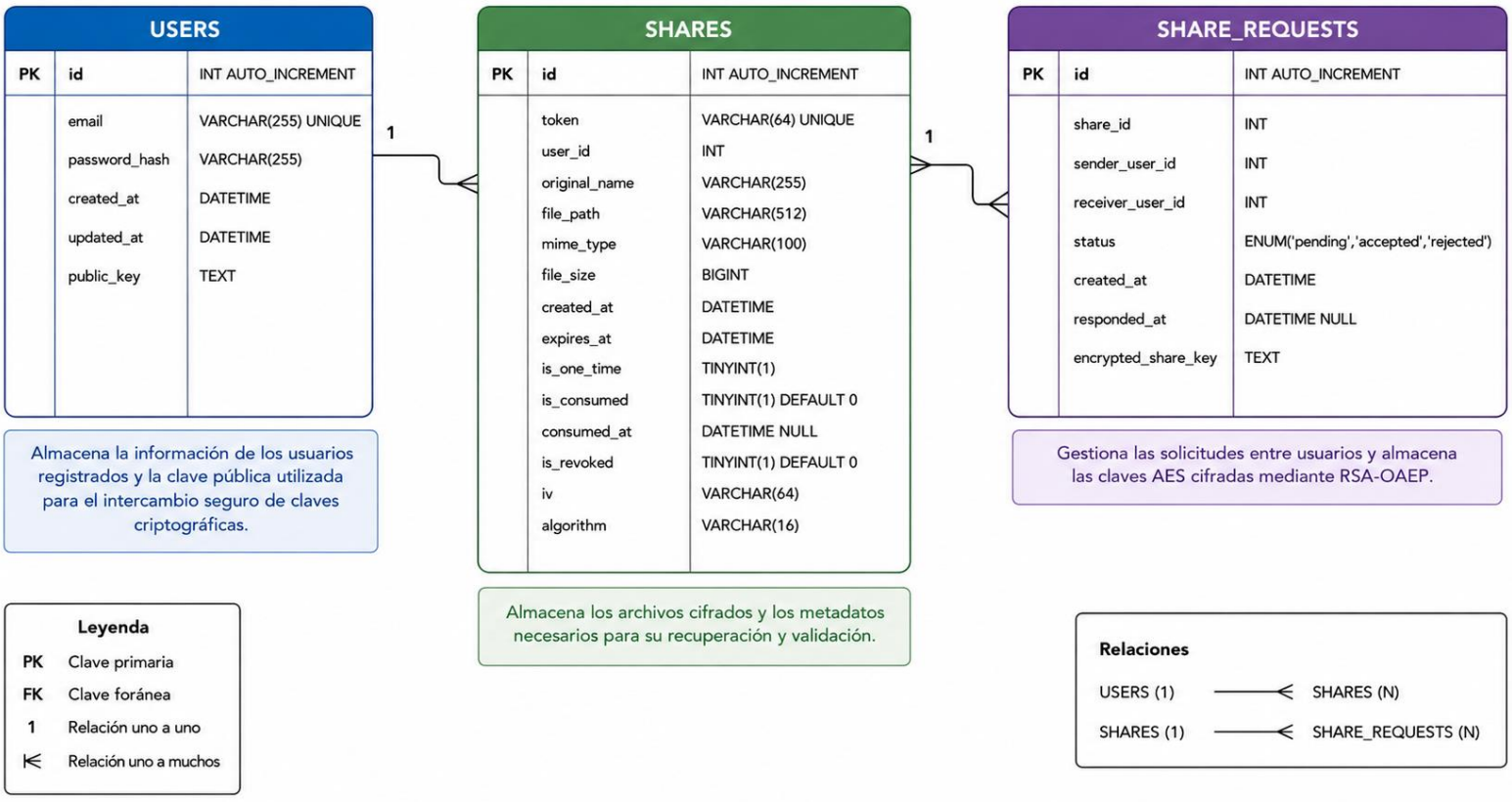

Figura 2. Diagrama de base de datos de la aplicación SecureShare

3.5 Flujo general de funcionamiento

El funcionamiento general del sistema comienza cuando el usuario selecciona un archivo desde la interfaz web.

A continuación, el archivo es cifrado localmente en el navegador utilizando mecanismos criptográficos implementados mediante JavaScript y la API Web Crypto. Una vez cifrado, el contenido es enviado al servidor, que únicamente almacena la versión cifrada del archivo.

Tras completarse la subida, el sistema genera un identificador único asociado al archivo cifrado. En el caso de usuarios registrados, la clave AES utilizada para cifrar el archivo se protege utilizando criptografía asimétrica. Para ello, la clave AES es cifrada mediante la clave pública del receptor y almacenada en el servidor únicamente en su versión cifrada. Posteriormente, el receptor utiliza su clave privada local para recuperar la clave AES original y generar dinámicamente el enlace final de descarga. Para usuarios no registrados el usuario deberá encontrar sus propios medios de transmitir el enlace, disminuyendo así la seguridad dependiendo esta del método que se use para transmitir el enlace de un solo uso.

Cuando el receptor accede al enlace, el archivo cifrado es descargado desde el servidor y posteriormente descifrado en el navegador utilizando la clave AES recuperada localmente mediante la clave privada del receptor, recuperando así el contenido original.

