



Formación Profesional
Océano Atlántico

MÓDULO PROYECTO

CICLO SUPERIOR DE DESARROLLO DE APLICACIONES WEB

Lucian Ioan Ilisei

DNI: Y0748122S

CURSO: 2024-2025

PROYECTO:

Este proyecto consiste en el desarrollo de una aplicación web fullstack que permite a los usuarios llevar un control detallado de sus finanzas personales. El sistema permite registrar ingresos y gastos, visualizar el saldo disponible, analizar el historial de movimientos y obtener gráficos e informes visuales de los gastos por categoría. Incluye funcionalidades como registro e inicio de sesión, análisis mensual, desgloses en porcentajes y visualización con gráficos interactivos.



Contenido

1	Identificación del proyecto	3
2	Introducción	4
3	Análisis previo, planificación y presupuesto	5
3.1	Aspectos generales de la aplicación	5
3.2	Especificación de requisitos	5
3.2.1	Requerimientos de sistema	5
3.2.2	Requerimientos de proceso o área de negocio (funcionales)	5
3.2.3	Requerimientos de interfaz gráfica	5
4	Fases de desarrollo	6
4.1.1	Fase 1:	6
4.1.2	Fase 2:	6
4.1.3	Fase 3	6
4.1.4	Fase 4:	6
4.1.5	Fase 5:	6
4.2	Planificación	6
4.3	Presupuesto	6
4.4	Entorno de trabajo (tecnologías de desarrollo y herramientas)	6
5	Diseño e implementación de la base de datos	7
5.1	Diseño conceptual (diagrama ER)	7
5.2	Normalización y diseño lógico (tablas y relaciones)	7
5.3	Implementación en SGBD	7
6	Diseño de interfaz de usuario	8
7	Estructura del proyecto: clases y elementos fundamentales del desarrollo	9
7.1	Estructura general	9
7.2	Clases y Elementos	9



8	Pruebas	10
9	Generación del JavaDoc	11
10	Generación y descripción de ejecutables	12
11	Manual de usuario	13
12	Desarrollo evolutivo de la aplicación	14
13	Bibliografía y documentación de referencia	15



1 Identificación del proyecto

Denominación	El Poder del Ahorro – Aplicación web de control de gastos e ingresos personales.
Autor	Lucian Ioan Ilisei
Curso	2º de Desarrollo de Aplicaciones Web (DAW)
Tipo de proyecto	Proyecto de Fin de Ciclo (TFG) de carácter práctico, enfocado al desarrollo de una aplicación web full-stack.
Fundamento	La gestión financiera personal es una necesidad creciente en la sociedad actual. Muchas personas desconocen con precisión en qué gastan su dinero y tienen dificultades para ahorrar. Este proyecto surge como una solución sencilla e intuitiva que permite al usuario registrar sus ingresos y gastos, visualizar estadísticas mensuales y analizar la distribución de su dinero para tomar mejores decisiones financieras.
Objetivos	<p>Desarrollar una aplicación web funcional con un diseño atractivo y responsivo.</p> <ul style="list-style-type: none">• Permitir a los usuarios registrar y clasificar sus ingresos y gastos.• Mostrar el saldo disponible, estadísticas de ahorro y análisis por categorías.• Proteger el acceso con autenticación mediante token JWT.• Implementar un backend seguro con Spring Boot y una base de datos MySQL.• Dockerizar toda la aplicación para su despliegue completo y simultáneo.
Destinatarios	El proyecto está orientado a cualquier usuario que quiera gestionar sus finanzas personales de



	forma sencilla, así como a desarrolladores o empresas interesadas en soluciones de gestión económica básica.
Lenguaje de programación	Frontend: TypeScript (Angular) Backend: Java (Spring Boot) Estilos: CSS con Bootstrap personalizado Scripting de contenedores: YAML (Docker Compose)
Base de datos	MySQL
Entregables	Código fuente completo (frontend y backend) Ficheros DockerFile y docker-compose.yml para despliegue del sistema Documentación técnica del proyecto Manual de usuario Presentación final del proyecto

2 Introducción

En la era digital, el control de las finanzas personales se ha convertido en una necesidad fundamental para garantizar una buena salud económica, especialmente entre jóvenes, estudiantes y trabajadores que buscan una mayor conciencia sobre sus hábitos de consumo. Sin embargo, muchas de las herramientas disponibles son complejas, impersonales o carecen de una visión clara y visual del estado financiero del usuario.

“El Poder del Ahorro” es una aplicación web diseñada para facilitar el seguimiento, la organización y el análisis de los movimientos financieros personales de una forma intuitiva, accesible y visualmente atractiva. La herramienta permite al usuario registrar ingresos y gastos en tiempo real, categorizarlos según su naturaleza (ocio, transporte, comida, etc.), y consultar tanto un historial detallado como gráficos que reflejan de forma visual la distribución del gasto mensual.

Además, el sistema ofrece funcionalidades como:

- Análisis porcentual por tipo de gasto.
- Gráficos circulares interactivos.
- Visualización de las transacciones más recientes con opción de expandir a historial completo.
- Protección mediante autenticación con JWT.
- Despliegue completo en contenedores Docker, lo que facilita su puesta en producción y portabilidad.

Desde el punto de vista técnico, este proyecto representa una solución full-stack moderna: el **frontend** está desarrollado en **Angular**, el **backend** en **Spring Boot**, y la **base de datos** en **MySQL**, todo orquestado a través de **Docker Compose** para una experiencia de desarrollo y despliegue óptima.

“El Poder del Ahorro” no solo pretende ser una aplicación útil para el día a día, sino también un ejemplo de integración de tecnologías, buenas prácticas de programación, organización modular del código y despliegue profesional. Es el resultado de un proceso de aprendizaje continuo, esfuerzo personal y aplicación práctica de los conocimientos adquiridos a lo largo del ciclo formativo de Desarrollo de Aplicaciones Web.

3 Análisis previo, planificación y presupuesto

3.1 Aspectos generales de la aplicación

“El Poder del Ahorro” es una aplicación web orientada a facilitar la gestión económica personal de forma sencilla y visual. La plataforma permite al usuario registrar ingresos y gastos, clasificarlos por categorías, visualizar su saldo disponible y realizar un seguimiento de su historial financiero a través de gráficos interactivos y análisis porcentuales.

