SMART FINANCE SOLUTIONS, Sistema integral de organización y análisis de finanzas personales

SMART FINANCE SOLUTIONS, An integrated system for personal finance organization and analysis

Trabajo de Fin de Grado
Curso 2025–2026

Autor
Daniela Valentina Valera Fuentes 9,5
Miguel Martínez-Almeida Nistal 9,5

Director
Ramón González del Campo Rodríguez Barbero

Colaborador
Ninguno

Grado en Ingeniería de Software
Facultad de Informática
Universidad Complutense de Madrid

SMART FINANCE SOLUTIONS, Sistema integral de organización y análisis de finanzas personales
SMART FINANCE SOLUTIONS, An integrated system for personal finance organization and analysis

Trabajo de Fin de Grado en Ingeniería de Software

Autor
Daniela Valentina Valera Fuentes 9,5
Miguel Martínez-Almeida Nistal 9,5

Director
Ramón González del Campo Rodríguez Barbero

Colaborador
Ninguno

Convocatoria: *Enero 2026*

Grado en Ingeniería de Software
Facultad de Informática
Universidad Complutense de Madrid

20 de ENERO de 2026

Dedicatoria

A nuestras familias, por soportar con paciencia nuestras noches de cafés interminables. A nuestros profesores, por enseñarnos más de lo que imaginábamos y no rendirse con nuestras preguntas absurdas. A nuestros amigos, por las risas, los momentos de tensión y las largas jornadas compartidas que hicieron más llevadero este camino.

Agradecimientos

Este proyecto no habría sido posible sin todas las personas que estuvieron a nuestro lado, dentro y fuera de las aulas. Es el resultado de años de constancia, de desarrollo personal, de aciertos y sobretodo, aprendizaje de nuestros errores.

Por eso, queremos agradecerle a nuestros profesores, por enseñarnos con paciencia y pasión contribuyendo a nuestro aprendizaje y crecimiento académico. A nuestro tutor, por su disponibilidad y orientación, al ser nuestro guía en cada etapa del proyecto.

A nuestras familias, por su apoyo incondicional y por enseñarnos el valor del esfuerzo y la constancia. A nuestros amigos, por su ánimo y compañía en los momentos más difíciles.

A todas las personas que han confiado en nosotros, cada línea de este trabajo lleva un pedacito de todos vosotros. Por eso, nuestro agradecimiento es enorme y sincero.

Resumen

SMART FINANCE SOLUTIONS, Sistema integral de organización y análisis de finanzas personales

Smart Finance Solutions es una aplicación web orientada a la gestión de las finanzas personales, cuyo objetivo es facilitar al usuario el control y seguimiento de sus ingresos, gastos y presupuestos de forma intuitiva y eficiente.

La plataforma proporciona herramientas de visualización y análisis financiero que apoyan la toma de decisiones económicas, permitiendo al usuario comprender su situación financiera de manera clara y estructurada. Asimismo, incorpora componentes sociales y elementos de gamificación que fomentan la interacción, el compromiso y la continuidad en el uso de la aplicación.

Smart Finance Solutions no solo busca simplificar la gestión del dinero, sino también ofrecer recomendaciones personalizadas basadas en los patrones de consumo y comportamiento financiero del usuario, sentando las bases para la integración de técnicas de análisis predictivo orientadas a una planificación económica más consciente y eficiente.

El desarrollo de la aplicación se ha llevado a cabo siguiendo una metodología ágil, lo que ha permitido una evolución iterativa del sistema y la incorporación progresiva de mejoras en función de los objetivos del proyecto.

Palabras clave

Finanzas, Gestión, Análisis predictivo.

Abstract

SMART FINANCE SOLUTIONS, An integrated system for personal finance organization and analysis

Smart Finance Solutions is a web application for managing personal finances, their main objective is to make it easier for users to keep track of their income, expenses, and budgets in a simple and practical way.

The platform offers tools to visualize and analyze finances, helping users understand their financial situation clearly and easily. It also includes social features and gamification elements that make using the app more fun and engaging.

Smart Finance Solutions doesn't just simplify money management—it also gives personalized tips based on how users spend and manage their money. This sets the stage for using predictive analysis to plan finances more smartly and efficiently.

The application was developed using an agile approach, which allowed the system to evolve step by step and gradually include improvements based on the project goals.

Keywords

Finance, Management, Predictive Analysis.

Índice

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

"No ahorres lo que te queda después de gastar,
gasta lo que te queda después de ahorrar"
— Warren Buffett

La creciente digitalización de la sociedad ha transformado la forma en que las personas gestionan su economía personal. El uso habitual de pagos electrónicos, suscripciones y servicios financieros digitales, entre otros, genera una gran cantidad de información que, si no se organiza adecuadamente, puede dificultar el control y la planificación financiera a medio y largo plazo.

En este contexto, surge la necesidad de herramientas que no solo permitan registrar información económica, sino que ayuden a interpretarla de manera clara y útil. La gestión financiera personal deja de ser un proceso meramente administrativo para convertirse en un apoyo fundamental en la toma de decisiones cotidianas, como el control del gasto, la planificación de objetivos económicos o la detección de hábitos de consumo poco eficientes.

Este Trabajo de Fin de Grado aborda dicha problemática mediante el desarrollo de una solución orientada al usuario, poniendo especial énfasis en la simplicidad, la claridad de la información y el apoyo a la toma de decisiones.

A lo largo de esta memoria se aborda el desarrollo de una solución orientada a la gestión de las finanzas personales, partiendo del análisis del contexto actual y de las necesidades que surgen en el ámbito de la planificación económica individual. Este enfoque inicial permite establecer las bases del proyecto, definir los objetivos perseguidos y justificar las decisiones adoptadas durante su desarrollo, tanto a nivel funcional como técnico.

1.1. Motivación

En la actualidad, la gestión de las finanzas personales se ha convertido en una necesidad fundamental para una gran parte de la población. El aumento del coste de vida, la diversidad de gastos recurrentes y la facilidad para realizar pagos digitales hacen que muchas personas pierdan el control sobre sus ingresos y gastos, lo que dificulta la planificación económica y la toma de decisiones financieras responsables. A pesar de la existencia de múltiples herramientas financieras, muchas de ellas resultan complejas, poco intuitivas o no se adaptan a las necesidades reales de los usuarios. Esta situación pone de manifiesto la importancia de contar con aplicaciones accesibles que permitan llevar un control claro y organizado de la economía personal.

La motivación principal de este Trabajo de Fin de Grado surge de la necesidad de desarrollar una solución que simplifique la gestión financiera, facilitando el seguimiento de ingresos, gastos y presupuestos de forma visual, comprensible y eficiente.

La incorporación de elementos como la gamificación, la interacción entre usuarios y la visualización clara del progreso económico contribuyen a aumentar la motivación del usuario y a mejorar su implicación en el seguimiento continuo de su situación financiera, transformando la gestión económica en un proceso más dinámico y accesible.

Por otro lado, la incorporación de herramientas de análisis responde a la motivación de ir más allá del simple registro de datos, aportando valor añadido al usuario mediante información útil que apoye la toma de decisiones económicas. De este modo, la aplicación no solo actúa como un registro financiero, sino como un apoyo para fomentar una planificación económica más consciente, responsable y sostenible.

El desarrollo de Smart Finance Solutions permite aplicar y consolidar conocimientos adquiridos a lo largo del grado, como el diseño de aplicaciones web, el desarrollo de sistemas backend y frontend, la gestión de bases de datos y la implementación de sistemas seguros de autenticación de usuarios. Además, ofrece la oportunidad de adentrarse en áreas desconocidas y adquirir nuevas competencias, aprendiendo técnicas, herramientas y metodologías que enriquecen la formación académica y profesional, como el uso de librerías especializadas o la implementación de técnicas de análisis predictivo, entre otras.

En conclusión, la motivación de este Trabajo de Fin de Grado se basa tanto en la relevancia social del problema abordado como en el interés personal y académico por desarrollar una aplicación web completa, práctica y orientada a mejorar la gestión de las finanzas personales mediante el uso de tecnologías actuales.

1.2. Objetivos

En este apartado se presentan los objetivos del desarrollo de Smart Finance Solutions, orientados a proporcionar una herramienta completa y flexible para la gestión de las finanzas personales y el análisis de la información económica del usuario.

