



MÓDULO PROYECTO

Ciclo Superior Desarrollo de Aplicaciones Web

Departamento: Informática y Comunicaciones

IES "María Moliner"

Curso: 2024/2025

Grupo: S2I

Proyecto: Pass Warriors

Miguel Guerrero Martín

Email: miguel.guemar@educa.jcyl.es

Tutor individual: Enrique Carballo Albarrán

Tutor colectivo: Enrique Carballo Albarrán

Fecha de presentación: Mayo 2025

Índice

1.	Descripción del proyecto	1
1.1.	Objetivos	2
1.2.	Metodología	3
1.3.	Entorno de trabajo.....	4
1.3.1.	Tecnologías	4
1.3.2.	Herramientas y lenguajes de programación.....	4
2.	Descripción general del producto	7
2.1.	Funcionalidades básicas	8
2.2.	Usuarios	8
2.3.	Sistemas accesibles.....	9
2.4.	Arquitecturas, modelos y técnicas.....	9
2.4.1.	Frontend.....	9
2.4.2.	Backend	10
3.	Despliegue	11
3.1.	Base de datos	12
3.2.	Backend	12
3.3.	Frontend.....	13
4.	Planificación y presupuesto.....	13

4.1.	Planificación	13
4.1.1.	Semana 1.....	14
4.1.2.	Semana 2.....	15
4.1.3.	Semana 3.....	15
4.1.4.	Semana 4.....	16
4.1.5.	Semana 5.....	16
4.1.6.	Semana 6.....	16
4.1.7.	Semana 7.....	16
4.1.8.	Semana 8.....	17
4.1.9.	Semana 9.....	17
4.1.10.	Semana 10.....	17
4.1.11.	Semana 11	17
4.2.	Presupuesto	18
4.2.1.	Costes materiales	18
4.2.2.	Costes profesionales.....	19
4.2.3.	Costes totales	19
5.	Documentación técnica	20
5.1.	Análisis del sistema.....	20
5.1.1.	Registro e inicio de sesión	21

5.1.2.	Navegación	22
5.1.3.	Almacenamiento de contraseñas	22
5.1.4.	Generador de contraseñas	23
5.1.5.	Auditoría de contraseñas	23
5.1.6.	Personalización de la interfaz	25
5.1.7.	Modificación de cuenta	25
5.2.	Riesgos	26
5.2.1.	Retrasos por falta de conocimiento.....	26
5.2.2.	Retrasos por fallas técnicas	27
5.3.	Diseño del sistema.....	27
5.3.1.	Modelo de datos	27
5.3.2.	Diseño de interfaz	30
5.4.	Implementación.....	35
5.4.1.	Backend	35
5.4.2.	Frontend.....	42
5.5.	Pruebas.....	48
6.	Manuales.....	49
6.1.	Manual de usuario.....	49
6.1.1.	Objetivos	49

6.1.2.	Estructura principal	49
6.1.3.	Navegación	52
6.1.4.	Registro e inicio de sesión.....	53
6.1.5.	Generador de contraseñas.....	57
6.1.6.	Baúl personal	59
6.1.7	Baúl compartido	63
6.1.8.	Auditoría de seguridad	64
6.2.	Manual técnico	71
6.2.1.	Objetivo.....	71
6.2.2.	Base de datos	71
6.2.3.	Backend	78
6.2.4.	Frontend.....	83
7.	Conclusiones.....	87
7.1	Futuras ampliaciones	87
8.	Bibliografía	88

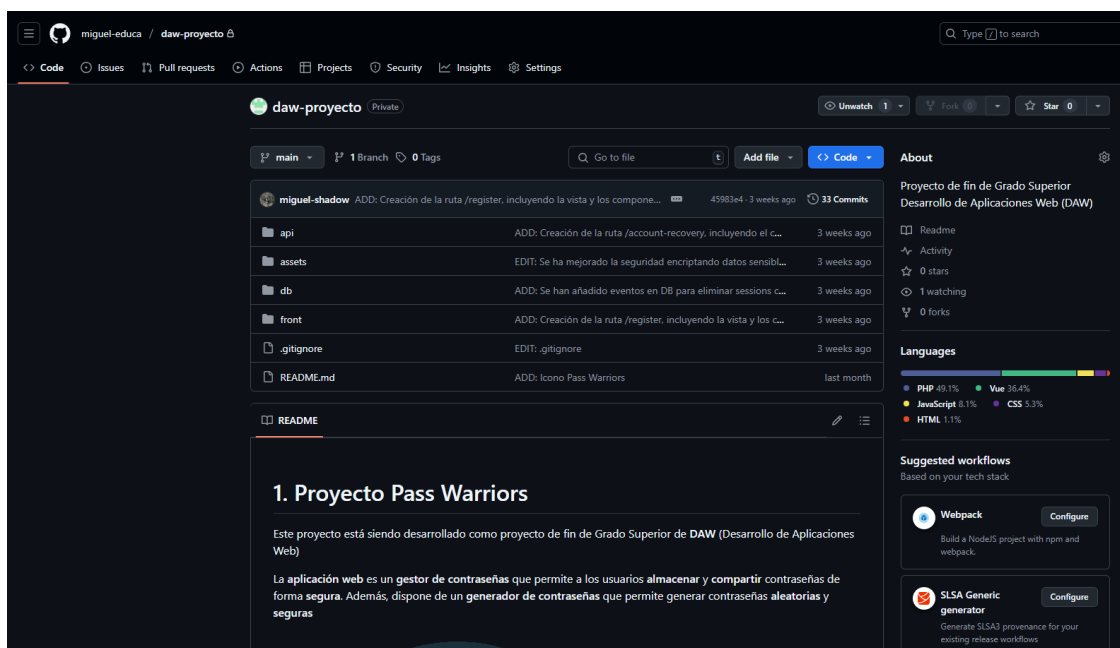
1. Descripción del proyecto

El tema escogido para el proyecto de fin de Grado Superior de DAW es la realización de una aplicación web. La aplicación web, llamada “**Pass Warriors**”, trata sobre un gestor de contraseñas que, mediante una única contraseña maestra, el usuario puede acceder a todas sus contraseñas almacenadas, pudiendo organizarlas en carpetas, editarlas, hacer revisiones de seguridad... Además, la aplicación dispone de un generador de contraseñas seguras para aumentar la seguridad de las contraseñas utilizadas por el usuario.

El usuario también puede compartir contraseñas con otros usuarios, teniendo permisos de lectura únicamente.



El código está disponible en **GitHub** (<https://github.com/miguel-educa/daw-proyecto>).



1.1. Objetivos

El objetivo principal del proyecto es **aplicar** los **conocimientos adquiridos** durante el curso para poder desarrollar una aplicación real que pueda utilizarse como un servicio.

En este caso, he decidido ayudar a los usuarios a **mejorar** la **seguridad** de sus contraseñas, permitiéndoles crear y almacenar contraseñas distintas y seguras sin necesidad de recordarlas todas, sino que solo deben recordar una única **contraseña maestra**.

Otro objetivo es desarrollar una **interfaz** personalizada de usuario que cumpla con los estándares de **accesibilidad** y **usabilidad**, permitiendo a cualquier tipo de usuario utilizar la aplicación de manera sencilla e intuitiva y en cualquier tipo de dispositivo

Otro objetivo del desarrollo de esta aplicación es aprender sobre el **manejo** de **sesiones** de usuario **seguras**, evitando la **suplantación** o el **robo** de **cookies**, además de establecer **duraciones** de **sesiones personalizadas** y **distinción** de **dispositivos**, es decir, saber si el dispositivo coincide con el que se inició sesión.

Cómo último objetivo, aprender distintos **métodos criptográficos** existentes para almacenar información cifrada de manera segura, teniendo en cuenta el objetivo del cifrado, si únicamente se debe **almacenar** y **comparar**, pero **no recuperar**, un elemento cifrado, si se desea **almacenar** y realizar una **búsqueda** de un elemento cifrado, o si se desea **almacenar** y **recuperar** un elemento cifrado.

1.2. Metodología

Aunque hay numerosas metodologías de trabajo, después de investigar y contrastar las ventajas y desventajas de las más populares, se ha decidido utilizar el **modelo en cascada** (*Waterfall*)

El modelo *Waterfall* sigue siendo una de las metodologías de trabajo más utilizadas. Se caracteriza por ser un proceso lineal, donde cada fase debe completarse antes de comenzar la siguiente. Su principal ventaja es que es un proceso muy estructurado y organizado, diferenciando cada fase del proceso para facilitar la planificación y la gestión. Esta estructura también dificulta la flexibilidad del proyecto, es decir, que los requisitos de la aplicación vayan cambiando durante el desarrollo.

El modelo en cascada se adapta correctamente a las necesidades de este proyecto, teniendo las siguientes fases:

- **Análisis:** Recopilación de requisitos del sistema y del producto, análisis de casos de uso y especificación de las funcionalidades que tendrá la aplicación.
- **Diseño:** En esta fase se tiene en cuenta los requisitos analizados en la fase anterior y se diseña la estructura de datos e interfaces.
- **Implementación:** En esta fase se comienza a desarrollar la aplicación, teniendo en cuenta el análisis y diseño anteriormente realizado.
- **Pruebas:** En esta fase se comprueba que se cumplen los requisitos recogidos durante la fase de análisis. También se comprueba errores, tanto de implementación como de carga, asegurando que la aplicación es fiable.
- **Despliegue:** En esta fase se proporciona la aplicación para que los usuarios finales puedan utilizarla.
- **Mantenimiento:** Esta fase permanece abierta durante la vida útil de la aplicación. En ella se realizan nuevas funcionalidades de la aplicación, mejoras de diseño o implementación, corrección de errores...

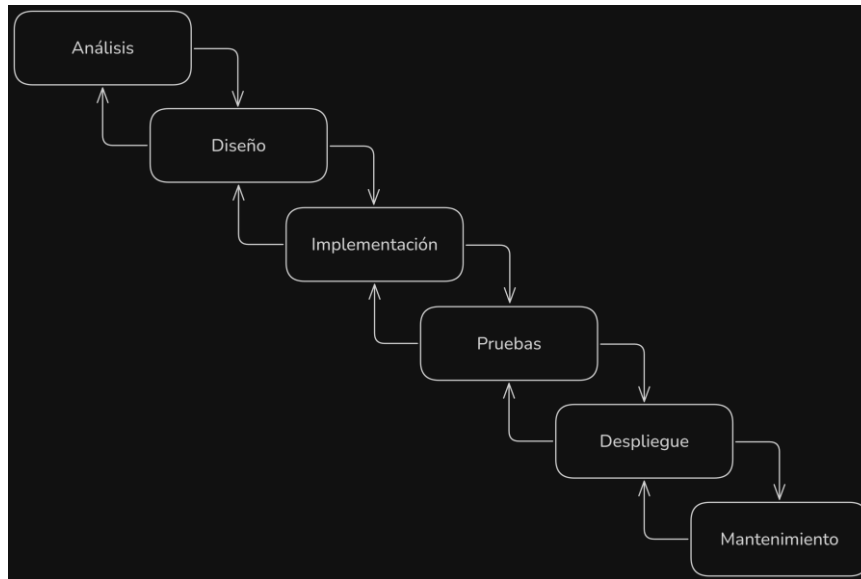


Ilustración 1: Diagrama metodología Waterfall

Las fases se complementan de manera **lineal**, es decir, de una fase se pasa a la siguiente, pero **sin saltarse** fases. El modelo clásico en cascada no permite regresar a la fase anterior, pero versiones más modernas sí que lo permiten, aumentando la flexibilidad del desarrollo si se modifica algún requisito, se detecta un error de planteamiento, etc. Aun así, la **idea principal** de este modelo sigue siendo que el flujo de desarrollo sea hacia delante de manera lineal, evitando el regreso a fases anteriores tomando el tiempo necesario para realizar los planteamientos correctos durante cada fase.

1.3. Entorno de trabajo

1.3.1. Tecnologías

El desarrollo del proyecto se ha llevado a cabo con un enfoque **cliente-servidor**, utilizado para aplicaciones web, y se ha utilizado una **arquitectura distribuida** mediante el uso de un **host** remoto.

1.3.2. Herramientas y lenguajes de programación

Para la organización y planificación del proyecto se ha utilizado la metodología **Kanban**. Consta de un tablero virtual en el que se añaden las tareas y subtareas a desarrollar, habiendo 3 estados, *por hacer*, *en desarrollo* y *hecho*.

También se ha utilizado **Git** como control de versiones del proyecto y **GitHub** como repositorio remoto. Gracias al control de versiones y el tablero *Kanban*, se puede llevar un seguimiento del desarrollo y las tareas realizadas.

Para la realización de esquemas y diagramas se ha utilizado draw.io. Para el diseño de las interfaces se ha utilizado [Figma](https://www.figma.com). Ambas herramientas son aplicaciones web.

El desarrollo se ha llevado a cabo en un ordenador con el sistema operativo **Windows 11**, aunque el destino final de la aplicación (despliegue) es implementarse en un **host remoto**, generalmente con un sistema operativo basado en **Linux**. Para la carga de archivos al *host* mediante SFTP se ha utilizado [FileZilla](https://filezilla-project.org).

Como editor de código se ha utilizado [Visual Studio Code](https://code.visualstudio.com), actualizada a su última versión.

Se ha utilizado **MySQL** (versión 8.4.2) como gestor de bases de datos relacional que almacena los datos producidos por la aplicación. Para el desarrollo en local se ha utilizado [DBEngine](https://dbeaver.io) (última versión). Esta herramienta permite la creación y gestión de distintas instancias de bases de datos (*MySQL*, *MariaDB*, *PostgreSQL* y *Redis*), permitiendo seleccionar que versión utilizar en cada instancia en un mismo entorno.

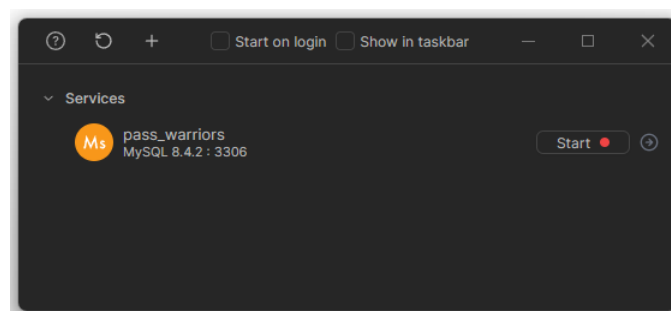


Ilustración 2: DBEngine

Como cliente de la base de datos se ha utilizado [DBeaver](https://dbeaver.io) (versión 25.0.4). Esta herramienta permite realizar conexiones a bases de datos, tanto locales como remotas, y realizar consultas, modificaciones y visualización de estructura o datos.

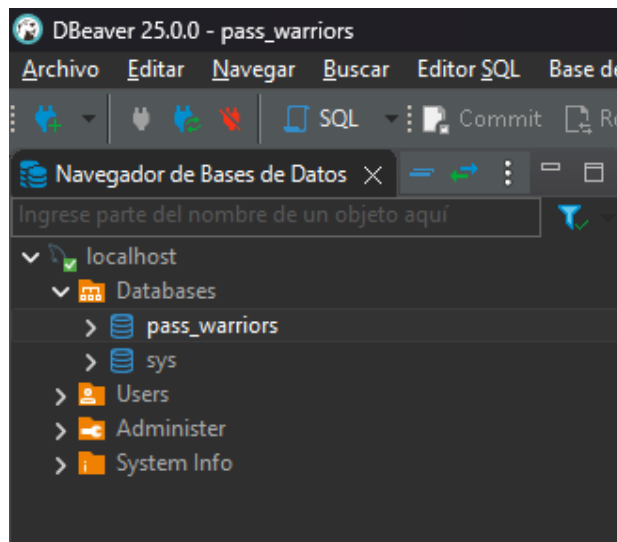


Ilustración 3: DBeaver

Para el desarrollo de la **API (backend)** de la aplicación se utiliza **PHP 8.2.12**. Como entorno de desarrollo se ha utilizado [XAMPP](#) (versión 8.2.12 de PHP), pero únicamente como servidor **Apache** con **PHP**.

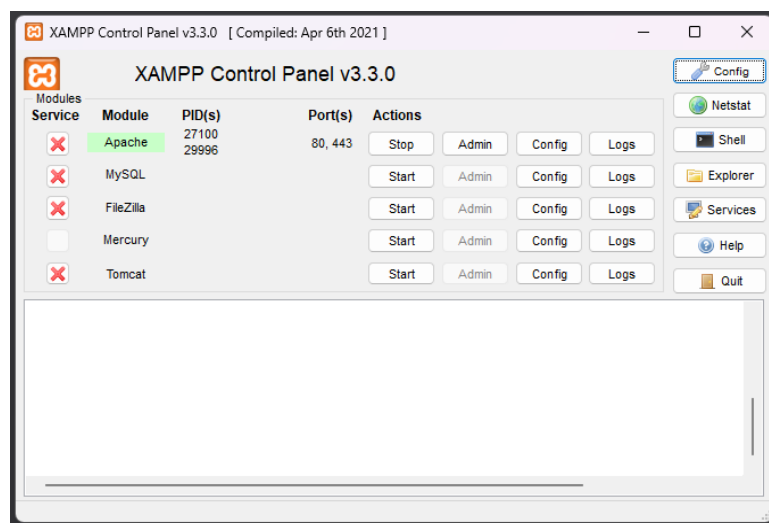


Ilustración 4: XAMPP

También se ha utilizado la librería externa [PHPGangsta/GoogleAuthenticator](#) para implementar la autenticación de doble factor mediante código temporal.

Para el desarrollo de la interfaz (*frontend*) se ha utilizado [Vue 3](#) como *framework* de desarrollo. Para utilizar esta tecnología, se debe instalar el entorno de ejecución [Node JS](#) (versión 22.12). Como gestor de paquetes de *Node*, se ha utilizado [PNPM](#)

(versión 9.15.4). *Vue 3* utiliza *HTML*, *CSS* y *JavaScript* para la creación de **componentes**. Otras tecnologías que se han utilizado en el desarrollo del *frontend* son:

- [Vue Router](#) para la gestión de rutas ***Single Page Application***.
- [Pinia](#) como almacén de estados entre componentes.
- [Bulma](#) como *framework* de *CSS*.
- [Font Awesome](#) como biblioteca de iconos.
- [Google Fonts](#) como biblioteca de fuentes de texto:
 - [Winky Rough](#) como fuente del logo
 - [Inter](#) como fuente principal
 - [Fira Code](#) como fuente auxiliar

2. Descripción general del producto

La aplicación web “*Pass Warriors*” es bastante autónoma y localizada, las funciones principales no dependen de aplicaciones externas.

La aplicación está preparada para **funcionar** con el *backend* y el *frontend* **ejecutándose** en un **mismo servidor**, o divididos en **distintos servidores**. La **base de datos** de la aplicación también puede **ejecutarse** en el **mismo servidor** o en otro **servidor distinto**. De esta manera, la aplicación puede escalar y dividir recursos y tráfico si la cantidad de usuarios aumenta.

Algunas funcionalidades sí dependen de **sistemas externos** (no disponemos de control sobre ellos) como **Google Fonts**, para la carga de fuentes personalizadas (aunque la aplicación seguiría funcionando completamente sin este sistema externo), o la API de [HavelBeenPwned](#), que permite saber si una contraseña ha sido filtrada en una filtración masiva de datos.

2.1. Funcionalidades básicas

La aplicación tendrá las siguientes funcionalidades básicas:

- **Registro o inicio de sesión** (pudiendo elegir la duración). Permite el acceso a la parte privada de la aplicación. También se puede **cerrar sesión** manualmente.
- **Generar contraseñas**, pudiendo personalizar la generación.
- **Almacenar contraseñas privadas**, incluyendo el nombre de usuario, 0 o hasta 5 enlaces y notas personalizadas
- **Almacenar contraseñas públicas**, pudiéndose compartir con otros usuarios.
- **Comprobar** la **seguridad** de las contraseñas almacenadas, mediante resúmenes, duplicaciones o filtraciones.
- **Personalizar** la interfaz de usuario, **tema** y ubicación de la **barra lateral** en dispositivos grandes.
- **Modificar detalles** de la **cuenta** como el nombre a mostrar, la contraseña maestra, generar un nuevo código de recuperación, activar la autenticación de doble factor...

2.2. Usuarios

La aplicación tiene 2 tipos de usuarios, **anónimos** y **autenticados**.

Los **usuarios anónimos** son aquellos que acceden a la aplicación sin registrarse o sin iniciar sesión. Tienen **acceso** a la página de **inicio**, a las páginas de **registro** e **inicio de sesión** y al **generador de contraseñas**.

Los **usuarios autenticados** son aquellos que acceden a la aplicación después de registrarse o iniciar sesión. Tienen **acceso a todas las funcionalidades** y páginas de la aplicación, excepto al registro e inicio de sesión. Aunque no existen tipos de

usuarios autenticados, todos tienen las mismas funciones, sí que hay una diferencia al compartir elementos. Los **usuarios** que **comparten** un elemento con otros usuarios pueden realizar **modificaciones** de éstos. Sin embargo, cuando el usuario es quién **recibe** el elemento compartido, únicamente puede **ver** los detalles, es decir, no puede realizar modificaciones.

2.3. Sistemas accesibles

La aplicación, al ser una **aplicación web**, puede ejecutarse desde **cualquier** tipo de **dispositivo** que disponga un **navegador** y **conexión** a **internet**. El navegador debe **permitir** la ejecución de código **JavaScript**, en caso contrario, la aplicación no funcionará y se mostrará un aviso.

2.4. Arquitecturas, modelos y técnicas

La aplicación implementa una arquitectura **cliente-servidor** basada en el modelo **SPA** (*Single Page Application*) con comunicación mediante **API REST**. Esta arquitectura permite una clara separación entre el *frontend* (cliente) y el *backend* (servidor), mejorando la escalabilidad y el mantenimiento. También permite reutilizar el *backend* en futuras aplicaciones nativas en dispositivos móviles o de escritorio.

La **comunicación** entre el *frontend* y el *backend* se realiza mediante **solicitudes HTTP** (GET, POST, DELETE...). El *backend*, a su vez, se comunica con la base de datos para obtener la información que devolverá al *frontend* en formato **JSON**.

2.4.1. Frontend

El *frontend* se ha basado en el modelo SPA. Este tipo de aplicaciones se **cargan una sola vez** en el navegador y, en vez de recargar toda la página mientras se navega por las distintas rutas de la aplicación, se **solicitan** los **datos** correspondientes y se **muestra** el contenido **dinámicamente**. De esta manera, se consigue una **navegación** más **fluida** para el usuario y mejora el rendimiento, **evitando sobrecargar** el servidor, debido a que el navegador solo descarga y renderiza los recursos necesarios.

Se utiliza **comunicación asíncrona** con el *backend* para solicitar los datos a mostrar evitando pausas incómodas al usuario mientras realiza operaciones en la aplicación.

La técnica utilizada en el *frontend* es el desarrollo de **componentes**. Un componente es una porción de código, diseño o estilo de la aplicación que se puede extraer y reutilizar en distintas partes de la aplicación, evitando la duplicación de código y facilitando el mantenimiento.

Para la comunicación entre componentes, se ha utilizado **almacenes de estados**. Esta utilidad permite sincronizar estados entre distintos componentes y mantener una conexión de los datos entre ellos, mejorando la experiencia de usuario.

2.4.2. Backend

El *backend* se ha basado en el modelo **MVC** (Moldeo-Vista-Controlador) para la creación de una API REST. La **Vista** se encarga de exponer o recibir la información (en formato JSON) enviada o solicitada por el *frontend* y que está almacenada en la base de datos. El **Controlador** se encarga de realizar las validaciones de la información enviada por el usuario, comprobar la sesión y los permisos del usuario y comunicar el Modelo con la Vista. El **Modelo** se encarga de comunicarse y realizar operaciones con la base de datos.

Una **API REST** es un tipo de aplicación que realiza intercambios de datos en formato **JSON** mediante peticiones HTTP como GET (solicitar recursos), POST (creación de recursos), DELETE (eliminación de recursos), PATCH (edición parcial de recursos)...

De esta manera se logra dividir las partes fundamentales de la aplicación, permitiendo escalar la aplicación, mejorando el mantenimiento y el rendimiento y posibilitando la migración de un tipo de servicio de almacenamiento (en este caso *MySQL*) a otro servicio (por ejemplo, a *PostgreSQL*).

3. Despliegue

Para el despliegue de esta aplicación se ha decidido utilizar un **hosting**, en concreto **IONOS**. El *hosting* de IONOS ofrece el acceso un servidor web (Apache) remoto mediante SSH, una base de datos, un servidor SFTP.

El servidor tiene el sistema operativo *Debian GNU/Linux 11*. El servidor tiene instalado un servidor Apache y también ofrece una base de datos y un servidor SFTP para la carga de archivos desde un cliente. IONOS también incluye un **dominio** para que los usuarios puedan acceder a la aplicación de una manera sencilla.

Desde el panel de control de IONOS tenemos acceso a los distintos servicios que ofrece el *hosting*.

- Espacio web: Permite ver, cargar o eliminar los recursos (ficheros y carpetas) del servidor desde un navegador. También muestra el espacio disponible.
- SFTP y SSH: Permite conectarse al servidor remotamente mediante SSH desde una terminal. SFTP permite la carga de archivos desde un cliente.
- Base de datos: Ofrece la creación de una base de datos *MySQL* (8.0) y acceso a ella mediante **PHPmyAdmin**, desde el propio *host*. Realiza copias de seguridad de la base de datos cada 7 días. También se pueden obtener las credenciales (nombre de usuario y contraseña) para acceder a la base de datos sin utilizar PHPmyAdmin, por ejemplo, desde el *backend*.
- PHP: Se puede seleccionar que versión de PHP se ejecuta en el servidor. En este caso se ha elegido la versión 8.4.