La interfaz está pensada para ser intuitiva, con un diseño atractivo y funcional, adaptado tanto para su uso en ordenadores como en pantallas medianas. El backend se encarga de gestionar la lógica de negocio y la persistencia de datos, garantizando la seguridad mediante autenticación JWT. Además, la arquitectura está diseñada para permitir una posible escalabilidad futura, gracias a la separación clara entre capas y al uso de contenedores Docker.

3.2 Especificación de requisitos

3.2.1 Requerimientos de sistema

- **Servidor Backend:**
 - Java 17 o superior
 - Spring Boot 3
 - Maven
 - Puerto: 8080
 - Autenticación JWT
 - Integración con base de datos MySQL
- **Base de datos:**
 - MySQL 8
 - PhpMyAdmin para gestión visual
 - Base de datos: tfg_db
 - Usuario: user, Contraseña: password
- **Frontend:**

- Angular 17
- Node.js 22
- Puerto: 4200
- Bootstrap + estilos personalizados
- **Contenedores Docker (Docker Compose):**
- Servicios: frontend, backend, mysql, phpmyadmin
- Red virtual común
- Persistencia de datos con volúmenes

3.2.2 Requerimientos de proceso o área de negocio (funcionales)

- Registro de usuarios y autenticación con JWT.
- Login seguro y redirección al dashboard.
- Visualización del saldo total.
- Registro de **ingresos y gastos**, con categoría seleccionable.
- Análisis porcentual por tipo de gasto.
- Visualización de los **últimos 3 movimientos** con opción de “ver más”.
- Historial completo de movimientos ordenado por fecha.
- Gráfico circular del desglose de gastos por categoría.
- Visualización del total gastado por categoría.
- Protección de rutas (dashboard y análisis) mediante AuthGuard.
- Acceso restringido a usuarios autenticados.

3.2.3 Requerimientos de interfaz gráfica

- Estética tipo **glassmorphism**, con fondo en gradiente y tarjetas transparentes.
- Diseño responsive para pantallas medianas.
- Modal flotante para añadir ingresos o gastos.
- Modal adicional para mostrar el **análisis financiero** y el gráfico circular.
- Elementos visuales animados (botones, tarjetas, entrada de modales).
- Mensajes de error y éxito con SweetAlert2.
- Inputs validados con mensajes claros para campos obligatorios o erróneos.
- Colores diferenciados para ingresos (verde) y gastos (rojo).



- Navegación simple e intuitiva:
- Login > Dashboard > Análisis / Historial
- Botón de cierre de sesión siempre accesible

4 Fases de desarrollo

4.1.1 Fase 1: Diseño de la estructura inicial del backend y la base de datos

- Creación del proyecto Spring Boot.
- Implementación de entidades `User` y `Transaction`.
- Configuración de la conexión con MySQL mediante Docker.
- Generación automática de tablas con `spring.jpa.hibernate.ddl-auto=update`.
- Desarrollo de los repositorios y servicios básicos.
- Configuración inicial de seguridad y JWT.

4.1.2 Fase 2: Desarrollo del sistema de autenticación

- Implementación de endpoints de login y registro.
- Generación de tokens JWT personalizados.
- Configuración de filtros de autorización y autenticación en Spring Security.
- Protección de rutas sensibles.
- Pruebas con Postman para validar el flujo completo.

4.1.3 Fase 3: Desarrollo del frontend y conexión con el backend

- Creación del proyecto Angular y diseño del login con glassmorphism.
- Configuración de rutas con `AuthGuard` para proteger el dashboard.
- Conexión a la API usando `HttpClient` y almacenamiento del token JWT.
- Creación de los formularios para registrar movimientos financieros.
- Implementación del dashboard con saldo, ingresos, gastos y botones.

4.1.4 Fase 4: Funcionalidades adicionales y análisis

- Implementación del historial de transacciones.
- Lógica para mostrar solo los 3 últimos movimientos con opción de “ver más”.
- Cálculo del análisis porcentual por categoría.
- Creación del modal de análisis y gráfico de tipo `pie`.
- Validación para impedir ingresos negativos o saldo negativo.

4.1.5 Fase 5:

- Creación de los `Dockerfile` para backend y frontend.
- Integración en un `docker-compose.yml` con MySQL y phpMyAdmin.
- Ajustes de entorno para que todo se levante con un solo comando.
- Verificación de puertos, comunicación entre contenedores y persistencia de datos.

4.2 Planificación

- Semana 1: Backend + MySQL + seguridad JWT
- Semana 2: Frontend + login/registro + dashboard básico
- Semana 3: Funcionalidades completas (historial, añadir movimientos)
- Semana 4: Análisis, gráficas y validaciones
- Semana 5: Docker, limpieza, pruebas finales y documentación

4.3 Presupuesto

Concepto	Coste estimado
Hosting (no necesario por Docker local)	0 €
Base de datos MySQL	0 € (open source)
Frameworks y herramientas	0 € (open source)
Total	0 €

4.4 Entorno de trabajo (tecnologías de desarrollo y herramientas)

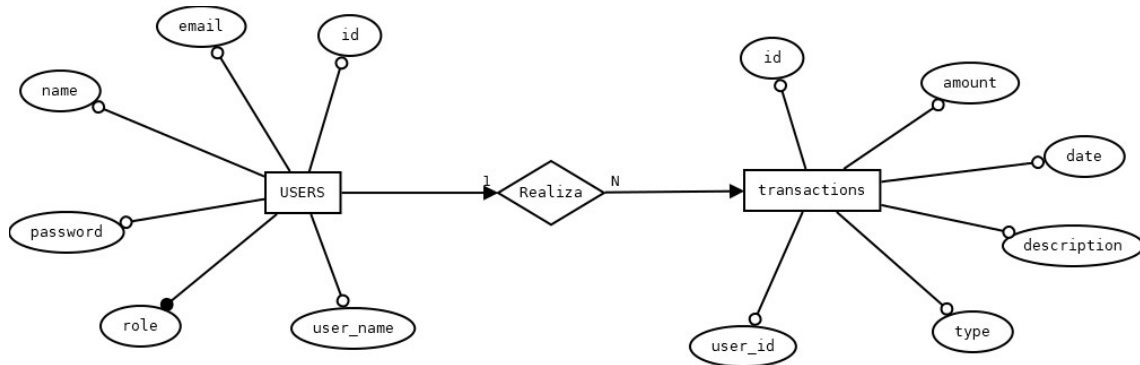
- **Frontend:**
 - Angular 17
 - Bootstrap
 - HTML/CSS (estilo glassmorphism)