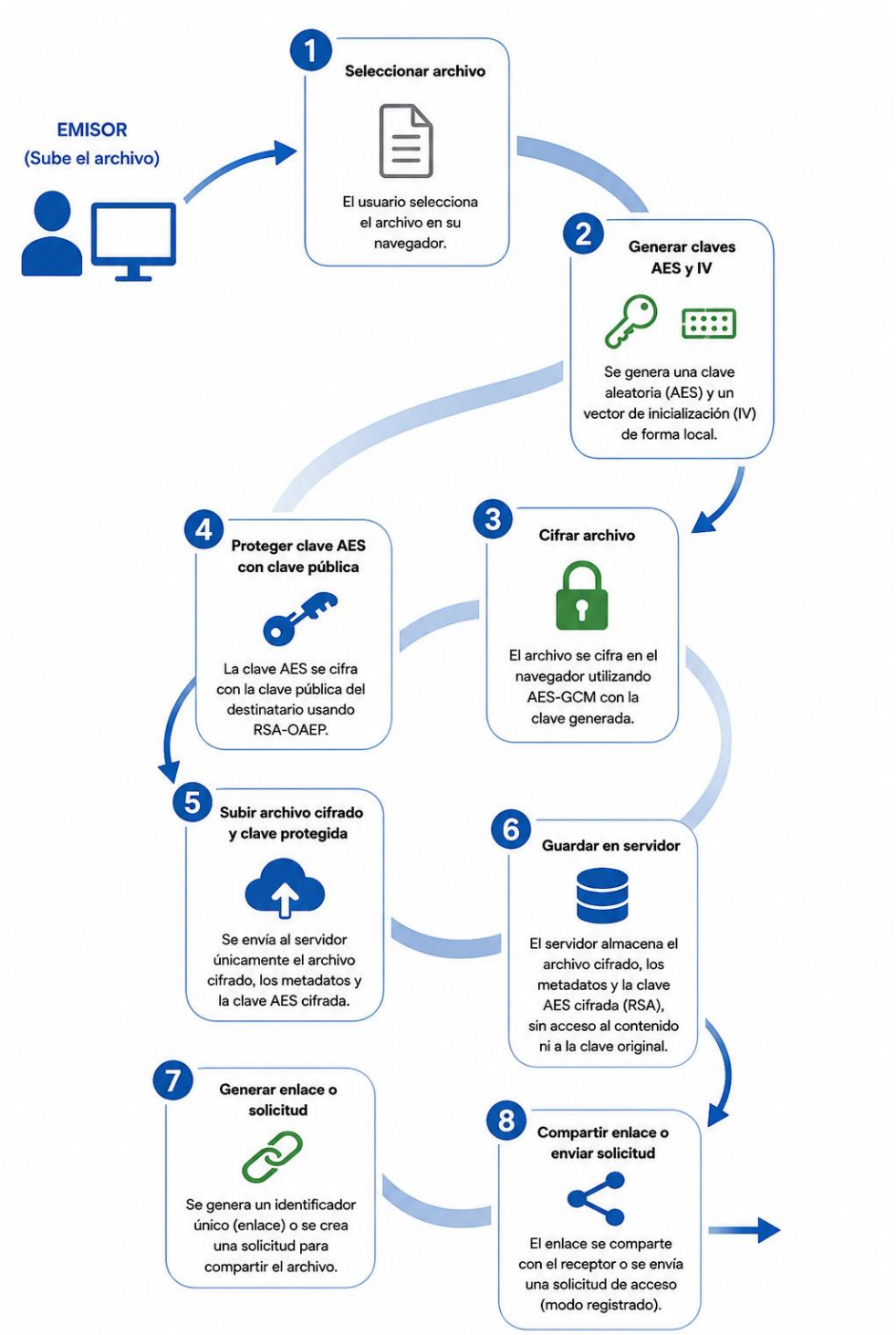

Figura 3. Flujo general del proceso del emisor en SecureShare

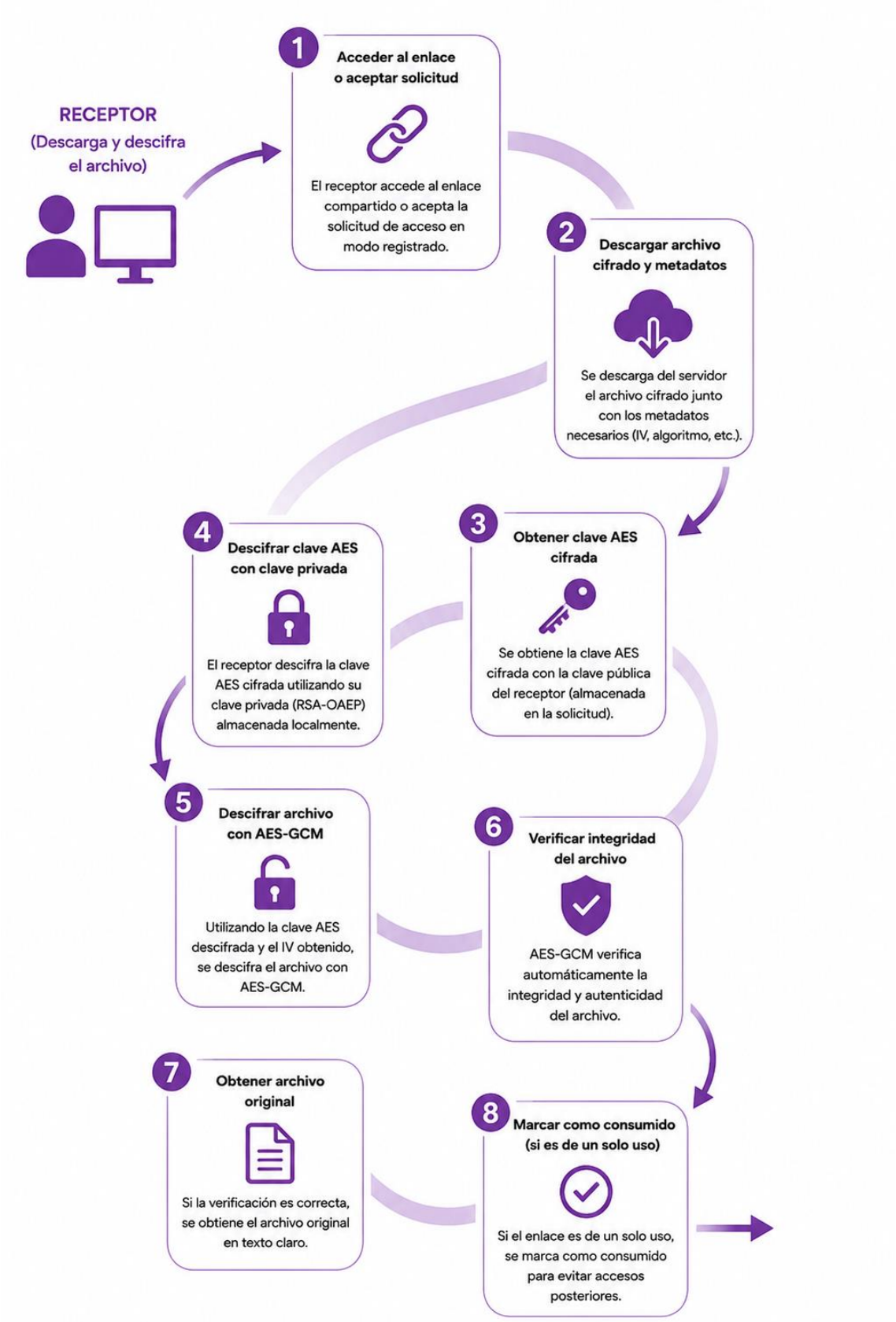

Figura 4. Flujo general del proceso del receptor en SecureShare.

Capítulo 4. Diseño de la solución

4.1 Diseño de la arquitectura

Durante el diseño del sistema se optó por una arquitectura cliente-servidor basada en la separación de responsabilidades entre el frontend, el backend y la base de datos.

La principal decisión de diseño consistió en trasladar las operaciones criptográficas al lado del cliente, permitiendo que el cifrado y descifrado de los archivos se realizaran directamente en el navegador del usuario. De este modo, el servidor únicamente recibe y almacena información cifrada, incorporando un mecanismo de intercambio seguro de claves criptográficas mediante criptografía asimétrica. Cada usuario registrado dispone de un par de claves pública/privada, permitiendo proteger las claves AES utilizadas durante la compartición de archivos entre usuarios.

Este enfoque reduce significativamente la exposición de datos sensibles y minimiza la dependencia del servidor como elemento de confianza para la protección de la información almacenada.

Por otro lado, el backend se diseñó con una estructura sencilla orientada principalmente a la gestión de enlaces, almacenamiento de archivos cifrados y control de acceso. Esto permitió mantener una separación clara entre la lógica de negocio y las operaciones criptográficas.

La arquitectura implementada facilita además la futura ampliación del sistema, permitiendo incorporar nuevas funcionalidades sin modificar el núcleo del mecanismo de cifrado extremo a extremo.

4.2 Diseño del sistema criptográfico

Uno de los aspectos más relevantes del proyecto fue el diseño del mecanismo de cifrado utilizado para proteger los archivos compartidos.

Se optó por implementar un modelo de cifrado extremo a extremo en el que el archivo es cifrado antes de abandonar el navegador del emisor. Para ello, se utilizaron funcionalidades criptográficas proporcionadas por la API Web Crypto disponible en navegadores modernos [1].

El sistema genera dinámicamente una clave simétrica para cada archivo compartido. Posteriormente, dicha clave es utilizada para cifrar el contenido mediante el algoritmo AES-GCM, ampliamente utilizado por ofrecer confidencialidad e integridad sobre los datos cifrados [7].

Una vez completado el cifrado, únicamente el archivo cifrado es enviado al servidor. En el modo invitado, la clave AES utilizada para cifrar el archivo se incorpora en el fragmento de URL (#), permitiendo que permanezca bajo control del usuario, evitando que sea enviada automáticamente al servidor durante las peticiones HTTP.

En el modo autenticado entre usuarios registrados, el sistema implementa un mecanismo adicional de intercambio seguro de claves. Para ello, cada usuario dispone de un par de claves

RSA-OAEP generado localmente en el navegador. La clave pública se almacena en la base de datos, mientras que la clave privada permanece exclusivamente en el navegador del usuario.

Cuando un usuario comparte un archivo con otro usuario registrado, la clave AES del archivo se cifra utilizando la clave pública del receptor. El servidor únicamente almacena dicha clave AES en su versión cifrada, sin disponer de acceso a la clave original ni a las claves privadas necesarias para recuperarla.

Como resultado de este diseño:

- el servidor nunca tiene acceso al archivo original,
- la clave de descifrado permanece bajo control del usuario,
- y el proceso de recuperación del archivo puede realizarse completamente desde el navegador del receptor.

Además, el sistema fue diseñado teniendo en cuenta la posibilidad de manejar archivos de distintos tamaños, utilizando un enfoque compatible con el procesamiento eficiente de datos binarios en JavaScript.

4.3 Diseño del sistema de enlaces y compartición

El sistema de compartición se diseñó utilizando enlaces únicos asociados a cada archivo subido al sistema.

Cada archivo compartido incorpora un token generado aleatoriamente que identifica de forma unívoca el recurso almacenado en el servidor. Este token es utilizado por el servidor para localizar el archivo cifrado y verificar las condiciones de acceso asociadas al mismo.

Con el objetivo de aumentar la seguridad, se implementaron mecanismos de:

- expiración configurable,
- enlaces de un solo uso,
- y control del estado de consumo del recurso.

En el caso de los enlaces de un solo uso, el sistema marca automáticamente el recurso como consumido tras la primera descarga válida, impidiendo accesos posteriores.

Además, se incorporaron dos modos de funcionamiento diferenciados:

- un modo invitado orientado a comparticiones rápidas,
- y un modo autenticado con funcionalidades adicionales de gestión entre usuarios registrados.

En el modo autenticado, el sistema permite enviar solicitudes de compartición internas a otros usuarios registrados mediante su dirección de correo electrónico. El receptor puede aceptar o rechazar dichas solicitudes desde su panel personal antes de acceder al enlace compartido.

Una vez aceptada la solicitud, el navegador del receptor utiliza su clave privada para recuperar la clave AES del archivo y generar localmente el enlace final de descarga y descifrado.

Este enfoque permite mantener un mayor control sobre la distribución de los enlaces y mejora la seguimiento de las comparticiones realizadas dentro de la plataforma.

4.4 Diseño del sistema de usuarios

El sistema de autenticación se diseñó con el objetivo de mantener un mecanismo sencillo, pero suficientemente seguro para un entorno académico y de pruebas.

Los usuarios registrados disponen de una cuenta asociada a un correo electrónico y una contraseña almacenada de forma segura mediante funciones hash. Además, durante el registro se genera localmente un par de claves criptográficas RSA-OAEP asociado a cada usuario. La clave pública se almacena en la base de datos para permitir el intercambio seguro de claves AES, mientras que la clave privada permanece exclusivamente en el navegador del usuario. Este enfoque evita almacenar material criptográfico sensible en texto claro dentro de la base de datos.

Una vez autenticado, el usuario puede acceder a funcionalidades adicionales respecto al modo invitado, como:

- la gestión de solicitudes de compartición,
- el envío de archivos a otros usuarios registrados,
- y el control del estado de las solicitudes recibidas.

La separación entre usuarios invitados y autenticados permitió simplificar el flujo básico de compartición rápida, manteniendo al mismo tiempo funcionalidades más avanzadas para usuarios registrados.

4.5 Decisiones tecnológicas

Las tecnologías seleccionadas durante el desarrollo del proyecto se eligieron teniendo en cuenta criterios de compatibilidad, simplicidad de despliegue y facilidad de integración entre componentes.

Para el frontend se utilizó HTML, CSS y JavaScript, permitiendo desarrollar una interfaz accesible desde cualquier navegador moderno sin necesidad de instalar software adicional.

La implementación de las operaciones criptográficas se realizó mediante la API Web Crypto, ya que proporciona soporte nativo para algoritmos criptográficos seguros directamente desde el navegador y evita depender de librerías externas innecesarias.

La Web Crypto API permitió implementar tanto algoritmos simétricos (AES-GCM) como algoritmos asimétricos (RSA-OAEP), facilitando la construcción del mecanismo de intercambio seguro de claves utilizado entre usuarios registrados.

En el backend se empleó PHP debido a su facilidad de integración con aplicaciones web tradicionales y su compatibilidad con entornos locales basados en XAMPP, utilizados durante el desarrollo del proyecto.

Por último, se utilizó MySQL como sistema gestor de base de datos para almacenar los metadatos asociados a los archivos compartidos, usuarios y solicitudes internas del sistema.

La combinación de estas tecnologías permitió desarrollar un prototipo funcional completamente operativo manteniendo una arquitectura relativamente sencilla y fácilmente reproducible en entornos locales.

Capítulo 5. Implementación

En este capítulo se describe el proceso de implementación de la aplicación SecureShare. Se detallan el desarrollo del frontend y del backend, los mecanismos de cifrado híbrido, el sistema de intercambio seguro de claves criptográficas, la gestión de usuarios y las funcionalidades de compartición implementadas. Además, se muestran diferentes capturas y fragmentos de código representativos de las principales funcionalidades desarrolladas.

5.1 Entorno del desarrollo

El desarrollo de SecureShare se realizó utilizando un entorno local basado en tecnologías web ampliamente utilizadas y compatibles con la mayoría de sistemas actuales. Para ello, se empleó XAMPP como entorno de ejecución local [5], permitiendo integrar servidor web Apache, PHP y base de datos MySQL en una misma plataforma de desarrollo.

La implementación del proyecto se llevó a cabo utilizando Visual Studio Code como editor principal de código fuente [6], facilitando la organización del proyecto, la edición de archivos y la depuración de errores durante el desarrollo. Además, se utilizaron diferentes extensiones orientadas al desarrollo web y al análisis de código PHP.

En cuanto a las tecnologías utilizadas, el frontend de la aplicación se desarrolló mediante HTML, CSS y JavaScript, mientras que el backend se implementó utilizando PHP [2]. Para la persistencia de datos se utilizó MySQL como sistema gestor de bases de datos [4]. Además, se utilizaron mecanismos de almacenamiento local del navegador (localStorage) para mantener las claves privadas RSA de los usuarios registrados.

El sistema de cifrado extremo a extremo se implementó utilizando la Web Crypto API proporcionada por los navegadores modernos. Esta tecnología permite realizar operaciones criptográficas directamente desde el navegador, incluyendo el cifrado de archivos y el intercambio seguro de claves.