En este caso, por motivos didácticos, se ha decidido desplegar en el mismo *hosting* tanto el *frontend* como el *backend*, pero los pasos a seguir serían similares si fuesen en distintos *hosts*.

3.1. Base de datos

Para desplegar la base de datos, nos conectamos desde *PHPmyAdmin* o cualquier otra opción proporcionada por el *host* y cargamos el *script* con la estructura de la base de datos.

3.2. Backend

Para desplegar el *backend* en el *host*, primero debemos verificar que se está utilizando la versión correcta de PHP en el *host*, en este caso 8.4 o superior.

Después, en el proyecto, se debe crear y modificar el archivo de configuración del *backend* llamado ***env.php*** (está disponible una plantilla). Este archivo contiene variables de entorno y configuraciones de seguridad que se deben establecer antes de cargar los archivos en el *host*, no se puede utilizar la misma configuración en local (durante el desarrollo) que en el despliegue.

- *DB_HOST*: IP o nombre del dominio de la base de datos.
- *DB_USER*: Nombre del usuario de la base de datos.
- *DB_PASSWORD*: Contraseña del usuario de la base de datos.
- *DB_NAME*: Nombre de la base de datos.
- *ENCRYPTION_PASSPHRASE*: Clave de cifrado con el que se cifraran los datos sensibles en la base de datos. Debe ser larga (mayor a 32 caracteres) de carácter aleatorio y nunca debe ser compartida.

Una vez modificado los parámetros correspondientes, el **despliegue** es tan sencillo como conectarse con FileZilla al servidor SFTP y arrastrar los archivos y carpetas del *backend* a la raíz servidor. Si el despliegue del *frontend* y *backend* es en el mismo servidor (como en este caso), se debe crear una carpeta llamada *“/api”* e introducir los archivos y carpetas del *backend* en ella. Así se mantienen separados ambos entornos y se facilita el mantenimiento.

3.3. Frontend

Para desplegar el *frontend*, primero se debe crear un archivo llamado “.env” (existe una plantilla). Este archivo contiene configuraciones necesarias para el funcionamiento de la aplicación:

- **VITE_API_URL:** IP o dominio del *backend*. Si se despliegan ambos entornos en el mismo *host*, se puede establecer en “/api”.

Una vez creado y modificado el archivo de configuración, mediante el comando “*pnpm run build*” desde una terminal. Este comando genera, en la carpeta “/dist”, los archivos y carpetas optimizados para producción. Mediante *FileZilla*, al igual que en el despliegue del *backend*, se deben arrastrar los archivos y carpetas generados a la raíz del servidor.

4. Planificación y presupuesto

4.1. Planificación

- Planificación inicial

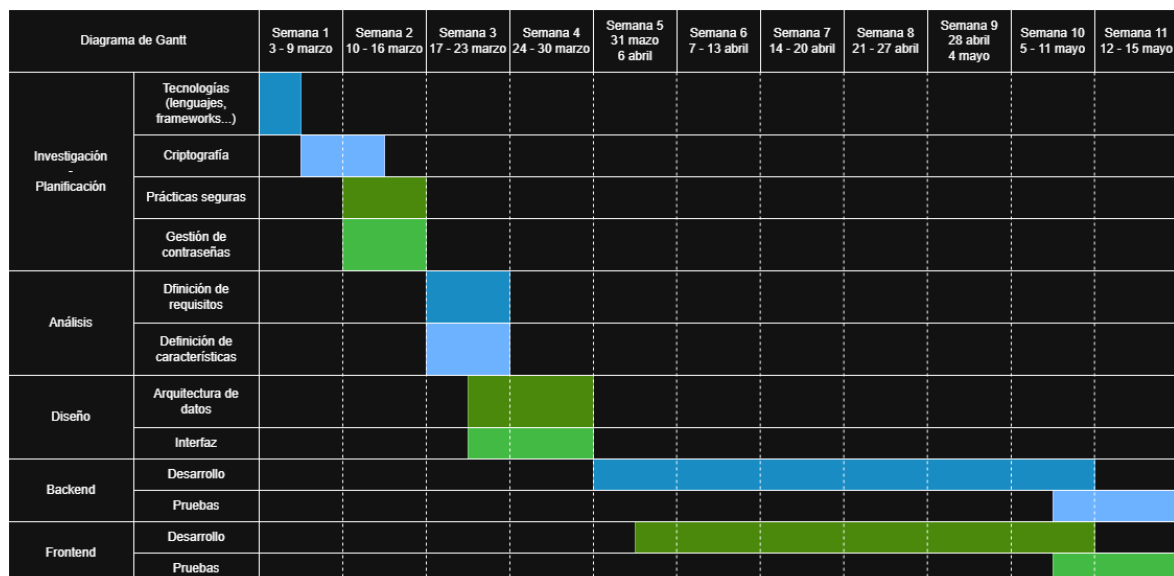


Ilustración 5: Diagrama de Gantt inicial

- Panificación final

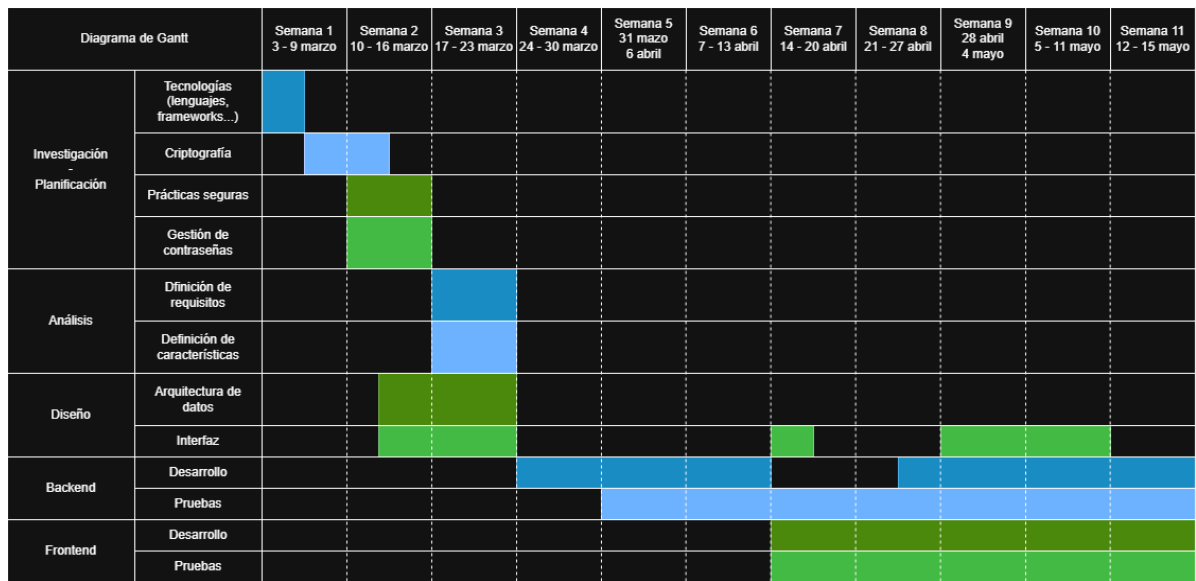


Ilustración 6: Diagrama de Gantt final

4.1.1. Semana 1

El comienzo de proyecto fue investigar y estudiar que tecnologías se iban a usar para el análisis y desarrollo del proyecto. Se investigó sobre qué *frameworks* se podrían utilizar, qué lenguajes, librerías o servicios externos podrían facilitar o mejorar el desarrollo, añadir funcionalidades, etc.

Durante la primera semana también se comenzó a investigar que tipos de criptografía existen para almacenar información de manera segura, dependiendo de la finalidad del elemento a almacenar. Para esta aplicación, el cifrado tiene 3 finalidades:

- **Cifrado unidireccional:** Cifrar elementos sin posibilidad de recuperar su valor original, pero sí compararlo.
- **Cifrado unidireccional determinista:** Cifrar elementos sin posibilidad de recuperar su valor original, pero el cifrado debe retornar siempre un mismo valor a una misma entrada. Esto permite realizar búsquedas de elementos sin comprometer su seguridad.
- **Cifrado reversible:** Cifrar elementos, pero con la posibilidad de recuperar su valor original.

4.1.2. Semana 2

Durante esta semana se finalizó la investigación relacionada con la criptografía y se comenzó a plantear que prácticas de seguridad se podrían implementar y cuál sería la mejor forma de gestionar las contraseñas y datos sensibles del usuario.

También se estudió que posibles *hostings* podrían servir la aplicación.

Aunque no estaba programado desde el inicio, también se comenzó a diseñar la estructura de datos, definiendo las entidades principales y como se relacionan entre ellas. Esta parte ayudó a entender a como realizar la gestión de contraseñas y datos de los usuarios, gestionar las sesiones, etc.

También se comenzó a pensar que posibles vistas tendría la aplicación y como sería su diseño.

4.1.3. Semana 3

Durante esta semana se definieron los requisitos y características que tendría la aplicación. Se enumeraron las funcionalidades, tanto de usuarios autenticados como anónimos, se desarrolló el flujo de autenticación y acceso a información privada, qué estructura tendría la información recibida...

Con los requisitos terminados, se pudo completar el diseño de la base de datos. Se crearon los diagramas modelo entidad/relación y el modelo relacional

correspondientes. Por último, se creó la estructura de la base de datos en código SQL y se desplegó la base de datos en el entorno de desarrollo.

También se desarrollaron los prototipos de la interfaz de todas las vistas que tendrá la aplicación en *Figma*.

4.1.4. Semana 4

Se comenzó el desarrollo del *backend* (API) de la aplicación. Para ello se realizó la instalación y configuración del entorno de desarrollo *XAMPP*. Posteriormente, se definió el formato que tendrán los recursos devueltos por la API para mantener una misma estructura entre todas las rutas.

Se creó la estructura del proyecto, siguiendo el modelo MVC, se creó la plantilla del archivo de configuración y también se crearon las clases de utilidad que se utilizarán en todo el proyecto.

4.1.5. Semana 5

Se comenzó a desarrollar las primeras rutas de la API, relacionadas con el registro e inicio de sesión de los usuarios, para poder comenzar con el desarrollo del *frontend* y desarrollar ambas partes simultáneamente. También se realizaron pruebas de las rutas creadas.

4.1.6. Semana 6

En esta semana se implementaron las primeras rutas privadas de la aplicación que retornan la información privada del usuario. Se probó que la autenticación funcionase correctamente y se retornaban los datos correctos.

4.1.7. Semana 7

Durante esta semana se produjo la primera toma de contacto con **VUE 3**, el *framework* que se utilizará para el desarrollo del *frontend* y **Bulma CSS**, otro *framework* para estilar la aplicación. Se realizaron numerosos ejercicios y distintas

pruebas, además de la lectura de la documentación oficial y otros recursos, para entender el funcionamiento y poder implementarlo correctamente en el proyecto.

También se ajustaron los prototipos del diseño de la interfaz, ajustándose a los nuevos conocimientos adquiridos.

4.1.8. Semana 8

Una vez conseguido dominar los fundamentos de VUE 3 y la gestión de estados entre componentes, se comenzó a integrar el *frontend* con el *backend*, pudiendo registrar usuarios e iniciar sesión. Se realizaron distintas pruebas para comprobar que todo funcionase correctamente.

4.1.9. Semana 9

Durante esta semana, se continuó con el desarrollo del *backend* y del *frontend*, simultáneamente. También se implementaron los diseños de los prototipos a la aplicación real y se desarrolló el generador de contraseñas.

4.1.10. Semana 10

Se desarrollaron más rutas de la API, relacionadas con la edición y eliminación de elementos. También se desarrollaron más vistas de la aplicación, como el baúl (personal y público) y las auditorías de seguridad. También se implementó la interacción con la API de ***HavelBeenPwned***, que proporciona información sobre contraseñas filtradas.

4.1.11. Semana 11

Última semana del desarrollo del proyecto. Se terminó la personalización de la interfaz por parte del usuario y se implementaron las últimas características. También se desplegó la aplicación completa en el *hosting* y se realizaron pruebas en el entorno de producción.

4.2. Presupuesto

Para el cálculo del presupuesto total se ha estimado teniendo en cuenta numerosos factores que se describen a continuación.

4.2.1. Costes materiales

En este factor se han tenido en cuenta los costes de programas, *hosting* y otros gastos relacionados con herramientas de desarrollo.

- **Hosting.** Se ha utilizado IONOS como *hosting* del proyecto. Se ha contratado el servicio **IONOS Hosting Plus**, que ofrece el *host* y el nombre de dominio incluidos por unos 11 €/mes, aunque el primer año se reduce a 1 €/mes.
- **Nombre de dominio.** El nombre de dominio se incluye en el *host* durante el primer año. Al finalizar el año, se añadiría al coste total, siendo unos 6 €/año por un dominio “.es”. Para este proyecto se ha comprado el dominio “*passwarriors.es*”. Incluye el certificado SSL.
- **Protección de dominio.** Es un servicio opcional que ofrece IONOS que permite proteger al dominio de accesos no autorizados, posibles ataques como el secuestro de DNS, prueba de propiedad... Durante el primer año es 1 €/año, después aumenta a 15 €/año.
- **Herramientas de desarrollo.** Todas las herramientas utilizadas para el desarrollo de la aplicación son completamente gratuitas o poseen un plan gratuito. Tanto el editor de código, herramientas y utilidades de diseño, esquemas y control de versiones como los *frameworks*, librerías y API utilizados.

4.2.2. Costes profesionales

En este apartado se tiene en cuenta cuanto hubiese costado tener un programador junior durante todo el desarrollo del proyecto, teniendo en cuenta el sueldo medio y las horas dedicadas al proyecto. Después de contrastar diversas fuentes y portales de empleo, se ha decidido tomar como referencia unos 1450 € brutos al mes (1200 € netos al mes). Teniendo en cuenta que al proyecto se han dedicado unas 500 horas durante 74 días (6,5 horas al día, incluyendo fines de semana y festivos) y que el precio/hora de un programador junior es de unos 9 €/hora, sale una estimación de **4500 €**, a los que habría que añadir un aumento por trabajo intensivo en fines de semana y festivos.

4.2.3. Costes totales

Teniendo en cuenta todos los costes descritos anteriormente, el coste total del proyecto estaría en torno a unos **5050 €**, teniendo en cuenta los precios completos del hosting de IONOS.

5. Documentación técnica

5.1. Análisis del sistema

Después de realizar el análisis de la aplicación, se han determinado los siguientes casos de uso:

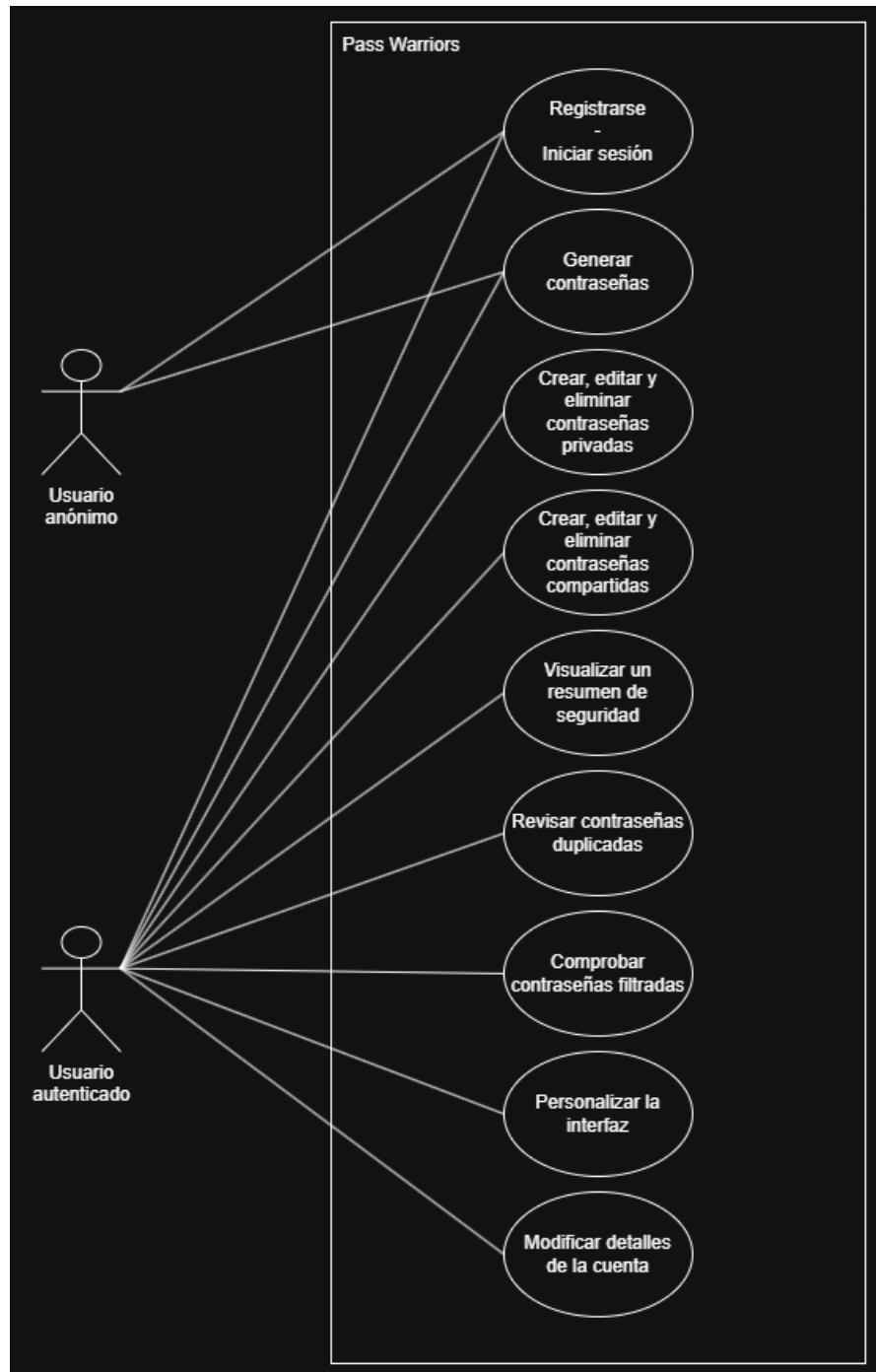


Ilustración 7: Diagrama de casos de uso

5.1.1. Registro e inicio de sesión

Los usuarios podrán **registrarse**, o **iniciar sesión** si ya se han registrado anteriormente, para acceder a todas las funcionalidades de la aplicación. Para iniciar sesión, se debe introducir el nombre de usuario y la contraseña maestra. El usuario puede elegir la duración de la sesión en el momento que inicia sesión, pudiendo elegir entre las siguientes opciones:

- **15 minutos:** Ideal en dispositivos no personales o de uso compartido.
- **30 minutos:** Ideal en dispositivos no personales o de uso compartido.
- **1 hora:** Ideal en dispositivos no personales o de uso compartido (es la opción predeterminada cuando el usuario se registra).
- **4 horas:** Ideal en dispositivos no personales.
- **8 horas:** Ideal en dispositivos no personales.
- **1 día:** Ideal en dispositivos personales.
- **1 semana:** Ideal en dispositivos personales, aunque con menor seguridad a accesos no autorizados
- **1 mes:** Ideal en dispositivos personales, aunque con menor seguridad a accesos no autorizados.
- **3 meses:** Ideal en dispositivos personales, aunque con menor seguridad a accesos no autorizados.

El usuario puede **cerrar sesión** cuando desee, sin necesidad de esperar a que caduque la sesión.

El usuario puede activar la **autenticación de doble factor** mediante **código temporal**, aumentando la seguridad de la cuenta. Al activarlo, en cada inicio de sesión se pedirá el código temporal generado por la aplicación correspondiente (*Google Authenticator, Microsoft Authenticator, Authy...*).

En caso de **no poder** iniciar sesión mediante nombre de usuario y contraseña maestra o no disponer del código de autenticación de doble factor, se podrá recuperar la cuenta mediante el **código de recuperación** generado al registrarse. Recuperar la cuenta elimina el doble factor de autenticación (se debe activar de nuevo) y se genera un nuevo código de recuperación, invalidando el anterior.

5.1.2. Navegación

El usuario, autenticado o anónimo, dispondrá de un **menú de navegación** (en la parte superior en dispositivos pequeños y en una barra en el lateral de la pantalla en dispositivos grandes) para **desplazarse** cómodamente entre las **distintas funcionalidades** de la aplicación.

El usuario anónimo podrá acceder al generador de contraseñas y a las páginas de inicio de sesión o registro.

El usuario autenticado podrá acceder a todas las funcionalidades, excepto al inicio de sesión o registro.

5.1.3. Almacenamiento de contraseñas

La **función principal** de la aplicación es el **almacenaje** seguro de contraseñas, siendo **accesibles** mediante la **contraseña maestra**. A esta sección de la aplicación se denomina **baúl**. Para acceder a esta funcionalidad, es necesario que el **usuario** esté **registrado**. Las contraseñas, denominadas **elementos**, pueden almacenar la propia contraseña, un nombre de usuario, 0 o hasta 5 enlaces y notas personalizadas, además, son identificadas mediante un **nombre**. Los elementos pueden ser de dos tipos, **privados** y **públicos**.

Se pueden **crear**, **editar** o **eliminar** la cantidad de elementos, tanto privados como públicos, que se deseen.

Los elementos privados, almacenados en el **baúl personal**, se pueden organizar en **carpetas** (sin límite de creación) facilitando la organización del baúl, pudiéndose **filtrar** por 1 o más carpetas simultáneamente. También se dispone de un **buscador** para realizar búsquedas por coincidencia parcial en el nombre del elemento.

Los elementos públicos pueden ser compartidos por el usuario o que otros usuarios puedan compartir con él. Estos elementos están disponibles en el **baúl compartido**, pudiendo editar o eliminar los elementos creados, pero únicamente pudiendo ver los elementos compartidos por otros usuarios. Se pueden **filtrar** los elementos que hayan sido compartidos por el usuario o que han sido compartidos con él, facilitando la gestión de ambos tipos. También se dispone de un **buscador** al igual que en el baúl privado, para realizar búsquedas por coincidencia parcial en el nombre del elemento.

Se puede **copiar** rápidamente el **nombre de usuario** o la **contraseña** del elemento (público o privado) sin acceder a los detalles. También se puede **abrir** el **primer enlace** almacenado en el elemento en una nueva pestaña, si contiene alguno. Para abrir el resto de los enlaces es necesario acceder a los detalles.

5.1.4. Generador de contraseñas

Es una funcionalidad fundamental para la **creación** de **contraseñas seguras**. Esta sección, disponible tanto para **usuarios autenticados** como **anónimos**, permite al usuario **generar contraseñas personalizadas**, pudiendo elegir la **longitud** de ésta (entre 5 y 50 caracteres). También el usuario puede **seleccionar** que tipo de **caracteres** debe **contener** (minúsculas, mayúsculas, números y caracteres especiales), pudiendo especificar la **cantidad mínima** de cada uno de los tipos, desde 0 hasta un máximo de 10.

En cada dispositivo se almacena un **historial** (independientes entre ellos) de las últimas 75 contraseñas generadas, siendo accesible incluso después de cerrar el navegador. Dicho historial puede ser **vaciado** cuando el usuario lo desee.

5.1.5. Auditoría de contraseñas

Es una funcionalidad muy interesante debido que permite al usuario autenticado **comprobar** rápidamente la **seguridad** de sus **contraseñas**. Este apartado se divide en 3 secciones.

Contraseñas duplicadas

Permite al usuario **visualizar** rápidamente las **contraseñas** que son **utilizadas** en **varios elementos**, permitiendo acceder a los elementos para su edición.

Contraseñas filtradas

Permite al usuario visualizar las **contraseñas** que se han encontrado en **filtraciones** masivas de **datos** pudiendo acceder a los elementos para su edición. Para este apartado, se realiza una conexión con la API de ***HavelBeenPwned***.

Resumen de seguridad

En esta sección, el usuario puede ver un **resumen** de los **requisitos de seguridad** de todas las contraseñas almacenadas. El resumen se divide en 3 apartados, seguridad **baja** (revisión urgente), **media** (revisión recomendable) y **alta**. Los requisitos que se tienen en cuenta en el resumen son:

- **Longitud** de la contraseña:
 - **Bajo**: menos de 8 caracteres.
 - **Medio**: entre 8 y 14 caracteres.
 - **Alto**: Más de 14 caracteres.
- **Caracteres mínimos**:
 - Al menos 2 **minúsculas**.
 - Al menos 2 **mayúsculas**.
 - Al menos 2 **números**.
 - Al menos 2 **caracteres especiales**.
- No contiene 3 o más **caracteres repetidos**.
- No contiene **secuencias** de **caracteres** (números o letras consecutivos, ascendente o descendente).

5.1.6. Personalización de la interfaz

El usuario podrá modificar el **tema de interfaz** que muestra la aplicación, **claro** u **oscuro**. Por defecto, se muestra el tema predefinido por el navegador. También, en los dispositivos que muestran la **barra lateral**, el usuario podrá elegir en qué lado aparece de la pantalla aparece, **derecho** o **izquierdo**.

Además, el diseño de la aplicación web es **responsive**, es decir, se adapta a cualquier tipo de pantalla y dispositivo.