- Garantizar la seguridad y confidencialidad de los datos mediante un sistema de registro e inicio de sesión seguro.
- Diseñar interfaces visuales e intuitivas que faciliten la comprensión y seguimiento de la situación económica del usuario.
- Desarrollar funcionalidades de gestión de presupuestos, ingresos y gastos, incluyendo clasificación, filtrado y análisis eficiente de la información financiera.
- Incorporar herramientas de análisis predictivo basadas en patrones de consumo, para apoyar una planificación económica consciente.
- Implementar un panel de control (dashboard) que centralice la información financiera, mostrando resúmenes visuales atractivos para el usuario.
- Incluir funcionalidades avanzadas como la gestión de carteras y monederos, definición de metas financieras y un sistema de gestión de gastos compartidos.
- Proporcionar acceso en tiempo real a la información del valor de mercado de criptomonedas y acciones, permitiendo al usuario tomar decisiones financieras más informadas.
- Fomentar la participación y constancia del usuario mediante elementos de gamificación y sociales, como foros, chat con amigos, reseñas, logros y notificaciones.
- Asegurar la escalabilidad y estabilidad del sistema para soportar un crecimiento del número de usuarios y del volumen de datos.
- Validar la aplicación mediante pruebas funcionales y de usabilidad, asegurando que cumpla con los requisitos del usuario y proporcione una experiencia atractiva y accesible.

1.3. Metodología del trabajo

Para el desarrollo de Smart Finance Solutions se ha seguido una metodología basada en Scrum, una de las metodologías ágiles más utilizadas en el desarrollo de software, que permite organizar el trabajo en ciclos iterativos, promoviendo la entrega continua de funcionalidades y la adaptación a cambios de forma rápida y eficiente.

Metodología ágil

- Se centra en entregas parciales y frecuentes, donde el producto se desarrolla en iteraciones cortas (sprints).
- Permite adaptar requisitos y funcionalidades conforme se obtiene retroalimentación de usuarios o se detectan nuevas necesidades.

Desarrollo Iterativo e Incremental

- **Iterativo:** se repite el ciclo de análisis, diseño, implementación y pruebas varias veces. Cada iteración mejora y refina lo hecho previamente.
- **Incremental:** se agregan funcionalidades nuevas de manera gradual, formando versiones cada vez más completas del sistema.

Esta metodología es ideal cuando los requisitos no están completamente claros al inicio o pueden cambiar. Con ella, se garantiza un desarrollo organizado, flexible y centrado en las necesidades del usuario, asegurando que la aplicación sea funcional, segura y de fácil uso.

Durante el proyecto se utilizaron herramientas de gestión de tareas y comunicación que facilitaron la coordinación del equipo y el seguimiento del progreso del desarrollo.

Al inicio se empleó **Trello** para organizar y priorizar tareas, y posteriormente se migró a **Jira**, conectándolo con un servidor de **Discord** que fomenta la comunicación. Esta integración permitía recibir notificaciones automáticas cada vez que se creaba una nueva tarjeta en Jira, manteniendo al equipo informado en tiempo real sobre avances, incidencias y nuevas tareas.

Además, esta combinación de herramientas permitió asignar responsabilidades de manera clara, hacer seguimiento de los sprints y documentar decisiones y cambios, asegurando que todo el equipo estuviera sincronizado y que las iteraciones cumplieran con los objetivos establecidos, reforzando la transparencia y la eficiencia del proceso de desarrollo.

1.4. Plan de trabajo

1.4.1. Especificación de requisitos

1. Análisis de requisitos

   El desarrollo del proyecto comenzó con una fase de análisis orientada a identificar las necesidades generales del usuario y definir el alcance funcional del sistema.

   Para ello, se realizó un estudio de aplicaciones similares de gestión financiera, lo que permitió identificar patrones comunes, buenas prácticas y oportunidades de mejora que sirvieron como referencia para el diseño inicial de Smart Finance Solutions.

   - **Requisitos funcionales:** definición de las capacidades que debía ofrecer la aplicación para permitir al usuario gestionar, analizar y consultar su información financiera de forma estructurada, así como interactuar con el sistema de manera eficiente y personalizada.
   - **Requisitos no funcionales:** identificación de criterios que garantizaran un funcionamiento fiable y fluido de la aplicación, garantizando la protección de los datos del usuario, una experiencia de uso ágil y a un diseño que permitiera la evolución futura del sistema sin comprometer su estructura.

   Este análisis permitió establecer una base flexible sobre la que diseñar e implementar las distintas funcionalidades del sistema, manteniendo la posibilidad de adaptación a nuevas necesidades detectadas durante el desarrollo.

2. Diseño del sistema

   Una vez definidos los requisitos, se abordó el diseño del sistema con el objetivo de establecer una estructura clara y coherente que facilitara tanto el desarrollo como el mantenimiento de la aplicación.

   - Definición de la arquitectura general de la aplicación web y de la estructura de la base de datos, asegurando una correcta organización de la información.
   - Diseño de interfaces orientadas a la experiencia de usuario (UX/UI), priorizando la claridad visual y la facilidad de navegación.
   - Elaboración de diagramas de flujo y modelos de datos para garantizar la correcta estructura del sistema.

3. Implementación

   La fase de implementación se llevó a cabo de forma progresiva, siguiendo un enfoque iterativo e incremental acorde con la metodología Scrum adoptada.

   - Desarrollo iterativo de las funcionalidades, organizando el trabajo en tareas y sprints que permitieran una evolución continua del sistema.

- Implementación progresiva de las distintas funcionalidades, priorizando aquellas de mayor valor para el usuario y permitiendo incorporar mejoras a lo largo del desarrollo.

4. Pruebas y validación

   Paralelamente al desarrollo, se realizaron tareas de prueba y validación con el fin de garantizar la calidad y fiabilidad de la aplicación.

   - Realización de pruebas unitarias e integración para verificar el correcto funcionamiento de los distintos módulos del sistema.
   - Pruebas de usabilidad con usuarios potenciales para detectar posibles mejoras en la interfaz y en la experiencia de uso.
   - Corrección de errores detectados y optimización del rendimiento general de la aplicación.

5. Documentación y presentación

   La última fase del plan de trabajo estuvo centrada en la documentación del proyecto y la preparación de su presentación final.

   - Elaboración de la memoria del Trabajo de Fin de Grado utilizando LaTeX, herramienta especialmente adecuada para la redacción de documentos técnicos de gran extensión. El uso de LaTeX ha permitido mantener una estructura clara y coherente, garantizar una alta calidad tipográfica y separar el contenido del formato, favoreciendo la escalabilidad y mantenibilidad de la documentación.
   - Preparación de la presentación final y del material de apoyo necesario para la defensa del proyecto. Esta fase incluyó la elaboración de una presentación estructurada y clara que resumiera los aspectos más relevantes del trabajo. Asimismo, se diseñaron elementos visuales de apoyo que facilitaran la comprensión del proyecto durante la exposición, prestando especial atención a la claridad, la coherencia visual y la gestión del tiempo, con el objetivo de comunicar de forma eficaz el trabajo realizado ante el tribunal.

Herramientas de trabajo

Para el desarrollo del proyecto se utilizaron diversas herramientas que facilitaron la implementación, el control del código y la coordinación del trabajo:

- **Entorno de desarrollo:** se utilizó **Visual Studio Code** como editor principal, debido a su versatilidad, ligereza y amplia disponibilidad de extensiones, así como a su integración con sistemas de control de versiones. Además de la familiaridad de la herramienta.
- **Control de versiones:** se emplearon **Git** y **GitHub** para gestionar el código fuente del proyecto, permitiendo el seguimiento de cambios, la recuperación de versiones anteriores y el mantenimiento de un historial completo del desarrollo.
- **Gestión y comunicación:** se utilizaron herramientas descritas anteriormente tales como **Trello**, **Jira** y **Discord** que permitieron una organización eficiente del trabajo y una comunicación fluida durante el desarrollo.
- **Pruebas y depuración:** se recurrió a las herramientas de desarrollo del navegador para la detección de errores y la validación del correcto funcionamiento de la aplicación.

Figura 1.1: Logo aplicación Visual Studio Code

Figura 1.2: Logo aplicación GitHub

1.4.2. Planificación

La planificación del proyecto se estructuró siguiendo los principios de Scrum, organizando el trabajo en sprints de duración variable que permitieran una entrega continua de funcionalidades.

Cada sprint incluía la definición de objetivos parciales, asignación de tareas y estimación de tiempos, asegurando que el desarrollo se mantuviera alineado con los requisitos y las prioridades del proyecto.

Aunque inicialmente se plantearon sprints semanales, estos se fueron adaptando en función de la carga de trabajo, la complejidad de las funcionalidades y la disponibilidad, lo que permitió una gestión más realista y flexible del proyecto.

Al final de cada sprint se realizaban reuniones de Sprint Review y Retrospectiva, donde se evaluaban los avances, se recogía la retroalimentación de los usuarios y se ajustaban los objetivos de los siguientes sprints.

Esta planificación iterativa permitió responder de manera flexible a cambios en los requisitos, optimizar el uso de recursos y mejorar continuamente la calidad de la aplicación.