Tecnología	**Función**	**Motivo de selección**
PHP 8	Backend	Integración sencilla con aplicaciones web y MySQL
MySQL	Base de datos	Fiabilidad y facilidad de integración
JavaScript	Lógica cliente	Necesario para implementar el cifrado local
Web Crypto API	Criptografía	Permite operaciones criptográficas nativas en el navegador
AES-GCM	Cifrado de archivos	Alto rendimiento y autenticación integrada
RSA-OAEP	Intercambio de claves	Permite compartir claves AES de forma segura
XAMPP	Entorno de desarrollo	Facilita el despliegue local durante el desarrollo
Visual Studio Code	Desarrollo	Entorno ligero y ampliamente utilizado

Tabla 1. Tecnologias utilizadas para el desarrollo de Secureshare

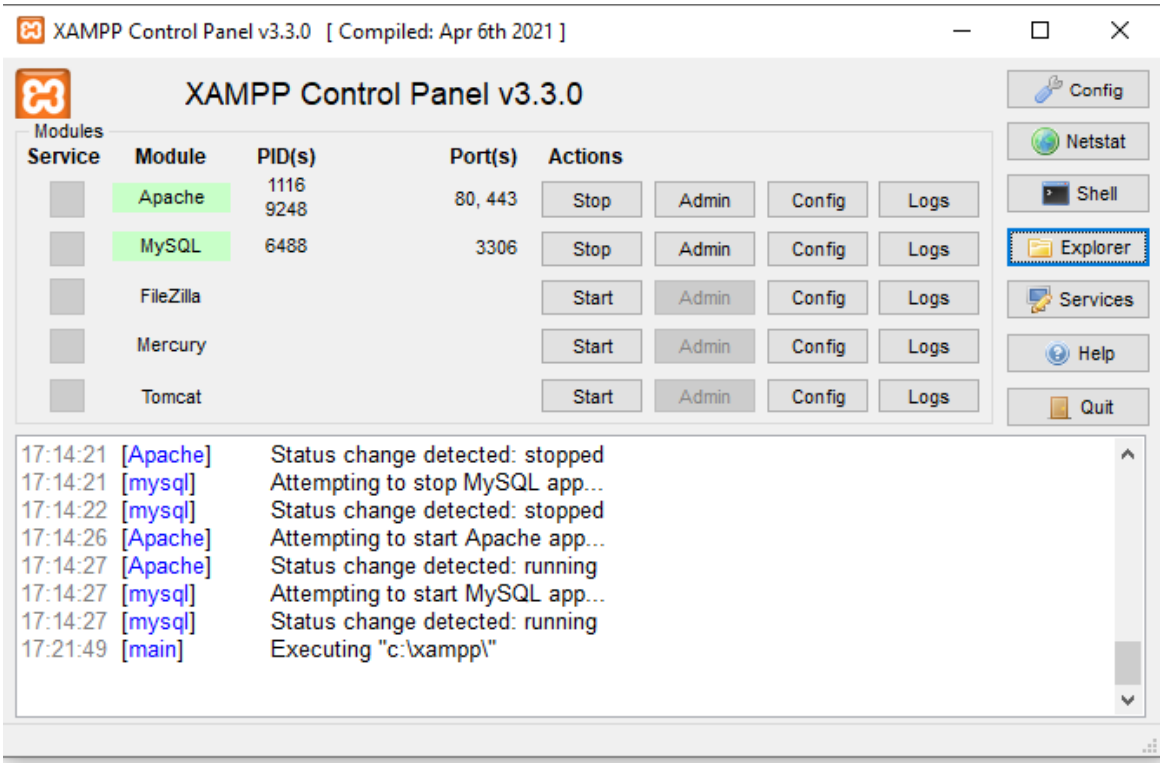

Figura 5. Entorno local de ejecución utilizado durante el desarrollo del proyecto mediante XAMPP.

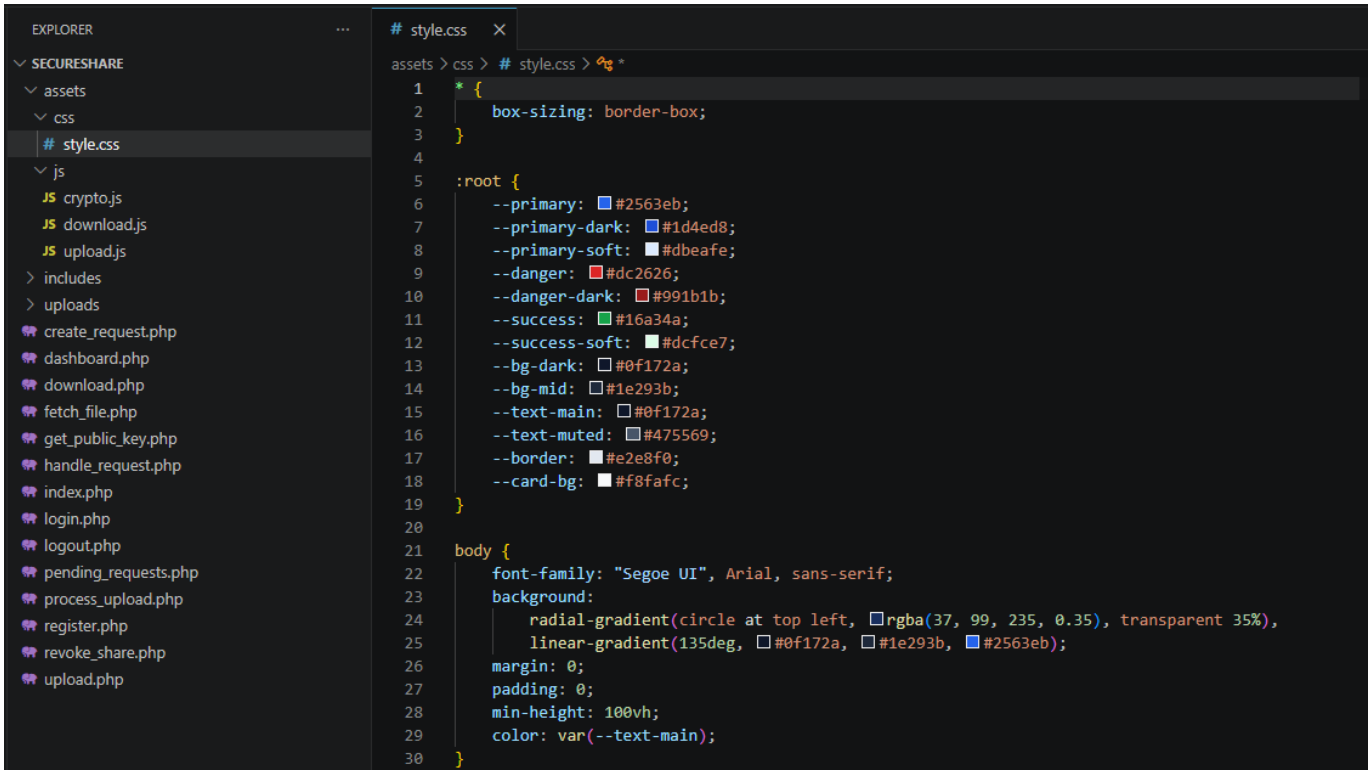

Figura 6. Entorno del proyecto en Visual Studio Code.

5.2 Estructura del proyecto

La aplicación se organizó siguiendo una estructura modular orientada a separar correctamente las distintas responsabilidades del sistema. De este modo, se diferenciaron claramente los componentes relacionados con la interfaz de usuario, la lógica de negocio, el almacenamiento y las operaciones criptográficas.

El proyecto principal se ubicó dentro del directorio `htdocs` de XAMPP, permitiendo su ejecución mediante Apache en un entorno local. Dentro del proyecto se organizaron diferentes carpetas y archivos según su funcionalidad.

Por un lado, la carpeta `assets` contiene los recursos relacionados con la interfaz gráfica y la lógica del cliente, incluyendo hojas de estilo CSS y scripts JavaScript encargados del cifrado, descifrado y subida de archivos. Dentro de los scripts JavaScript también se implementaron las operaciones criptográficas necesarias para la compartición segura de archivos entre usuarios registrados.

La carpeta `includes` agrupa los archivos reutilizables del backend, como la conexión a la base de datos, la gestión de sesiones y las funciones auxiliares utilizadas en diferentes partes del sistema.

Además, el sistema dispone de diferentes páginas PHP responsables de funcionalidades concretas, como el inicio de sesión, el registro de usuarios, la subida de archivos, la descarga segura o la gestión de solicitudes internas entre usuarios registrados.

Por último, los archivos cifrados subidos por los usuarios se almacenan en el servidor dentro de un directorio específico, manteniendo siempre el contenido en formato cifrado.

La estructura del proyecto se organizó separando los recursos del cliente, los archivos auxiliares del servidor y las páginas principales de la aplicación, como se observa en la Figura 7.

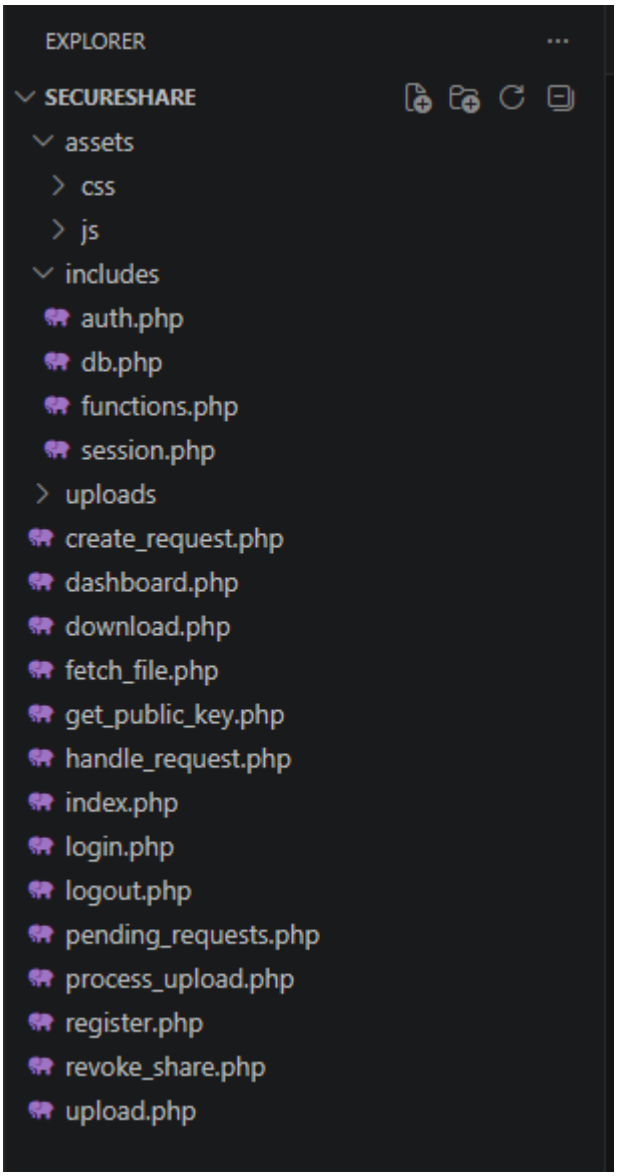

Figura 7. Estructura de carpetas y archivos principales del proyecto SecureShare.

5.3 Implementación del frontend

El frontend de la aplicación fue diseñado con el objetivo de proporcionar una interfaz sencilla, intuitiva y accesible para el usuario, manteniendo al mismo tiempo una apariencia moderna orientada a aplicaciones web actuales.