5.1.7. Modificación de cuenta

El usuario autenticado podrá realizar las siguientes modificaciones de la cuenta:

- **Cambiar** el **nombre** a mostrar (no el nombre de usuario).
- **Ver** o **generar** un nuevo **código de recuperación**, invalidando el código anterior.
- **Cambiar** la **contraseña maestra**.
- **Cambiar** la duración de la sesión actual (mismas opciones que en el inicio de sesión).
- **Activar** o **desactivar** la **autenticación** de **doble factor** mediante **código temporal**. En el momento de activar, se debe verificar el código generado en la aplicación correspondiente para confirmar la activación.
- **Vaciar** baúl personal. **Eliminando** todas las **carpetas** y **contraseñas**.
- **Vaciar** baúl compartido. **Eliminando** todas las **contraseñas compartidas** a otros usuarios, pero no las que otros usuarios nos han compartido.
- **Eliminar** la cuenta. Esta opción elimina todos los datos de la cuenta, tanto carpetas como contraseñas privadas y públicas, evitando poder iniciar sesión de nuevo.

5.2. Riesgos

Durante el análisis surgieron distintos riesgos, tanto durante el desarrollo como del despliegue, que son los siguientes:

5.2.1. Retrasos por falta de conocimiento

Debido a la utilización de tecnologías y conceptos que no han sido utilizadas durante el curso, supone un riesgo su utilización debido a que se necesita tiempo para aprender a manejarlas. Este tiempo podría ser mayor del esperado, provocando un retraso en el desarrollo.

Durante este proyecto, los posibles riesgos que han surgido son los siguientes:

Comprender y manejar correctamente el cifrado de datos, entendiendo las distintas funcionalidades del cifrado y evitar al máximo los errores en este apartado, que podrían afectar a la seguridad del sistema.

Relacionado con la seguridad, también existe el riesgo de accesos o interacciones con la base de datos no autorizadas (*SQL Injection*). Para evitar este problema, que supone un serio problema de seguridad, se utilizan sentencias preparadas y validaciones de datos antes de realizar conexiones y operaciones con la base de datos.

Aprendizaje de *frameworks* nuevos. En este caso, se ha aprendido a desarrollar una aplicación con *VUE 3* y *Bulma CSS* consiguiendo crear una SPA mediante la utilización de componentes reutilizables y que se conectan entre sí, compartiendo estados. Durante este aprendizaje se produjo un pequeño retraso debido a la gran complejidad que supuso aprender el correcto manejo de estados entre distintos componentes.

Manejo de sesiones. Debido a la separación del *frontend* y el *backend*, se debió investigar las numerosas formas distintas que existen para el control de sesiones y autenticación de usuario. El riesgo de una mala implementación de esta funcionalidad podría suponer un alto peligro de seguridad en la aplicación.

5.2.2. Retrasos por fallas técnicas

Al mezclar distintas tecnologías se pueden producir errores de compatibilidad entre las distintas partes de la aplicación. Durante el desarrollo de la aplicación, no se ha sufrido ningún retraso bajo este pretexto.

También existen riesgos en el momento del despliegue, como incompatibilidad de versiones o tecnologías, pérdida de datos o problemas con la disponibilidad de la aplicación. Durante la fase de investigación, se decidió utilizar IONOS como *hosting* porque presentaba soluciones a estos posibles riesgos. Las versiones de PHP, Apache y MySQL son compatibles con las utilizadas durante el desarrollo. Además, se realizan copias de seguridad de la base de datos cada 7 días (pudiéndose ampliar a copias diarias mediante una suscripción) y disponibilidad global mediante el uso de *CDNs*, distribuyendo la aplicación entre distintos servidores de todo el mundo para garantizar un acceso rápido y alta disponibilidad.

5.3. Diseño del sistema

5.3.1. Modelo de datos

Se ha decidido utilizar un modelo relacional de datos mediante MySQL para gestionar la información generada y consumida por la aplicación. Este modelo permite estructurar los datos relacionándose entre sí, facilitando la integridad referencial.

Entidades

- **User.** Representa a un usuario registrado en la aplicación.
- **Session.** Representa la sesión de un usuario.
- **Folder.** Representa una carpeta creada por un usuario que contendrá las contraseñas.
- **Password.** Representa una contraseña. También puede almacenar un nombre de usuario, enlaces o notas.

Relaciones entre entidades

En la aplicación se producen las siguientes relaciones entre las distintas entidades:

- **User-Session.** Un usuario puede tener 0, 1 o múltiples sesiones. Cada sesión únicamente puede pertenecer a un usuario.
 - La relación es de tipo **1:N**.
- **User-Folder.** Un usuario puede crear 0, 1 o múltiples carpetas. Una carpeta puede pertenecer a un único usuario.
 - La relación es de tipo **1:N**.
- **User-Password.** Un usuario puede crear 0, 1 o varias contraseñas y una contraseña puede pertenecer a un único usuario.
 - La relación es de tipo **1:N**.

Además, un usuario puede compartir 0, 1 o múltiples contraseñas con otro usuario. Del mismo modo, a un usuario le pueden compartir 0, 1 o varias contraseñas.

- La relación es de tipo **N:M**, por tanto, se crea una nueva entidad denominada **Shared Password**, que representa la relación entre una contraseña con el usuario compartido.
- **Folder-Password.** Una carpeta puede tener 0, 1 o muchas contraseñas. Una contraseña puede pertenecer a una carpeta o a ninguna.

Cómo identificadores únicos de las entidades, se ha decidido utilizar UUID v4 (Universally Unique Identifier versión 4). Es una cadena aleatoria de 32 caracteres hexadecimales separados por guiones en grupos de 8, 4, 4, 4 y 12 caracteres. Permite identificar a una entidad de manera global, incluso fuera de su propia tabla y añade una capa de seguridad extra, debido a que es bastante complicado poder adivinar los IDs de las demás entidades.

Todas las acciones de **actualización** o **eliminación** en elementos relacionados son en **cascada**, es decir, si se elimina o se actualiza un registro “*padre*”, se elimina o se actualiza el registro “*hijo*”, excepto en la relación *Folder-Password*. En esta relación, al momento de **eliminarse** una carpeta, el atributo referenciado en la contraseña se establece en “**null**”, indicando que no pertenece a ninguna carpeta.

Diagramas

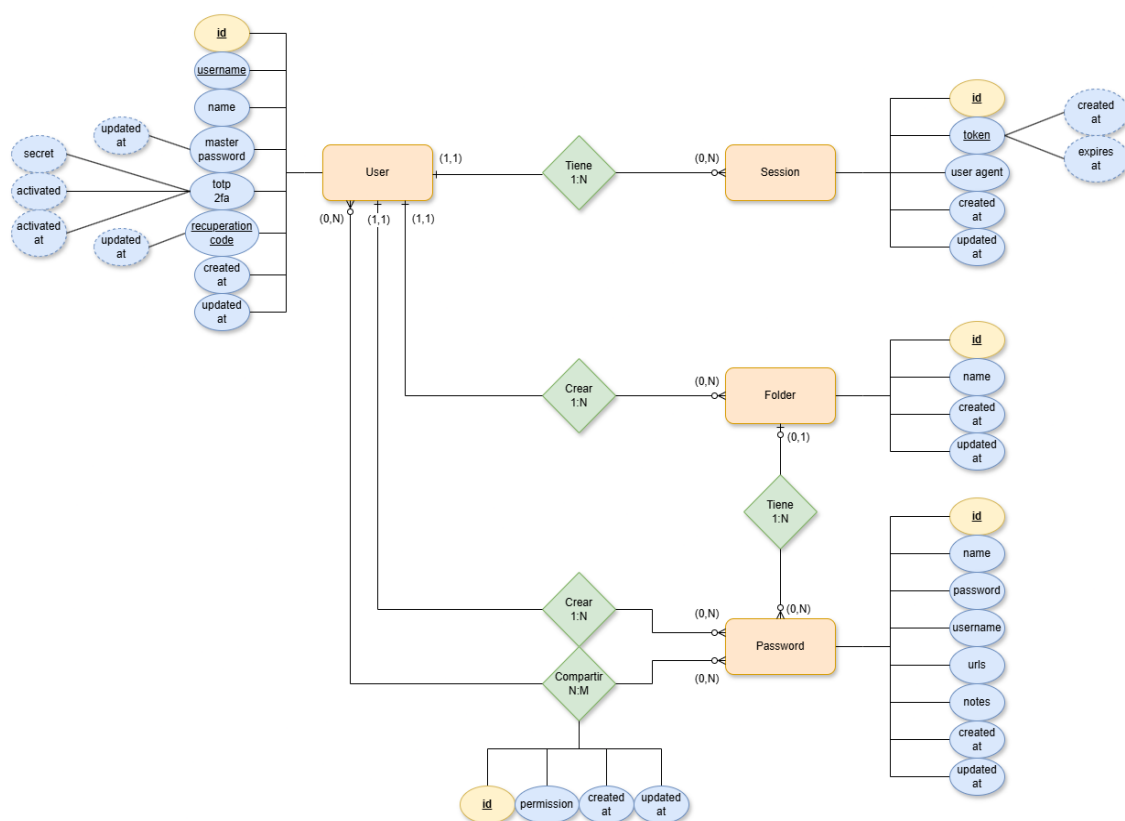


Ilustración 8: Diagrama Entidad/Relación

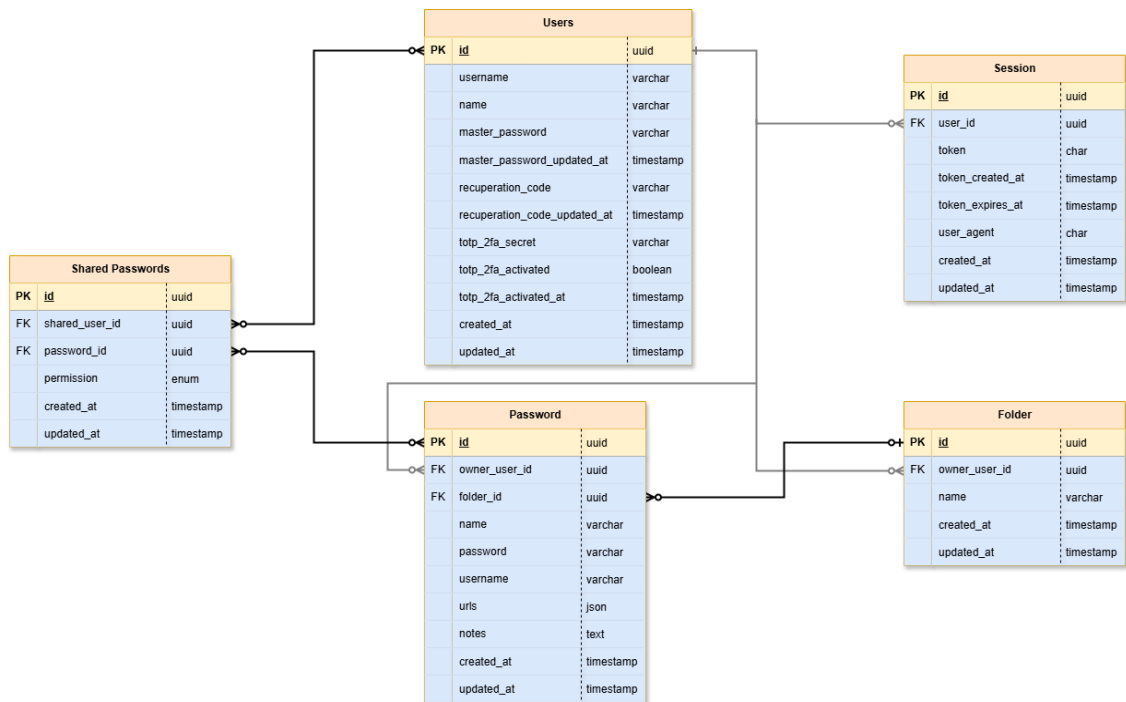


Ilustración 9: Diagrama modelo relacional

5.3.2. Diseño de interfaz

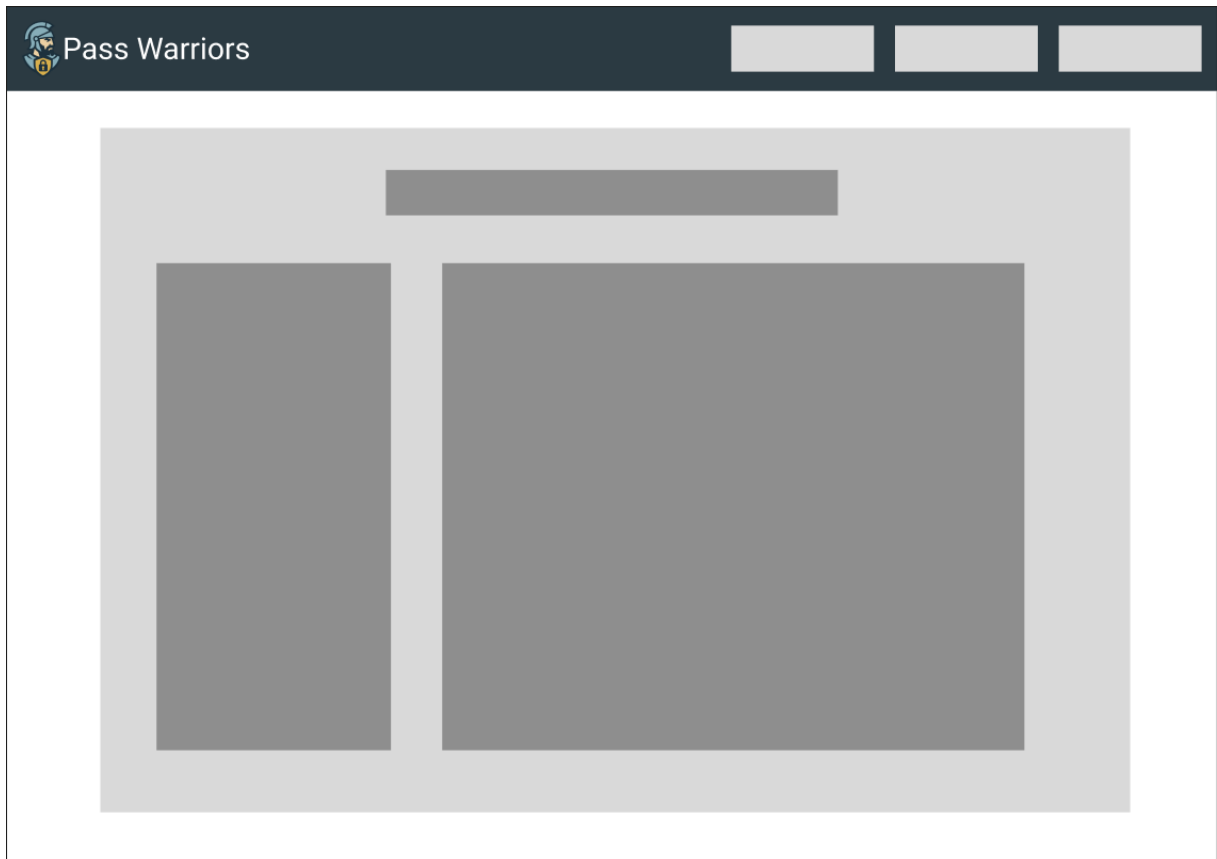
Para el diseño de la interfaz se ha utilizado, en primer lugar, papel y boli, para desarrollar los primeros bocetos de los prototipos y, posteriormente, se ha utilizado Figma para prototipar la interfaz final.

Se ha utilizado una paleta de colores con tonalidades azules, diseñando un tema oscuro y un tema claro y manteniendo una misma estética en todas las secciones de la aplicación. Para facilitar el mantenimiento y los estilos, se ha utilizado el *framework Bulma CSS* y variables CSS para almacenar los colores seleccionados.

Prototipos

A continuación, se muestran los prototipos hechos con *Figma*. Se puede acceder a todas las vistas desde el menú de navegación (superior o en barra lateral).

- Página de inicio (usuario anónimo). Se muestra un breve resumen de las funcionalidades de la aplicación.



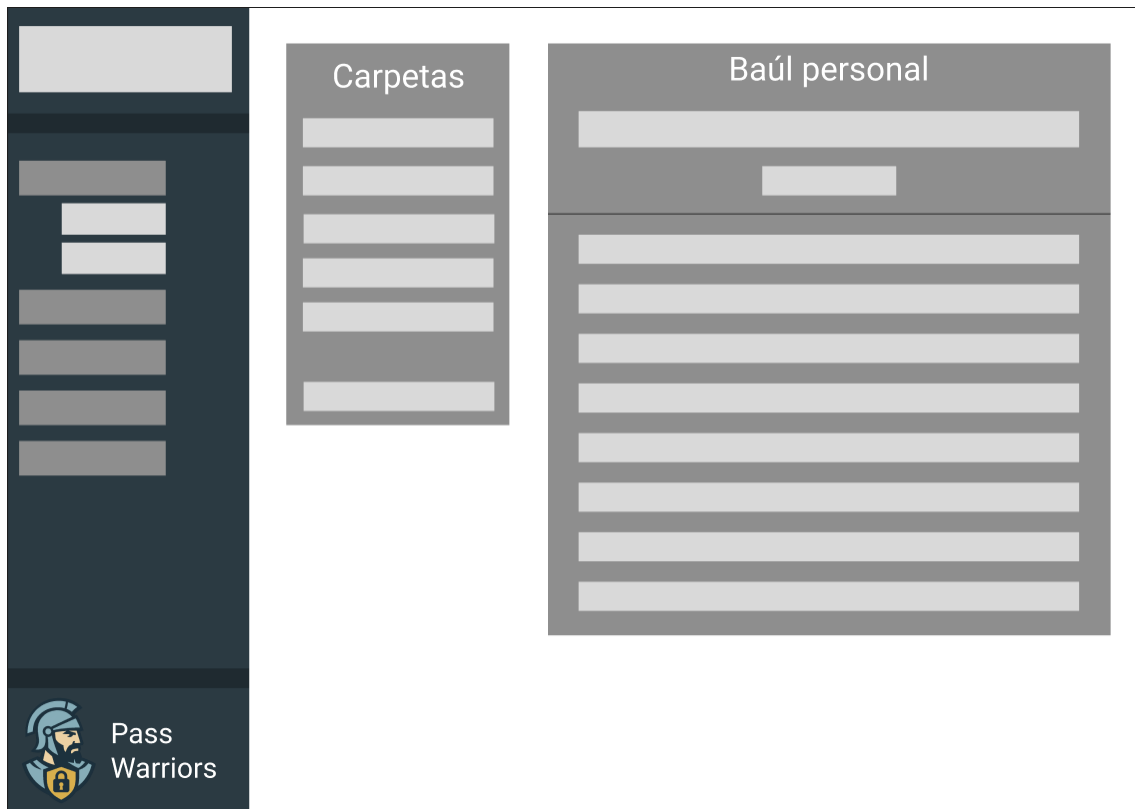
- Página de inicio de sesión o registro (usuario anónimo).

The image shows a web browser window with the 'Pass Warriors' logo in the top left corner. The main content area is titled 'Login/Registro' and features a form with four horizontal input fields stacked vertically. Below these fields is a single button. The entire form is centered within a light gray container.

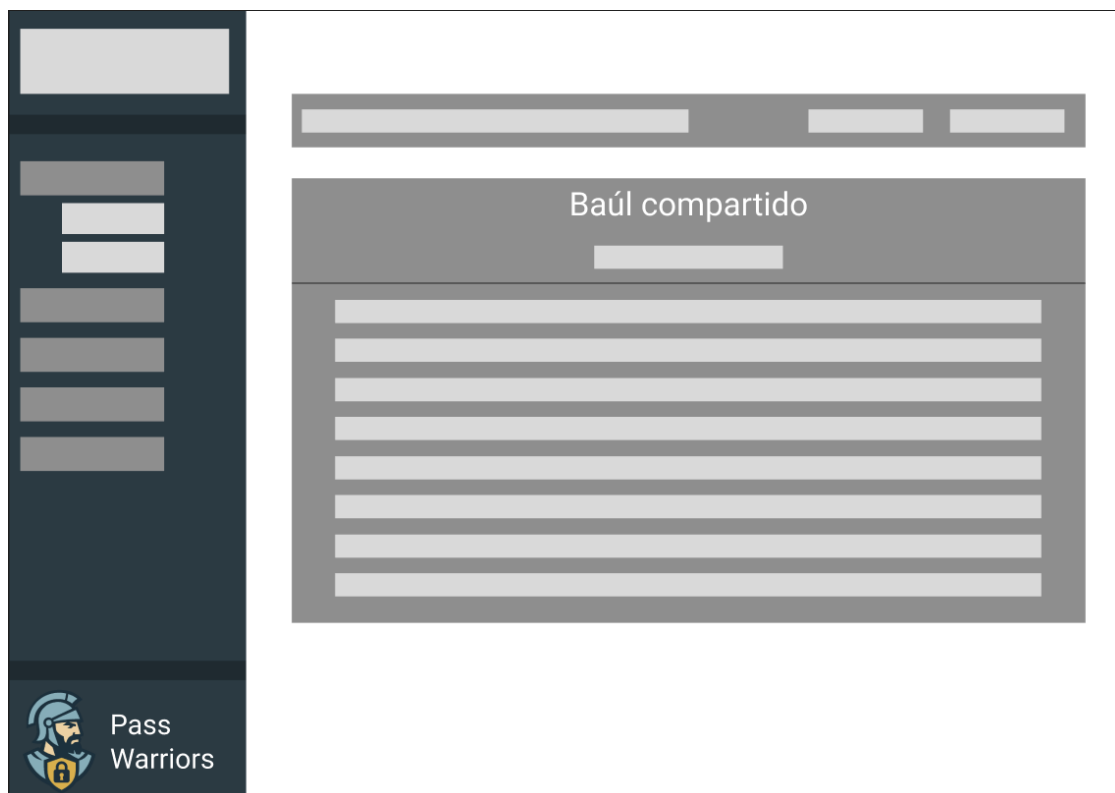
- Generador de contraseñas (usuario anónimo).

The image shows a web browser window with the 'Pass Warriors' logo in the top left corner. The main content area is titled 'Generador de contraseñas'. The form includes a large input field at the top, followed by a smaller input field. Below these is a horizontal line, and then another smaller input field. At the bottom, there is a large container with four smaller input fields arranged in a 2x2 grid.

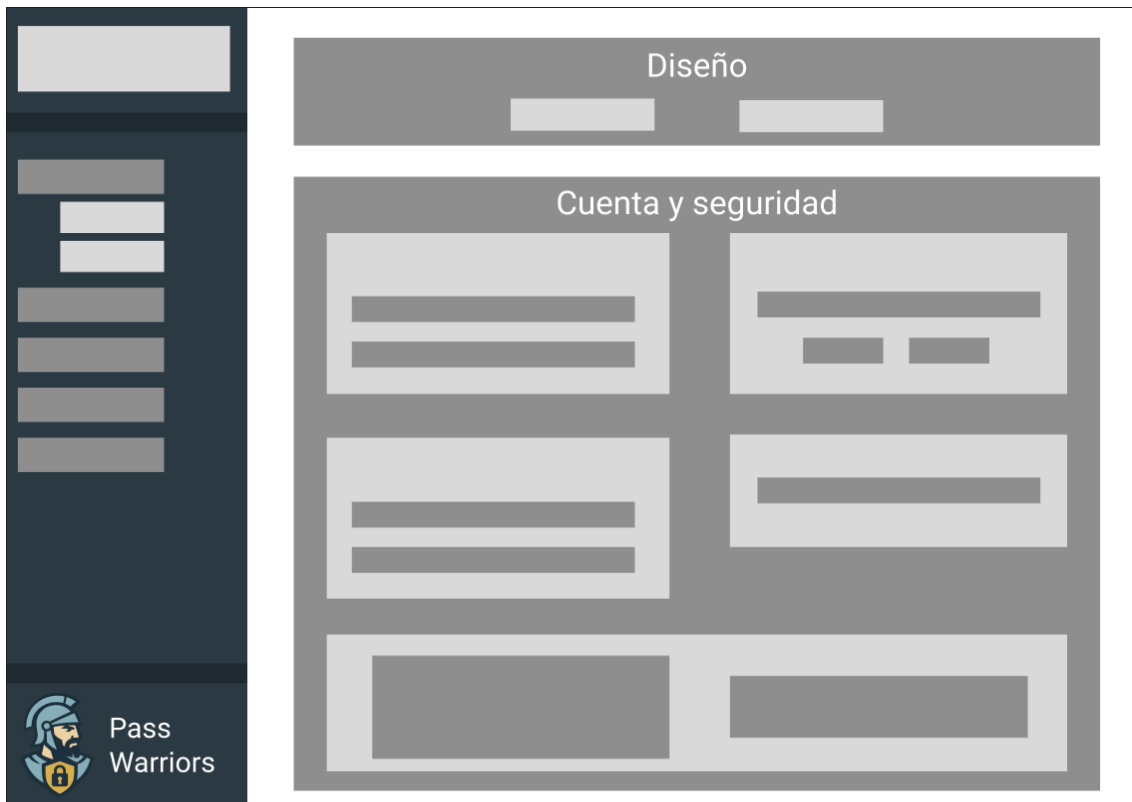
- Baúl personal (usuario autenticado).