1.4.3. Desarrollo

Durante la fase de desarrollo, las funcionalidades de Smart Finance Solutions se implementaron de forma incremental, priorizando en las primeras iteraciones aquellas que permitían al usuario utilizar la aplicación de manera efectiva desde el inicio, como la gestión básica de su información financiera. Cada iteración dio lugar a versiones funcionales del sistema, que eran evaluadas y mejoradas de forma continua.

Seguidamente, se procedió a implementar aquellas funcionalidades que aportaban diferenciación en el mercado y un valor añadido al usuario, asegurando que la aplicación ofreciera características relevantes y competitivas. Al mismo tiempo, se realizaban ajustes y mejoras sobre las funcionalidades ya implementadas, refinando su comportamiento y optimizando la experiencia de uso.

Se realizaron pruebas de manera regular con el objetivo de detectar errores de forma temprana y asegurar un funcionamiento estable de la aplicación. Además, se llevaron a cabo pruebas de usabilidad con usuarios potenciales, lo que permitió ajustar la interfaz y mejorar la experiencia de navegación.

El desarrollo iterativo e incremental permitió que cada versión del sistema fuera funcional y probada, asegurando que la aplicación evolucionara de manera coherente con los objetivos iniciales del proyecto, incorporando mejoras y refinando requisitos conforme avanzaba el desarrollo.

Capítulo 2

Estado de la Cuestión

2.1. Aplicaciones similares

El sector de las aplicaciones de gestión de finanzas personales ha experimentado un crecimiento significativo en los últimos años, impulsado por la digitalización de los servicios financieros, la diversificación de los medios de pago y la necesidad de los usuarios de disponer de herramientas que faciliten el control y la planificación de su economía personal. En este contexto, han surgido numerosas aplicaciones que abordan la gestión financiera desde diferentes enfoques, tales como el control de gastos, la planificación presupuestaria, la inversión o la gestión compartida de gastos.

Para el desarrollo de Smart Finance Solutions se han analizado diversas aplicaciones representativas del sector, que han servido como referencia para identificar buenas prácticas, limitaciones existentes y oportunidades de mejora. Entre las aplicaciones estudiadas destacan **YNAB (You Need a Budget)**, **Empower**, **Fintonic** y **Tricount**, así como el uso tradicional de **hojas de cálculo (Excel)**, ampliamente extendido entre los usuarios.

YNAB (You Need a Budget) se centra en la planificación activa del presupuesto, promoviendo una metodología basada en la asignación consciente de cada ingreso a categorías específicas de gasto. Su enfoque educativo ayuda a los usuarios a adquirir hábitos financieros más responsables y a mantener un control estricto de su economía.

Figura 2.1: Logo aplicación YNAB

No obstante, YNAB presenta una curva de aprendizaje elevada para usuarios que buscan una gestión más sencilla e inmediata. Además, opera bajo un modelo de suscripción de pago, lo que puede suponer una barrera de entrada. La aplicación no incorpora funcionalidades relacionadas con inversiones, análisis predictivo

avanzado, elementos sociales o gamificación, limitando su alcance a la planificación presupuestaria tradicional.

De YNAB se ha tomado como referencia su enfoque en la claridad y la concienciación del usuario sobre su situación financiera. Smart Finance Solutions adopta esta filosofía, pero simplificando el proceso de gestión presupuestaria y reduciendo la carga metodológica, con el objetivo de ofrecer una experiencia más inmediata y accesible, especialmente para usuarios que buscan un control financiero sin una curva de aprendizaje elevada.

Empower es una plataforma orientada principalmente a la gestión patrimonial y la inversión. Ofrece una visión consolidada de activos y pasivos, análisis avanzados de carteras de inversión y herramientas para evaluar la diversificación y los costes asociados a los productos financieros.

Si bien proporciona funcionalidades muy completas en el ámbito de la inversión, su enfoque está dirigido a usuarios con un patrimonio elevado, y muchas de sus prestaciones avanzadas requieren la contratación de servicios de gestión patrimonial. Como consecuencia, resulta menos accesible para usuarios que buscan una solución integral para la gestión financiera cotidiana.

En este sentido, Smart Finance Solutions toma como referencia a Empower en lo relativo a la gestión de inversiones y a la visualización conjunta de activos y pasivos del usuario, pero adaptadas a un enfoque más sencillo y accesible, orientado a usuarios sin un alto nivel de especialización financiera. El objetivo es ofrecer una visión clara y comprensible del estado financiero global del usuario.

Figura 2.2: Logo aplicación Empower

Fintonic, por su parte, es una aplicación española de gestión financiera personal que se ha consolidado como una de las plataformas más completas y populares en el mercado español e iberoamericano.

La aplicación permite la conexión automática con entidades financieras, lo que facilita la sincronización en tiempo real de cuentas bancarias, tarjetas, préstamos y otros productos financieros. A partir de esta información, Fintonic realiza una categorización automática de los movimientos, apoyada en algoritmos que aprenden del comportamiento del usuario y mejoran progresivamente la precisión de dicha clasificación.

Uno de los aspectos más destacados de Fintonic es su sistema avanzado de visualización y análisis de gastos, que ofrece informes detallados sobre patrones de consumo, comparativas mensuales y anuales, y gráficos claros que permiten al usua-

rio comprender fácilmente su situación financiera. Además, la aplicación incorpora un sistema de alertas inteligentes, contribuyendo a una mayor transparencia y control del gasto.

En cuanto al ahorro, Fintonic también incluye un sistema de metas financieras, que permite establecer objetivos de ahorro y realizar un seguimiento básico de su progreso.

Un elemento claramente diferencial de Fintonic es la integración de un sistema de préstamos y productos financieros. La plataforma actúa como intermediaria. Este enfoque convierte a Fintonic no solo en una herramienta de control, sino también en un canal de acceso a financiación, ampliando su alcance funcional más allá del análisis de gastos.

Desde el punto de vista del modelo de negocio, Fintonic opera bajo un enfoque freemium. La versión gratuita incluye la mayoría de las funcionalidades esenciales, mientras que la versión Premium ofrece servicios adicionales como análisis más avanzados, reportes personalizados y mejoras en el sistema de predicción.

En este sentido, Smart Finance Solutions toma a Fintonic como una de sus principales referencias, pero busca ir un paso más allá incorporando funcionalidades sociales, gamificación y gestión de inversiones en acciones y criptomonedas que permiten ampliar el alcance de la aplicación.

Figura 2.3: Logo aplicación fintonic

Tricount está orientada a la gestión de gastos compartidos entre grupos de usuarios, siendo especialmente utilizada en contextos como viajes, convivencias o eventos. Su funcionalidad principal consiste en registrar gastos realizados por distintos participantes y calcular automáticamente los saldos pendientes entre ellos, facilitando la división equitativa de los pagos.

La aplicación destaca por su sencillez de uso y su enfoque colaborativo. Esta simplicidad la convierte en una herramienta muy eficaz para situaciones puntuales de gestión financiera en grupo. Sin embargo, su funcionalidad se limita exclusivamente al reparto de gastos, careciendo de herramientas avanzadas de análisis financiero, presupuestos, inversiones o seguimiento global de la economía personal.

Tricount resulta relevante como referencia para Smart Finance Solutions a nivel colaborativo, permitiendo el reparto equitativo de pagos dentro de una plataforma más amplia.

Figura 2.4: Logo aplicación tricount

Por último, muchas personas continúan utilizando **hojas de cálculo (Excel)** como herramienta para gestionar sus finanzas personales. El uso de hojas de cálculo permite un alto grado de control y personalización, siendo una opción habitual para usuarios con conocimientos básicos de ofimática que desean adaptar su sistema de gestión financiera a medida. Además, no requiere suscripciones ni conexión con entidades financieras, lo que puede resultar atractivo para determinados perfiles de usuario.

No obstante, este enfoque presenta claras desventajas, requiere un alto grado de implicación por parte del usuario y carece de automatización, análisis avanzado, alertas inteligentes y visualizaciones dinámicas, lo que limita su eficacia a medio y largo plazo.

Figura 2.5: Logo software Excel

Smart Finance Solutions toma como referencia estas soluciones existentes, pero propone una aproximación integradora que combina control financiero, análisis predictivo avanzado, gestión de inversiones, funcionalidades sociales y una experiencia de usuario intuitiva, todo ello en una única plataforma accesible y orientada a usuarios con distintos niveles de conocimiento financiero.

2.2. Tabla comparativa con aplicaciones existentes

A continuación se presenta una tabla comparativa que resume las principales funcionalidades de las soluciones analizadas en el estado de la cuestión.

Esta comparativa permite identificar de forma clara las diferencias y similitudes entre Smart Finance Solutions y otras aplicaciones relevantes del mercado, destacando especialmente aquellos aspectos funcionales que aportan un valor diferencial.