La Figura 8 muestra la pantalla principal de la aplicación, desde la cual el usuario puede elegir entre utilizar SecureShare sin cuenta o acceder mediante una cuenta registrada.

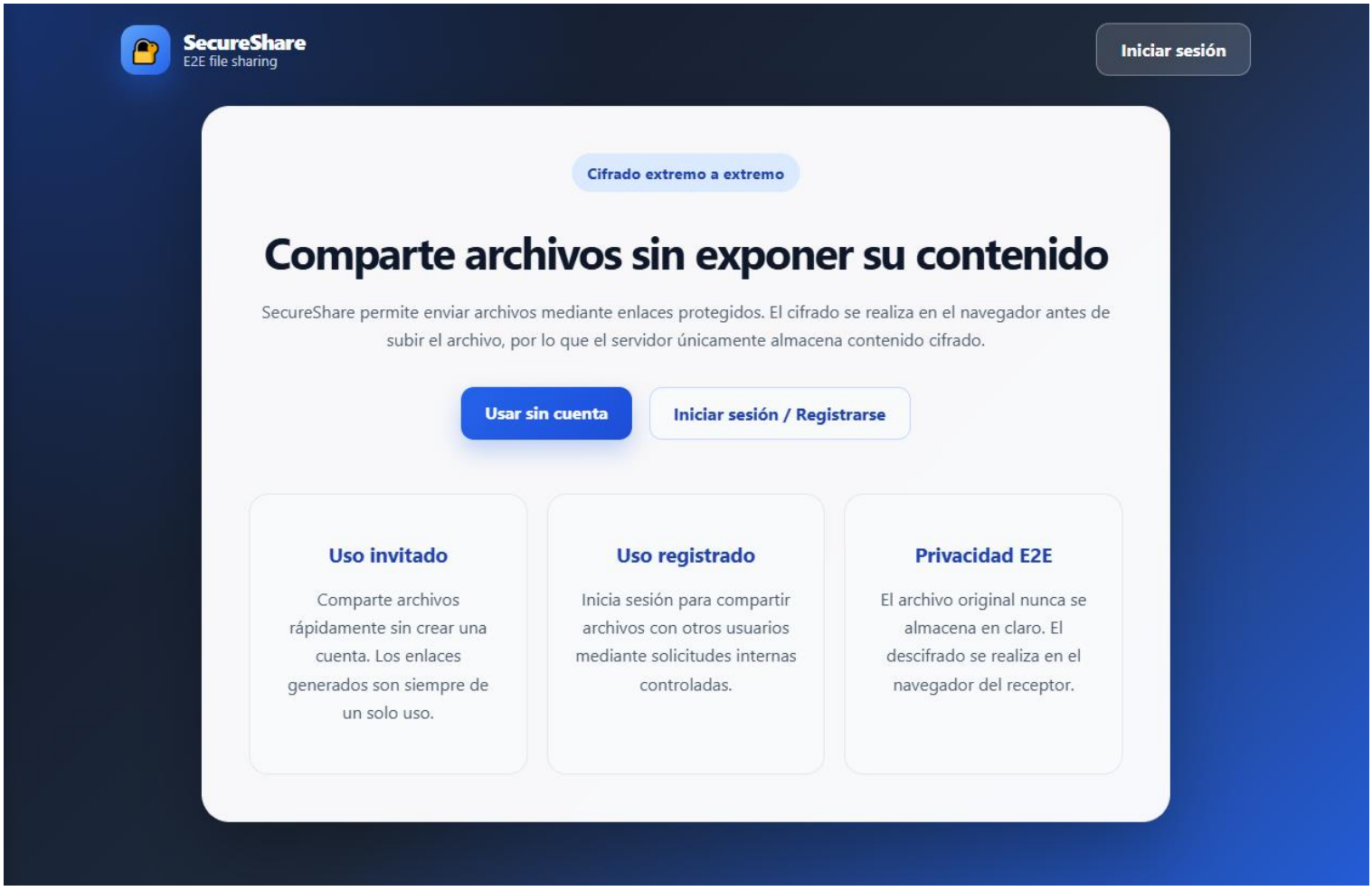

Figura 8. Pantalla principal de SecureShare con acceso al modo invitado y al modo registrado.

La interfaz se desarrolló utilizando HTML y CSS, complementándose con JavaScript para implementar la lógica de interacción y las operaciones criptográficas ejecutadas en el navegador.

Entre las principales vistas implementadas se encuentran la página principal de bienvenida, las pantallas de inicio de sesión y registro, el panel de usuario autenticado, el sistema de subida de archivos y la página de descarga segura.

Figura 9. Interfaz de subida de archivos con sistema drag and drop y configuración del enlace.

Durante el desarrollo se realizaron mejoras progresivas orientadas a optimizar la experiencia de usuario, incorporando elementos visuales como tarjetas informativas, barras de progreso, zonas de arrastre de archivos (drag and drop), indicadores visuales de seguridad y componentes adaptados a diferentes tamaños de pantalla.

La aplicación también incorpora retroalimentación visual durante operaciones críticas, como el cifrado o la subida de archivos, mostrando al usuario el estado del proceso mediante mensajes y barras de progreso dinámicas.

Figura 10. Barra de progreso mostrada durante el proceso de cifrado y subida del archivo.

Una vez finalizado el proceso de cifrado y subida del archivo, la aplicación genera automáticamente un enlace seguro asociado al recurso compartido. En el modo invitado, dicho

enlace incorpora la clave AES necesaria para el descifrado en el fragmento de la URL, permitiendo que únicamente el receptor que posea el enlace completo pueda recuperar el contenido original. En el caso de usuarios registrados, el enlace completo se genera dinámicamente en el navegador del receptor tras recuperar localmente la clave AES protegida mediante RSA-OAEP.

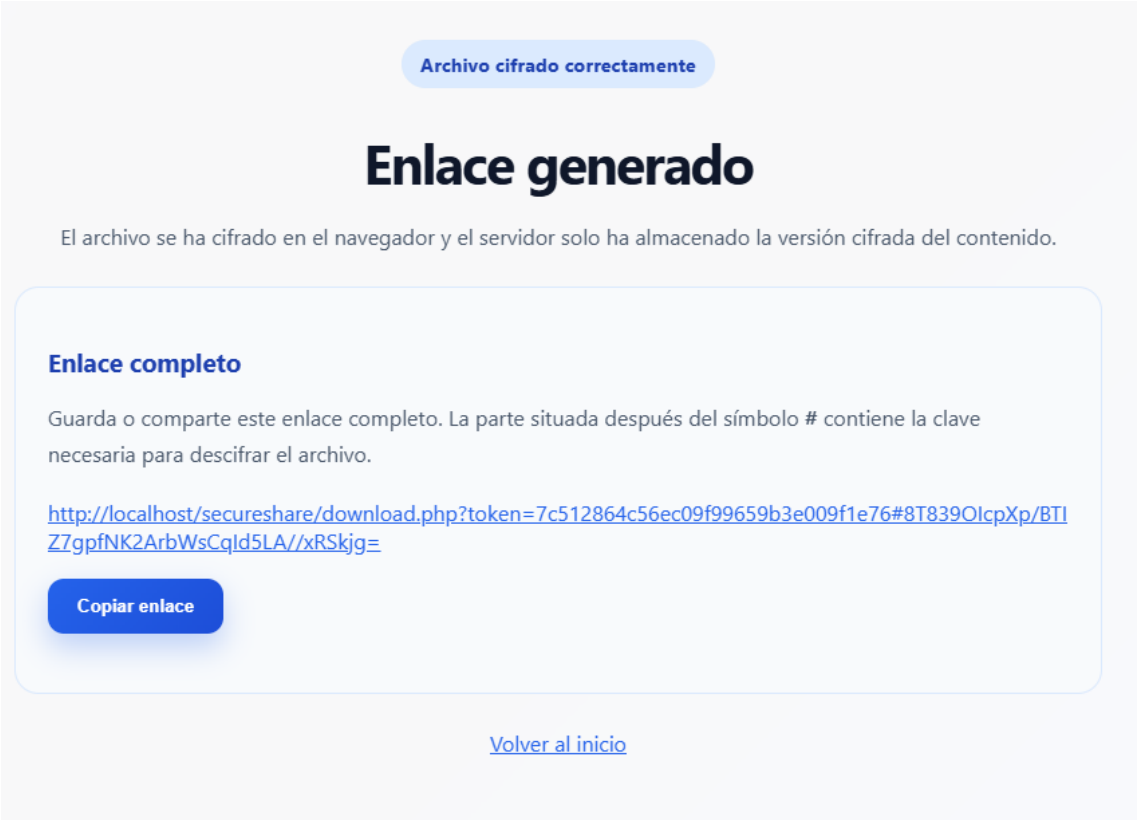

Figura 11. Resultado final del proceso de compartición (en modo invitado) tras generar correctamente el enlace seguro del archivo cifrado.

5.4 Implementación del backend

El backend de SecureShare se implementó utilizando PHP, encargándose de gestionar la lógica de negocio de la aplicación y la comunicación con la base de datos.

Entre sus principales responsabilidades se encuentran la gestión de usuarios y sesiones, el almacenamiento de archivos cifrados, la generación y validación de enlaces de descarga, el control de expiración y revocación de enlaces, así como la administración de solicitudes internas entre usuarios registrados. Además, el backend participa en la gestión de la información necesaria para la compartición segura entre usuarios registrados.

La autenticación de usuarios se desarrolló mediante sesiones PHP, permitiendo mantener el estado de autenticación entre diferentes páginas de la aplicación. Las contraseñas se almacenan utilizando funciones de hash seguras proporcionadas por PHP, evitando almacenar información sensible en texto claro.

Para las operaciones con la base de datos se utilizaron consultas preparadas (prepared statements), reduciendo el riesgo de ataques de inyección SQL y mejorando la seguridad general del sistema.

La aplicación almacena la información relacionada con enlaces, archivos y estados de compartición en una base de datos MySQL estructurada mediante diferentes tablas relacionadas entre sí. Además de la información necesaria para gestionar usuarios, archivos y solicitudes de compartición.

☐	1	**id**	int(11)	
☐	2	**token**	varchar(100)	utf8mb4_general_ci
☐	3	**file_path**	varchar(255)	utf8mb4_general_ci
☐	4	**original_name**	varchar(255)	utf8mb4_general_ci
☐	5	**mime_type**	varchar(100)	utf8mb4_general_ci
☐	6	**file_size**	bigint(20)	
☐	7	**created_at**	datetime	
☐	8	**expires_at**	datetime	
☐	9	**is_one_time**	tinyint(1)	
☐	10	**is_consumed**	tinyint(1)	
☐	11	**consumed_at**	datetime	
☐	12	**is_revoked**	tinyint(1)	
☐	13	**password_protected**	tinyint(1)	
☐	14	**iv**	text	utf8mb4_general_ci
☐	15	**algorithm**	varchar(50)	utf8mb4_general_ci
☐	16	**user_id**	int(11)	
☐	17	**share_key**	text	utf8mb4_general_ci

Figura 12. Tabla principal de compartición de archivos almacenada en MySQL mediante phpMyAdmin.