- ng2-charts (para gráficas)
- **Backend:**
 - Java 17
 - Spring Boot 3
 - Spring Security + JWT
 - Maven
- **Base de datos:**
 - MySQL 8
 - PhpMyAdmin (interfaz de gestión)
- **Contenedores:**
 - Docker + Docker Compose
- **Herramientas adicionales:**
 - Postman (pruebas API)
 - Visual Studio Code (frontend)
 - Visual Studio Code (backend)
 - GitHub (control de versiones)
 - SweetAlert2 (alertas visuales)

5 Diseño e implementación de la base de datos

5.1 Diseño conceptual (diagrama ER)



5.2 Normalización y diseño lógico (tablas y relaciones)

5.3

tfg_db users	tfg_db transactions
id : int	id : bigint
email : varchar(255)	amount : double
name : varchar(255)	date : datetime(6)
password : varchar(255)	description : varchar(255)
role : varchar(255)	type : varchar(255)
user_name : varchar(12)	user_id : int

Implementación en SGBD

```

CREATE TABLE `users` (
  `id` int NOT NULL,
  `email` varchar(255) NOT NULL,
  `name` varchar(255) NOT NULL,
  `password` varchar(255) NOT NULL,
  `role` varchar(255) DEFAULT NULL,
  `user_name` varchar(12) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
    
```

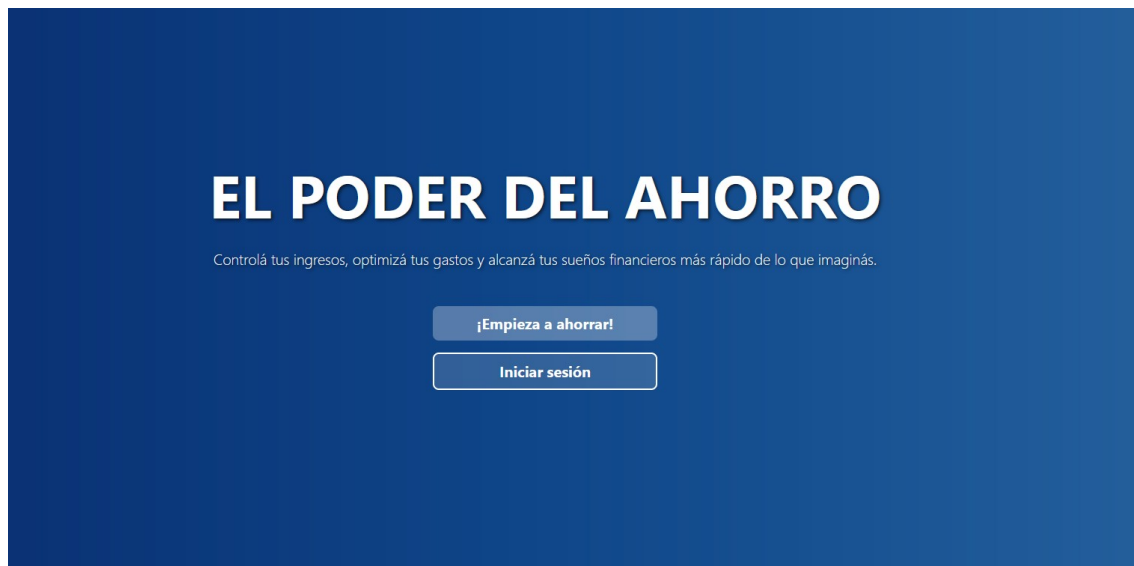
```

CREATE TABLE `transactions` (
  `id` bigint NOT NULL,
  `amount` double NOT NULL,
  `date` datetime(6) DEFAULT NULL,
  `description` varchar(255) DEFAULT NULL,
  `type` varchar(255) NOT NULL,
  `user_id` int DEFAULT NULL
) ;
    
```

6 Diseño de interfaz de usuario

Pantalla de Inicio (Home)

- Breve bienvenida al usuario.
- Enlace visual a registro o inicio de sesión.
- Estética cuidada con gradientes y tipografía clara.



Registro e Inicio de Sesión

- Formularios centrados en pantalla con efecto glass.
- Inputs validados con feedback visual.
- Uso de SweetAlert2 para notificaciones de éxito/error.
- Responsive y adaptado a móvil.



Iniciar Sesión

Entrar

¿No tienes cuenta? [Regístrate](#)

Inicia sesión'."/>

Crear Cuenta

Nombre completo

Nombre de usuario

Email

Contraseña

Crear cuenta

¿Tienes cuenta? [Inicia sesión](#)

Panel de Control (Dashboard)

- Mostrado del **saldo disponible, ingresos totales y gastos totales**.
- Botones animados para añadir ingresos/gastos con colores diferenciados.
- Historial de transacciones con colores (verde para ingresos, rojo para gastos).
- Visualización limitada a los **últimos 3 movimientos** con opción “Ver más”.



Modal de Añadir Movimiento

- Selector de tipo de movimiento (Ingreso o Gasto).
- Selector de categoría contextual (según tipo).
- Validación del importe (mínimo 0.01).
- Feedback visual claro al guardar.