Funcionalidad	**Smart Finance**	**YNAB**	**Empower**	**Fintonic**	**Excel**
Registro de ingresos y gastos	Sí	Sí	Sí	Sí	Sí
Categorización flexible	Sí	Sí	Automática	Automática	Manual
Presupuestos avanzados	Sí	Sí	Limitado	Básico	Manual
Metas financieras	Sí	Sí	Limitado	Sí	Manual
Gastos compartidos	Sí	Limitado	No	No	Manual
Análisis predictivo	Sí	No	Básico	Premium	No
Predicción de ingresos y gastos	Sí	No	Limitado	No	No
Detección de anomalías	Sí	No	No	No	No
Acceso a información de mercado (acciones)	Sí	No	Sí	No	Manual
Acceso a información de mercado (criptomonedas)	Sí	No	Limitado	No	Manual
Gestión de inversiones	No	No	Sí	No	Manual
Activos y pasivos	Sí	No	Sí	No	Manual
Sistema social y mensajería	Sí	No	No	No	No
Gamificación	Sí	No	No	No	No
Dashboard personalizable	Parcial	Parcial	Sí	Parcial	No
Automatización	Media	Media	Alta	Alta	Nula

Tabla 2.1: Comparativa de funcionalidades entre Smart Finance Solutions y aplicaciones existentes en el mercado

Capítulo 3

Descripción del Trabajo

En este apartado se describe de forma detallada el funcionamiento general de la aplicación, así como las funcionalidades que ofrece a los usuarios. El objetivo es presentar una visión clara de las características implementadas y de cómo estas se organizan en función del tipo de acceso a la plataforma.

La aplicación distingue entre usuarios no autenticados y usuarios autenticados, ofreciendo diferentes niveles de funcionalidad en cada caso. Por un lado, los usuarios no logueados pueden acceder a las funcionalidades básicas, permitiéndoles conocer las características principales de la aplicación antes de crear una cuenta. Por otro lado, los usuarios logueados disponen de acceso completo a todas las funcionalidades de la aplicación.

A continuación, se detallan de manera estructurada las funcionalidades disponibles para cada tipo de usuario, describiendo su propósito y comportamiento dentro del sistema.

3.1. Funcionalidades para Usuarios No Autenticados

Los usuarios que acceden a la aplicación sin autenticarse disponen de un conjunto limitado de funcionalidades diseñadas para permitirles conocer la plataforma y decidir si desean registrarse. Estas funcionalidades proporcionan una visión general de las capacidades del sistema y facilitan el proceso de registro e inicio de sesión.

3.1.1. Página Principal y Navegación

La página principal actúa como punto de entrada a la aplicación y proporciona información relevante sobre la plataforma.

Los usuarios no autenticados pueden visualizar hasta seis reseñas aleatorias de otros usuarios, que incluyen calificaciones por estrellas y comentarios, ofreciendo una perspectiva de la experiencia de otros usuarios con la aplicación. Adicionalmente, se muestra una sección corta de preguntas frecuentes que ayudan a resolver dudas

comunes sobre el funcionamiento de la plataforma.

La navegación permite acceder a las secciones de registro, inicio de sesión y recuperación de contraseña, así como a información adicional sobre las características y propuesta de valor de la aplicación. También pueden acceder a la vista completa de reseñas.

Figura 3.1: Ventana de página principal

3.1.2. Sistema de Autenticación

El sistema de autenticación está diseñado para ser seguro y fácil de usar. El proceso de registro permite crear una cuenta nueva mediante un formulario que recopila información esencial. El sistema implementa validación en tiempo real que verifica la información mediante peticiones AJAX, proporcionando retroalimentación inmediata al usuario. Si por ejemplo, el nombre de usuario seleccionado ya está en uso, el sistema sugiere automáticamente alternativas disponibles.

La seguridad de las contraseñas está garantizada mediante requisitos mínimos: al menos seis caracteres, una mayúscula, un número y un carácter especial. Las contraseñas se almacenan utilizando técnicas de hashing, nunca en texto plano. Los usuarios pueden opcionalmente subir una foto de perfil durante el registro, sino, se le asigna una por defecto.

El proceso de inicio de sesión permite autenticarse utilizando el nombre de usuario o el correo electrónico junto con la contraseña. El sistema proporciona mensajes de error descriptivos que ayudan al usuario a identificar problemas específicos, como credenciales incorrectas o cuentas no registradas. Tras un inicio de sesión exitoso, el

usuario es redirigido automáticamente a su dashboard personal.

Para casos en los que el usuario ha olvidado su contraseña, el sistema implementa un proceso de recuperación seguro mediante tokens temporales. El usuario introduce su correo electrónico asociado a la cuenta, y el sistema genera un token seguro con un tiempo de expiración. Se envía automáticamente un correo electrónico con un enlace de recuperación. Por razones de seguridad, el sistema no revela si el correo existe en la base de datos, mostrando siempre el mismo mensaje de confirmación.

El enlace de recuperación permite restablecer la contraseña mediante un formulario que valida que el token sea válido y no haya expirado. El usuario puede establecer una nueva contraseña que debe cumplir los mismos requisitos que en el registro, y el sistema confirma el cambio exitoso proporcionando un enlace de regreso al inicio de sesión.