El backend también se encarga de verificar diferentes restricciones asociadas a los enlaces compartidos, incluyendo:

- comprobación de expiración,
- enlaces de un solo uso,
- revocación manual,
- y validación de acceso.

Antes de permitir la descarga de un archivo, el backend verifica diferentes restricciones de seguridad relacionadas con la expiración, revocación y consumo del enlace.

```
if ($share["is_revoked"])
if ($share["is_consumed"])
if (strtotime($share["expires_at"]) < time())
```

Figura 13. Validaciones de seguridad implementadas en el backend para el control de acceso a enlaces.

Además, el servidor nunca realiza operaciones de descifrado sobre los archivos almacenados, limitándose únicamente a guardar y servir contenido previamente cifrado por el cliente.

5.5 Implementación del cifrado y descifrado

Uno de los componentes principales del proyecto es el sistema de cifrado implementado directamente en el navegador mediante JavaScript y las funciones proporcionadas por la Web Crypto API, incluyendo SbutleCrypto.encrypt [8]. Cuando el usuario selecciona un archivo para compartir, el sistema genera localmente una clave criptográfica AES-GCM de 256 bits. Posteriormente, el archivo es leído en el navegador y cifrado antes de ser enviado al servidor.

El algoritmo AES-GCM fue seleccionado debido a que proporciona confidencialidad e integridad de los datos mediante un mecanismo de autenticación incorporado. Además, la generación aleatoria de vectores de inicialización evita reutilización de parámetros criptográficos entre diferentes archivos compartidos.

El servidor únicamente recibe la versión cifrada del contenido junto con los metadatos necesarios para gestionar el enlace, sin disponer nunca de acceso al archivo original en texto claro.

Para el proceso de descifrado, el receptor descarga el archivo cifrado desde el servidor y utiliza la clave incluida en el fragmento del enlace para recuperar el contenido original directamente en el navegador.

En el modo invitado, la clave AES utilizada durante el cifrado permanece exclusivamente en el fragmento de la URL tras el símbolo #, evitando que sea enviada automáticamente al servidor durante las peticiones HTTP.

Por otro lado, en el modo de usuarios registrados se implementó un mecanismo de intercambio seguro de claves basado en criptografía asimétrica RSA-OAEP. Durante el registro, el navegador genera localmente un par de claves pública/privada asociado al usuario.

La clave pública se almacena en la base de datos para permitir el intercambio seguro de claves entre usuarios, mientras que la clave privada permanece exclusivamente almacenada en el navegador mediante localStorage.

Cuando un usuario comparte un archivo con otro usuario registrado, la clave AES utilizada para cifrar el archivo se cifra utilizando la clave pública del receptor. El servidor únicamente almacena dicha clave AES en su versión cifrada dentro de la tabla share_requests mediante el campo encrypted_share_key, sin disponer nunca de acceso a la clave original ni a las claves privadas necesarias para recuperarla.

Posteriormente, el receptor utiliza su clave privada local para descifrar la clave AES y generar dinámicamente el enlace final de descarga y descifrado desde su navegador.

De este modo, se mantiene el modelo de cifrado extremo a extremo definido durante la fase de diseño para ambos modos de funcionamiento de la aplicación. Adicionalmente, el sistema utiliza vectores de inicialización (IV) generados aleatoriamente para cada operación criptográfica, incrementando la seguridad del proceso de cifrado.

```
// Generar clave AES
async function generateKey() {
    return await window.crypto.subtle.generateKey(
        {
            name: "AES-GCM",
            length: 256
        },
        true,
        ["encrypt", "decrypt"]
    );
}
```

Figura 14. Código utilizado en crypto.js para la generación de claves AES-GCM mediante Web Crypto API.

```
// Cifrar archivo
async function encryptFile(file, key) {
```

Figura 15. Implementación del cifrado local de archivos mediante AES-GCM.

5.6 Gestión de usuarios y sesiones

El sistema incorpora un mecanismo de autenticación que permite diferenciar entre usuarios invitados y usuarios registrados.

Los usuarios registrados pueden iniciar sesión mediante correo electrónico y contraseña, junto con los mecanismos criptográficos asociados a su cuenta, accediendo a funcionalidades adicionales como el historial de archivos, la gestión de solicitudes y la compartición interna entre usuarios.

La gestión de sesiones se implementó utilizando sesiones PHP, almacenando la información básica del usuario autenticado y restringiendo el acceso a determinadas páginas mediante comprobaciones de autenticación y garantizando el acceso a las funcionalidades disponibles para usuarios autenticados entre usuarios registrados.

Del mismo modo, el sistema implementa mecanismos básicos de validación de formularios y control de acceso para evitar operaciones no autorizadas.

Figura 16. Pantalla de inicio de sesión para usuarios registrados.

Tras autenticarse correctamente, el usuario accede a un panel privado desde el que puede gestionar enlaces, solicitudes y archivos compartidos.

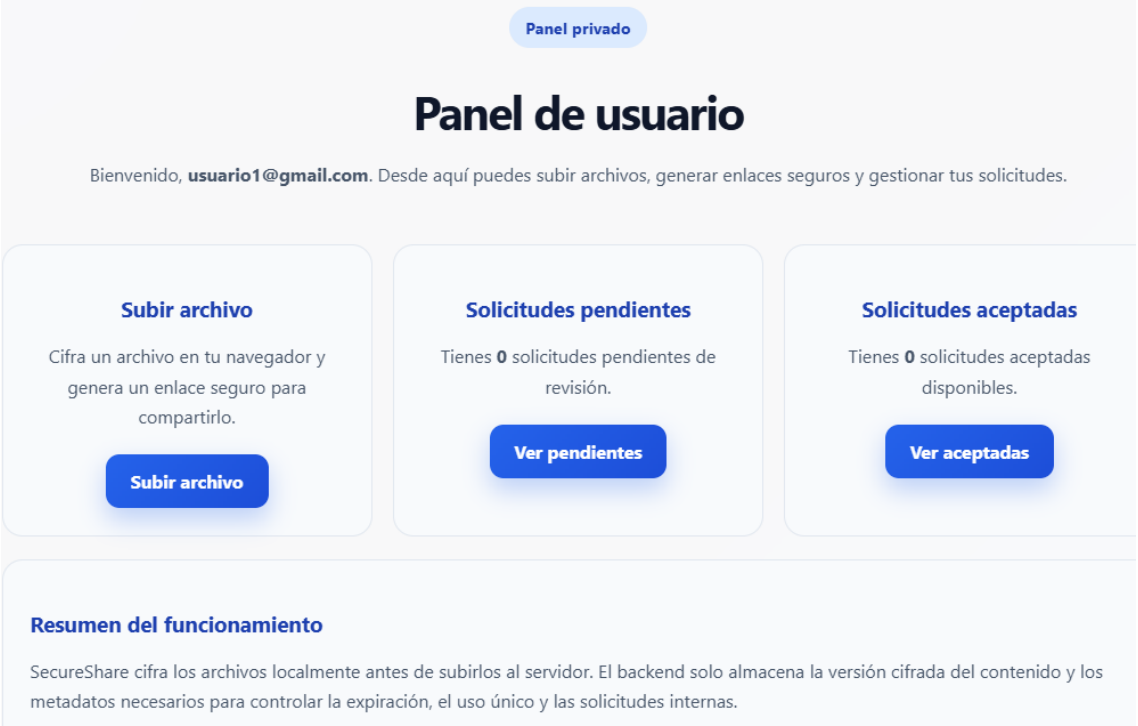

Figura 17. Panel privado de usuario autenticado en SecureShare.

5.7 Gestión de enlaces, solicitudes e historial

La aplicación permite generar enlaces únicos asociados a cada archivo compartido, incorporando diferentes mecanismos de control orientados a mejorar la seguridad.

Entre las funcionalidades implementadas destacan:

- enlaces de un solo uso,
- expiración configurable,
- revocación manual,
- historial de archivos,
- y solicitudes internas entre usuarios registrados.

El sistema mantiene un registro de los archivos generados por cada usuario, mostrando información como el estado del enlace, la fecha de expiración o el tipo de acceso configurado.

Por otro lado, los usuarios registrados pueden enviar solicitudes internas de compartición a otros usuarios del sistema, permitiendo aceptar o rechazar el acceso desde el panel privado. Cuando una solicitud es aceptada, el navegador del receptor recupera la clave AES cifrada asociada al archivo compartido y la descifra localmente utilizando su clave privada RSA. Posteriormente, el sistema genera dinámicamente el enlace final de descarga y descifrado del archivo.

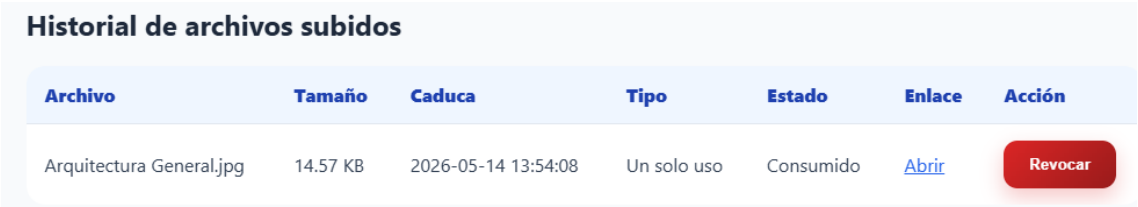

Figura 18. Historial de archivos con opción a revocar enlaces, reabrir enlaces no consumidos y comprobar estado.

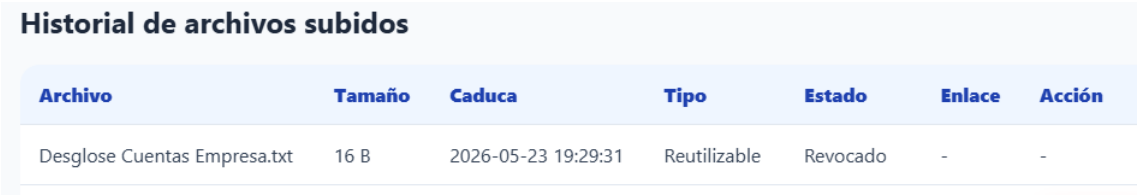

Figura 19. Como aparece al revocar un enlace en SecureShare.

Figura 20. Gestión de solicitudes pendientes de compartición entre usuarios registrados.

Figura 21. Interfaz al aceptar una solicitud en SecureShare.

5.8 Mejoras visuales e interacción con el usuario

Durante las últimas fases del desarrollo se realizaron diferentes mejoras visuales orientadas a ofrecer una experiencia de usuario más moderna y profesional.

Entre ellas destacan:

- diseño responsive,
- uso de tarjetas visuales,
- barras de progreso,
- indicadores visuales de seguridad,
- animaciones ligeras,
- y zonas drag and drop para facilitar la subida de archivos.

También se mejoró la organización visual del panel de usuario y del historial de archivos, incorporando tablas y componentes reutilizables que facilitan la navegación y comprensión de la información mostrada.

Estas mejoras permitieron aproximar la apariencia de la aplicación a soluciones web reales orientadas al usuario final.