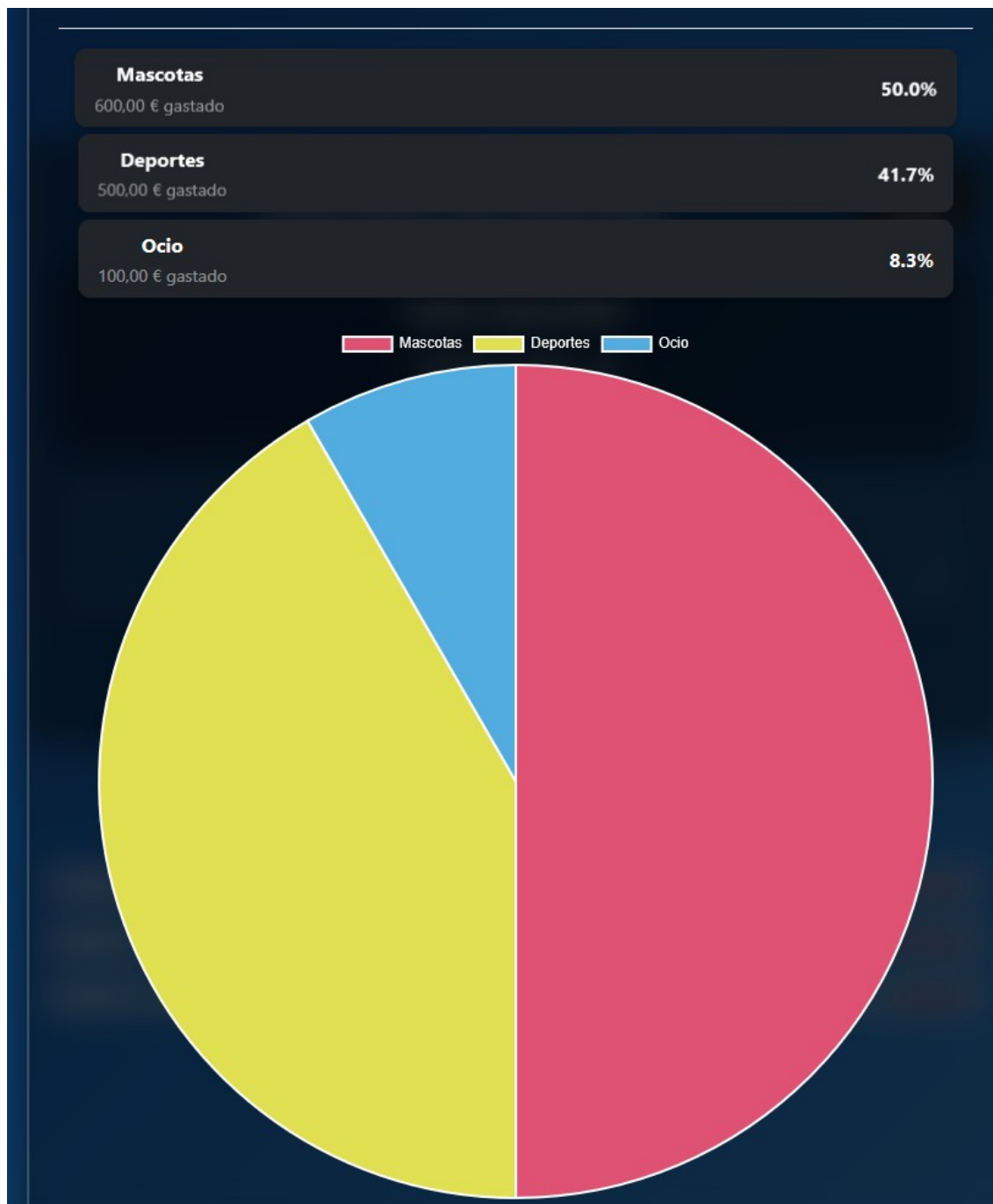


A screenshot of a web application modal titled "Añadir Ingreso" (Add Income). The modal has a dark blue background with a white border. It contains a text input field with the value "200", a dropdown menu with the selected option "Venta", and a blue button labeled "Guardar" (Save). A close button (X) is in the top right corner.

A screenshot of a web application modal titled "Añadir Gasto" (Add Expense). The modal has a dark blue background with a white border. It contains a text input field with the value "300", a dropdown menu with the selected option "Tecnología", and a blue button labeled "Guardar" (Save). A close button (X) is in the top right corner.

Modal de Análisis

- Lista de categorías con porcentaje y total gastado.
- Opción de visualizar un gráfico circular dinámico con `Chart.js` y `ng2-charts`.
- Modal completamente estilizado y accesible.



7 Estructura del proyecto: clases y elementos fundamentales del desarrollo

7.1 Estructura general

El proyecto se divide en dos grandes bloques: **Backend (Spring Boot)** y **Frontend (Angular)**, organizados en carpetas separadas y dockerizados para facilitar el despliegue conjunto.

Backend (/backend)

- `/entities`: Contiene las clases `User` y `Transaction`, que representan las entidades del modelo de datos.
- `/repositories`: Interfaces que extienden `CrudRepository` para acceder a la base de datos (`UserRepository`, `TransactionRepository`).
- `/services`: Contiene la lógica de negocio. Aquí se encuentra `TransactionService` y `UserService`.
- `/controllers`: API REST. Controladores que gestionan las peticiones HTTP (`UserController`, `TransactionController`).
- `/auth`: Lógica de autenticación JWT (`JwtAuthenticationFilter`, `JwtAuthorizationFilter`, `JwtUtil`, `SpringSecurityConfig`).

• **Frontend (/frontend)**

- `/app/pages/`: Componentes como `login`, `register`, `dashboard`, `home`.
- `/app/services/`: Servicios para comunicar con el backend.
- `/app/guards/`: `AuthGuard` protege rutas como el `dashboard`.
- Uso de **componentes standalone**, **Bootstrap** y **SweetAlert2**.
- Integración de `ng2-charts` y `Chart.js` para análisis gráfico.

7.2 Clases y Elementos

Backend

- User:
 - id, userName, email, password, role
 - Relación **1:N** con Transaction
- Transaction:
 - id, amount, date, description, type
 - Relación con User mediante @ManyToOne
- UserRepository, TransactionRepository:
 - Acceso a los datos con métodos automáticos.
- TransactionService:
 - Guarda movimientos y genera análisis mensual y por categoría.
- UserController / TransactionController:
 - Exponen endpoints /api/users, /api/transactions.
- Seguridad:
 - SpringSecurityConfig, JWT filters, token en headers.

Frontend

- login.component.ts y register.component.ts:
 - Formularios reactivos, validación, llamada al backend.
- dashboard.component.ts:
 - Carga y muestra saldo, movimientos, análisis de gastos, modal de añadir movimiento, y modal de análisis gráfico.
 - Contiene lógica para calcular el porcentaje por categoría y limitar la vista a los últimos 3 movimientos.
- auth.guard.ts:
 - Protege rutas privadas si no hay token válido.

8 Pruebas

8.1 Tipo de pruebas realizadas

En este proyecto se han llevado a cabo **pruebas funcionales manuales**, centradas en verificar que las funcionalidades principales del sistema cumplen con los requisitos establecidos.