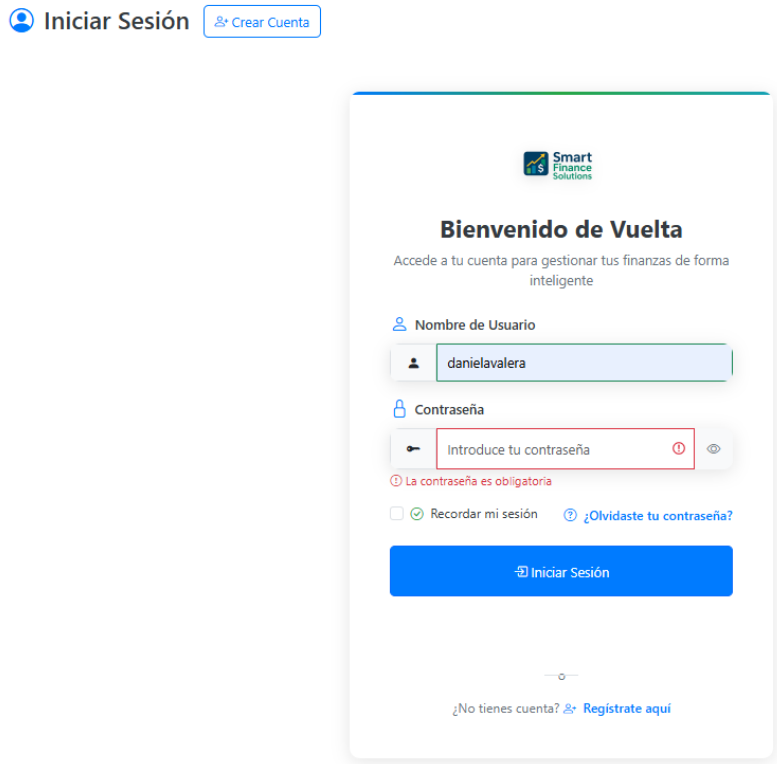

Figura 3.2: Ventana de inicio de sesión

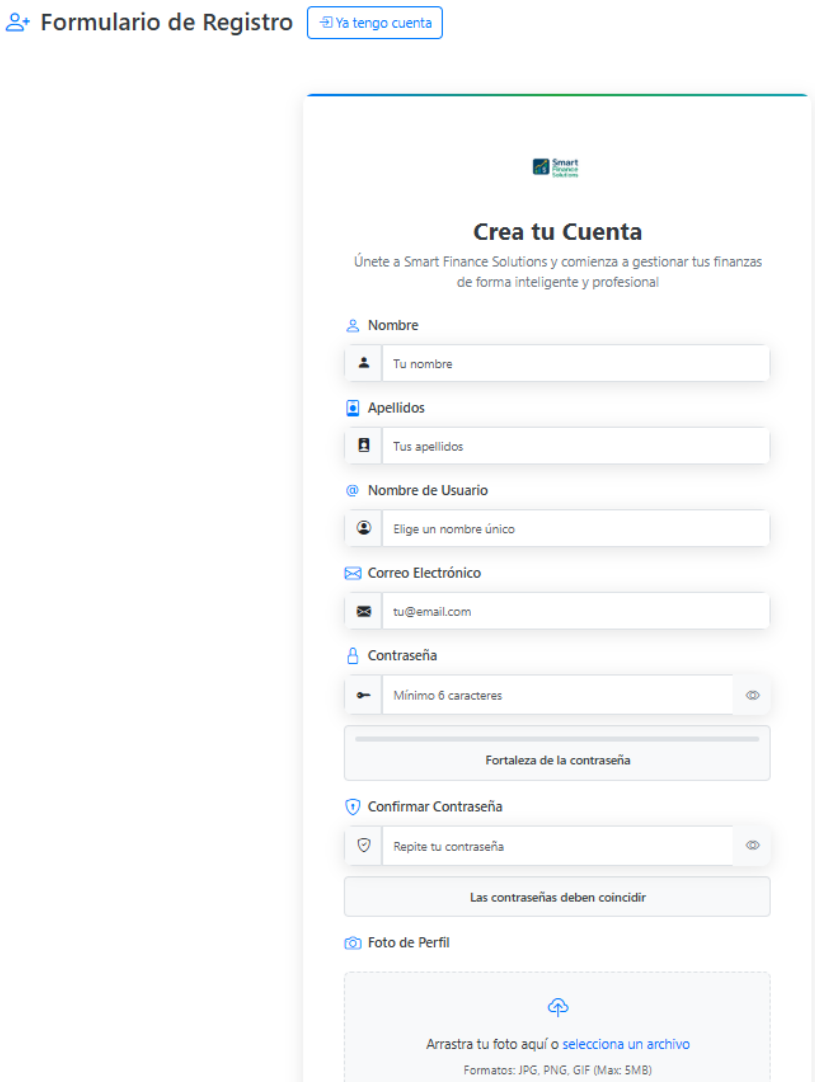

Figura 3.3: Ventana de registro

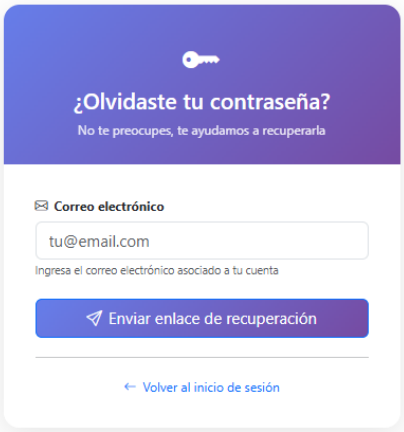

Figura 3.4: Ventana de recuperación de contraseña

3.1.3. Visualización de Contenido Público

Los usuarios no autenticados pueden acceder a la sección de reseñas, donde pueden visualizar todas las reseñas publicadas por los usuarios de la plataforma.

Esta funcionalidad permite a los visitantes conocer las opiniones y experiencias de otros usuarios, proporcionando información valiosa para tomar la decisión de registrarse. Las reseñas se muestran ordenadas por fecha, con las más recientes primero, e incluyen calificaciones por estrellas, comentarios y la información del usuario que las publicó.

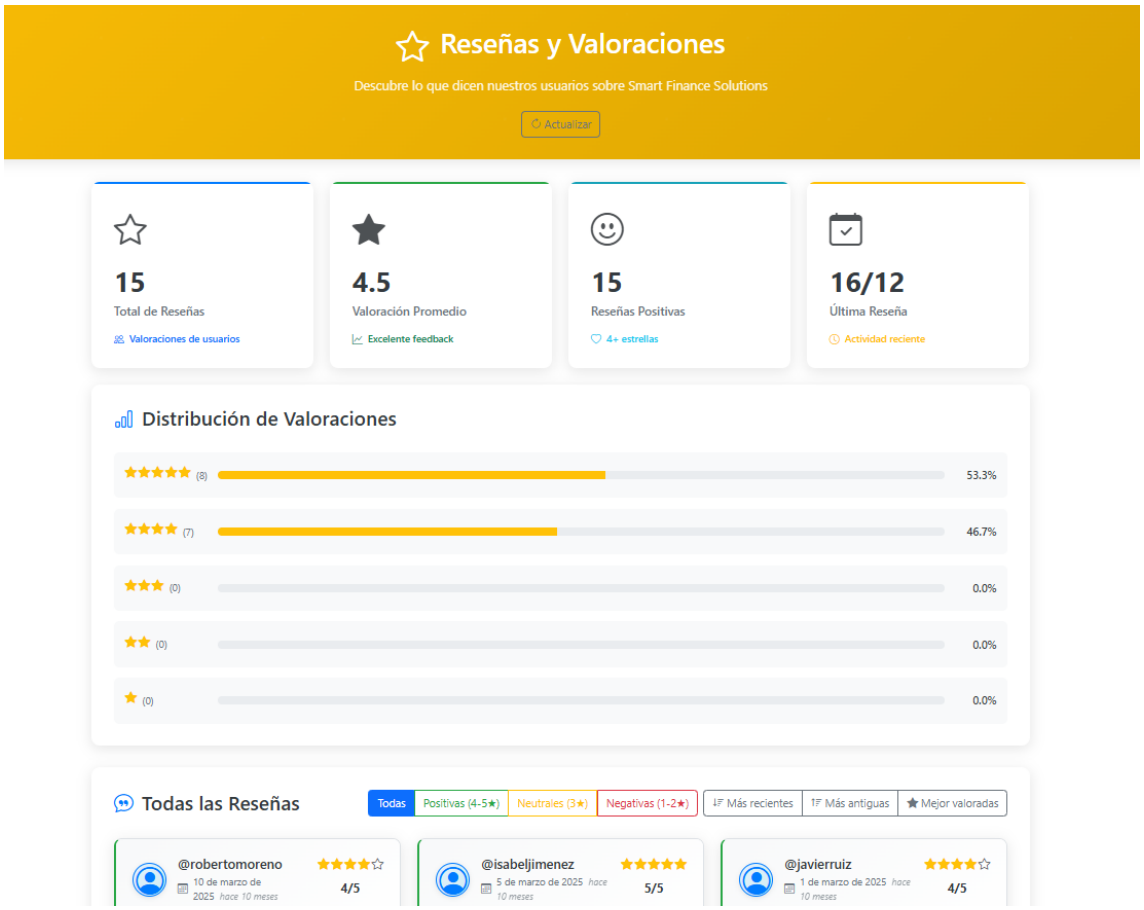

Figura 3.5: Ventana de todas las reseñas

3.2. Funcionalidades para Usuarios Autenticados

Una vez que el usuario se autentica en el sistema, tiene acceso a un conjunto completo de funcionalidades diseñadas para gestionar sus finanzas personales de manera integral. Estas funcionalidades se organizan en módulos especializados que cubren diferentes aspectos de la gestión financiera, desde el registro básico de movimientos hasta análisis avanzados y colaboración con otros usuarios.

3.2.1. Dashboard y Visión General

El dashboard principal proporciona una visión consolidada de la situación financiera del usuario.