Capítulo 6. Pruebas

6.1 Pruebas Funcionales

Con el objetivo de verificar el correcto funcionamiento de la aplicación SecureShare, se realizaron diferentes pruebas funcionales sobre las principales características implementadas durante el desarrollo. Estas pruebas permitieron validar tanto las funcionalidades orientadas al usuario como los mecanismos de seguridad incorporados en el sistema.

Las pruebas se realizaron en un entorno local utilizando XAMPP como servidor web y MySQL como sistema gestor de base de datos. También se emplearon diferentes navegadores y tamaños de pantalla para comprobar el comportamiento general de la aplicación y la correcta adaptación de la interfaz.

A continuación, se muestran algunas de las pruebas funcionales realizadas durante el desarrollo del proyecto.

ID	Funcionalidad evaluada	Descripción de la prueba	Resultado esperado	Resultado obtenido
PF-01	Acceso a la página principal	Acceder a la aplicación desde el navegador	Visualización correcta de la página principal	Correcto
PF-02	Registro de usuario	Crear una cuenta mediante correo y contraseña	Cuenta registrada correctamente	Correcto
PF-03	Validación de correo	Introducir un correo con formato incorrecto	Mostrar mensaje de error	Correcto
PF-04	Confirmación de contraseña	Introducir contraseñas diferentes en el registro	Bloquear el registro y mostrar error	Correcto
PF-05	Inicio de sesión	Acceder con credenciales válidas	Acceso al dashboard	Correcto
PF-06	Credenciales incorrectas	Intentar iniciar sesión con contraseña errónea	Mostrar mensaje de error	Correcto
PF-07	Gestión de sesión	Acceder al dashboard sin autenticación	Redirección al login	Correcto
PF-08	Cierre de sesión	Finalizar sesión del usuario	Eliminación de sesión y retorno al inicio	Correcto
PF-09	Subida de archivo	Seleccionar un archivo y generar enlace	Archivo cifrado y almacenado correctamente	Correcto
PF-10	Drag and drop	Arrastrar archivo sobre la zona de subida	Detección correcta del archivo	Correcto
PF-11	Barra de progreso	Subir un archivo grande	Actualización dinámica del progreso	Correcto

ID	Funcionalidad evaluada	Descripción de la prueba	Resultado esperado	Resultado obtenido
PF-12	Generación de clave AES	Iniciar proceso de cifrado	Generación local de clave AES-GCM	Correcto
PF-13	Cifrado local	Cifrar archivo antes de enviarlo	Archivo cifrado en navegador	Correcto
PF-14	Generación de enlace	Finalizar subida correctamente	Generación de enlace seguro	Correcto
PF-15	Inclusión de clave en URL	Generar enlace completo	Inclusión de clave tras el símbolo #	Correcto
PF-16	Generación de claves RSA	Registro de usuario registrado	Generación local de claves pública/privada	Correcto
PF-17	Intercambio seguro de claves	Compartir archivo entre usuarios registrados	Clave AES cifrada mediante RSA-OAEP	Correcto
PF-18	Descarga de archivo	Abrir enlace válido	Descarga y descifrado correcto	Correcto
PF-19	Descifrado local	Recuperar archivo desde navegador	Archivo original recuperado	Correcto
PF-20	Enlace caducado	Intentar acceder tras expiración	Bloqueo del acceso	Correcto
PF-21	Enlace consumido	Reutilizar enlace de un solo uso	Acceso denegado	Correcto
PF-22	Revocación manual	Revocar enlace desde dashboard	Enlace deshabilitado	Correcto
PF-23	Historial de archivos	Consultar archivos generados	Visualización correcta del historial	Correcto
PF-24	Solicitud interna	Compartir archivo con otro usuario registrado	Solicitud creada correctamente	Correcto
PF-25	Solicitud pendiente	Consultar solicitudes recibidas	Visualización correcta	Correcto
PF-26	Aceptación de solicitud	Aceptar solicitud recibida	Acceso habilitado al enlace	Correcto
PF-27	Rechazo de solicitud	Rechazar solicitud recibida	Bloqueo del acceso compartido	Correcto
PF-28	Responsive design	Acceder desde resolución móvil	Adaptación correcta de la interfaz	Correcto
PF-29	Navegación general	Navegar entre diferentes vistas	Funcionamiento correcto de enlaces y navegación	Correcto
PF-30	Visualización de estados	Consultar estados de enlaces	Visualización correcta de activo, consumido o revocado	Correcto
PF-31	Protección de archivos	Verificar almacenamiento cifrado en servidor	El servidor solo almacena archivos cifrados	Correcto
PF-32	Validación backend	Acceder con token inexistente	Mostrar error de acceso	Correcto

Tabla 2. Pruebas funcionales realizadas en SecureShare con descripción y resultados.

Los resultados obtenidos durante las pruebas funcionales permitieron verificar el correcto comportamiento de las principales funcionalidades implementadas en la aplicación. Además, se verificó el correcto funcionamiento de los mecanismos de seguridad y control incorporados durante el desarrollo, así como del sistema de compartición implementado

6.2 Pruebas de Seguridad

Además de las pruebas funcionales, se realizaron diferentes comprobaciones orientadas a verificar el correcto funcionamiento de los mecanismos de seguridad incorporados en SecureShare. Estas pruebas no constituyen una auditoría de seguridad completa, pero permiten validar que las principales medidas implementadas funcionan correctamente dentro del alcance del proyecto.

Las pruebas se centraron especialmente en la protección de credenciales, el control de acceso a zonas privadas, la validación de enlaces compartidos y la verificación del modelo de seguridad definido durante el diseño.

Protección de credenciales de usuario

El sistema de autenticación de SecureShare evita almacenar contraseñas en texto claro dentro de la base de datos. Durante el registro de usuarios, las contraseñas se procesan mediante funciones de hash proporcionadas por PHP, mientras que durante el inicio de sesión se verifica la contraseña introducida mediante la función password_verify [13].

De esta forma, aunque un atacante consiguiera acceso a la base de datos, no obtendría directamente las contraseñas originales de los usuarios.

La Figura 22 muestra el proceso de verificación de credenciales durante el inicio de sesión, donde se comprueba la contraseña introducida frente al hash almacenado en la base de datos.

```php
if (password_verify($password, $user["password_hash"])) {
    $_SESSION["user_id"] = $user["id"];
    $_SESSION["user_email"] = $user["email"];
```

Figura 22. Verificación de credenciales mediante password_verify durante el inicio de sesión.

Protección frente a inyección SQL

Para reducir el riesgo de inyección SQL, las operaciones con la base de datos se implementaron utilizando consultas preparadas [9]. En lugar de concatenar directamente los datos introducidos por el usuario dentro de las consultas SQL, se emplean sentencias preparadas y parámetros vinculados mediante bind_param.

Este mecanismo permite separar la estructura de la consulta de los valores introducidos por el usuario, reduciendo el riesgo de manipulación maliciosa de las consultas.

La Figura 23 muestra un ejemplo de consulta preparada utilizada durante el inicio de sesión, donde el correo electrónico introducido por el usuario se vincula como parámetro y no se concatena directamente en la consulta SQL.

```php
$stmt = $conn->prepare("SELECT id, email, password_hash FROM users WHERE email = ?");
$stmt->bind_param("s", $email);
$stmt->execute();
```

Figura 23. Uso de consultas preparadas para la autenticación de usuarios.

Control de acceso mediante sesiones

Las páginas privadas de la aplicación están protegidas mediante comprobaciones de sesión. Para ello, se implementó una función auxiliar require_login, que verifica si existe un usuario autenticado antes de permitir el acceso a determinadas páginas, como el panel privado o la gestión de solicitudes.

En caso de que un usuario no autenticado intente acceder directamente a una página protegida, el sistema lo redirige a la página de inicio de sesión.

La Figura 24 muestra la función utilizada para proteger las rutas privadas de la aplicación mediante la comprobación de sesión activa.

```php
<?php
require_once "session.php";

function require_login() {
    if (!isset($_SESSION["user_id"])) {
        header("Location: login.php");
        exit;
    }
}
```

Figura 24. Función require_login utilizada para restringir el acceso a páginas privadas.

Validación de enlaces compartidos

El backend realiza diferentes comprobaciones antes de permitir el acceso a un archivo compartido. Entre ellas se encuentran la validación del token, la comprobación de expiración, la verificación de enlaces consumidos y el control de enlaces revocados.

Estas validaciones permiten impedir el acceso a recursos que ya no deberían estar disponibles, ya sea porque han caducado, porque se han consumido en modo de un solo uso o porque el usuario propietario los ha revocado manualmente desde su panel.

La Figura 25 muestra las validaciones realizadas antes de permitir la descarga de un archivo compartido, comprobando si el enlace ha sido revocado, consumido o ha caducado.

```php
if ($share["is_revoked"]) {
    die("Este enlace ha sido revocado.");
}

if ($share["is_consumed"]) {
    die("Este enlace ya ha sido consumido.");
}

if (strtotime($share["expires_at"]) < time()) {
    die("Este enlace ha caducado.");
}
```

Figura 25. Validaciones de seguridad aplicadas sobre enlaces compartidos.

Validación del modelo de cifrado y protección de archivos

Una de las comprobaciones más relevantes fue verificar que el servidor no recibe ni almacena el archivo original en texto claro. Durante la subida, el archivo es cifrado en el navegador mediante la Web Crypto API y solo posteriormente se envía al servidor.

Estas pruebas permitieron comprobar que el servidor almacena únicamente archivos previamente cifrados y los metadatos necesarios para gestionar los enlaces compartidos.

Estas comprobaciones permitieron verificar el correcto funcionamiento del modelo de cifrado definido durante la fase de diseño, garantizando que los archivos permanecen protegidos durante su almacenamiento y compartición.

La Figura 26 muestra la generación del enlace completo, donde la clave de descifrado se incorpora tras el símbolo #, permaneciendo fuera de las peticiones enviadas al servidor.

Figura 26. Enlace generado con clave de descifrado incluida en el fragmento de URL.

Resultado de las pruebas de seguridad

Los resultados obtenidos permitieron verificar que los principales mecanismos de seguridad implementados funcionan de forma adecuada dentro del alcance del proyecto. En concreto, se comprobó la protección de credenciales mediante hash, el uso de consultas preparadas, la restricción de acceso mediante sesiones, la validación de enlaces y el correcto funcionamiento de los mecanismos de protección implementados [11].

No obstante, al tratarse de un prototipo académico, existen mejoras futuras que podrían reforzar la seguridad del sistema, como la incorporación de tokens CSRF, limitación de intentos de inicio de sesión, autenticación multifactor o mecanismos avanzados de monitorización.

6.3 Pruebas de Usabilidad

Además de las pruebas funcionales y de seguridad, se realizaron diferentes comprobaciones relacionadas con la experiencia de usuario y la facilidad de utilización de la aplicación. Estas pruebas permitieron evaluar aspectos visuales y de interacción orientados a mejorar la usabilidad general del sistema.

Durante el desarrollo se realizaron diferentes mejoras progresivas sobre la interfaz con el objetivo de ofrecer una navegación más clara, moderna e intuitiva. Entre ellas destacan la reorganización visual de los componentes, la utilización de tarjetas informativas, la incorporación de indicadores visuales de seguridad y la implementación de elementos dinámicos de interacción.