8.2 Pruebas realizadas y resultados

Funcionalidad	Entrada	Resultado esperado	Resultado obtenido	Estado
Registro usuario	de Formulario datos válidos	con Usuario correctamente	registrado Usuario registrado	✅ OK
Registro datos inválidos	con Campos vacíos o incorrectos	Mensaje de error	Validación y error mostrado	✅ OK
Inicio de sesión	Usuario registrado credenciales correctas	y Acceso al dashboard	Acceso correcto	✅ OK
Inicio de sesión con error	Contraseña incorrecta	Mensaje de error	Error mostrado	✅ OK
Añadir ingreso	Formulario cantidad categoría	con Ingreso y saldo	sumado al Ingreso registrado	✅ OK
Añadir gasto	Formulario cantidad categoría	con Gasto y saldo	restado del Gasto registrado	✅ OK

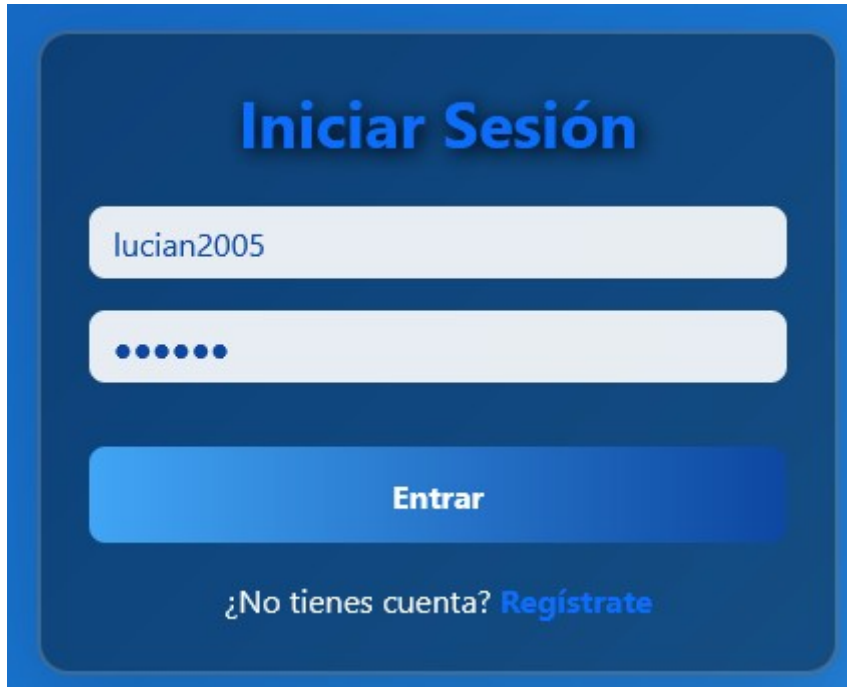


Funcionalidad	Entrada	Resultado esperado	Resultado obtenido	Estado
Control de saldo	Saldo nunca debe ser negativo	El saldo se mantiene en 0 como mínimo	Comprobado y correcto	✓ OK
Ver historial	Dashboard	Muestra los últimos 3 movimientos con opción "ver más"	Funciona correctamente	✓ OK
Modal de análisis	Click en botón "Análisis"	Muestra porcentaje de Datos y gráfico gastos por categoría cargados		✓ OK
Cierre de sesión	Click en botón	Usuario redirigido al login	Funciona correctamente	✓ OK

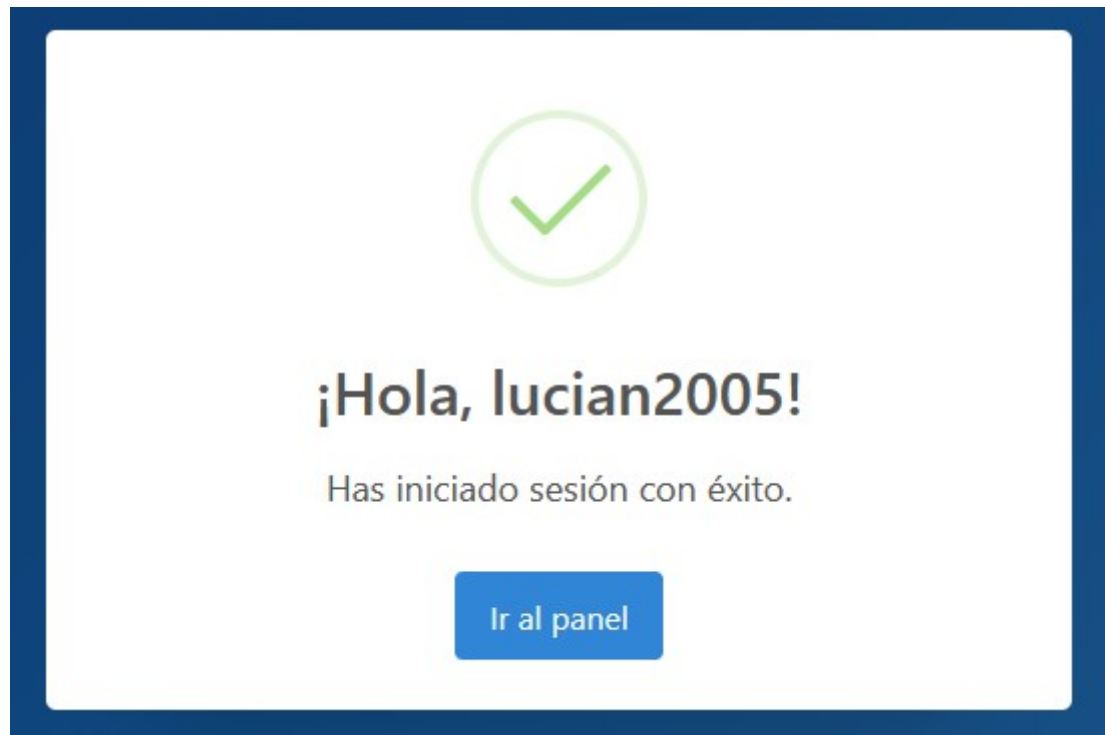
9 Manual de usuario

1. Inicio de sesión

Al acceder a la aplicación, se muestra una pantalla de inicio de sesión donde el usuario debe introducir su nombre de usuario y contraseña previamente registrados. En caso de datos incorrectos, se mostrará un mensaje de error.



The screenshot shows a login interface with a dark blue background. At the top, the text "Iniciar Sesión" is displayed in a large, bold, light blue font. Below this, there are two input fields: the first contains the username "lucian2005" and the second contains a masked password represented by six dots. A prominent blue button with the text "Entrar" is positioned below the password field. At the bottom, there is a link that reads "¿No tienes cuenta? [Regístrate](#)".