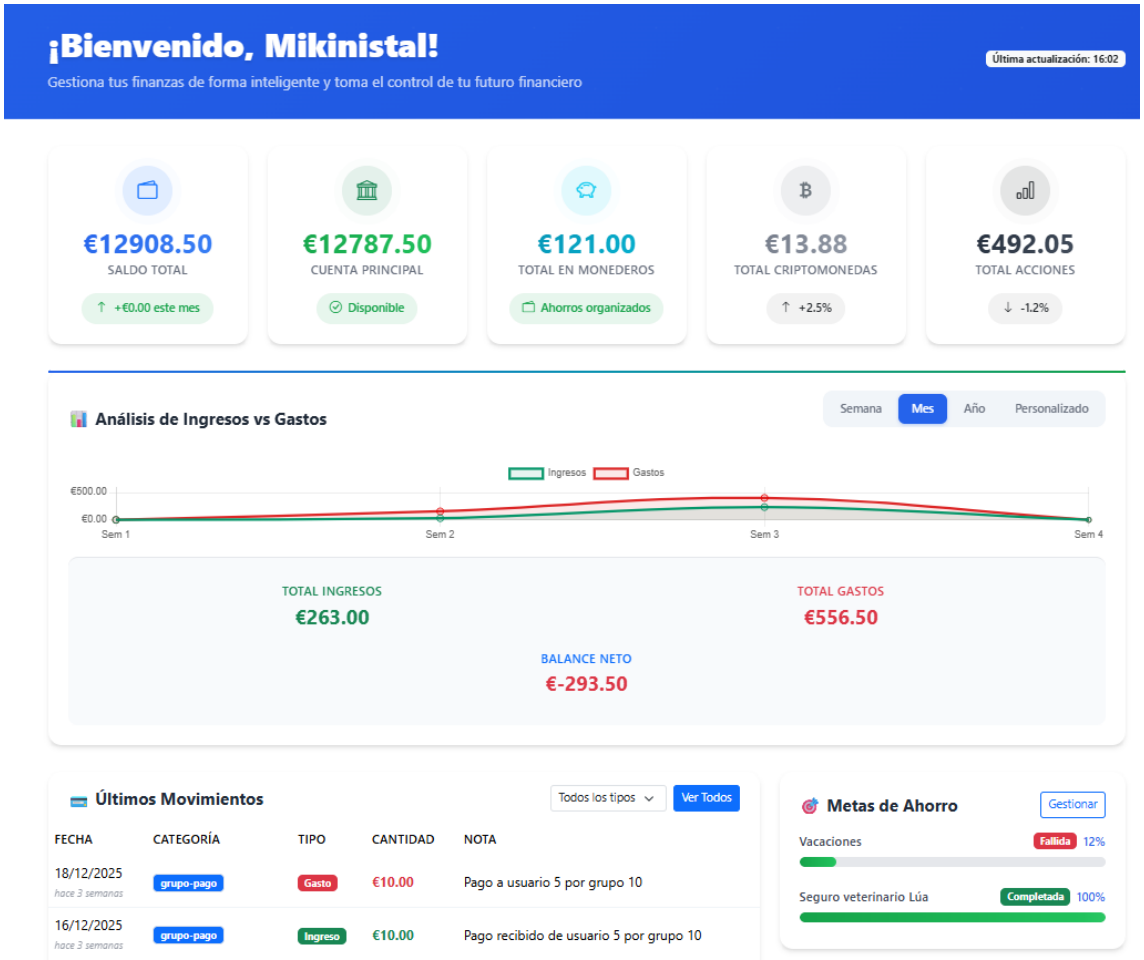

Figura 3.6: Ventana principal, Dashboard

La interfaz presenta un resumen financiero mediante viñetas que muestran las métricas clave: saldo total (suma de cuenta principal y el total en monederos), saldo de la cuenta principal, total acumulado en monederos virtuales y el valor total de las inversiones en criptomonedas y acciones. El valor de las inversiones se presenta de manera separada y no se integra en el total de las cuentas, al tratarse de

activos financieros no líquidos que no deben considerarse disponibles para la gestión cotidiana del usuario.

Una gráfica interactiva permite visualizar el análisis de ingresos vs gastos en diferentes períodos. Esta visualización ayuda a los usuarios a identificar tendencias en sus finanzas y tomar decisiones informadas sobre su gestión económica.

La sección de últimos movimientos muestra los diez movimientos más recientes, sean ingresos o gastos, ordenados descendentemente por fecha. Desde esta vista, el usuario puede acceder rápidamente a la gestión completa de movimientos.

Adicionalmente, se muestra un resumen de las metas de ahorro del usuario, con acceso directo a la gestión completa de metas mediante un botón dedicado.

3.2.2. Gestión Financiera Integral

El módulo de gestión financiera constituye el núcleo funcional de la aplicación, permitiendo a los usuarios registrar y gestionar todos sus movimientos económicos de manera organizada y categorizada.

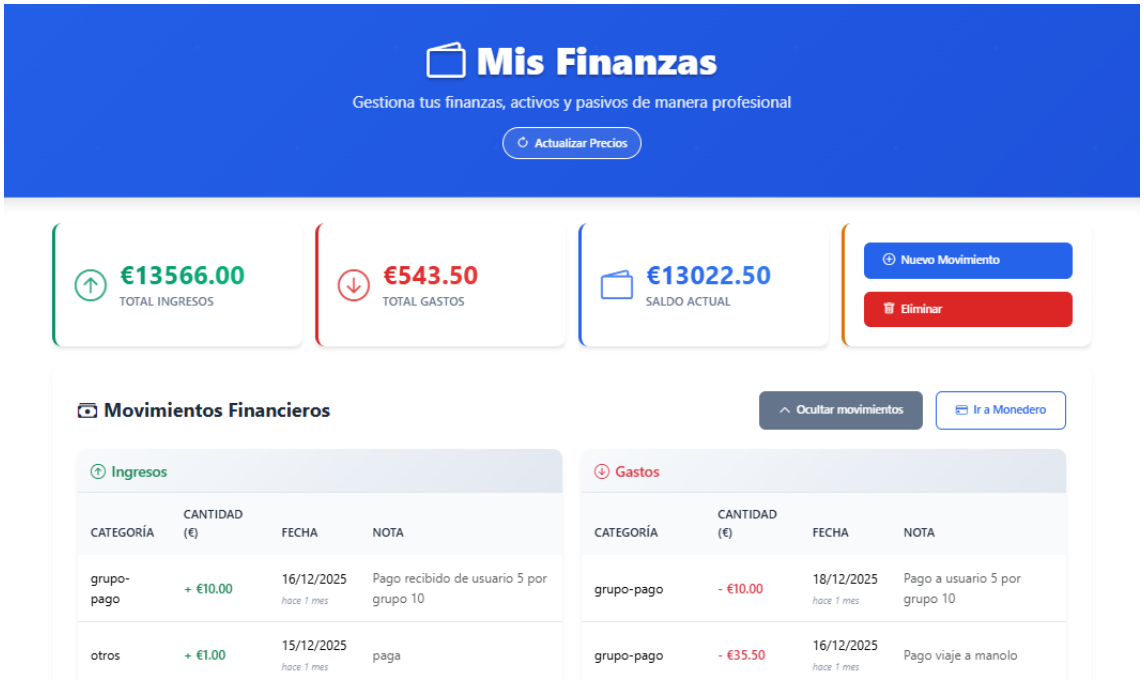

Figura 3.7: Ventana de finanzas

Registro de Movimientos

El sistema permite registrar tanto ingresos como gastos con información detallada. Los movimientos se pueden visualizar en detalle, ordenados por fecha más reciente, y la interfaz muestra el total de ingresos y gastos, así como el saldo actual disponible. Esta organización facilita el seguimiento del historial financiero completo y permite identificar patrones de gasto y fuentes de ingreso.

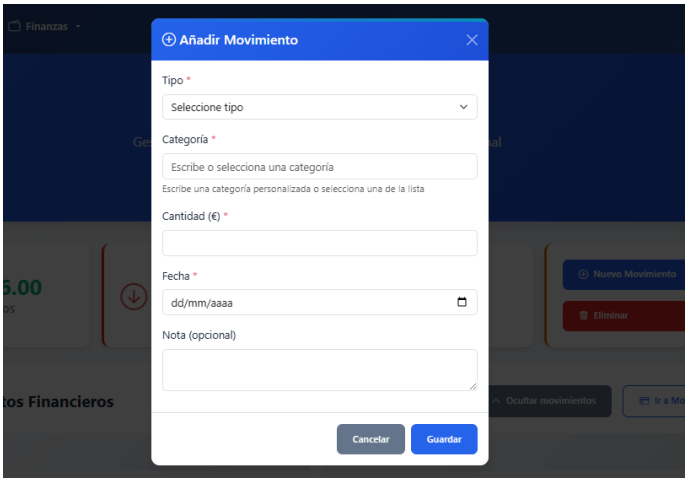

Figura 3.8: Ventana para añadir un nuevo movimiento

Gestión de Activos y Pasivos

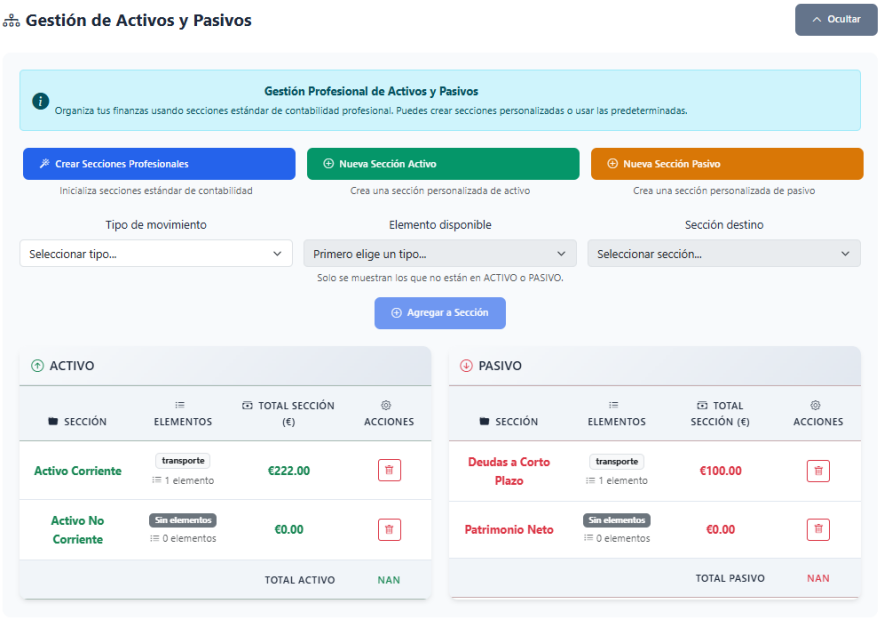

Figura 3.9: Ventana gestión de activos y pasivos

El sistema implementa una funcionalidad profesional para la gestión de activos y pasivos, creando automáticamente secciones predeterminadas organizadas por categorías. Los usuarios pueden crear secciones personalizadas adicionales y gestionar elementos dentro de cada sección.