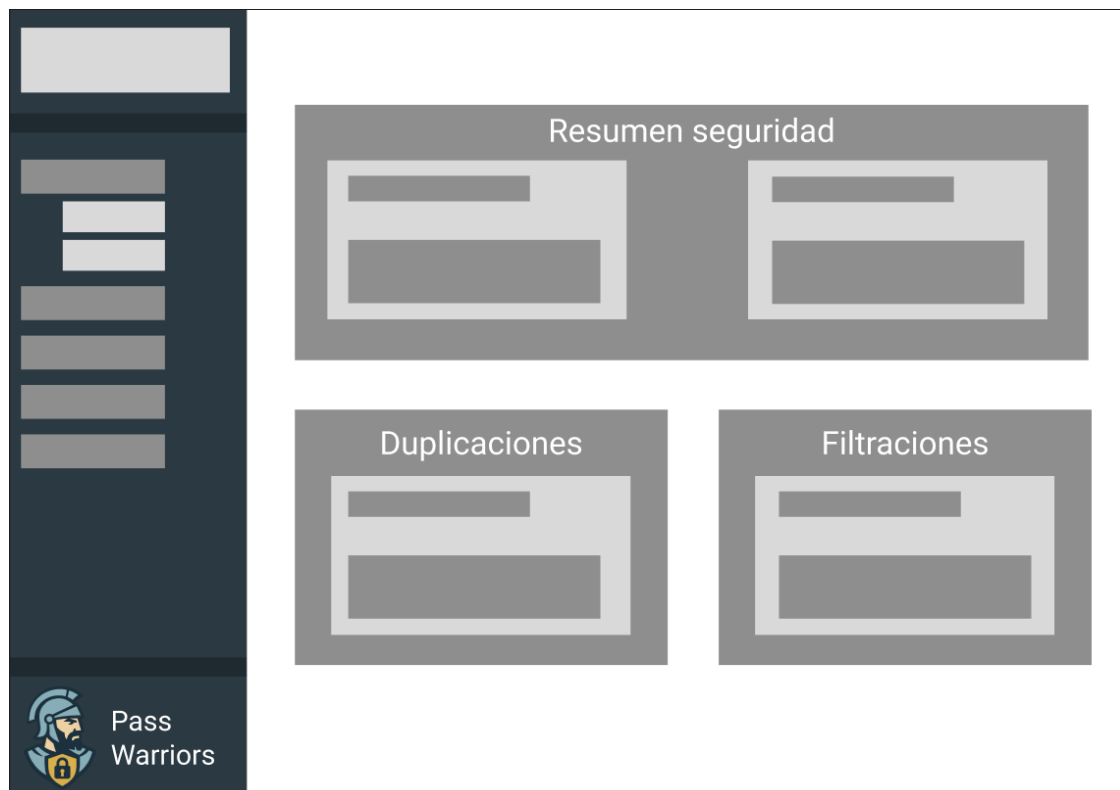
- Baúl compartido (usuario autenticado).



- Ajustes (usuario autenticado).



- Auditoría de contraseñas (usuario autenticado).



5.4. Implementación

Para el desarrollo del proyecto se han utilizado 2 entornos de desarrollo.

5.4.1. Backend

Para el desarrollo del *backend* se ha utilizado XAMPP y DBEngine como entornos de desarrollo.

XAMPP ofrece un servidor Apache que puede ejecutar código PHP. DBEngine permite crear y administrar distintas instancias de distintas bases de datos (en este caso MySQL 8.4.2). Como cliente de la base de datos se ha utilizado DBeaver.

Cómo editor de código se ha utilizado Visual Studio Code. También se ha utilizado Git como control de versiones y GitHub como repositorio remoto.

Se ha utilizado el modelo MVC y programación orientada a objetos. Cada ruta de la API (**vista**) posee un **controlador**, quién se encarga de la autenticación y validación de datos, y un **modelo**, que se encarga de realizar la conexión e interactuar con la base de datos.

A parte de los controladores y los modelos, se han creado utilidades extra para facilitar el desarrollo del proyecto:

- **Esquemas:** Permiten verificar los datos enviados por el cliente antes de realizar operaciones con la base de datos. Cada entidad tiene su propio esquema, de esta manera se facilita la validación de datos de cada entidad.


```

1 public static function validate(array $data): array {
2     $result = [
3         "data" => [],
4         "errors" => []
5     ];
6
7     self::validateName($data, $result);
8     self::validateFolder($data, $result);
9     self::validatePassword($data, $result);
10    self::validateUsername($data, $result);
11    self::validateUrls($data, $result);
12    self::validateNotes($data, $result);
13
14    return $result;
15 }
16
17 private static function validateName(array $data, array &$result): void {
18     $name = isset($data[PasswordsModel::COL_NAME]) ? trim($data[PasswordsModel::COL_NAME]) : null;
19
20     // Comprobar que existe el campo
21     if ($name === null) {
22         $result["errors"][] = "" . PasswordsModel::COL_NAME . " es obligatorio";
23         return;
24     }
25
26     // Comprobar formato
27     if (!preg_match(self::NAME_REGEX, $name))
28     {
29         $result["errors"][] = "" . PasswordsModel::COL_NAME . " debe tener una longitud entre 1 y 50
30         caracteres (ambos incluidos).
31         Se admite cualquier carácter. Los espacios sobrantes al principio y final del '"
32         . PasswordsModel::COL_NAME . " son eliminados";
33         return;
34     }
35
36     $result["data"]["name"] = $name;
37 }

```

Ilustración 10: Ejemplo del esquema de Password

- **Herramientas:** Proporcionan utilidades extra:
 - **Base de datos:** Permite realizar conexiones a una base de datos e interactuar con ella (recuperar, editar, añadir o eliminar información) mediante transacciones. Se ha utilizado PDO, que proporciona una capa de abstracción al conectarse con la base de datos y operar con ella.

- **Encriptación:** Proporciona funciones relacionadas con la encriptación. Hashear una cadena, comprobar una cadena con un hash, encriptar o desencriptar una cadena, generar tokens de sesión, códigos de recuperación, UUIDs...

```
1 public static function encrypt(string $string): string {
2     $ivLength = openssl_cipher_iv_length(CIPHER_ALGORITHM);
3     $iv = openssl_random_pseudo_bytes($ivLength);
4
5     $encryptedString = openssl_encrypt(
6         data: $string,
7         cipher_algo: CIPHER_ALGORITHM,
8         passphrase: ENCRYPTION_PASSPHRASE,
9         options: OPENSSL_RAW_DATA,
10        iv: $iv
11    );
12
13    return base64_encode("$encryptedString::$iv");
14 }
15
16
17 public static function decrypt(string $encryptedString): string|bool {
18     [ $encryptedData, $iv ] = explode(
19         "::$",
20         base64_decode($encryptedString),
21         2
22     );
23
24     return openssl_decrypt(
25         data: $encryptedData,
26         cipher_algo: CIPHER_ALGORITHM,
27         passphrase: ENCRYPTION_PASSPHRASE,
28         options: OPENSSL_RAW_DATA,
29         iv: $iv
30     );
31 }
32
33
34 public static function generateUUIDv4(): string {
35     $bytes = random_bytes(16);
36
37     // Versión 4
38     $bytes[6] = chr(
39         ord($bytes[6]) & 0x0f | 0x40
40     );
41     $bytes[8] = chr(
42         ord($bytes[8]) & 0x3f | 0x80
43     );
44
45     return vsprintf(
46         "%s%s-%s-%s-%s-%s",
47         str_split(
48             bin2hex($bytes),
49             4
50         )
51     );
52 }
```

Ilustración 11: Ejemplos de métodos de la clase Encrypt

- **Solicitud y Respuesta:** La clase “**Request**” contiene métodos y atributos relacionados con la petición HTTP como, por ejemplo, obtener el método HTTP, recuperar *user agent* del dispositivo, obtener el valor de una cookie, obtener el cuerpo de la petición... La clase “**Response**” contiene métodos y atributos relacionados con la respuesta HTTP, por ejemplo, establecer el código de respuesta HTTP, establecer o añadir los datos a enviar, añadir los errores encontrado durante la validación, establecer o eliminar cookies, mostrar el JSON correspondiente manteniendo la misma estructura en todas las rutas de la API...

```
1 public function showResponseAndExit(HttpStatusCode $httpCode): void {
2     http_response_code($httpCode→value);
3
4     header("X-Powered-By:");
5     header("Content-Type: application/json; charset=UTF-8");
6     header('Access-Control-Allow-Credentials: true');
7     header("Access-Control-Allow-Headers: Content-Type, Authorization, X-Requested-With");
8     header("Access-Control-Allow-Methods: GET, POST, OPTIONS, PATCH, DELETE");
9     header("Access-Control-Allow-Origin: " . ($_SERVER["HTTP_ORIGIN"] ?? "*"));
10    header("Cache-Control: no-store, no-cache, must-revalidate");
11    header("X-Content-Type-Options: nosniff");
12    header("Content-Security-Policy: default-src 'none'");
13    header("X-Frame-Options: DENY");
14
15    $hasErrors = count($this→errors) > 0;
16    $flags = JSON_UNESCAPED_UNICODE | JSON_UNESCAPED_SLASHES;
17
18    echo json_encode(
19        [
20            "service_name" ⇒ SERVICE_NAME,
21            "success" ⇒ !$hasErrors,
22            "data" ⇒ !$hasErrors ? $this→data : null,
23            "errors" ⇒ $hasErrors ? $this→errors : null
24        ],
25        $flags
26    );
27    exit;
28 }
```

Ilustración 12: Ejemplo de establecer cabeceras y mostrar el JSON en la clase Response

- **Códigos HTTP:** Enumerado con los códigos de respuesta HTTP que se utilizan en la API (200, 201, 401, 404...).
- **Métodos HTTP:** Enumerado con los métodos HTTP que se utilizan en la aplicación (GET, POST, PATCH, DELETE...).
- **Duración de sesión:** Enumerado con las posibles duraciones de sesión que el usuario puede elegir.

Se ha utilizado la librería externa [PHPGangsta/GoogleAuthenticator](#) para realizar la autenticación de doble factor mediante código temporal. Esta librería permite generar un secreto, crear el código QR correspondiente que puede ser leído por una aplicación de autenticación (Google Authenticator o Authy) y comprobar el código generado por la aplicación. La aplicación genera un código de 6 dígitos cada 30 segundos, añadiendo una capa de seguridad a la cuenta del usuario.

```
1  $ga = new PHPGangsta_GoogleAuthenticator();
2
3  // Generar secreto
4  $secret = $ga->createSecret(32);
5
6  // Generar código QR
7  $qrUrl = $ga->getQRCodeGoogleUrl('Pass Warriors', $secret);
8
9  // Obtener código temporal
10 $code = $ga->getCode($secret);
11
12 // Comparar código temporal con el introducido por el usuario
13 $code === $userCode;
```

Ilustración 13: Ejemplo de cómo utilizar la librería Google Authenticator

Para realizar la autenticación del usuario, se ha utilizado cookies de sesión que almacenan el token de sesión correspondiente. La cookie no es accesible mediante JavaScript, proporcionando seguridad a la sesión. Además, solo se pueden compartir mediante conexiones HTTPS.

Esquema de la estructura de archivos y carpetas

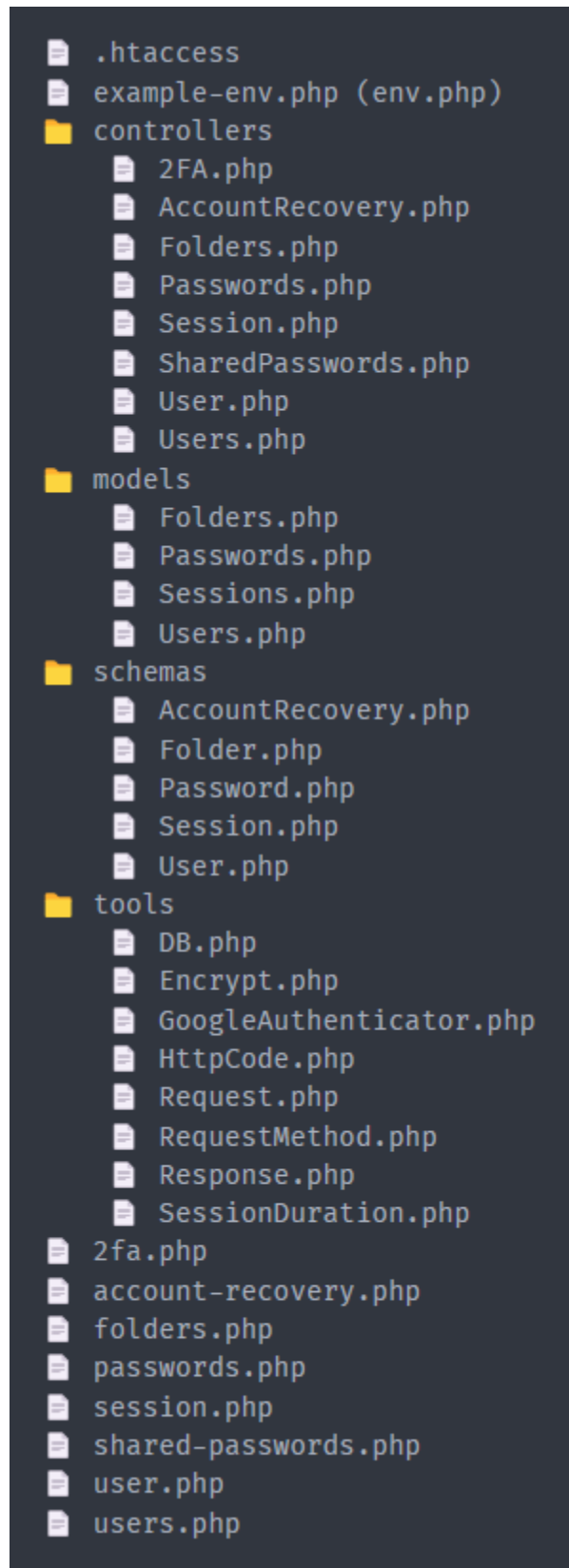


Ilustración 14: Estructura de archivos (backend)

Encriptación

Se han utilizado distintos métodos de encriptación dependiendo de la finalidad de la contraseña:

- **Cifrado unidireccional:** Cifrar elementos sin posibilidad de recuperar su valor original, pero sí compararlo.

PHP ofrece una función nativa “*password_hash()*” que cifra la contraseña mediante un algoritmo de cifrado. El resultado de la función es irreversible (no se puede obtener la cadena original) y genera distintas salidas para una misma entrada, fundamental para proporcionar privacidad al elemento cifrado.

En este caso se ha utilizado el algoritmo “*PASSWORD_BCRYPT*”. Es un algoritmo de cifrado de alta velocidad y de alto nivel de seguridad muy utilizado en la actualidad.

Para poder verificar una cadena con un hash, se utiliza la función “*password_verify(<cadena_comprobar>, <cadena_hasheada>)*”.

En la aplicación se utiliza este método para encriptar la contraseña maestra, que es la contraseña con la que el usuario accede a la aplicación.

- **Cifrado unidireccional determinista:** Cifrar elementos sin posibilidad de recuperar su valor original, pero el cifrado debe retornar siempre un mismo valor a una misma entrada. Esto permite realizar búsquedas de elementos sin comprometer su seguridad.

Para este método se ha utilizado el algoritmo SHA-256. Este algoritmo genera una cadena 64 caracteres hexadecimales. Además, para una misma entrada, genera la misma salida.

En la aplicación se utiliza este método para almacenar el token de sesión.

- **Cifrado reversible:** Cifrar elementos, pero con la posibilidad de recuperar su valor original. Este método se ha utilizado para almacenar las contraseñas del usuario, el código de recuperación y el secreto del doble factor de autenticación ya que son valores que se deben poder recuperar.

Para este método se han utilizado las funciones nativas de PHP “*openssl_encrypt*” y “*openssl_decrypt*”. Como algoritmo de cifrado se ha utilizado el “*AES-256-CTR*” que es un algoritmo de cifrado simétrico que utiliza una clave de 256bits. Utiliza el modo CTR, tomando un vector de inicialización de 128 bits.

5.4.2. Frontend

Para el desarrollo del *frontend* se ha utilizado el entorno de ejecución **Node JS**, con el administrador de paquetes **PNPM** para la implementación del *framework* **VUE 3**.

Node JS permite la ejecución de código JavaScript y PNPM permite instalar las dependencias y librerías necesarias para el desarrollo del proyecto. Estas dependencias se especifican en el archivo “*package.json*”.

Se ha utilizado un modelo basado en Rutas, Vistas y Componentes (similar a MVC). Las **Rutas** determinan las vistas que el usuario puede acceder. La **Vista** contiene los componentes que serán renderizados cuando el usuario acceda a la ruta. Los **Componentes** son porciones de código, HTML y estilos con funcionalidades específicas que se puede reutilizar en distintas partes de la aplicación.

VUE 3 permite la creación de aplicaciones **SPA** gracias a **VUE Router**, una dependencia que permite controlar las vistas de la aplicación mediante un enrutador. De esta manera se dispone de un único archivo HTML y, mediante JavaScript, VUE renderiza y muestra los componentes necesarios en cada vista consiguiendo una navegación fluida y un mejor rendimiento para el usuario.

VUE 3 también se encarga de optimizar recursos estáticos, como imágenes, CSS, fuentes, etc. Para mejorar el rendimiento de la aplicación.

Otra dependencia importante es **Pinia**. Esta dependencia permite crear almacenes (*stores*) que almacenan estados y permiten sincronizar componentes, que son independientes entre sí, de una manera sencilla.

La conexión con el *backend* se ha realizado mediante solicitudes HTTP utilizando **fetch**, una API de JavaScript que permite realizar comunicaciones asíncronas.

A parte del enrutador, vistas y componentes, se han creado utilidades extra para facilitar el desarrollo del proyecto:

- **Portapapeles.** Proporciona la utilidad de copiar texto en el portapapeles del usuario utilizando la API de JavaScript “Clipboard” o, en caso de que no esté disponible, mediante la acción de seleccionar y copiar texto.
- **Interacción con el backend.** Son distintas clases que permiten interaccionar con el Backend de una manera sencilla, pedir información, enviar recursos a crear, editar o eliminar, validaciones de datos... de las distintas entidades de la aplicación
- **Generador de contraseñas.** Contiene clases que se utilizan en el generador, por ejemplo, guardar o cargar la configuración personalizada, obtener los caracteres disponibles, ver, almacenar o eliminar una contraseña en el historial...
- **Tema:** Permite cambiar o recuperar el tema actual de la aplicación, cambiar de lado la barra lateral...
- **Contraseña:** Además de realizar peticiones HTTP al *backend*, contiene el algoritmo que se utiliza al realizar la revisión de seguridad de las contraseñas, comprobar filtraciones con la API de *HaveIBeenPwned*....


```

1  static async getSecurityPasswords() {
2      const groupedPasswords = await PasswordTools.getPasswordsGroupByPassword()
3      const securityPasswords = {
4          1: [],
5          2: [],
6          3: [],
7      }
8
9      for (const item of groupedPasswords) {
10         const result = this.getSecurityPasswordRating(item[0])
11
12         item.push(result)
13         securityPasswords[result.rating].push(item)
14     }
15
16     return Object.entries(securityPasswords)
17 }

```

Ilustración 15: Revisión de seguridad de contraseñas

```

1  static getSecurityPasswordRating(password) {
2      const result = {
3          rating: 0,
4          length: 1,
5          hasLower: false,
6          hasUpper: false,
7          hasNumber: false,
8          hasSpecial: false,
9          hasRepeatedChar: false,
10         hasCharSecuencias: false,
11     }
12
13     // Longitud
14     if (password.length > 7) {
15         result.rating++
16         result.length = 2
17     }
18     if (password.length > 14) {
19         result.rating++
20         result.length = 3
21     }
22
23     // Tipos de caracteres
24     if (password.match(/[a-z]/g)?.length > 1) {
25         result.rating++
26         result.hasLower = true
27     }
28     if (password.match(/[A-Z]/g)?.length > 1) {
29         result.rating++
30         result.hasUpper = true
31     }
32     if (password.match(/[0-9]/g)?.length > 1) {
33         result.rating++
34         result.hasNumber = true
35     }
36     if (password.match(/^[a-zA-Z0-9]/g)?.length > 1) {
37         result.rating++
38         result.hasSpecial = true
39     }
40
41     // Caracteres repetidos o secuencias
42     if (this.hasDuplicatedChars(password)) {
43         result.rating--
44         result.hasRepeatedChar = true
45     }
46     if (this.hasCharSecuencias(password)) {
47         result.rating--
48         result.hasCharSecuencias = true
49     }
50
51     // Ajustar rating a 1, 2 o 3
52     result.rating = Math.max(Math.floor(result.rating / 2), 1)
53
54     return result
55 }

```

Ilustración 16: Algoritmo de cálculo de seguridad

```

1  static hasDuplicatedChars(password) {
2      const chars = {}
3
4      for (let i = 0; i < password.length; i++) {
5          const char = password[i]
6
7          if (!chars[char]) {
8              chars[char] = 1
9              continue
10         }
11
12         chars[char]++
13
14         if (chars[char] > 2) {
15             return true
16         }
17     }
18
19     return false
20 }
21
22 static hasCharSecuencias(password) {
23     if (password.length < 3) return false
24
25     let ascCounter = 1,
26         descCounter = 1
27
28     for (let i = 0; i < password.length - 1; i++) {
29         const charCodeDiff = password.charCodeAt(i + 1) - password.charCodeAt(i)
30
31         if (charCodeDiff === 1) {
32             ascCounter++
33             descCounter = 1
34         } else if (charCodeDiff === -1) {
35             descCounter++
36             ascCounter = 1
37         } else {
38             ascCounter = 1
39             descCounter = 1
40         }
41
42         if (ascCounter ≥ 3 || descCounter ≥ 3) return true
43     }
44
45     return false
46 }

```

Ilustración 17: Métodos de apoyo del algoritmo

Esquema de la estructura de archivos

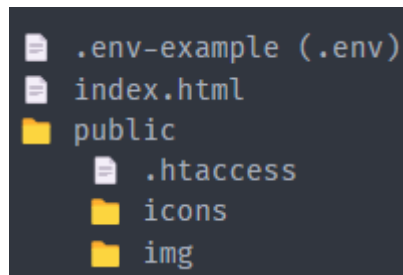


Ilustración 18: .env, index y public

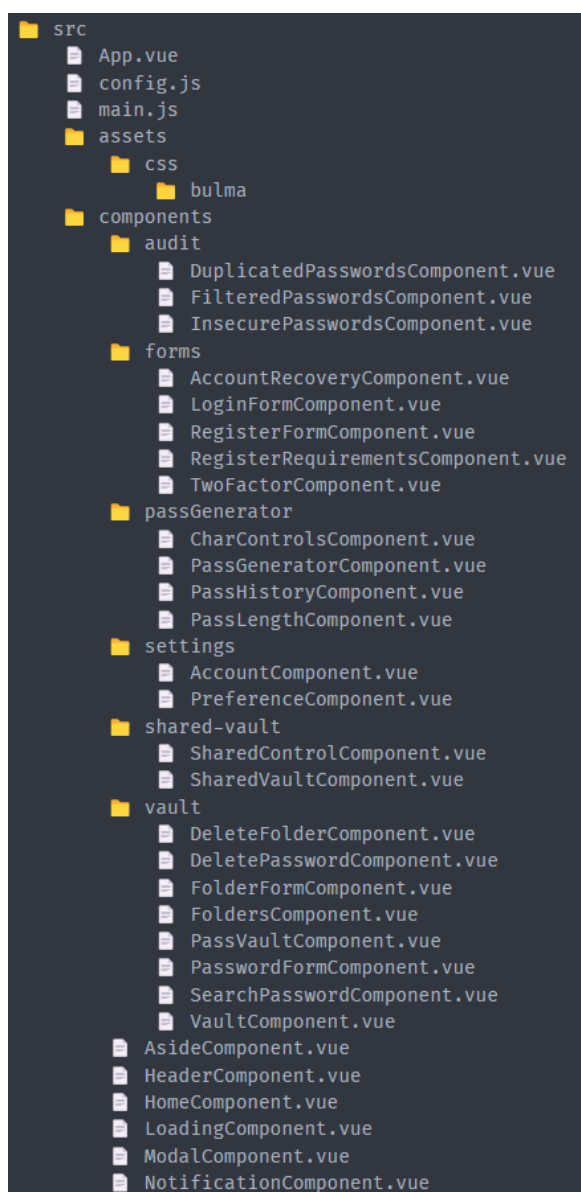


Ilustración 19: Carpeta src

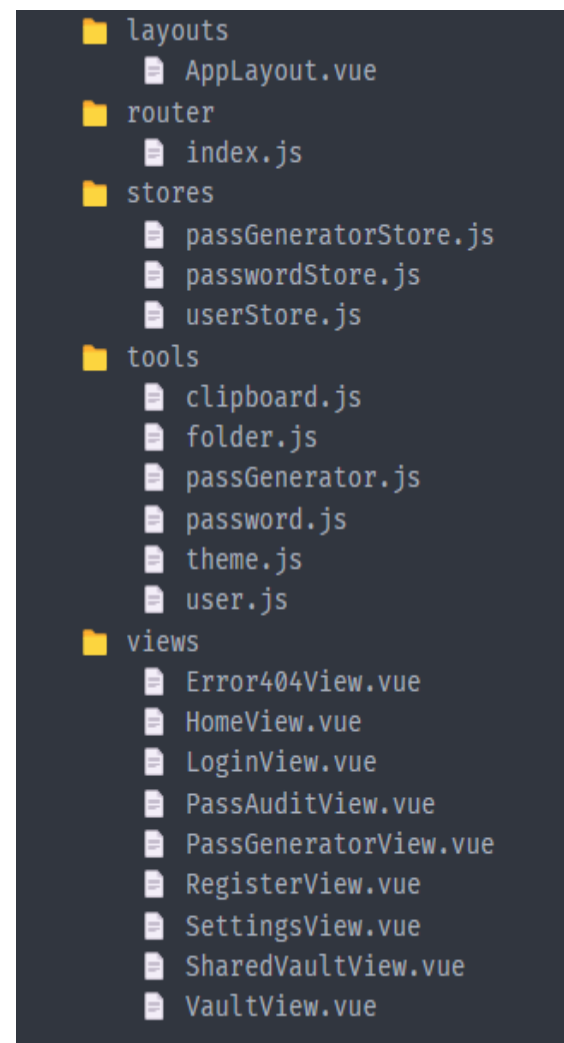


Ilustración 20: Continuación carpeta src

5.5. Pruebas

Se han realizado pruebas tanto del *backend* como del *frontend* durante todo el desarrollo, comprobando que las funcionalidades implementadas cumpliesen con los requisitos.