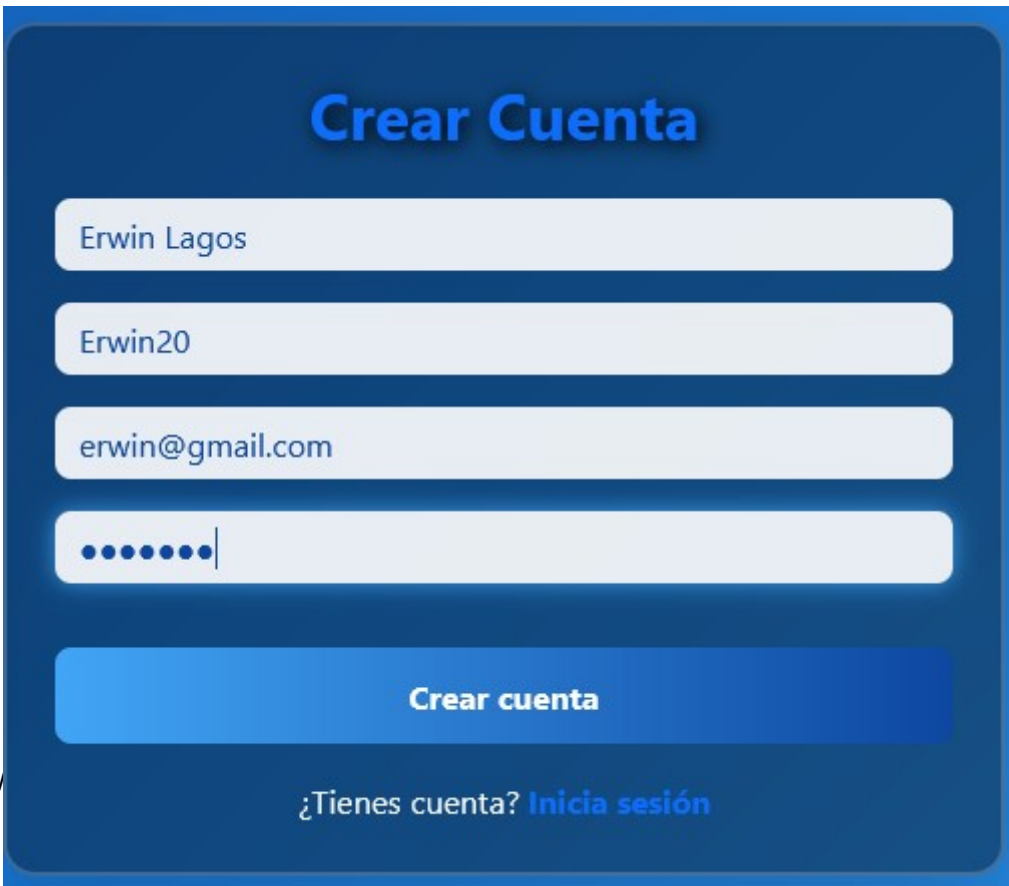


D

2. Registro de usuario

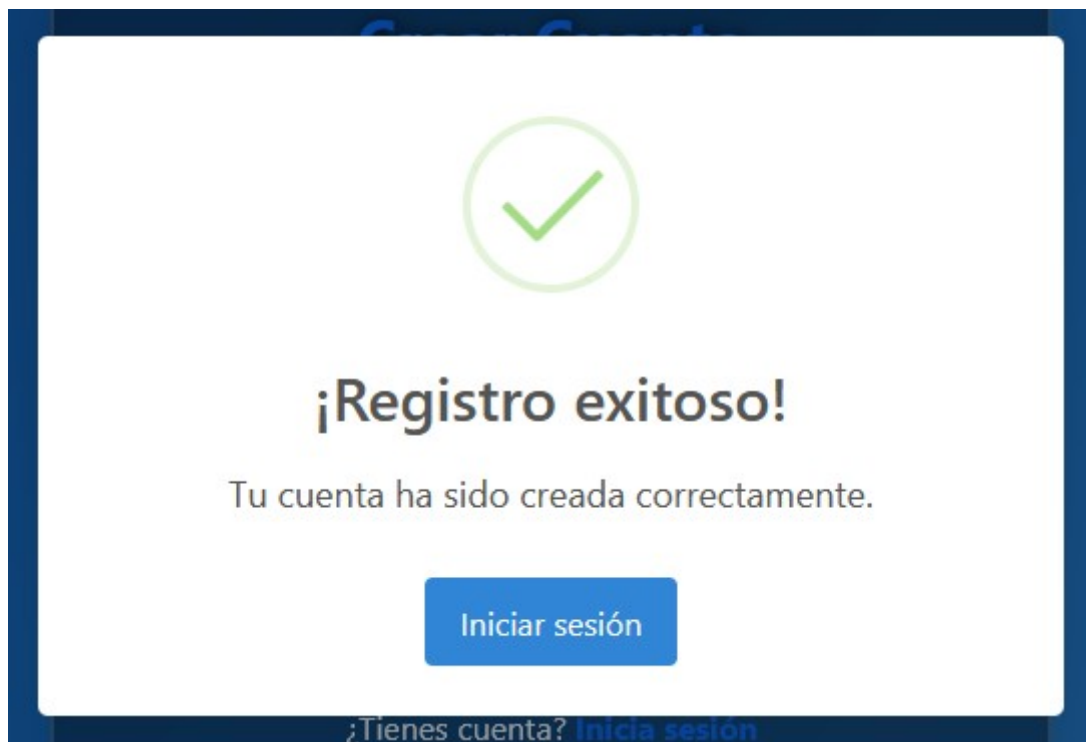
Si el usuario no tiene cuenta, puede acceder al formulario de registro donde deberá introducir su nombre completo, nombre de usuario, correo electrónico y una contraseña.

Todos los campos son obligatorios y están validados.

A screenshot of a "Crear Cuenta" (Create Account) form. The title "Crear Cuenta" is at the top in a large, bold, blue font. Below it are four input fields: the first contains "Erwin Lagos", the second contains "Erwin20", the third contains "erwin@gmail.com", and the fourth contains seven dots representing a password. Below these fields is a large blue button with the text "Crear cuenta" in white. At the bottom, there is a link that says "¿Tienes cuenta? [Inicia sesión](#)".