La funcionalidad permite convertir movimientos existentes en elementos de gestión de activos o pasivos, o añadir elementos directamente a las secciones sin partir de movimientos previos. Los usuarios pueden gestionar, editar y eliminar elementos individuales, así como eliminar secciones completas con todos sus elementos asociados, manteniendo un control total sobre su patrimonio.

Portfolio de Inversiones

El sistema permite gestionar dos tipos de inversiones: criptomonedas y acciones. Para cada tipo, los usuarios pueden añadir y eliminar la inversión, registrando información como el símbolo o nombre del activo, la cantidad adquirida y la fecha de transacción. El sistema calcula automáticamente el valor total del portfolio sumando la cantidad por el precio de compra para cada inversión. Esta funcionalidad proporciona una visión consolidada de las inversiones del usuario y facilita el seguimiento de su cartera de inversión.

Figura 3.10: Ventana Portfolio de Acciones

Figura 3.11: Ventana Portfolio de Criptomonedas

3.2.3. Monederos Virtuales

El módulo de monederos permite a los usuarios crear múltiples contenedores financieros para diferentes propósitos, como ahorros para vacaciones, fondos de emergencia o proyectos específicos. Los usuarios pueden crear, editar y eliminar monederos según sus necesidades. Cuando se elimina un monedero, el saldo se devuelve automáticamente a la cuenta principal, manteniendo la integridad de los fondos.

Una característica destacada es la configuración de transferencias automáticas, que permite que los monederos se financien automáticamente desde los ingresos del usuario según una frecuencia configurada. El sistema valida que exista saldo suficiente antes de permitir la creación de monederos con cantidad inicial, garantizando que las operaciones sean viables financieramente.

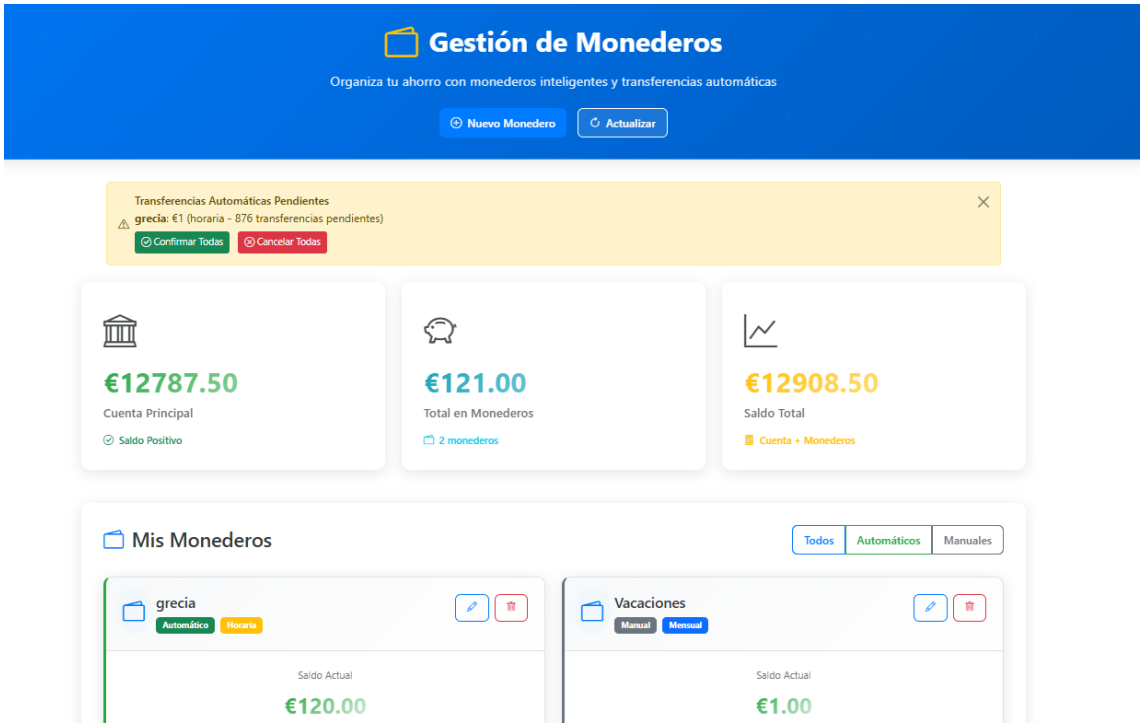

Figura 3.12: Ventana de monederos

3.2.4. Metas de Ahorro

El módulo de metas permite a los usuarios establecer objetivos financieros específicos y hacer seguimiento de su progreso. Las metas pueden estar en diferentes estados: nueva (sin progreso y no vencida), en progreso o completada (cuando el progreso alcanza o supera el 100 %).

Al crear una meta, el usuario puede asociarla a un monedero existente o crear un nuevo monedero específicamente para esa meta, agilizando el proceso de organización. El sistema permite activar notificaciones del progreso para recibir actualizaciones sobre el avance hacia la meta.

El progreso de cada meta se recalcula automáticamente basándose en el saldo del monedero asociado o de la cuenta principal. Los usuarios pueden editar las metas

y extender la fecha límite si una meta ha fallado, creando una nueva oportunidad para alcanzarla. El sistema genera notificaciones cuando se extiende una fecha límite o cuando se elimina una meta, manteniendo al usuario informado de los cambios importantes.

El sistema de notificaciones para metas es especialmente completo: se crean notificaciones automáticas cuando se alcanzan hitos importantes de progreso (25 %, 50 %, 75 % y 100 %), cuando se acerca la fecha límite y cuando una meta vence sin completarse. Estas notificaciones ayudan a mantener al usuario motivado y consciente del estado de sus objetivos financieros.

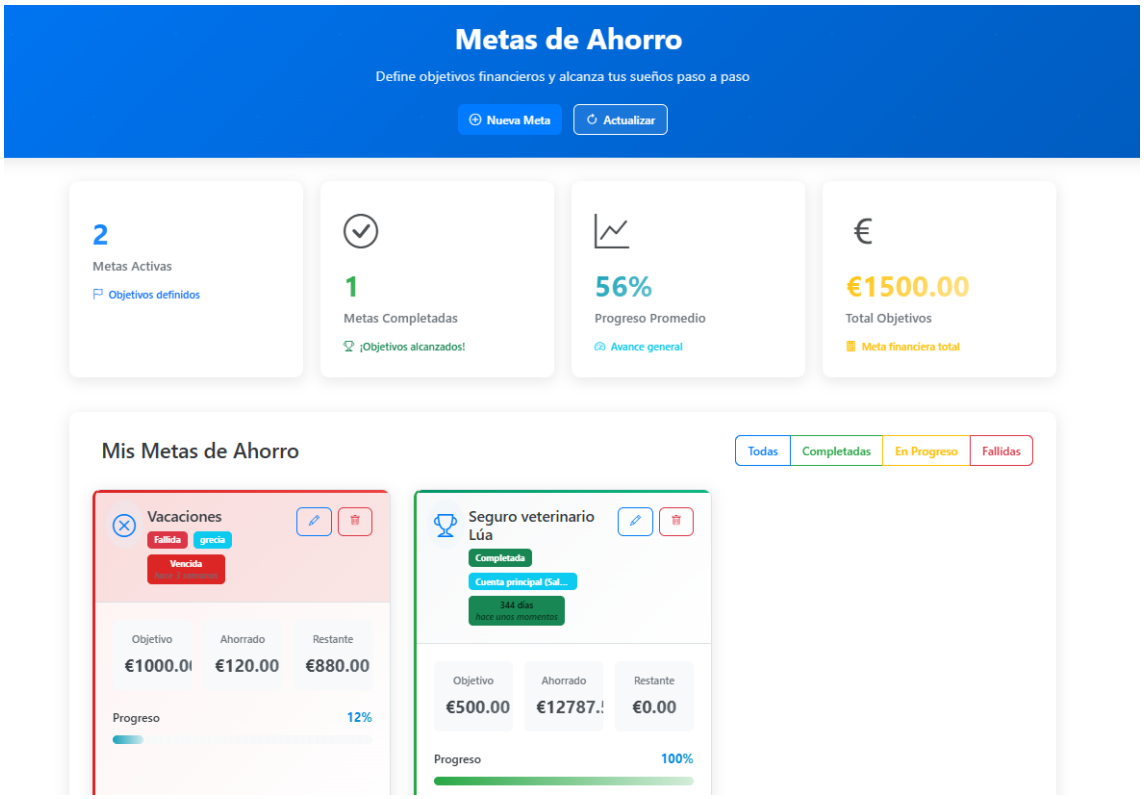

Figura 3.13: Ventana de metas de ahorro

3.2.5. Colaboración Financiera en Grupos

El módulo de grupos financieros implementa funcionalidades de colaboración similares a aplicaciones como TriCount, permitiendo a los usuarios crear grupos y gestionar gastos compartidos de manera eficiente.