Se han realizado pruebas de **creación de usuarios**, introduciendo valores válidos y no válidos. Cuando todos los valores eran válidos, se ha creado el usuario correctamente en la base de datos, se ha mostrado el JSON con la información del usuario creado y se ha creado la cookie de sesión con duración de 1 hora.

Cuando los valores eran incorrectos, se muestran correctamente los mensajes de error de los datos con errores.

Se ha comprobado a **iniciar sesión**, introduciendo los datos de un usuario existente, y se ha probado a iniciar sesiones de diferentes duraciones. Se ha creado correctamente la cookie correspondiente en cada intento.

También se ha comprobado a introducir un usuario válido pero una contraseña incorrecta y viceversa, mostrando en ambos casos el mensaje de error correspondiente. También se ha intentado ejecutar SQL Injection, pero no se ha conseguido obtener acceso.

Se ha comprobado el correcto funcionamiento de la **verificación de doble factor**, funcionando correctamente. Al introducir credenciales válidas se muestra el formulario de 2FA. Se ha probado a introducir códigos incorrectos o caducados, mostrando correctamente el mensaje de error y ha introducir el código correcto, iniciando sesión y creando la cookie de sesión correspondiente.

Por último, relacionado con sesiones, se ha comprobado a **acceder** a la aplicación con una **sesión caducada**, no permitiendo entrar y solicitando el usuario y la contraseña de nuevo.

Se ha probado **a crear carpetas y elementos**, tanto privados como compartidos, con datos correctos y se ha comprobado que se han creado correctamente en la base de datos y se muestra la información resultante al usuario.

Se han realizado pruebas con valores incorrectos al crear una carpeta o elemento, mostrando correctamente el error correspondiente.

Se ha probado el **algoritmo de cálculo de seguridad** con numerosas cadenas, obteniendo resultados bastante convincentes. También se ha comprobado que la búsqueda de contraseñas duplicadas funcionase correctamente. También se ha probado a buscar contraseñas en filtraciones de datos mediante la API de *HaveIBeenPwned*, obteniendo resultados esperados.

Por último, se ha comprobado el **diseño en distintos dispositivos**, adaptándose correctamente a la pantalla. También se ha comprobado que la personalización del tema y de la barra lateral funcionase correctamente.

6. Manuales

6.1. Manual de usuario

6.1.1. Objetivos

El objetivo de este manual es enseñar al usuario como utilizar las distintas funcionalidades que ofrece la aplicación.

6.1.2. Estructura principal

Todas las páginas de la aplicación tienen una estructura similar, dependiendo del tipo de usuario y dispositivo:

Usuarios anónimos



Ilustración 21: Estructura

1. Barra superior: Contiene el menú de navegación.
2. Contenido. Contiene el contenido de la página.

Usuarios autenticados



Ilustración 22: Estructura usuarios autenticados

1. Barra lateral: Contiene el menú de navegación y el nombre del usuario autenticado.
2. Información del usuario autenticado.
3. Contenido de la página.

Dispositivos móviles

En dispositivos con pantalla pequeña, la estructura es igual tanto para usuarios autenticados como anónimos.

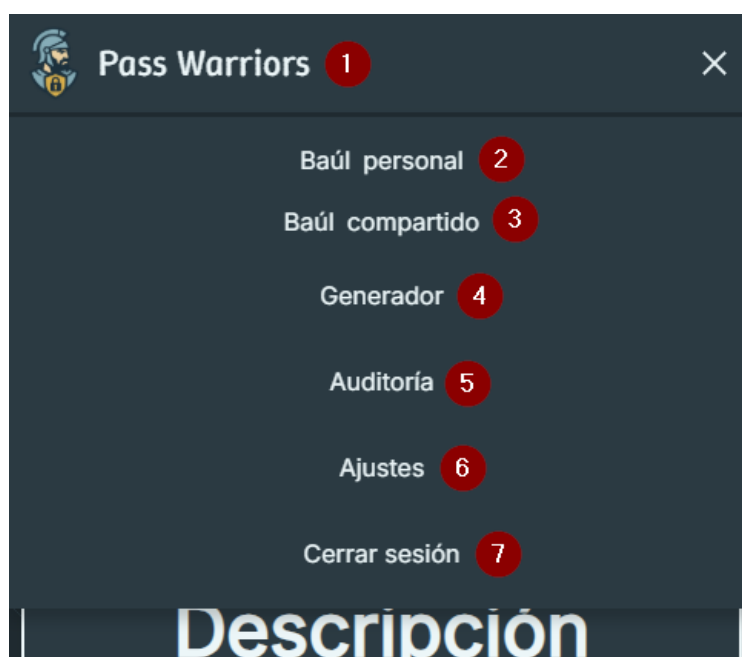
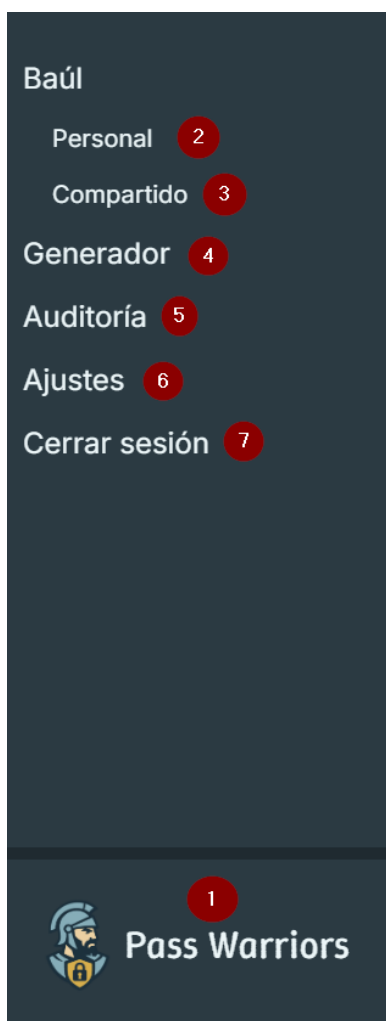


1. Barra superior, contiene el menú de navegación.
2. Muestra u oculta el menú de navegación.
3. Contenido de la página.

6.1.3. Navegación

Para navegar entre las distintas partes de la aplicación, existe un menú de navegación. Clicando en cada sección, podrás acceder a todas las funcionalidades.

En dispositivos con pantalla reducida, se debe clicar en el icono de 3 barras horizontales para acceder al menú de navegación. Pulsar de nuevo en la X para cerrarlo



1. Acceder a la página de inicio
2. Acceder al baúl personal
3. Acceder al baúl compartido
4. Acceder al generador de contraseñas
5. Acceder a la auditoría de seguridad
6. Acceder a los ajustes (preferencias y cuenta)
7. Cierra la sesión del usuario

6.1.4. Registro e inicio de sesión

Registro

Para darse de alta en la aplicación, se debe registrar, para ello acceda desde el menú de navegación a **Registrarse**.

The screenshot shows a registration form titled "Registrarse" in a dark-themed application. The form includes the following fields and elements:

- Nombre de usuario:** A text input field with a red box around the label.
- Nombre:** A text input field with a red box around the label.
- Contraseña:** A password input field with a red box around the label and a toggle icon.
- Confirmar contraseña:** A password input field with a red box around the label and a toggle icon.
- Registrarse:** A button at the bottom of the form with a red box around it.
- Mostrar requisitos:** A link below the password fields with a red box around it.
- ¿Ya tiene cuenta? iniciar sesión:** A link below the "Mostrar requisitos" link.

The top navigation bar contains three links: "Generador", "Registrarse" (highlighted with a red box), and "Iniciar sesión".

Ilustración 23: Registro

Debe rellenar el formulario. Los requisitos son los siguientes (se pueden visualizar desde el propio formulario clicando en **Mostrar requisitos**, en la parte inferior):

- **Nombre de usuario:** Debe tener una longitud entre **1** y **30** caracteres (ambos incluidos). Sólo se admiten **letras** del alfabeto inglés, tanto **minúsculas** como **mayúsculas**, y **barra baja** (_). No puede comenzar directamente con una barra baja o con un número. No se diferencian las mayúsculas de minúsculas.

No puede haber **dos nombres** de usuario **iguales**, es decir, si un usuario tiene registrado ese nombre de usuario, no se puede utilizar.

Se utiliza para iniciar sesión.

- **Nombre:** Debe tener una longitud entre **1** y **50** caracteres (ambos incluidos). Se admite cualquier carácter.
- **Contraseña maestra:** Debe tener una longitud entre **8** y **50** caracteres. Debe contener al menos una letra **minúscula** y una letra **mayúscula** (alfabeto inglés), un **número** y alguno de los siguientes símbolos especiales _- ,;!.@*&\#%+\$/. No se admiten otros caracteres.

Se utiliza para iniciar sesión.

Una vez rellenado los datos cumpliendo los requisitos, clique en **Registrarse**. Si se ha registrado con éxito, se le redirigirá al baúl personal. En caso contrario, se mostrará el correspondiente mensaje de error.

Se le mostrará el código de recuperación de la cuenta. Este código debe apuntarlo o almacenarlo en un lugar seguro. Le permitirá recuperar la cuenta en caso de olvidar la contraseña maestra (se puede volver a ver en los ajustes).

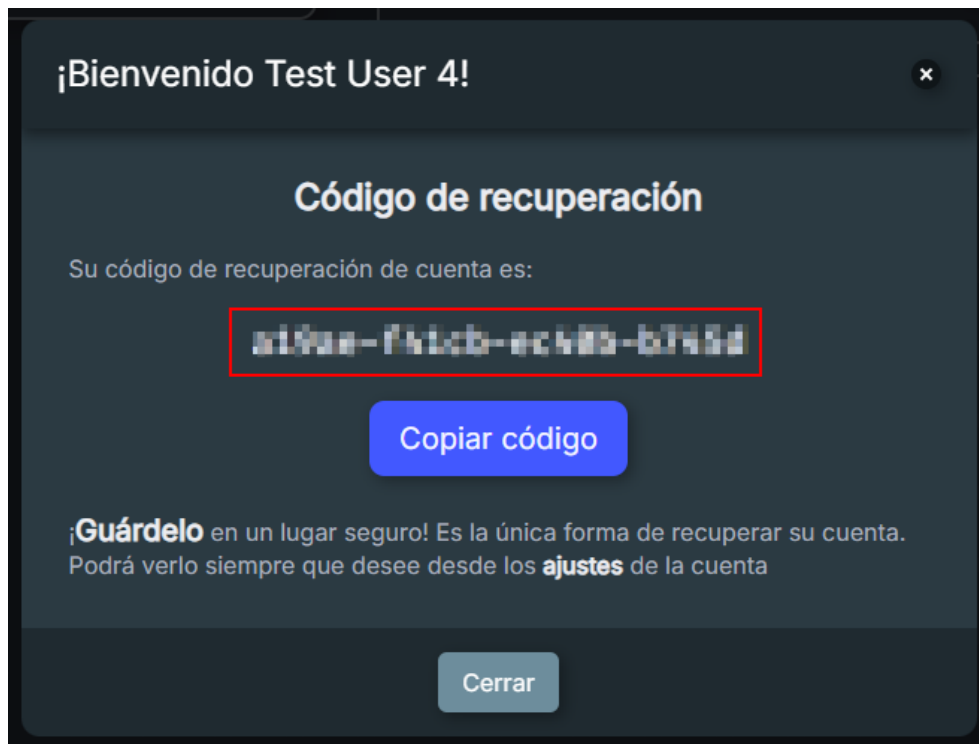


Ilustración 24: Código de recuperación

Inicio de sesión

Si ya se ha registrado en la aplicación, puede iniciar sesión con el nombre de usuario y la contraseña utilizadas en el registro. Para ello, acceda al apartado **Iniciar sesión** en el menú de navegación.

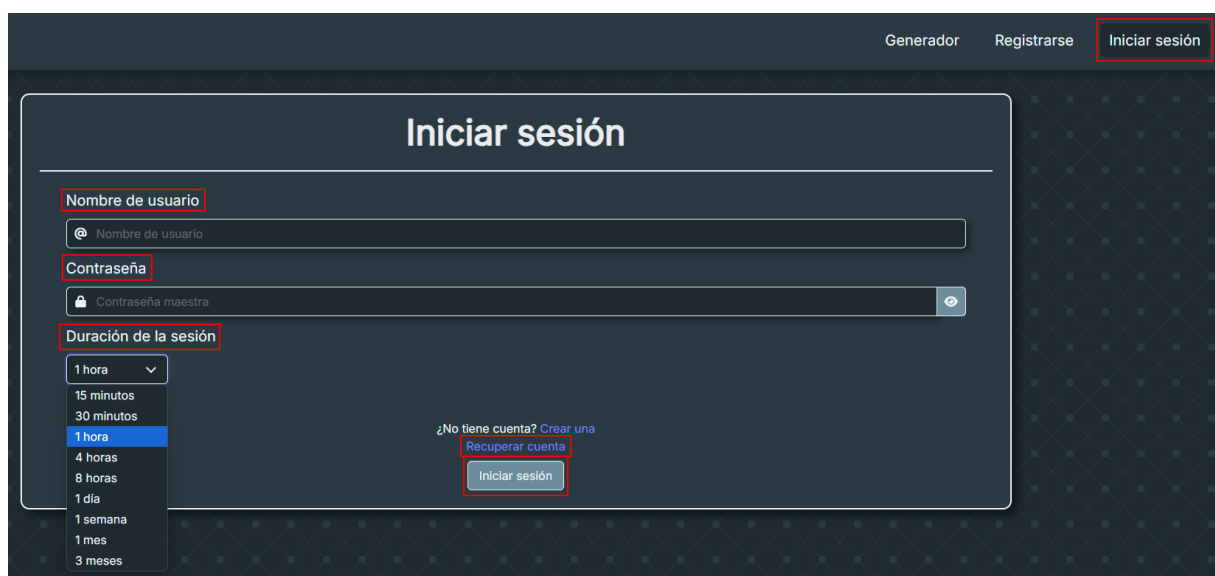


Ilustración 25: Inicio de sesión

Introduzca el nombre de usuario, la contraseña y una duración de sesión (por defecto es 1 hora). Una vez rellenado el formulario, pulse en Iniciar sesión. Si las credenciales son correctas, será redirigido al baúl personal. Si no, se mostrará el error correspondiente.

Si no puede acceder, puede clicar en **Recuperar cuenta**.

Recuperación de cuenta

Al clicar en Recuperar cuenta, se mostrará un nuevo formulario, deberá introducir el nombre de usuario, el código de recuperación y la nueva contraseña.

El formulario de recuperación de cuenta se presenta en un modal con un fondo oscuro. En la parte superior izquierda, el título "Recuperar cuenta" está en un color claro, y a la derecha hay un botón de cierre con una 'X'. El formulario contiene cuatro campos de entrada, cada uno con un icono a la izquierda y un placeholder: "Nombre de usuario" (icono @), "Código de recuperación" (icono escudo), "Nueva contraseña" (icono candado) y "Confirmar contraseña" (icono candado). Los campos de contraseña tienen un botón de visibilidad (ojo) a la derecha. En la parte inferior del modal, hay un botón rectangular con el texto "Recuperar".

Recuperar cuenta

Nombre de usuario

@ Nombre de usuario

Código de recuperación

Escudo Código

Nueva contraseña

Candado Contraseña

Confirmar contraseña

Candado Contraseña

Recuperar

Ilustración 26: Recuperación de cuenta

Si el nombre de usuario y el código de recuperación son correctos y la nueva contraseña cumple los requisitos, accederá de nuevo a la aplicación. Se generará un código nuevo de recuperación y se invalidará el último.

Cerrar sesión

Haga clic en Cerrar sesión en el menú de navegación para cerrar la sesión manualmente, sin esperar que caduque la sesión.

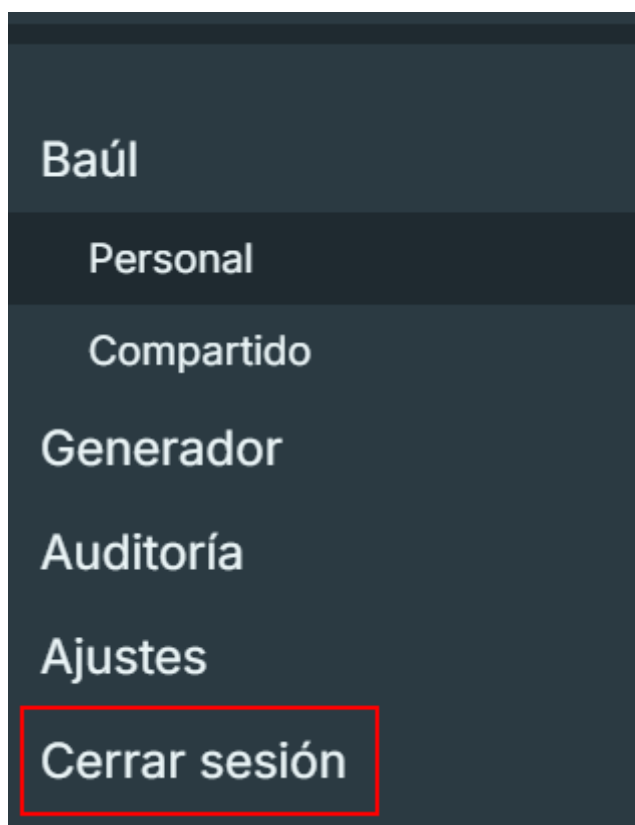


Ilustración 27: Cierre de sesión

6.1.5. Generador de contraseñas

En esta página, accesible tanto por usuarios autenticados como anónimos, podrá generar contraseñas aleatorias y seguras. Para acceder, pulse en Generador en el menú de navegación.

Puede personalizar la contraseña:

- Longitud: Número de caracteres de la contraseña.
- Caracteres mínimos: Cantidad de caracteres que desea que contenga la contraseña como mínimo (desde 0 hasta 10).
 - Minúsculas
 - Mayúsculas
 - Números
 - Caracteres especiales

Si no se selecciona ningún carácter (todos han sido puestos a 0), se activarán las minúsculas.

Clique en **Generar contraseña** para que aparezca la contraseña generada en la parte superior.

Ilustración 28: Generador de contraseñas

En el botón del historial (1), podrá visualizar las últimas 75 contraseñas generadas en el dispositivo. Si clicas en Vaciar, se limpiará el historial.

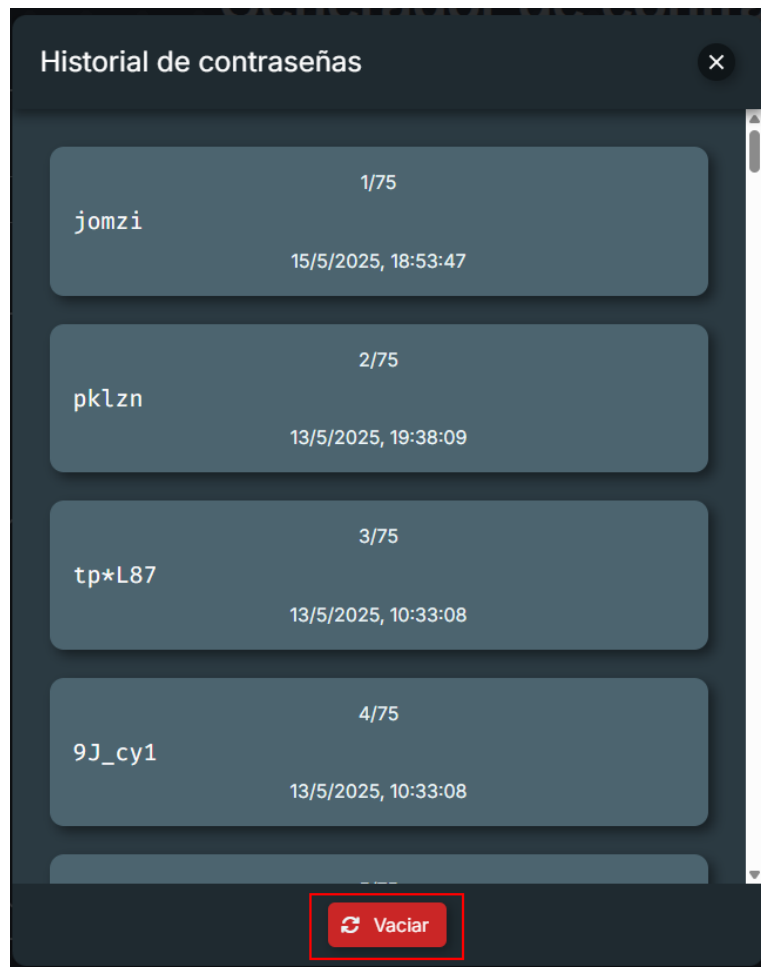


Ilustración 29: Historial de contraseñas generadas

6.1.6. Baúl personal

Almacena los elementos privados. Para ello, puede acceder desde Personal (en Baúl) desde el menú de navegación o al iniciar sesión.

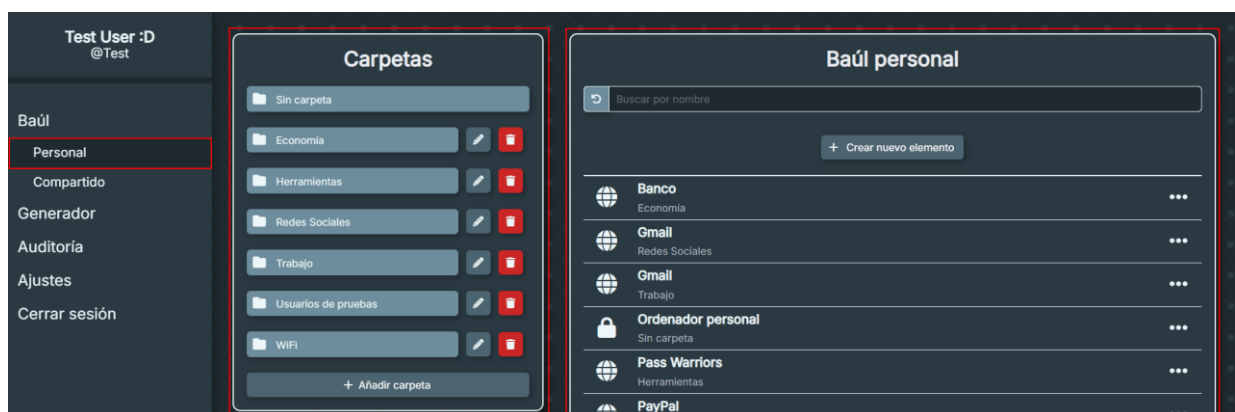


Ilustración 30: Baúl personal

Podrá crear las carpetas que desee, clicando en Añadir carpeta. No se pueden repetir los nombres de carpeta. Puede editar o eliminar (se mostrará confirmación) cualquier carpeta (excepto Sin carpeta), clicando en el lápiz o en la papelera, respectivamente.

Puede filtrar los elementos por carpeta clicando cada nombre de carpetas, pudiendo seleccionar 0 (se muestran todas los elementos), 1 carpeta o varias simultáneamente.

El baúl personal se divide en dos apartados. En la parte superior se encuentra el buscador. Podrá buscar elementos mediante búsqueda parcial por el nombre, es decir, que el nombre del elemento contenga la cadena introducida. Si clicla en la flecha de la izquierda, limpiará el buscador.

Para añadir un nuevo elemento, clique en Crear nuevo elemento. Se mostrará un formulario.

Desde el propio formulario se puede acceder al generador de contraseñas.

Añadir nuevo elemento

Carpeta

Sin carpeta

Nombre del elemento*

Nombre

Nombre de usuario

@ Nombre de usuario

Contraseña

Contraseña

Abrir generador

URLs

URL 1

Notas

Añadir notas

Añadir elemento

Ilustración 31: Creación de elementos

El nombre del elemento es obligatorio, el resto de campos son opcionales. Puede añadir desde 0 hasta 5 links. Pulse en Añadir elemento para crearlo.

Los elementos que tienen al menos un enlace almacenado se muestran con el icono de un globo terráqueo. Los que no, muestran un candado.



Ilustración 32: Iconos contraseñas

Mediante los 3 puntos situados a la derecha de cada elemento, podrá acceder a las acciones rápidas:

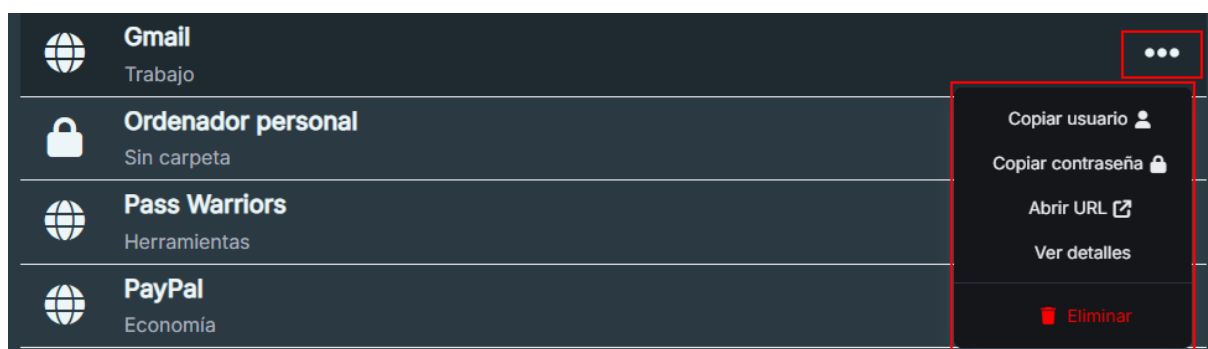


Ilustración 33: Opciones rápidas

Puede copiar el nombre de usuario (si hay uno almacenado), copiar la contraseña (si hay una almacenada), abrir el primer enlace en una nueva pestaña (si hay uno almacenado), ver los detalles para editarlos o eliminar la contraseña. Se mostrará confirmación.