DAW

e 35



3. Panel de control (dashboard)

Una vez iniciada la sesión, se accede al panel principal donde se muestra:

- **Saldo disponible**
- **Total de ingresos**
- **Total de gastos**

También se encuentran botones para añadir nuevos ingresos o gastos, así como un botón para ver un análisis detallado.



4. Añadir ingreso o gasto

Al hacer clic en "Añadir ingreso" o "Añadir gasto", se abre un modal donde el usuario puede introducir:

- La cantidad (€)
- La categoría (por ejemplo, comida, transporte, nómina, etc.)

Una vez completado el formulario, se guarda y actualiza el balance.



Añadir Ingreso

150

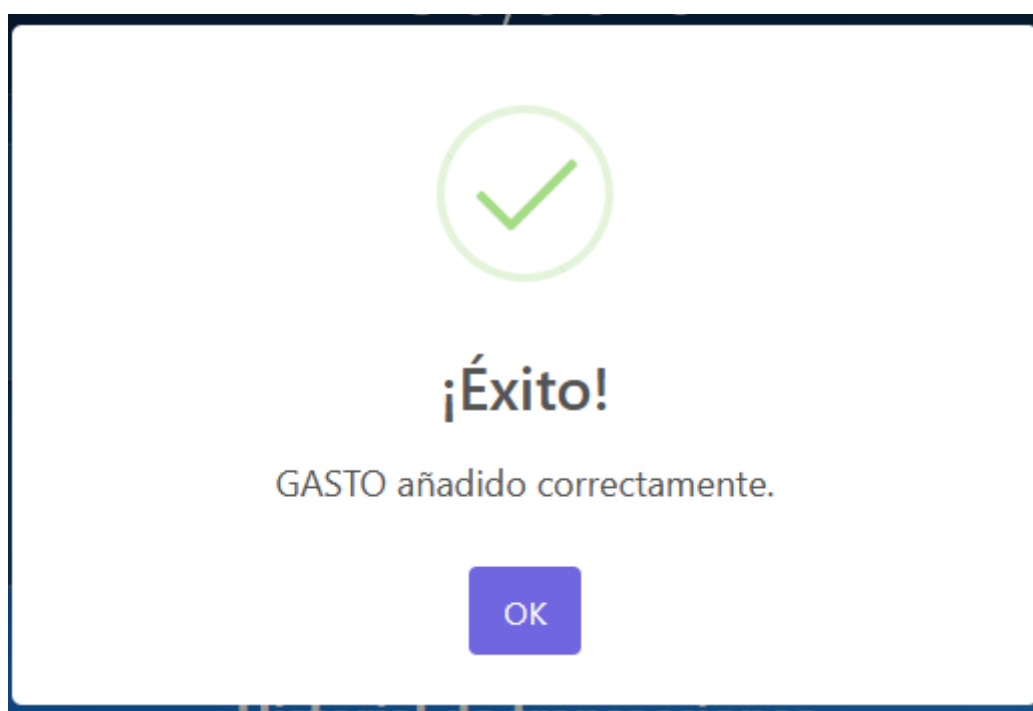
Nómina

Guardar





A screenshot of a mobile application's 'Añadir Gasto' (Add Expense) modal. The modal has a dark blue background with a white title bar at the top containing the text 'Añadir Gasto' and a close button (X). Below the title bar, there are two input fields: a numeric field with the value '200' and a dropdown menu with the selected option 'Mascotas'. At the bottom of the modal is a large blue button with the text 'Guardar' (Save).



5. Historial de transacciones

En la parte inferior del panel, se muestra el historial de transacciones. Por defecto, se muestran las 3 últimas operaciones. El usuario puede hacer clic en "Ver más" para desplegar todas las transacciones registradas.



Historial de transacciones	
GASTO — Mascotas	200,00 €
INGRESO — Nómina	150,00 €
GASTO — Deportes	500,00 €
Ver más	

Historial de transacciones	
GASTO — Mascotas	200,00 €
INGRESO — Nómina	150,00 €
GASTO — Deportes	500,00 €
GASTO — Mascotas	400,00 €
GASTO — Mascotas	200,00 €
GASTO — Ocio	100,00 €
INGRESO — Venta	1500,00 €
Ver menos	

6. Análisis de gastos

Desde el botón "Análisis", se accede a un modal que muestra un desglose de los gastos:

- Porcentaje por categoría
- Total gastado por categoría
- Representación visual en gráfico circular