Uno de los aspectos trabajados fue la simplificación del proceso de subida de archivos. Para ello, se implementó una zona de arrastre de archivos (*drag and drop*), permitiendo seleccionar archivos tanto mediante exploración tradicional como arrastrándolos directamente sobre la interfaz.

La Figura 27 muestra la interfaz de subida de archivos mediante una zona drag and drop implementada para simplificar la interacción del usuario.

Figura 27. Sistema de subida de archivos mediante drag and drop.

Adicionalmente, se incorporó retroalimentación visual durante las operaciones de cifrado y subida de archivos. De esta forma, el usuario puede conocer el estado del proceso en tiempo real mediante mensajes informativos y barras de progreso dinámicas.

Estas mejoras permiten reducir la incertidumbre durante operaciones que pueden requerir varios segundos, especialmente en archivos de gran tamaño.

La Figura 28 muestra la visualización del progreso durante el proceso de cifrado y subida de archivos.

Figura 28. Barra de progreso mostrada durante el cifrado y subida de archivos.

También se reorganizó el panel principal de usuario utilizando componentes visuales reutilizables y una distribución basada en tarjetas, permitiendo acceder de forma sencilla a las principales funcionalidades de la aplicación.

Esta estructura facilita la navegación entre las diferentes secciones del sistema y mejora la comprensión general de la información mostrada al usuario.

La Figura 29 muestra la organización visual del panel principal de usuario tras las mejoras de interfaz implementadas durante el desarrollo.

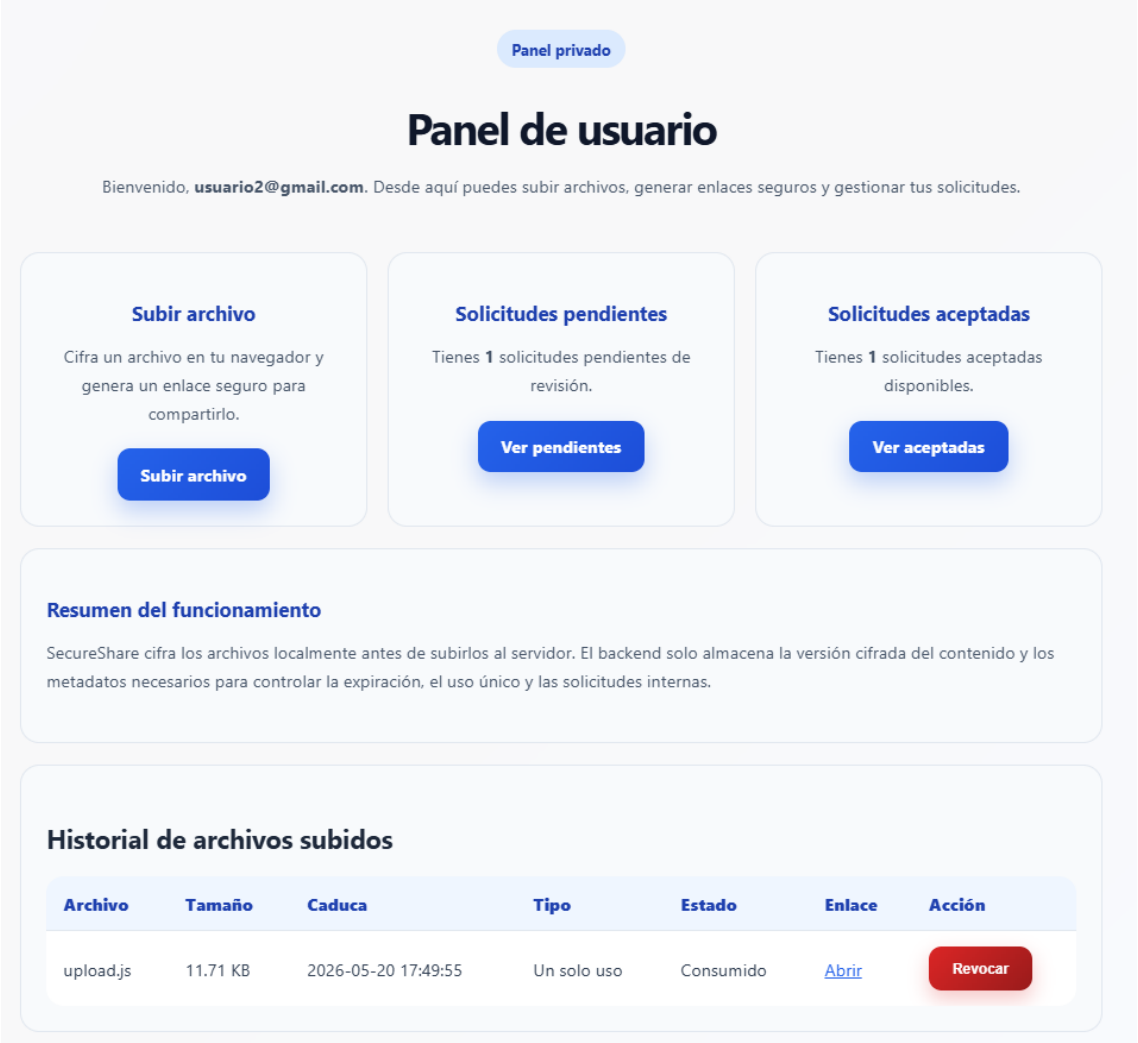

Figura 29. Panel principal de usuario tras las mejoras visuales implementadas.

Por otra parte, se incorporaron mensajes visuales relacionados con la seguridad del sistema, incluyendo indicadores de cifrado, estados de procesamiento y confirmaciones de generación de enlaces seguros.

Estos elementos permiten transmitir al usuario información relevante sobre el funcionamiento interno de la aplicación sin necesidad de conocimientos técnicos avanzados.

En el caso de los usuarios registrados, la plataforma permite realizar comparticiones internas mediante solicitudes. En cambio, en el modo invitado el intercambio del enlace debe realizarse a través de medios externos. La Figura 30 muestra la pantalla de generación del enlace seguro una vez finalizado el proceso de cifrado del archivo.

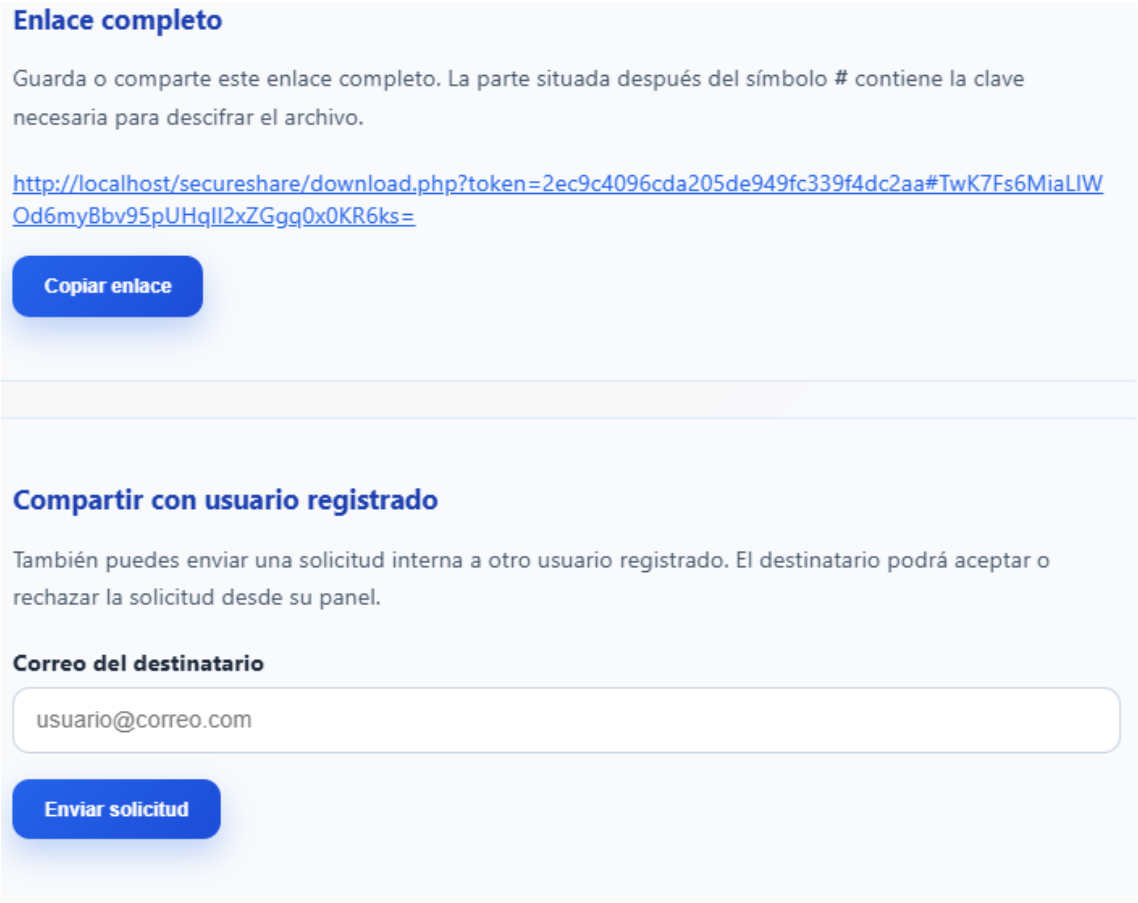

Figura 30. Generación de enlace seguro tras finalizar el cifrado del archivo.

Las pruebas de usabilidad realizadas permitieron comprobar que la aplicación presenta una interfaz clara y sencilla de utilizar, facilitando la interacción del usuario durante los diferentes procesos implementados en el sistema. Adicionalmente, las mejoras visuales incorporadas durante las últimas fases del desarrollo permitieron aproximar la apariencia final de la aplicación a soluciones web modernas orientadas al usuario final.

6.4 Limitaciones detectadas y mejoras futuras

A pesar de que SecureShare cumple con los objetivos planteados inicialmente y permite compartir archivos mediante cifrado extremo a extremo, durante el desarrollo se identificaron diferentes limitaciones técnicas, funcionales y de seguridad susceptibles de mejora en futuras versiones del sistema.

Limitaciones de Seguridad

Desde el punto de vista de la seguridad, el sistema presenta diversas limitaciones que podrían abordarse en futuras versiones.

Una de las principales limitaciones es que el sistema de autenticación implementado únicamente utiliza correo electrónico y contraseña, sin incorporar mecanismos adicionales de verificación como la autenticación multifactor (MFA). Aunque las contraseñas se almacenan utilizando funciones hash seguras, siguiendo las recomendaciones de OWASP para el almacenamiento

seguro de credenciales [10]. De este modo se utiliza la incorporación de un segundo factor de autenticación permitiría incrementar significativamente la seguridad del acceso a las cuentas de usuario [12].

Asimismo, la aplicación no implementa actualmente mecanismos avanzados de protección frente a ataques automatizados, como limitación de intentos de inicio de sesión (rate limiting) o sistemas de bloqueo temporal ante múltiples autenticaciones fallidas. Estas medidas permitirían reducir el riesgo asociado a ataques de fuerza bruta sobre las cuentas registradas.