El formulario de edición es el mismo que el de creación. Para guardar los datos editados, clique en Editar elemento.

6.1.7 Baúl compartido

Para acceder a esta sección, clique en Compartido (sección Baúl) en el menú de navegación.

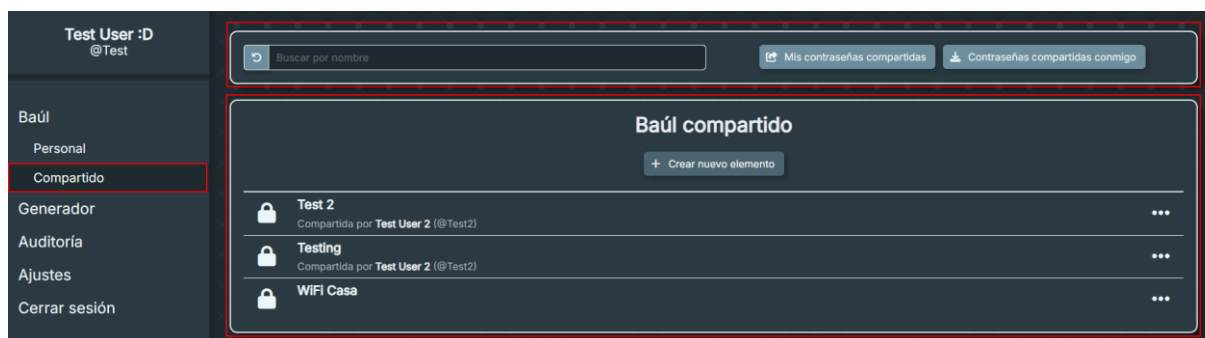


Ilustración 34: Baúl compartido

El buscador, situado en la parte superior, funciona de la misma forma que en el baúl personal. Los botones de la derecha, Mis contraseñas compartidas y Contraseñas compartidas conmigo, permiten filtrar los elementos creados por el usuario o que han sido compartidos con él. Sólo se puede seleccionar uno al mismo tiempo.

Para añadir un nuevo elemento, clique en Crear nuevo elemento. Se mostrará un formulario similar al del baúl personal, pero con una nueva sección.

Añadir nuevo elemento

Usuarios a compartir*

Test

Test User 2 (@Test2) Test User 4 (@Test_4)

Compartido con

Test User 3 (@Test3)

Nombre del elemento*

Nombre

Nombre de usuario

@ Nombre de usuario

Contraseña

Contraseña

[Abrir generador](#)

Añadir elemento

Ilustración 35: Crear elemento

En el buscador introduzca el nombre de usuario (con el que se inicia sesión) del usuario al que quiere compartir la contraseña. Cuando aparezcan los resultados, clique en el usuario. Se moverá a la sección Compartido con (si clic de nuevo, se eliminará de esta sección). Puede añadir los usuarios que desee para compartir. Para crear y compartir el elemento clique en Añadir elemento.

6.1.8. Auditoría de seguridad

En esta sección podrá realizar auditorías de seguridad de sus contraseñas personales guardadas. Para acceder, clique en Auditoría en el menú de navegación.



Ilustración 36: Auditoría de seguridad

Clique en el botón de la sección que desee ejecutar.

Seguridad de contraseñas

Muestra un informe con un resumen de seguridad de las contraseñas. Se tienen en cuenta diversos factores como la longitud de la contraseña, contiene minúsculas, mayúsculas, números o caracteres especiales, caracteres repetidos y secuencias (ascendentes o descendentes) de caracteres.

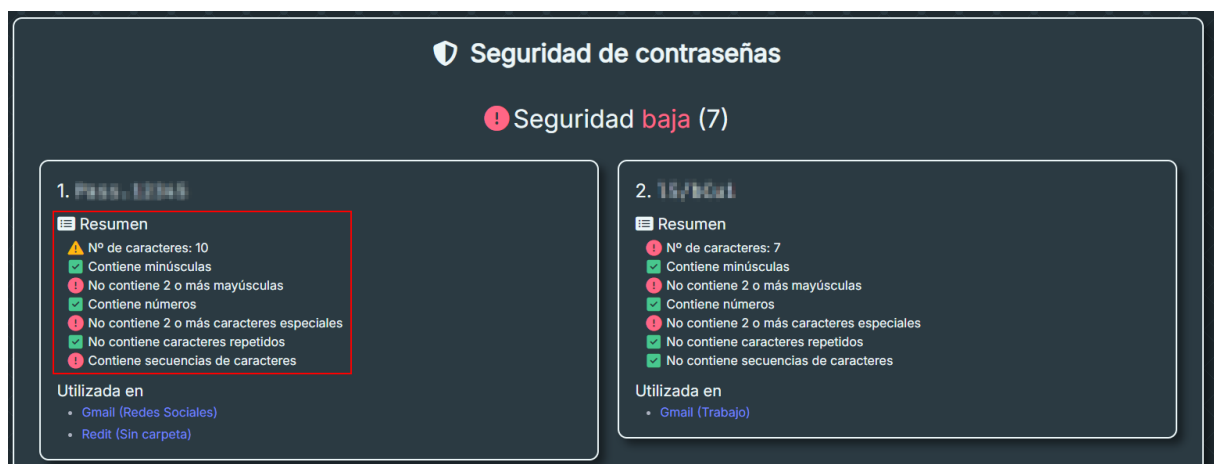


Ilustración 37: Resumen de seguridad

Los resultados se dividen entre seguridad **baja** (revisión urgente), **media** (revisión recomendada) y **alta**.

Contraseñas repetidas

Se mostrará las contraseñas que se utilizan en varios elementos, pudiendo clicar en el elemento para editar su valor.

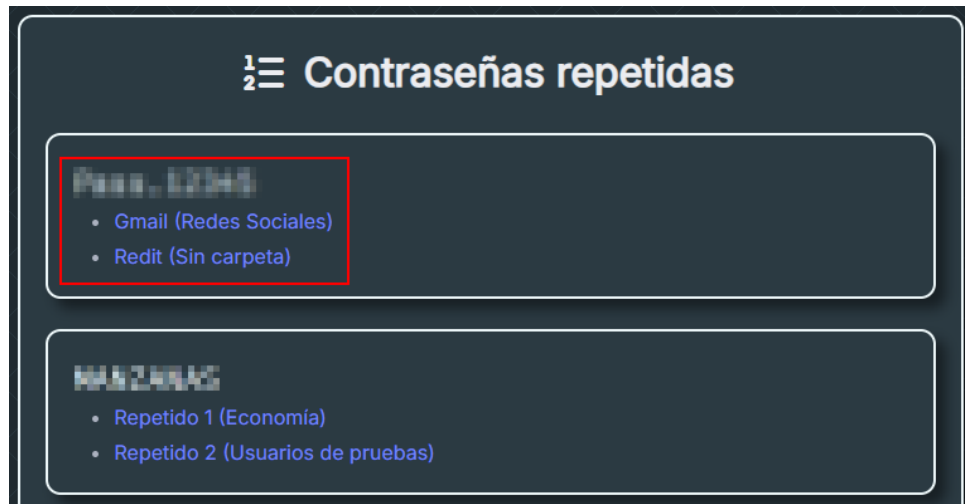


Ilustración 38: Contraseñas duplicadas

Contraseñas filtradas

Se mostrará un resumen con las contraseñas que se han encontrado en filtraciones y la cantidad de veces. Puede acceder al elemento clicando directamente en el nombre del elemento (recomendado modificarlas).

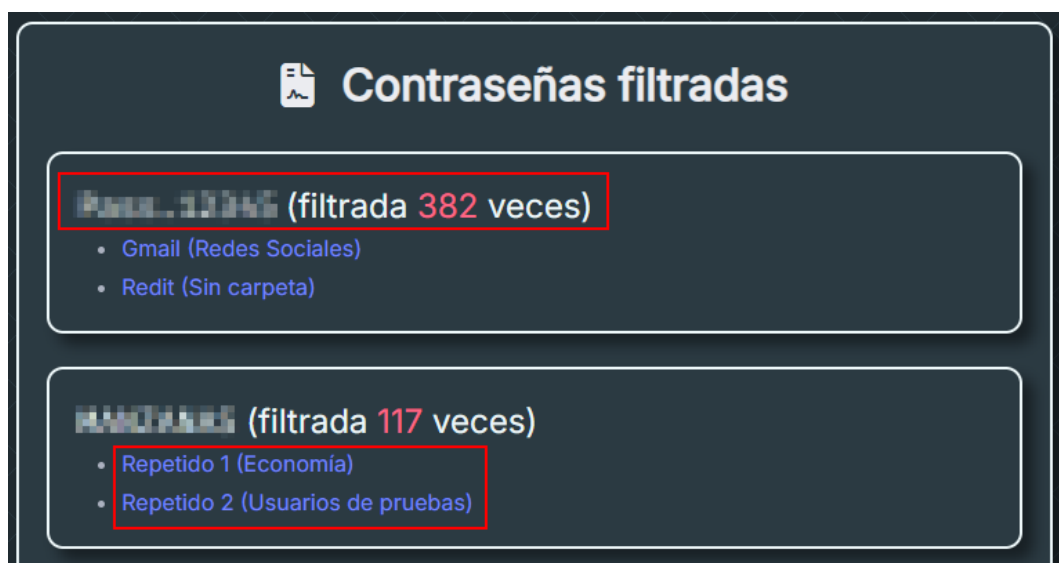


Ilustración 39: Contraseñas filtradas

6.1.9. Personalización de interfaz

Puede personalizar la interfaz a su gusto. Puede cambiar el tema de la aplicación (modo claro u oscuro) y cambiar la barra lateral de lado (en dispositivos que muestren la barra lateral). Para ello, acceda a los Ajustes desde el menú de navegación.

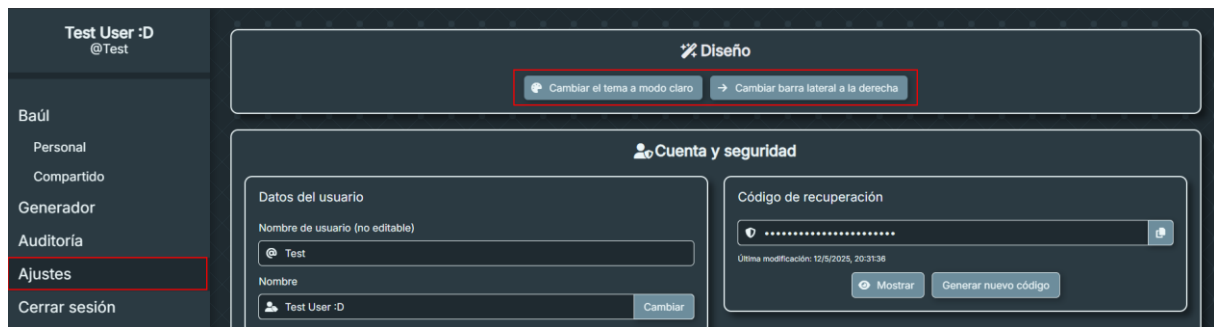


Ilustración 40: Diseño de interfaz

6.1.10. Seguridad de cuenta

En esta sección de los ajustes, podrá modificar detalles de su cuenta o configurar distintas opciones de seguridad.

Nombre

Puede editar su nombre, únicamente el nombre identificativo, el nombre de usuario no se puede editar. Para ello, introduzca su nuevo nombre y pulse en Cambiar.

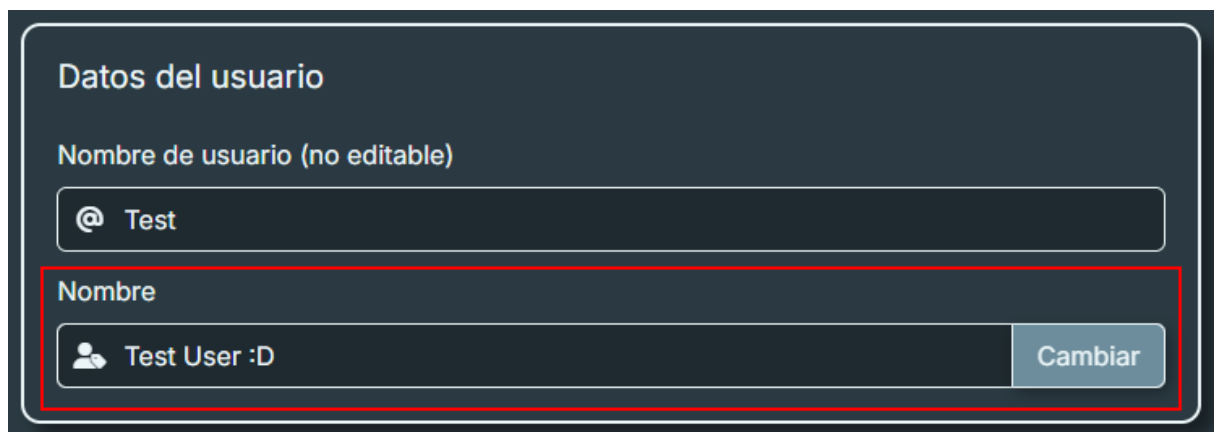


Ilustración 41: Modificación de cuenta

Contraseña maestra

Puede editar la contraseña maestra, para ello, introduzca la nueva contraseña y su confirmación. Pulse en Cambiar.

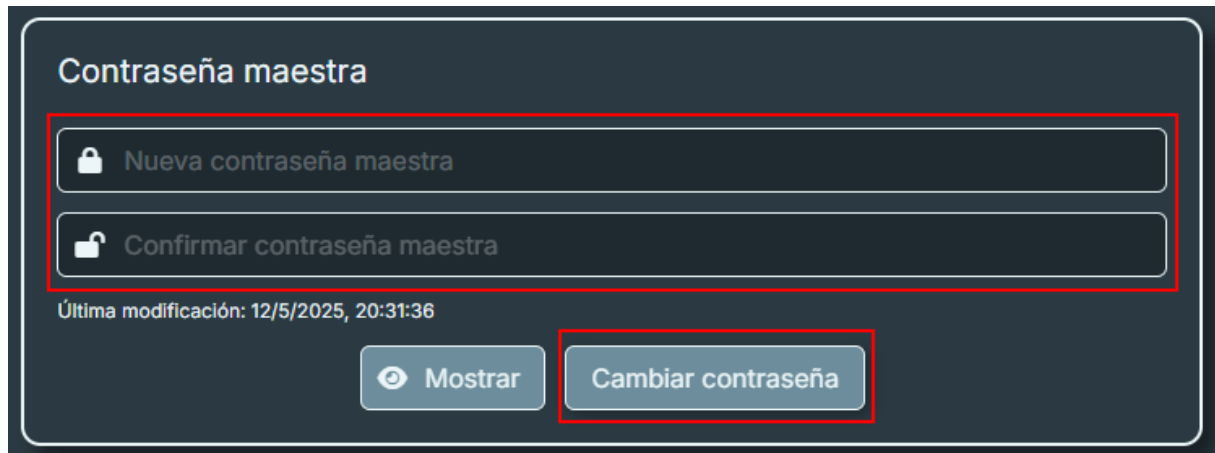


Ilustración 42: Modificación de cuenta

Código de recuperación

En este apartado puede ver su código de recuperación actual o generar uno nuevo, invalidando el anterior. Para ello, pulse en Mostrar o Generar nuevo código.

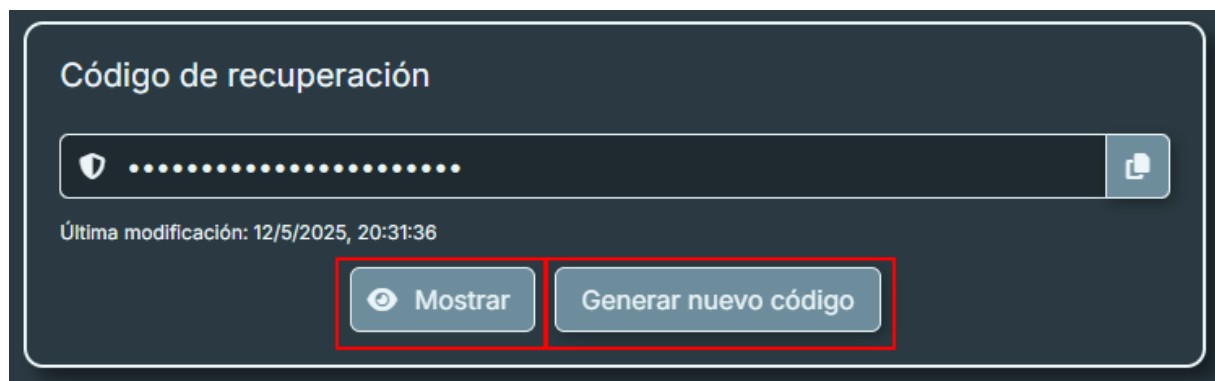


Ilustración 43: Modificación de cuenta

Cambiar duración de la sesión

Puede cambiar la duración de la sesión actual, pudiendo seleccionar una de las mismas opciones que durante el inicio de sesión. Esto modifica la duración de la sesión respecto a la hora actual, es decir, si la sesión caducaba en 2 semanas y cambias a una hora, la sesión se cerrará en 1 hora, no se suma a la sesión anterior.

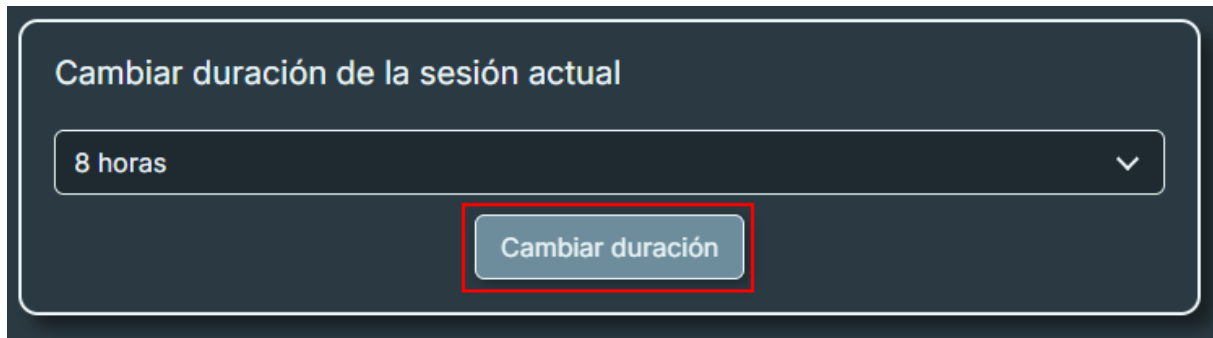


Ilustración 44: Modificación de cuenta

Doble factor de autenticación

Puede activar la autenticación de doble factor mediante código temporal. Para ello, clique en Activar.

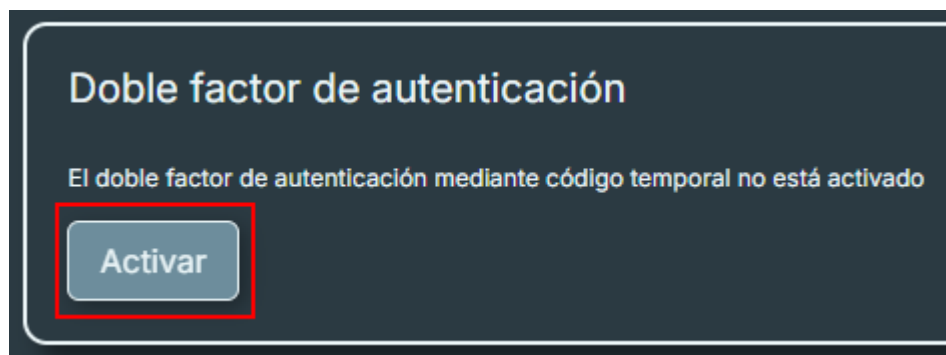


Ilustración 45: Activación 2FA

Se le mostrará un código QR. Escanee el código con una aplicación de generación de códigos temporales (Google Authenticator, Microsoft Authenticator, Authy...). Si no puede escanear el QR, introduzca manualmente la clave de la parte inferior.

Si recupera la cuenta mediante código de recuperación no se solicitará la autenticación de doble factor. Además, al recuperar la cuenta se desactiva, deberá activarlo de nuevo siguiendo los pasos descritos anteriormente.

6.2. Manual técnico

6.2.1. Objetivo

El objetivo de este manual es explicar los pasos a seguir para poder instalar y ejecutar la aplicación en un entorno local, ya sea para desarrollo o pruebas. Esta guía está orientada para usuarios con sistema operativo Windows 10 u 11.

6.2.2. Base de datos

Para la instalación de la base de datos se ha utilizado [DBEngine](#). Se debe descargar el instalador para Windows y ejecutarlo. Después de seguir los pasos correspondientes indicados en el instalador, procederemos a crear una instancia de MySQL 8.4.2. Para ello, se debe clicar en el + situado en la parte superior izquierda. En el desplegable, seleccionaremos MySQL.

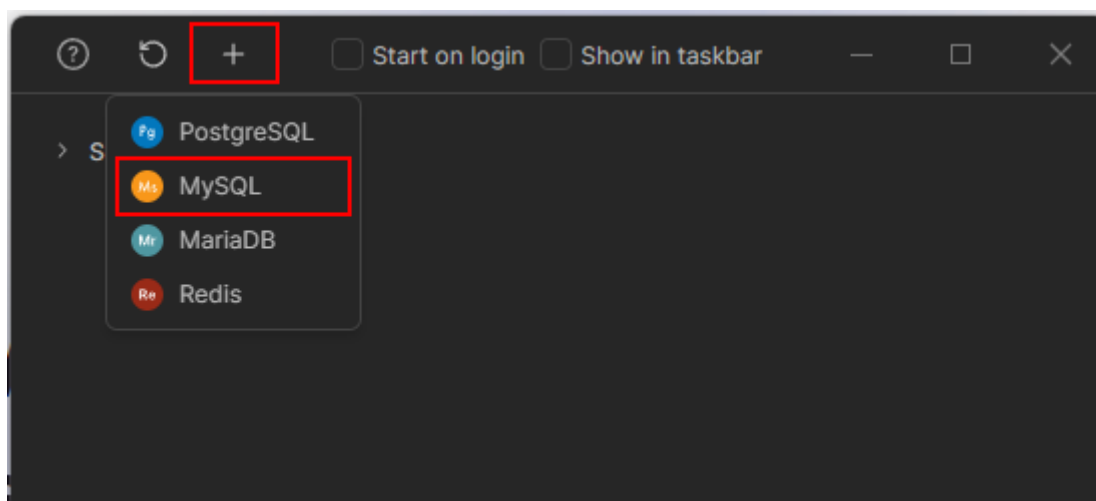


Ilustración 49: Crear DB MySQL

Se abrirá una nueva ventana, en ella podremos seleccionar la versión (seleccionar “8.4.2 – x64” o “8.4.0 – x64”), introducir el nombre de la base de datos (por ejemplo “pass_warriors”), el puerto (dejar por defecto, “3306”), el socket (dejar por defecto), la ruta donde se almacenará la información (puede elegir la ruta que se desee o dejar la ruta por defecto). Al pulsar en “Create”, se creará la instancia de la base de datos. El usuario para acceder a la base de datos, como se indica en la parte inferior, es “root” y sin contraseña.

Version 8.4.2 - x64

Name pass_warriors

Port 3306

Socket /tmp/database_server.sock
The socket path (optional)

Data Path Select a folder...
DBngin will use the default location if you leave it empty

Config Select a file... -
The database custom config file (optional)

☒ Disable bin log

☐ Automatically start service on Login

Default user of MySQL is root with no password.

Cancel Create

Ilustración 50: Configuración

Para iniciar la base de datos, se debe clicar en “start”. Una vez iniciada, el punto rojo se cambiará a verde. Para detener la base de datos, clicar en “stop” y el punto volverá a ser de color rojo

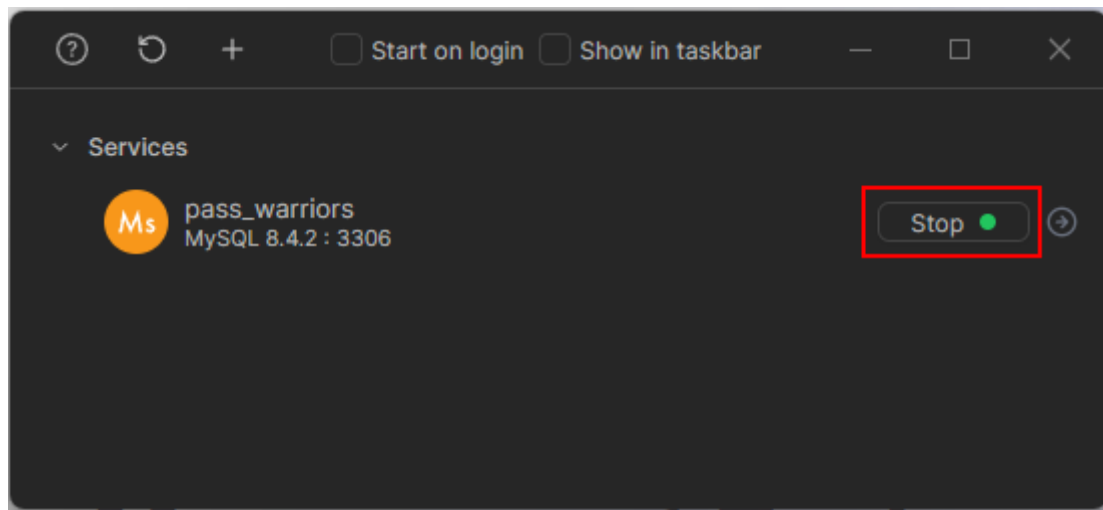


Ilustración 51: Arranque

Ya tenemos una base de datos funcional, para acceder a ella y realizar operaciones, se ha de descargar un cliente. En este caso, se utilizará [*DBeaver*](#). Para ello, se debe descargar el instalador de la página oficial y seguir los pasos del asistente.