Análisis de Gastos

Mascotas

800,00 € gastado

57.1%

Deportes

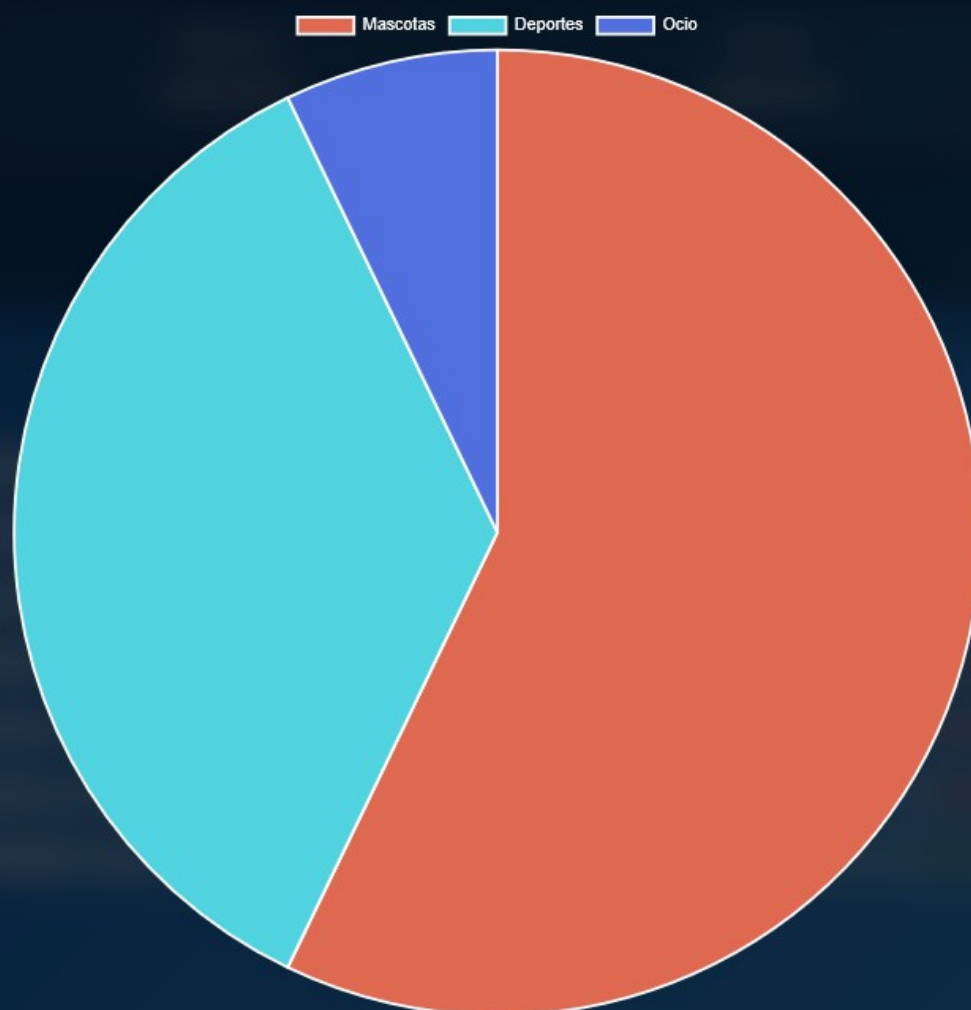
500,00 € gastado

35.7%

Ocio

100,00 € gastado

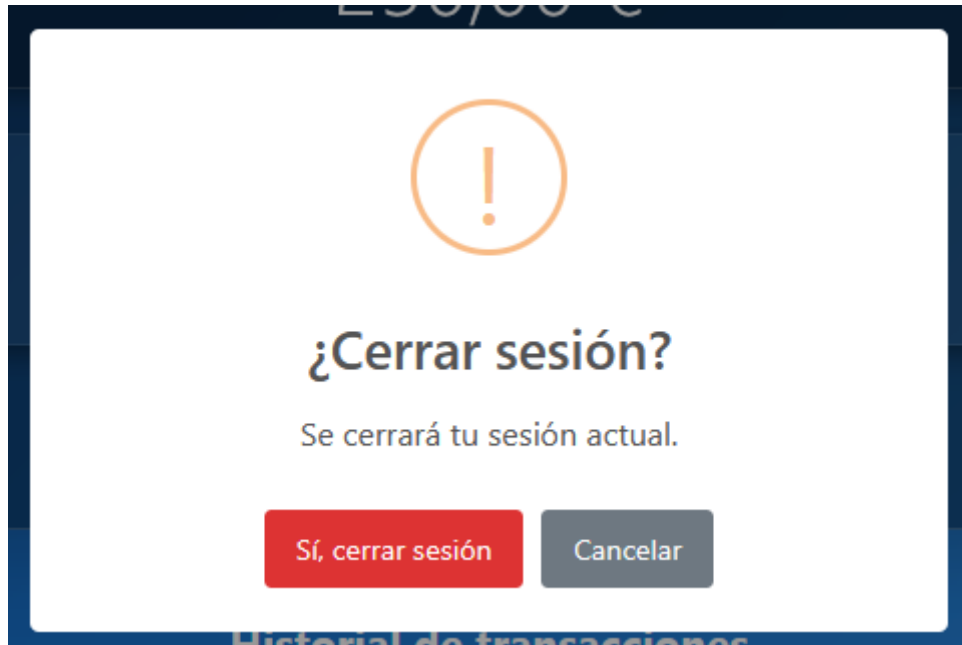
7.1%



7. Cierre de sesión

El usuario puede cerrar sesión desde el botón ubicado en la parte superior derecha del dashboard.

Se mostrará una confirmación y, al aceptarla, el sistema redirige al inicio de sesión.



10 Desarrollo evolutivo de la aplicación

La aplicación ha sido desarrollada de forma incremental, comenzando con una versión básica que fue evolucionando a lo largo del proyecto mediante sucesivas mejoras funcionales, técnicas y visuales. A continuación se detalla cómo ha evolucionado la aplicación desde su concepción inicial hasta su versión final.

Versión inicial

En la primera versión del proyecto se implementó lo esencial:

- Registro e inicio de sesión con validación.
- Almacenamiento de usuarios en base de datos.
- Panel con saldo total, ingresos y gastos acumulados.
- Añadir ingresos o gastos de forma sencilla con su tipo y cantidad.

Mejoras funcionales

Posteriormente se añadieron nuevas funcionalidades que enriquecen la experiencia de usuario:

- **Historial de transacciones** con visualización ordenada y limitada por defecto (solo los 3 últimos).
- **Botón “Ver más”** para consultar el historial completo.
- **Selección de categoría** al añadir cada ingreso o gasto.
- Modal con **análisis de gastos** mostrando porcentajes y totales por categoría.

Análisis visual

Se incorporó un gráfico circular dinámico que representa la distribución de gastos por categorías, permitiendo una mejor interpretación de los datos. Esto supuso la integración de bibliotecas externas como `ng2-chart.js`.

Estilo visual

Durante el desarrollo se aplicó un diseño coherente y moderno, con:

- Fondos con gradiente.
- Efecto *glassmorphism* en los componentes principales.

- Animaciones sutiles para mejorar la experiencia de usuario.

Contenedores Docker

En fases finales se dockerizó tanto el backend como el frontend junto con la base de datos MySQL y PhpMyAdmin, permitiendo ejecutar toda la aplicación con un único docker -compose.

Posibles mejoras futuras

Aunque el producto final cumple con todos los requisitos, se han identificado posibles mejoras futuras:

- Soporte multicuenta o grupos familiares.
- Filtros por fechas o categorías.
- Exportar los datos a PDF o Excel.
- Incorporación de notificaciones por objetivos de ahorro.

11 Bibliografía y documentación de referencia

Durante el desarrollo del proyecto se ha consultado y utilizado la siguiente documentación y recursos:

- **Curso de Udemey:**
 - *Desarrollo web fullstack con Spring Boot, Angular y MySQL*. Este curso fue fundamental para entender la estructura de una aplicación fullstack, así como la integración entre frontend y backend.
- **ChatGPT de OpenAI:**
 - Utilizado como asistente para resolver dudas puntuales, optimizar fragmentos de código, mejorar la redacción técnica de la documentación, estructurar los componentes del frontend y backend, y obtener buenas prácticas de programación.