Otra limitación detectada es la ausencia de mecanismos específicos de protección frente a ataques CSRF (Cross-Site Request Forgery). Aunque determinadas operaciones requieren autenticación previa, la incorporación de tokens CSRF permitiría reforzar la protección frente a peticiones maliciosas realizadas desde sitios externos.

Desde el punto de vista de la gestión de archivos, el sistema actualmente almacena únicamente la versión cifrada de los contenidos, sin realizar análisis adicionales sobre los archivos subidos por los usuarios. En futuras versiones podría incorporarse integración con motores antivirus o mecanismos de análisis de contenido orientados a detectar archivos potencialmente maliciosos.

Por último, las claves privadas RSA generadas durante el registro permanecen almacenadas localmente en el navegador mediante localStorage. Esto implica que la pérdida del navegador, la limpieza del almacenamiento local o el cambio de dispositivo pueden impedir recuperar el acceso a archivos compartidos previamente.

Limitaciones técnicas

El proyecto fue desarrollado principalmente como un prototipo funcional orientado a un entorno académico y de demostración. Por ello, no se han implementado mecanismos avanzados de escalabilidad, balanceo de carga o despliegue distribuido en la nube.

Del mismo modo, las pruebas realizadas se han centrado en la validación funcional y de seguridad del sistema, sin abordar escenarios de alta concurrencia o pruebas de carga orientadas a evaluar el comportamiento de la aplicación ante un elevado número de usuarios simultáneos.

En un entorno de producción real sería necesario incorporar infraestructuras más robustas capaces de soportar múltiples usuarios concurrentes, grandes volúmenes de almacenamiento y mecanismos avanzados de disponibilidad y recuperación ante fallos.

Limitaciones funcionales

En relación con las funcionalidades implementadas, actualmente el sistema no incorpora mecanismos avanzados de gestión de usuarios como recuperación de contraseñas mediante correo electrónico, verificación de cuentas o administración centralizada de usuarios. Estas funcionalidades mejorarían considerablemente la experiencia de uso y la gestión general de la plataforma.

Por otro lado, el sistema de compartición desarrollado se centra en enlaces individuales y solicitudes internas simples entre usuarios registrados. En futuras versiones podrían añadirse funcionalidades adicionales como carpetas compartidas, compartición múltiple avanzada, permisos granulares de acceso o notificaciones automáticas relacionadas con la descarga y utilización de los archivos compartidos.

Finalmente, también podrían incorporarse mecanismos avanzados de monitorización y auditoría, incluyendo registros detallados de actividad, alertas de seguridad y sistemas de trazabilidad orientados a detectar comportamientos anómalos o accesos sospechosos.

Líneas de evolución futura

En conjunto, estas posibles mejoras muestran diferentes líneas de evolución futura para el proyecto, permitiendo ampliar tanto las capacidades funcionales como los mecanismos de seguridad incorporados en la aplicación. La incorporación de estas funcionalidades acercaría SecureShare a soluciones profesionales de compartición segura de archivos orientadas a entornos reales de producción.

Capítulo 7. Conclusiones

El desarrollo de SecureShare ha permitido cumplir los objetivos planteados al inicio del proyecto, dando como resultado una aplicación web funcional orientada a la compartición segura de archivos mediante cifrado extremo a extremo. A lo largo del trabajo se diseñó e implementó un sistema capaz de cifrar archivos directamente en el navegador antes de enviarlos al servidor, evitando que el backend tenga acceso al contenido original almacenado.

Durante el desarrollo se implementaron diferentes funcionalidades relacionadas tanto con la seguridad como con la experiencia de usuario. Entre ellas destacan la generación de enlaces seguros, los enlaces de un solo uso, la expiración configurable, la revocación manual de accesos, la gestión de usuarios registrados y el sistema de solicitudes internas de compartición entre usuarios.

Desde el punto de vista técnico, el proyecto ha permitido aplicar de forma práctica conocimientos relacionados con el desarrollo web frontend y backend, la gestión de bases de datos, la autenticación mediante sesiones y la integración de mecanismos criptográficos modernos utilizando la Web Crypto API. Adicionalmente, se han reforzado conceptos relacionados con la protección de credenciales, el control de acceso y el diseño de aplicaciones orientadas a la privacidad de la información.

Uno de los aspectos más relevantes del proyecto ha sido la implementación de un modelo de cifrado extremo a extremo basado en criptografía híbrida AES-GCM y RSA-OAEP. Gracias a este enfoque, los archivos se cifran localmente en el navegador antes de ser enviados al servidor, mientras que las claves AES utilizadas para proteger cada archivo se intercambian de forma segura mediante criptografía asimétrica entre usuarios registrados.

De este modo, se mantiene el modelo de cifrado extremo a extremo definido previamente, garantizando la confidencialidad de la información almacenada. Esta arquitectura permitió aproximar el funcionamiento del sistema a soluciones modernas centradas en la protección de la privacidad y la confidencialidad de la información.

Más allá de los aspectos relacionados con la seguridad, el proyecto también evolucionó considerablemente desde el punto de vista visual y de interacción con el usuario. Durante las últimas fases del desarrollo se incorporaron diferentes mejoras orientadas a ofrecer una experiencia más moderna, intuitiva y accesible.

Por otra parte, el desarrollo del proyecto también permitió adquirir experiencia en la resolución de problemas técnicos reales surgidos durante la implementación, especialmente en aspectos relacionados con la integración entre frontend y backend, el tratamiento de archivos de gran tamaño, la gestión de sesiones y la comunicación entre JavaScript y PHP.

De igual forma, el proceso completo de análisis, diseño, implementación y validación realizado durante el proyecto ha permitido aplicar competencias relacionadas con la resolución de problemas de ingeniería, el diseño de soluciones software orientadas a la seguridad y la realización de pruebas funcionales y de validación sobre sistemas reales.

A pesar de los resultados obtenidos, el proyecto presenta diferentes posibilidades de evolución futura. Entre las posibles líneas de evolución destacan la incorporación de autenticación multifactor, mecanismos avanzados de monitorización y auditoría, integración con servicios cloud, sistemas de recuperación de contraseñas, análisis antivirus de archivos y funcionalidades avanzadas de compartición y gestión de permisos.

En conclusión, SecureShare ha permitido desarrollar un prototipo funcional que combina conceptos de desarrollo web y ciberseguridad en una aplicación real orientada a la protección de la información. El proyecto ha supuesto una experiencia de aprendizaje especialmente enriquecedora. Su desarrollo ha permitido aplicar de forma práctica numerosos conocimientos adquiridos durante la titulación relacionados con el desarrollo software, la ciberseguridad y la protección de la información, así como profundizar en tecnologías y mecanismos de seguridad ampliamente utilizados en entornos reales.

Bibliografía

[1] Mozilla Developer Network (MDN), "Web Crypto API," https://developer.mozilla.org/en-US/docs/Web/API/Web_Crypto_API. [Online].

[2] PHP Documentation Group, "PHP Manual," https://www.php.net/manual/en/. [Online].

[3] OWASP Foundation, "OWASP Top Ten," https://owasp.org/www-project-top-ten/. [Online].

[4] MySQL, "MySQL 8.0 Reference Manual," https://dev.mysql.com/doc/. [Online].

[5] Apache Friends, "XAMPP," https://www.apachefriends.org/. [Online].

[6] Microsoft, "Visual Studio Code," https://code.visualstudio.com/. [Online].

[7] National Institute of Standards and Technology (NIST), "Advanced Encryption Standard (AES)," FIPS PUB 197, November 2001.

[8] Mozilla Developer Network (MDN), "SubtleCrypto.encrypt() method," https://developer.mozilla.org/en-US/docs/Web/API/SubtleCrypto/encrypt. [Online].

[9] OWASP Foundation, "SQL Injection Prevention Cheat Sheet," https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html. [Online].

[10] OWASP Foundation, "Password Storage Cheat Sheet," https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html. [Online].

[11] Oracle Corporation, "MySQL Documentation: Prepared Statements," https://dev.mysql.com/doc/. [Online].

[12] PHP Documentation Group, "password_hash," https://www.php.net/manual/en/function.password-hash.php. [Online].

[13] PHP Documentation Group, "password_verify," https://www.php.net/manual/en/function.password-verify.php. [Online].

ANEXO A. CONTRIBUCIÓN A LOS OBJETIVOS DE DESARROLLO SOSTENIBLE (ODS)

La Agenda 2030 de las Naciones Unidas establece 17 Objetivos de Desarrollo Sostenible (ODS) orientados a promover un desarrollo económico, social y medioambiental sostenible. Aunque SecureShare es un proyecto centrado en el ámbito de la ciberseguridad y el desarrollo de software, sus características contribuyen de forma directa e indirecta a varios de estos objetivos.

ODS 4. Educación de calidad

El desarrollo de SecureShare ha permitido aplicar de forma práctica conocimientos adquiridos durante la titulación relacionados con el desarrollo web, la criptografía, la ciberseguridad, la gestión de bases de datos y la ingeniería del software.

Asimismo, el proyecto constituye un ejemplo real de aplicación de tecnologías y mecanismos de seguridad ampliamente utilizados en entornos profesionales, como el cifrado extremo a extremo, la criptografía híbrida basada en AES-GCM y RSA-OAEP, la autenticación de usuarios y la protección de la información. De este modo, el trabajo contribuye al aprendizaje práctico y a la adquisición de competencias técnicas relacionadas con las tecnologías de la información y las comunicaciones.

ODS 9. Industria, innovación e infraestructura

El ODS 9 promueve el desarrollo de infraestructuras resilientes, la innovación tecnológica y el fortalecimiento de la industria mediante la incorporación de nuevas soluciones digitales. SecureShare contribuye a este objetivo mediante el diseño e implementación de una aplicación web orientada a la protección segura de la información compartida a través de Internet.

La utilización de tecnologías modernas como la Web Crypto API, junto con la implementación de mecanismos de cifrado extremo a extremo, intercambio seguro de claves criptográficas y control de acceso a recursos, representa una aplicación práctica de la innovación tecnológica en el ámbito de la seguridad digital.

Además, el proyecto fomenta el desarrollo de infraestructuras digitales más seguras y fiables, contribuyendo a mejorar la protección de la información frente a accesos no autorizados durante los procesos de almacenamiento y compartición de archivos.

ODS 16. Paz, justicia e instituciones sólidas

El ODS 16 tiene entre sus objetivos promover sociedades más seguras, transparentes y confiables. En el ámbito digital, la protección de la privacidad y de la información constituye un elemento fundamental para alcanzar dichos objetivos.

SecureShare contribuye a este ODS mediante la implementación de mecanismos destinados a garantizar la confidencialidad de los datos compartidos, reduciendo la exposición de información sensible frente a terceros y favoreciendo la protección de la privacidad de los usuarios.

La utilización de técnicas criptográficas modernas permite reforzar la confianza en los sistemas digitales, fomentar buenas prácticas relacionadas con la ciberseguridad y promover el uso responsable de las tecnologías de la información. Asimismo, el proyecto pone de manifiesto la importancia de incorporar medidas de protección de datos desde las fases iniciales del diseño de aplicaciones y servicios digitales.