Una vez instalado, procedemos a abrir la aplicación. Para crear una conexión, se debe clicar en un icono de un enchufe con un +, en la parte superior izquierda.

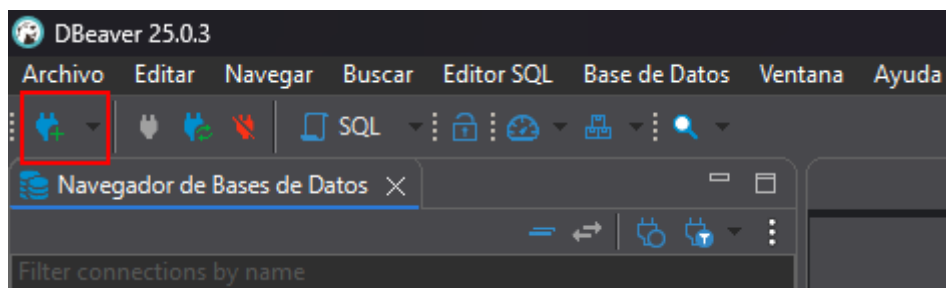


Ilustración 52: Crear conexión

Se abrirá un selector de bases de datos, seleccione MySQL y pulse siguiente.

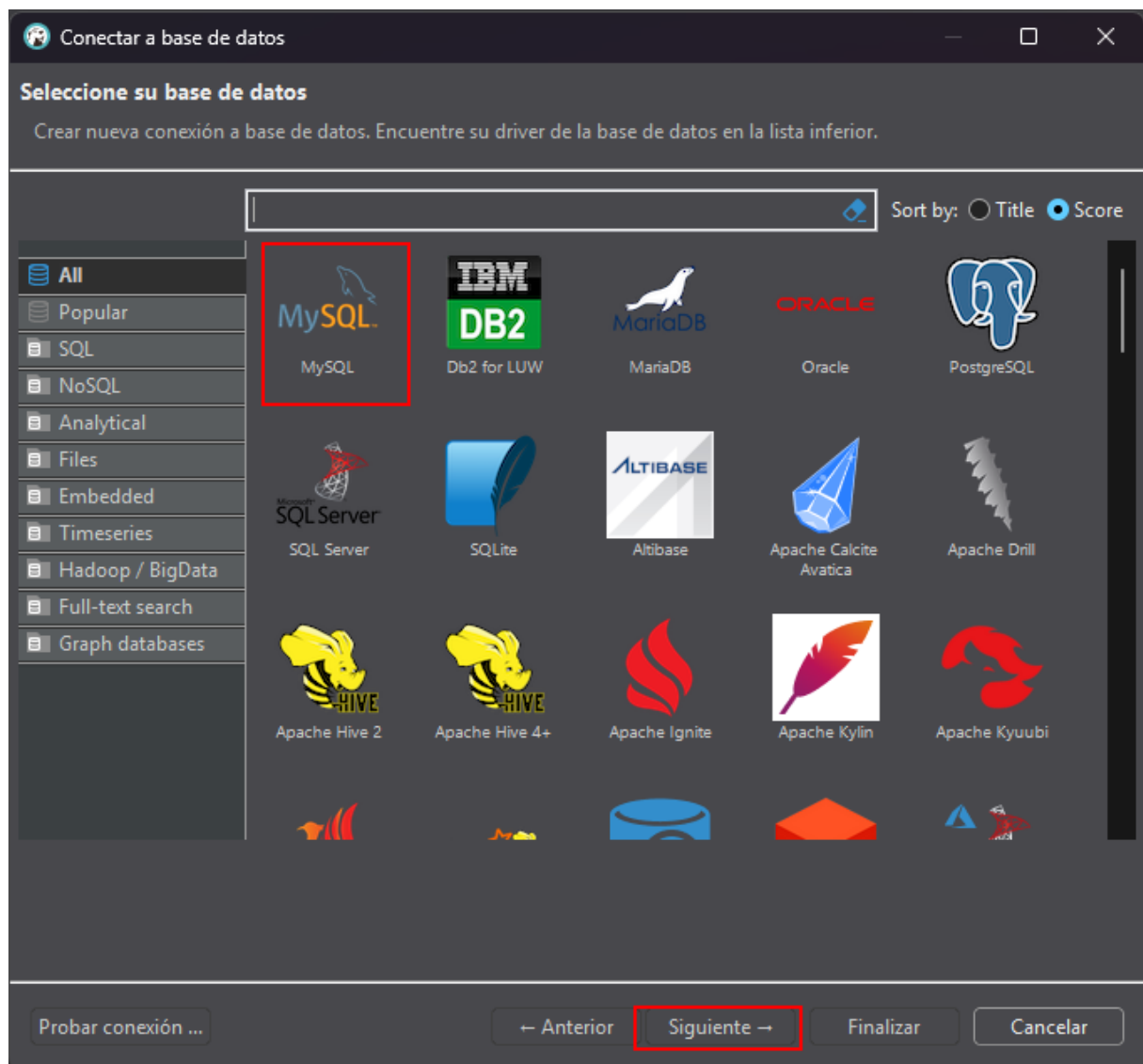


Ilustración 53: Selector de conector

En la nueva pestaña, se puede introducir el servidor al que nos vamos a conectar. Para ello, en Server Host introduzca “localhost” y deje el puerto 3306. Como nombre de usuario introduzca “root” y en contraseña no introduzca ningún valor. Una vez completado los pasos anteriores, pulse finalizar.

The screenshot shows the 'Conectar a base de datos' (Connect to database) dialog box in MySQL Workbench. The 'General' tab is selected. The 'Server' section shows 'Connect by: Host' selected, with 'Server Host' set to 'localhost' and 'Port' set to '3306'. The 'Authentication (Database Native)' section shows 'Nombre de usuario' set to 'root' and 'Contraseña' empty. The 'Advanced' section shows 'Server Time Zone' set to 'Auto-detect' and 'Local Client' set to 'MariaDB Binaries'. At the bottom, the 'Finalizar' button is highlighted with a red box.

Ilustración 54: Configuración

Opcionalmente, puede renombrar la conexión clicando con el clic derecho en el navegador de Bases de Datos.

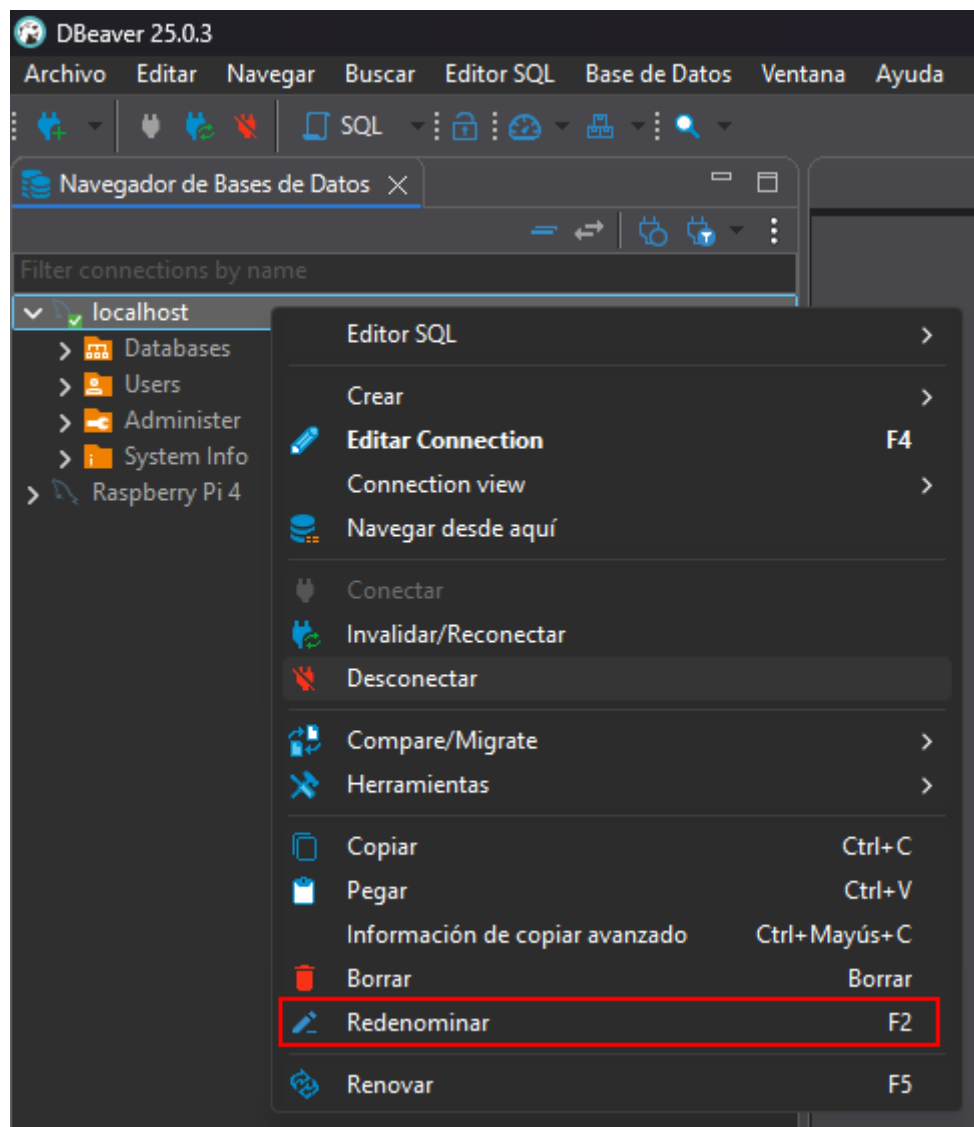


Ilustración 55: Cambio de nombre

Para conectarse a la base de datos, haga doble clic en el nombre de la conexión o clique en el desplegable a la izquierda del nombre, se debe mostrar un tick verde.

Para crear la base de datos, se debe ejecutar el script “db_pass_warriors_structure.sql”, almacenado en la carpeta “/db”. Para ello, copie el contenido del archivo y vuelva a DBeaver. En la parte superior, clique en SQL, se abrirá un editor de código.

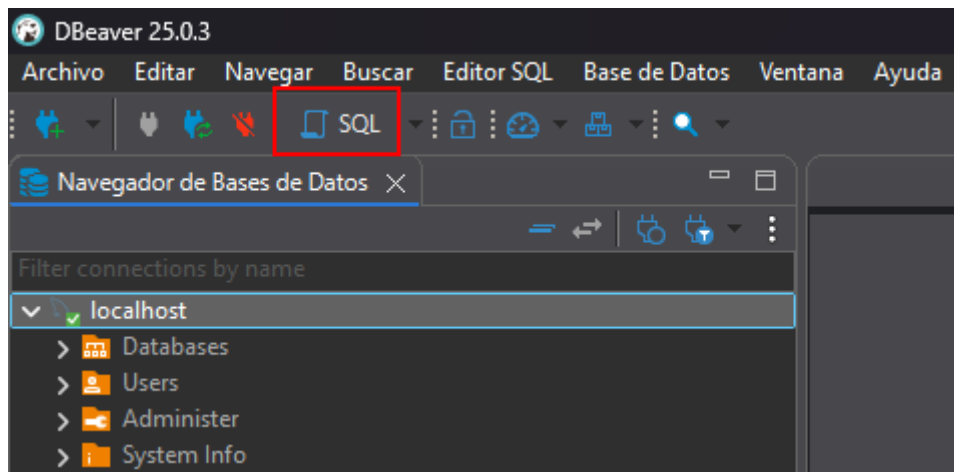


Ilustración 56: Abrir editor de código

En él pegue todo el contenido del script y clique en el pergamino que contiene una flecha, en la parte superior izquierda del editor.

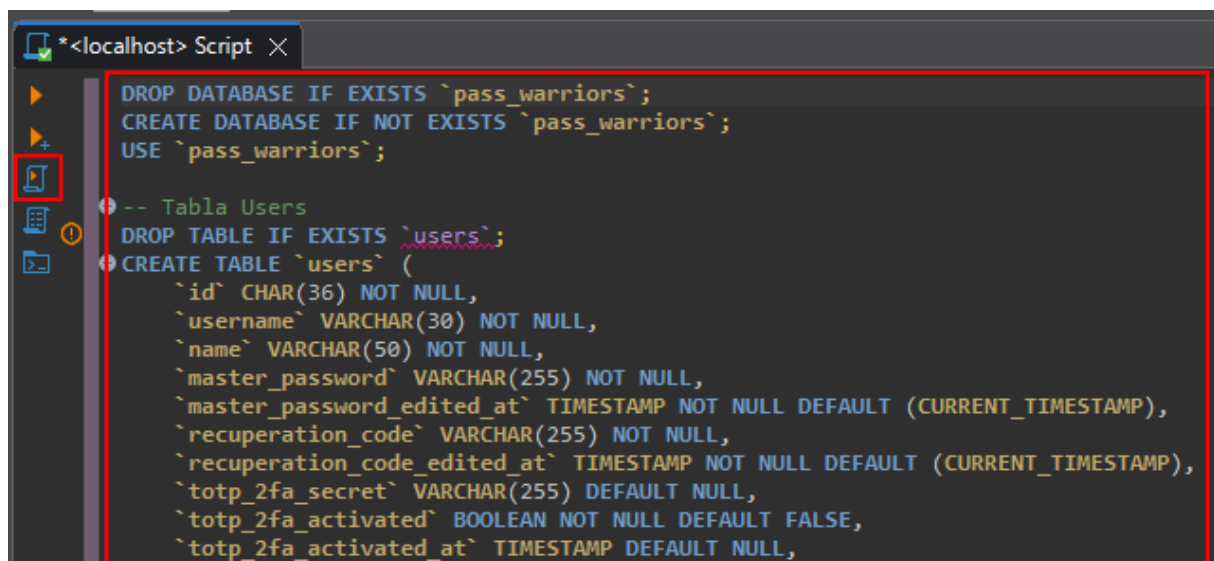


Ilustración 57: Copiar y ejecutar Script

Una vez finalizada la ejecución del script, debe realizar clic derecho en el nombre de la bse de datos (en Navegador de Bases de Datos) y clicar en “Renovar”. Deberá aparecer la base de datos “pass_warriors”.

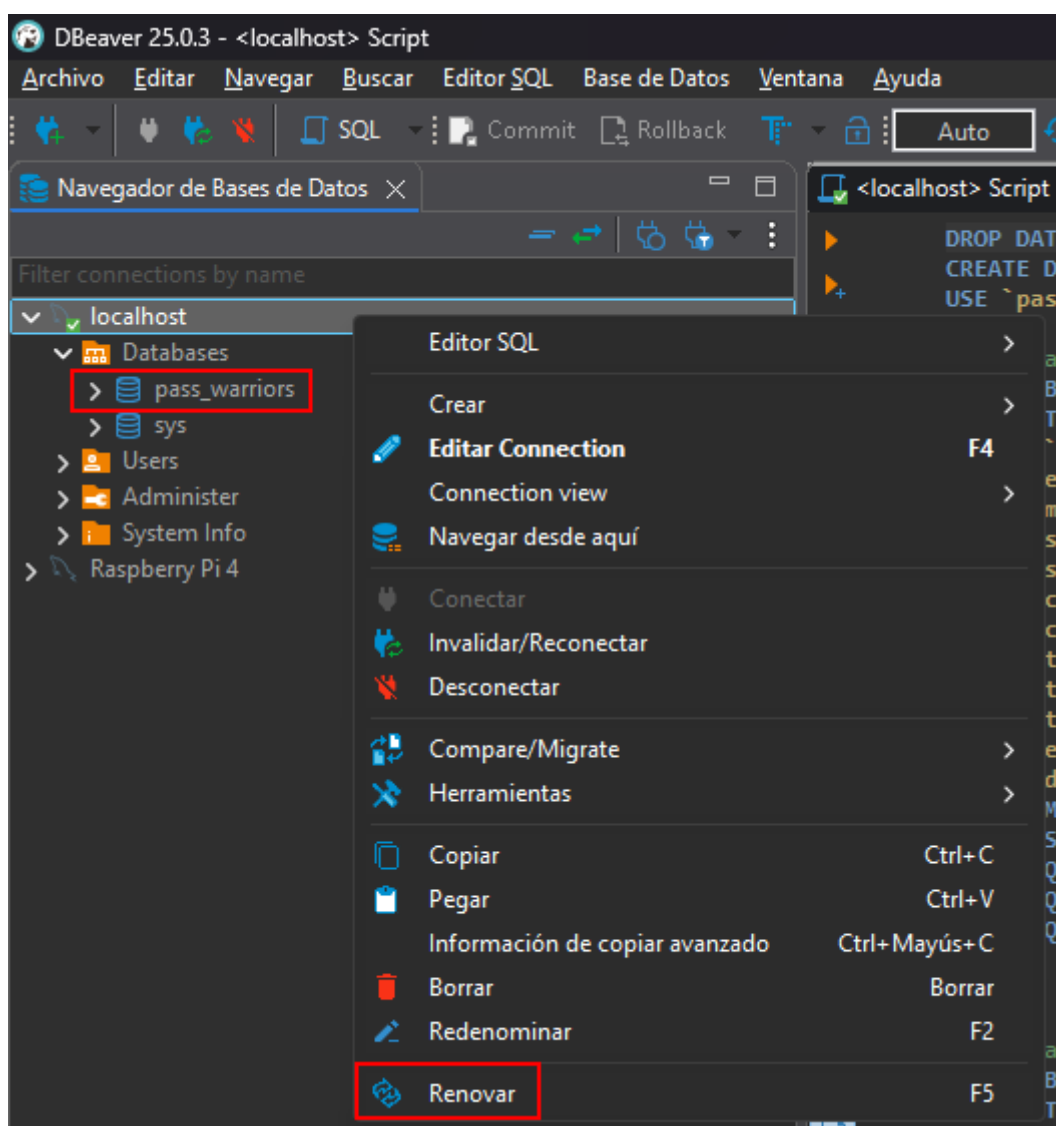


Ilustración 58: Comprobar

Ya tendríamos preparada la base de datos.

6.2.3. Backend

Para la instalación del backend, es necesario instalar el entorno de desarrollo [XAMPP](#), para ello, debe descargar el instalador de la página oficial y seguir los pasos del asistente. Puede instalar el paquete completo o únicamente el servidor Apache con PHP (recomendado). Una vez instalado XAMPP, ejecute, con permisos de administrador, el panel de control.

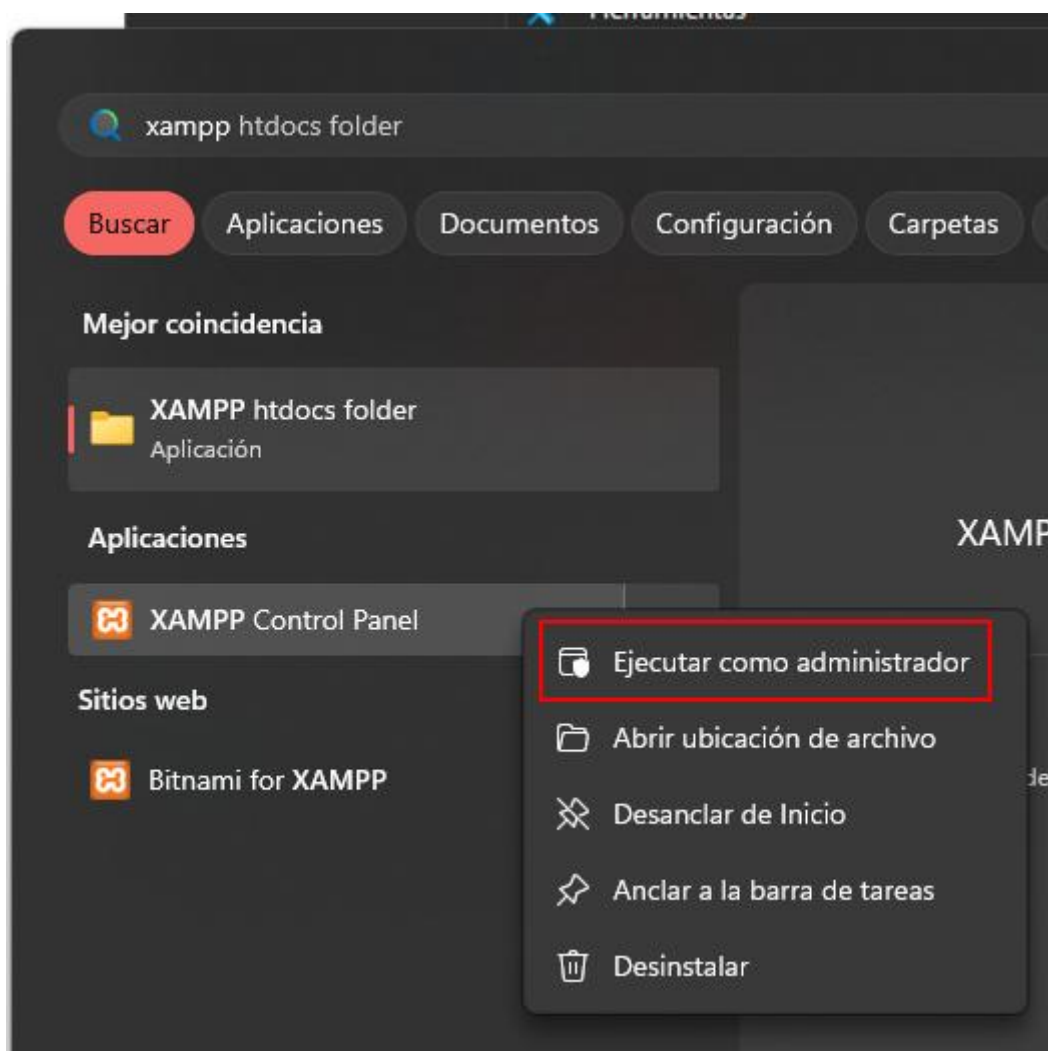


Ilustración 59: Abrir panel de control

En el panel de control, clique en Start del servicio Apache. Deberá mostrarse el nombre con un fondo verde indicando que se está ejecutando correctamente. Pulse en “Stop” para detener el servidor Apache.

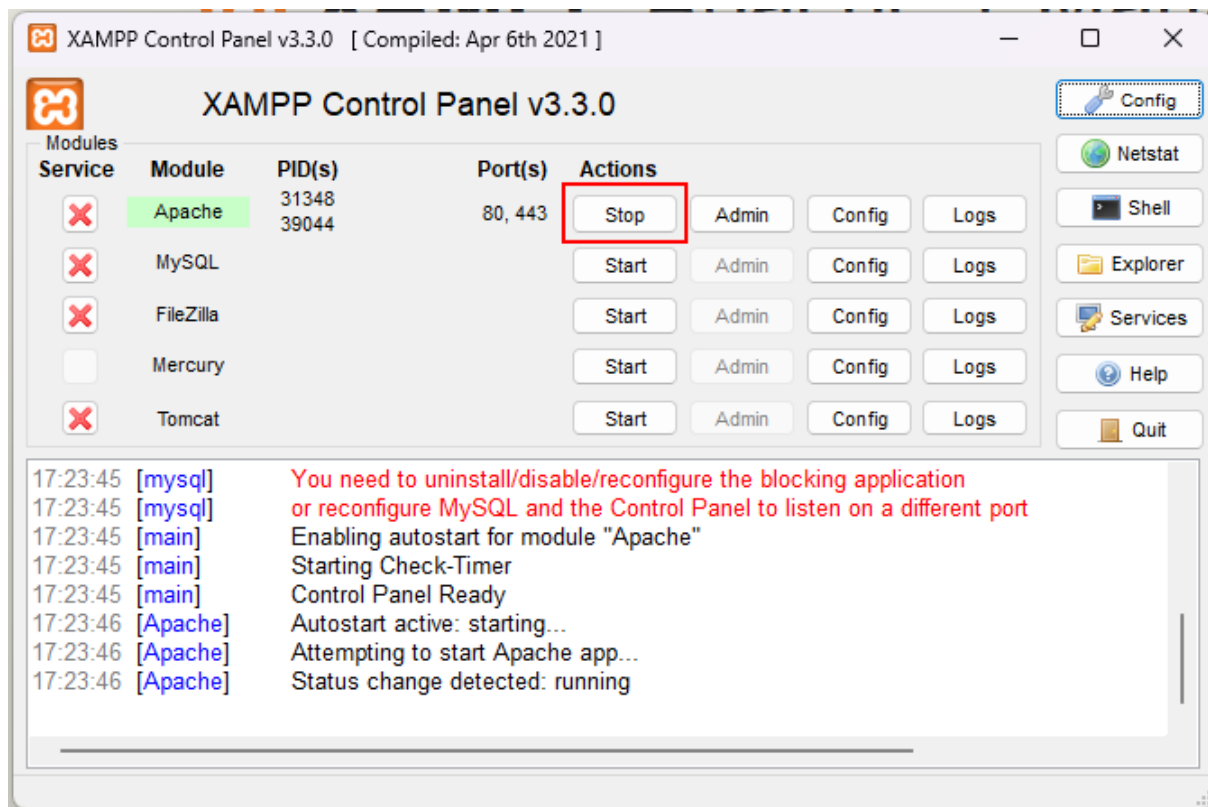


Ilustración 60: Arranque

Una vez que el servidor Apache funciona correctamente, se debe arrastrar la carpeta “/api” completa (no el contenido) a la carpeta htdocs creada en XAMPP (generalmente la ruta es “C:/xampp/htdocs”).

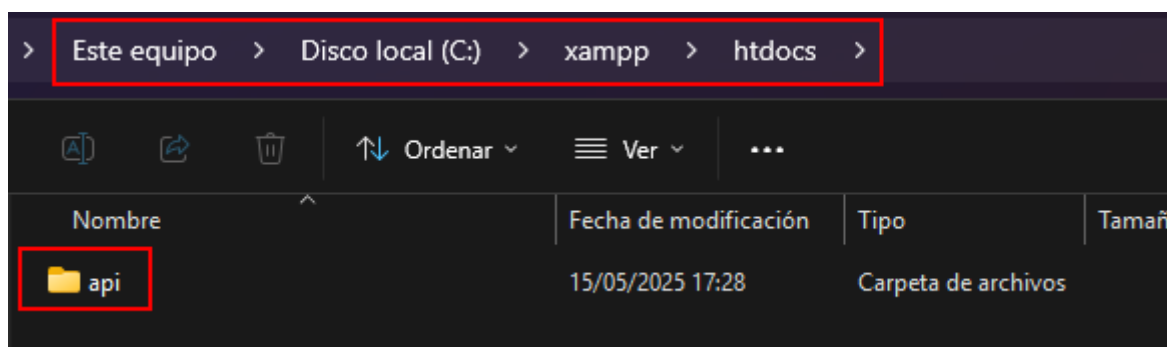
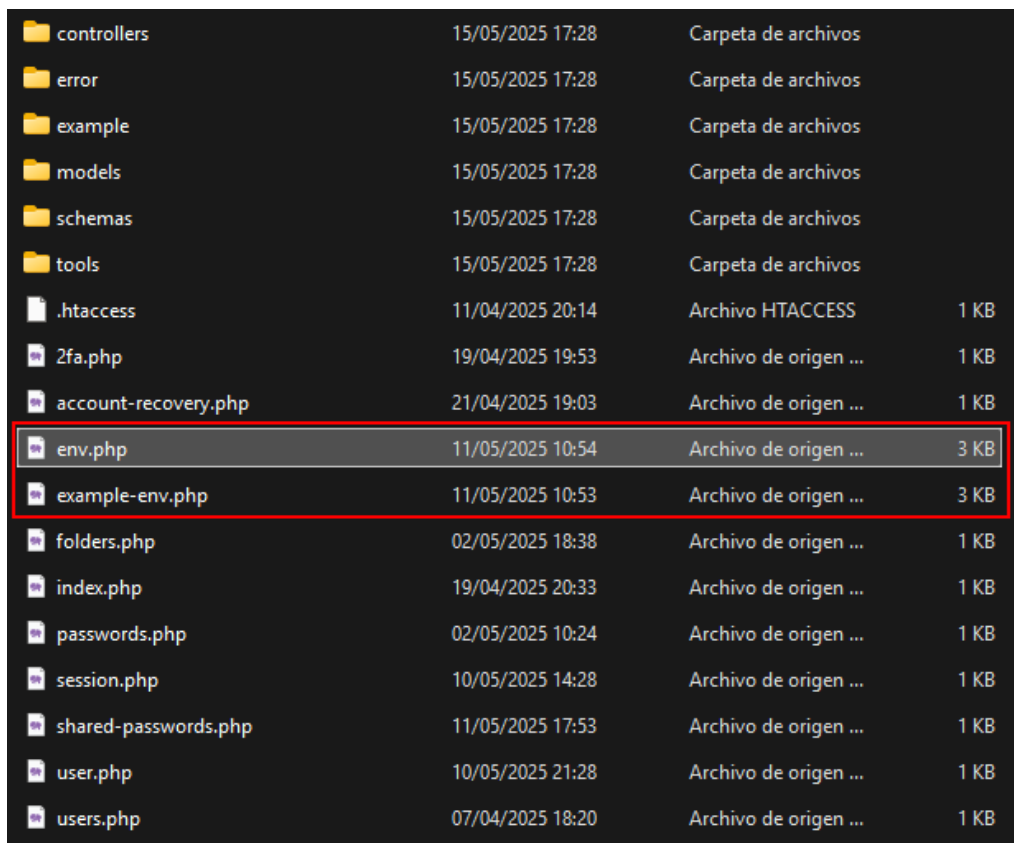


Ilustración 61: Copiar carpeta en htdocs

Para probar que funcione correctamente, debe acceder a la carpeta “/api” y realizar una copia del archivo “example-env.php”, guardándolo como “env.php”. Ábralo con un editor de código o de texto.

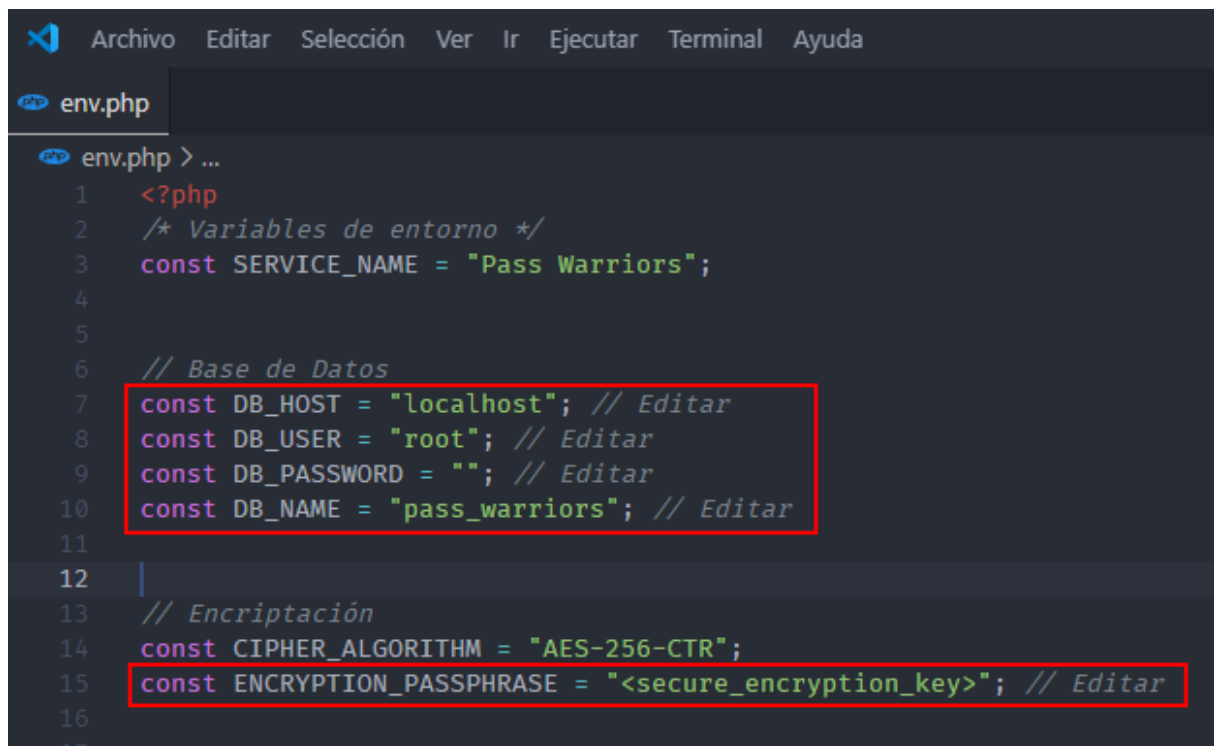


controllers	15/05/2025 17:28	Carpeta de archivos	
error	15/05/2025 17:28	Carpeta de archivos	
example	15/05/2025 17:28	Carpeta de archivos	
models	15/05/2025 17:28	Carpeta de archivos	
schemas	15/05/2025 17:28	Carpeta de archivos	
tools	15/05/2025 17:28	Carpeta de archivos	
.htaccess	11/04/2025 20:14	Archivo HTACCESS	1 KB
2fa.php	19/04/2025 19:53	Archivo de origen ...	1 KB
account-recovery.php	21/04/2025 19:03	Archivo de origen ...	1 KB
env.php	11/05/2025 10:54	Archivo de origen ...	3 KB
example-env.php	11/05/2025 10:53	Archivo de origen ...	3 KB
folders.php	02/05/2025 18:38	Archivo de origen ...	1 KB
index.php	19/04/2025 20:33	Archivo de origen ...	1 KB
passwords.php	02/05/2025 10:24	Archivo de origen ...	1 KB
session.php	10/05/2025 14:28	Archivo de origen ...	1 KB
shared-passwords.php	11/05/2025 17:53	Archivo de origen ...	1 KB
user.php	10/05/2025 21:28	Archivo de origen ...	1 KB
users.php	07/04/2025 18:20	Archivo de origen ...	1 KB

Ilustración 62: Crear archivo env.php

Con el editor, edite las siguientes variables:

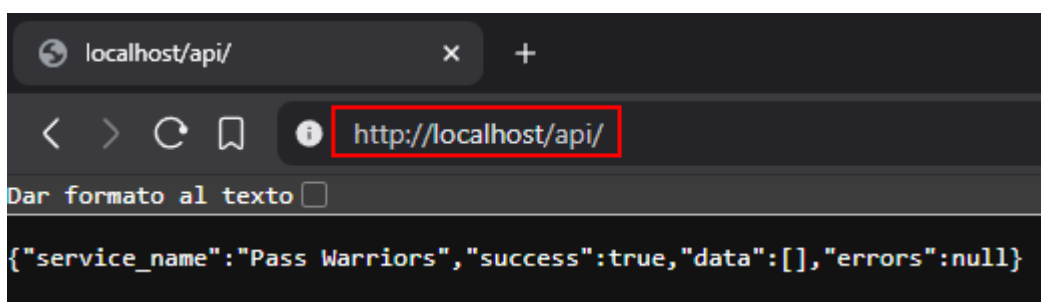
- DB_HOST: Sustituya “<db_host>” por “localhost”
- DB_USER: Sustituya “<db_user>” por “root”
- DB_PASSWORD: Sustituya “<db_password>” por “”
- DB_NAME: Sustituya “<db_name>” por “pass_warriors”
- ENCRYPTION_PASSPHRASE: Sustituya “<secure_encryption_key>” por su *passphrase* de encriptación. Introduzca cualquier cadena de caracteres.



```
env.php > ...
1  <?php
2  /* Variables de entorno */
3  const SERVICE_NAME = "Pass Warriors";
4
5
6  // Base de Datos
7  const DB_HOST = "localhost"; // Editar
8  const DB_USER = "root"; // Editar
9  const DB_PASSWORD = ""; // Editar
10 const DB_NAME = "pass_warriors"; // Editar
11
12
13 // Encriptación
14 const CIPHER_ALGORITHM = "AES-256-CTR";
15 const ENCRYPTION_PASSPHRASE = "<secure_encryption_key>"; // Editar
16
17
```

Ilustración 63: Edición de env.php

Ya tendríamos preparado el backend. Para comprobar su funcionamiento, abra un navegador web e introduzca <http://localhost/api/> en la url. Deberá visualizar un JSON.



```
localhost/api/
< > ↻
http://localhost/api/
Dar formato al texto
{"service_name": "Pass Warriors", "success": true, "data": [], "errors": null}
```

Ilustración 64: Comprobación

6.2.4. Frontend

Para la instalación del *frontend*, deberá instalar el entorno de ejecución [Node JS](#). Se debe instalar la versión 22 y PNPM como gestor de paquetes. Puede utilizar la instalación mediante Powershell o mediante instalador (recomendado).



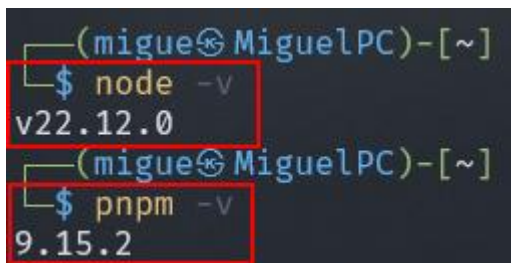
The screenshot shows the 'Descarga Node.js®' page. At the top, it says 'Obtiene Node.js®' followed by a dropdown menu set to 'v22.15.1 (LTS)'. This is followed by 'para' and a dropdown menu set to 'Windows', then 'usando' and a dropdown menu set to 'fnm', and finally 'con' and a dropdown menu set to 'pnpm'. Below this, there is a code block with the following commands:

```
1 # Descarga e instala fnm:
2 winget install Schniz.fnm
3
4 # Descarga e instala Node.js:
5 fnm install 22
6
7 # Verifica la versión de Node.js:
8 node -v # Debería mostrar "v22.15.1".
9
10 # Descarga e instala pnpm:
11 corepack enable pnpm
12
13 # Verifica versión de pnpm:
14 pnpm -v
```

Below the code block, it says 'PowerShell' and a button 'Copiar al portapapeles'. Below that, it says '"fnm" is a cross-platform Node.js version manager. If you encounter any issues please visit [fnm's website](#)'. Below this, it says 'O obtiene una versión pre compilada de Node.js® para' followed by a dropdown menu set to 'Windows', then 'usando la arquitectura' and a dropdown menu set to 'x64'. At the bottom, there are two buttons: 'Windows Installer (.msi)' and 'Standalone Binary (.zip)'. The 'Windows Installer (.msi)' button is highlighted with a red box.

Ilustración 65: Descarga Node JS

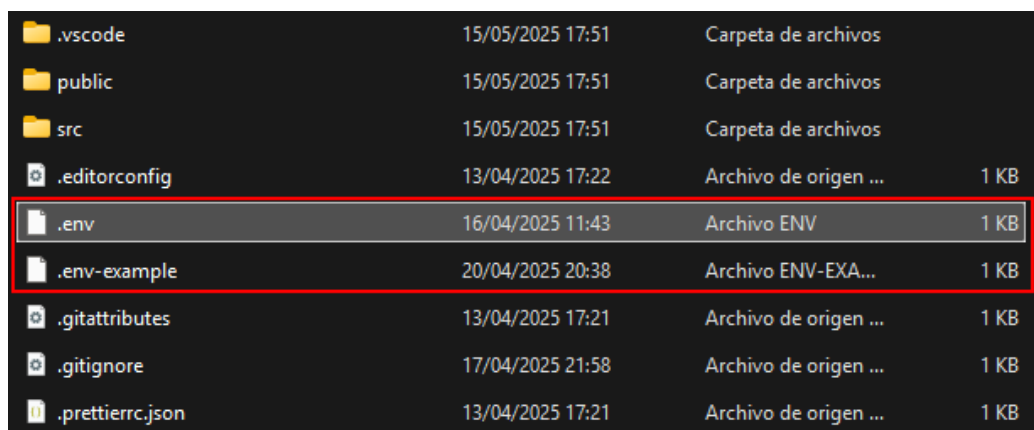
Una vez que tenga instalado Node JS, comprueba en una terminal que están instalados correctamente Node y PNPM. Si PNPM muestra un error, pruebe a seguir la siguiente guía: [Instalar pnpm con Corepack](#).



```
(migue@MiguelPC)-[~]  
$ node -v  
v22.12.0  
(migue@MiguelPC)-[~]  
$ pnpm -v  
9.15.2
```

Ilustración 66: Comprobación

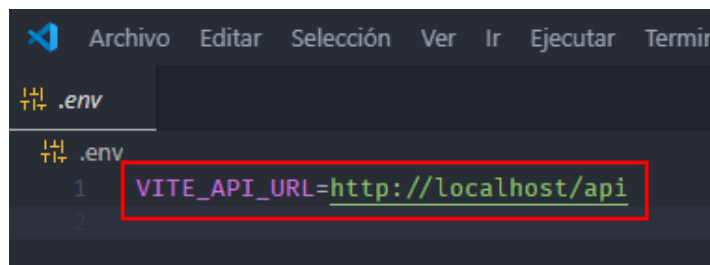
Una vez terminada la instalación, acceda a la carpeta “/frontend”. Copie el archivo “.env-example” y guárdelo como “.env”. Ábralo con un editor de texto o código.



folder	.vscode	15/05/2025 17:51	Carpeta de archivos	
folder	public	15/05/2025 17:51	Carpeta de archivos	
folder	src	15/05/2025 17:51	Carpeta de archivos	
file	.editorconfig	13/04/2025 17:22	Archivo de origen ...	1 KB
file	.env	16/04/2025 11:43	Archivo ENV	1 KB
file	.env-example	20/04/2025 20:38	Archivo ENV-EXA...	1 KB
file	.gitattributes	13/04/2025 17:21	Archivo de origen ...	1 KB
file	.gitignore	17/04/2025 21:58	Archivo de origen ...	1 KB
file	.prettierrc.json	13/04/2025 17:21	Archivo de origen ...	1 KB

Ilustración 67: Creación de .env

Modifique el valor de VITE_API_URL, sustituyendo “<api_url>” por “http://localhost/api”, sin las comillas.



```
Archivo  Editor  Selección  Ver  Ir  Ejecutar  Terminar  
+ .env  
+ .env  
1 VITE_API_URL=http://localhost/api  
2
```

Ilustración 68: Edición de .env

Una vez editado el archivo, acceda desde una terminal a la carpeta “/frontend”.

Ejecute el comando “pnpm install”.

```
(migue@MiguelPC)-[~/Documents/proyecto/frontend]
$ pnpm install
Lockfile is up to date, resolution step is skipped
Packages: +246
+++++
```

Ilustración 69: Instalación de dependencias

Una vez terminada la instalación de los paquetes y dependencias, ejecute el comando “pnpm run dev”. Al finalizar, debe mostrar una URL indicando el puerto en el que se aloja la aplicación, generalmente “<http://localhost:5173/>”.

```
VITE v6.2.4 ready in 5722 ms
→ Local: http://localhost:5173/
→ Network: use --host to expose
→ Vue DevTools: Open http://localhost:5173/__devtools__/ as a separate window
→ Vue DevTools: Press Alt(⌘)+Shift(⇧)+D in App to toggle the Vue DevTools
→ press h + enter to show help
[@vue/compiler-sfc] `defineEmits` is a compiler macro and no longer needs to be imported.
```

Ilustración 70: Ejecución del proyecto

Ahora, puede acceder desde un navegador a la URL proporcionada, mostrando la aplicación. En este momento, podrá crear un usuario y probar las funcionalidades de la aplicación.

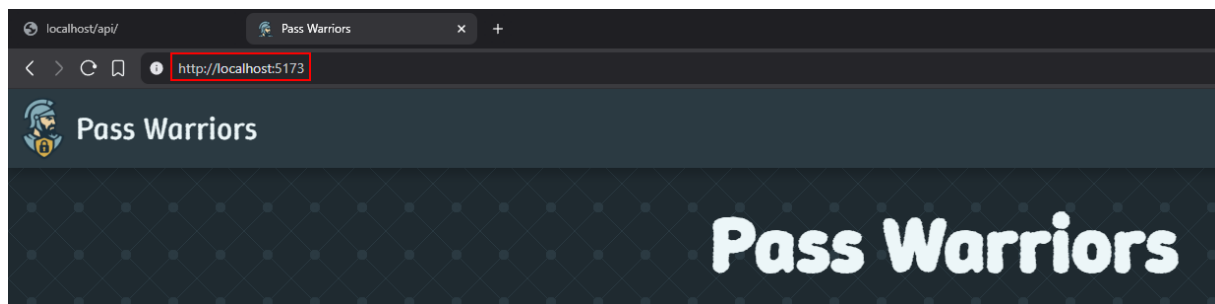


Ilustración 71: Comprobación

Si accede a “http://localhost:5173/_devtools_/”, podrá ver las herramientas de desarrollador, donde se muestran los componentes utilizados y como se relacionan, las páginas, tiempos de carga, etc.

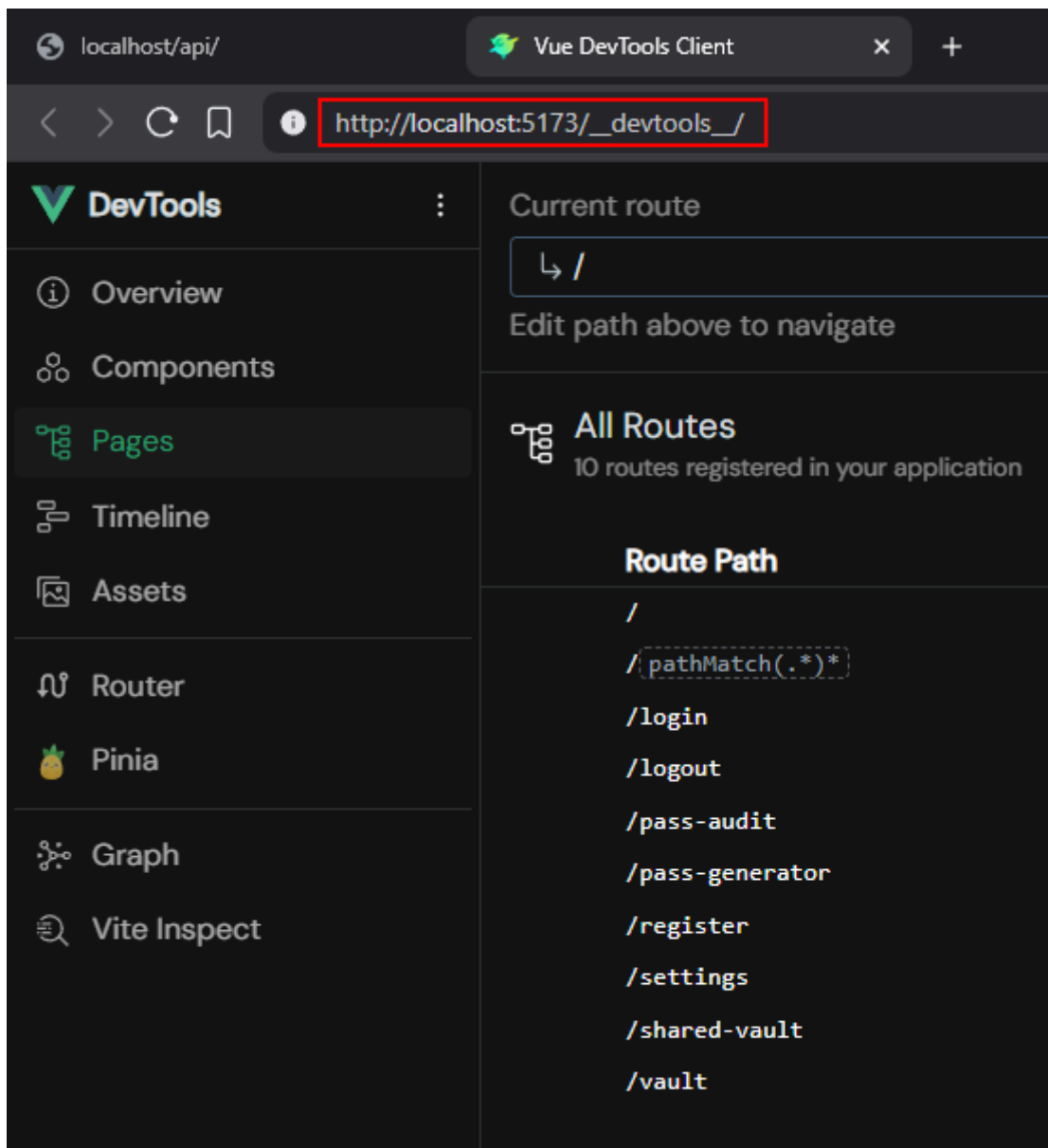


Ilustración 72: Dev Tools

7. Conclusiones

El desarrollo del proyecto me ha ayudado a comprender y asentar los conocimientos adquiridos durante el curso. También, mediante el uso de tecnologías nuevas y que no ha sido vistas en clase, me ha servido para mejorar la búsqueda de información y lectura de documentación.

Además, me ha servido para entender de una mejor manera como es el desarrollo completo de un proyecto real, desde la idea, pasando por el análisis y el diseño, hasta el desarrollo, pruebas y despliegue.

7.1 Futuras ampliaciones

Debido al corto período de duración del desarrollo del proyecto, algunas de las ideas que tenía en mente o se me han ocurrido mientras se producía el desarrollo, no se han podido implementar.

Una de las principales sería mejorar el sistema de compartición de contraseñas, añadiendo un sistema de “amigos” para poder seleccionar que usuarios pueden compartir contraseñas contigo.

Otra idea que se me ocurrió durante el desarrollo y no se ha implementado, es mostrar un resumen de cuantas contraseñas has compartido con un usuario o que ellos han compartido contigo.

También, me hubiese gustado añadir una integración más completa entre elementos privados y compartidos, es decir, que puedas compartir un elemento personal sin la necesidad de crear un nuevo elemento compartido o viceversa, dejar de compartir un elemento, pero sin eliminarlo.

8. Bibliografía

Se han utilizado fuentes y documentación oficiales, como blogs de compartición de código y el uso de inteligencia artificial:

- <https://vuejs.org/guide/introduction>
- <https://pinia.vuejs.org/core-concepts/>
- <https://router.vuejs.org/guide/>
- <https://www.php.net/manual/es/>
- <https://www.php.net/manual/es/function.openssl-encrypt.php>
- <https://bulma.io/documentation>
- <https://docs.fontawesome.com/>
- <https://stackoverflow.com/questions>