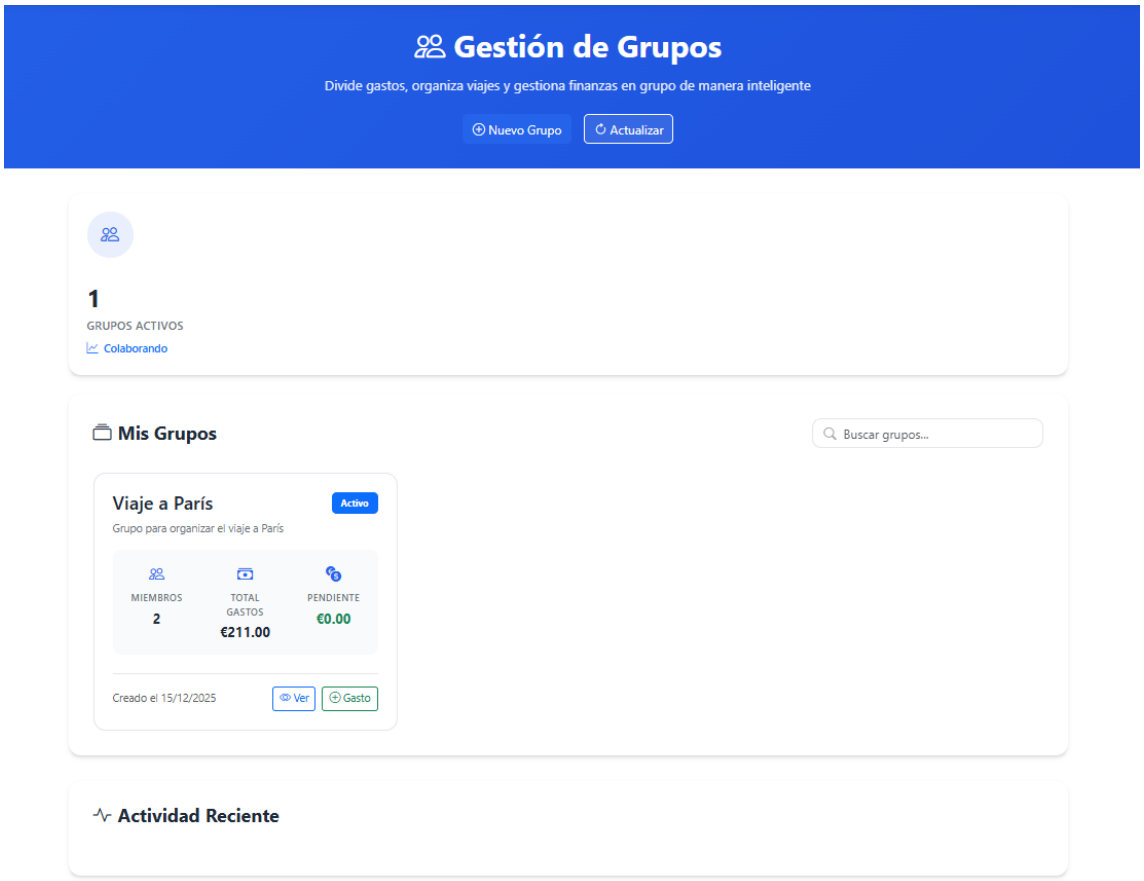

Figura 3.14: Ventana de grupos de amigos

Creación y Gestión de Grupos

Al crear un grupo, el usuario puede añadir amigos aceptados como miembros. El sistema verifica automáticamente que todos los miembros propuestos sean amigos del usuario, garantizando que solo se incluyan usuarios con relaciones establecidas. El usuario que ha creado el grupo se añade automáticamente como administrador del grupo, con capacidad para gestionar miembros y eliminar el grupo si es necesario.

Gestión de Gastos Compartidos

La vista detallada del grupo muestra información completa sobre todos los gastos registrados, los miembros del grupo y los balances calculados automáticamente. Los balances se presentan tanto a nivel general del grupo como individualmente por cada miembro, mostrando claramente qué ha pagado cada persona y qué debe. El sistema mantiene un historial completo de los pagos realizados entre miembros.

Una funcionalidad importante es la opción de rebalancear el grupo, recalcula todos los balances y deudas considerando los pagos realizados. Esto asegura que la información mostrada siempre refleje el estado actual de las finanzas del grupo.

3.2.6. Sistema de Amistades y Comunicación

La aplicación incluye un sistema completo de relaciones sociales y comunicación que permite a los usuarios interactuar entre sí.

Gestión de Amistades

El módulo de amigos organiza las relaciones en tres secciones claras: amigos aceptados, solicitudes de amistad recibidas y solicitudes enviadas. Desde esta interfaz, los usuarios pueden ver las estadísticas de sus amigos, eliminarlos de su lista o iniciar un chat directo.

El sistema incluye una funcionalidad de búsqueda de usuarios por nombre de usuario o correo electrónico, con un mínimo de dos caracteres para iniciar la búsqueda. La búsqueda excluye automáticamente al usuario actual y a usuarios que ya son amigos o tienen solicitudes pendientes, evitando duplicados y simplificando la experiencia.

Cuando un usuario envía una solicitud de amistad, el sistema crea automáticamente un mensaje y una notificación en el buzón del destinatario, facilitando la comunicación y el seguimiento de las solicitudes.

Una característica adicional es que se permite marcar amigos como cercanos, lo que habilita la visualización de información completa para esos contactos, mientras que para otros amigos solo se muestra información básica.

Figura 3.15: Ventana de contactos

Buzón de Mensajes

El buzón implementa un sistema de mensajería asíncrona que permite a los usuarios enviar mensajes que pueden ser leídos posteriormente. La interfaz muestra mensajes recibidos con un contador de mensajes no leídos y mensajes enviados, proporcionando una visión clara del estado de la comunicación.

Los usuarios pueden enviar mensajes mediante el nombre de usuario del destinatario y tienen la posibilidad de adjuntar archivos a cada mensaje. El sistema crea automáticamente una notificación para el destinatario cuando se recibe un nuevo

mensaje.

La eliminación de mensajes es lógica: los mensajes no se eliminan físicamente de la base de datos, sino que se marcan como eliminados para el usuario que realiza la acción. Si solo uno de los usuarios elimina un mensaje, este permanece visible para el otro usuario, manteniendo la integridad de la conversación.

El buzón también gestiona las solicitudes de amistad, permitiendo aceptar o rechazar solicitudes directamente desde el mensaje. Cuando se acepta una solicitud, el usuario puede opcionalmente marcar al nuevo amigo como cercano, estableciendo el nivel de visibilidad de la información desde el inicio de la relación.

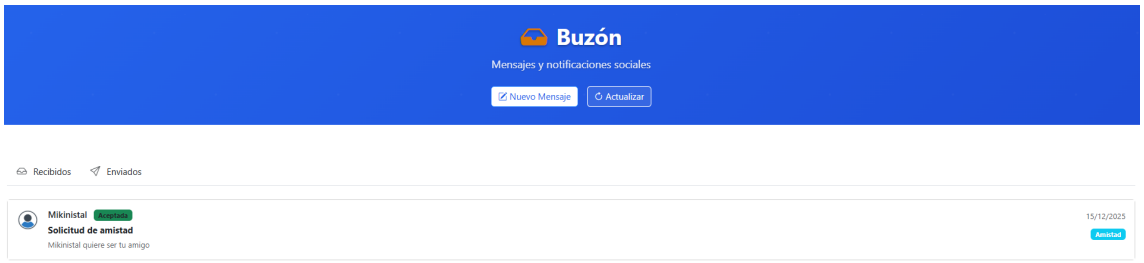

Figura 3.16: Ventana de buzón de mensajes

Chat en Tiempo Real

El sistema de chat permite comunicación en tiempo real entre usuarios que son amigos. Los chats se crean automáticamente al enviar el primer mensaje, y el sistema mantiene un historial completo de todas las conversaciones.

Al acceder a un chat, se visualiza el historial completo de mensajes con ese amigo específico. Los mensajes se marcan automáticamente como leídos al acceder al chat y el contador de mensajes no leídos se actualiza en tiempo real. Si el destinatario no está actualmente en el chat, el sistema crea una notificación para alertarle del nuevo mensaje.

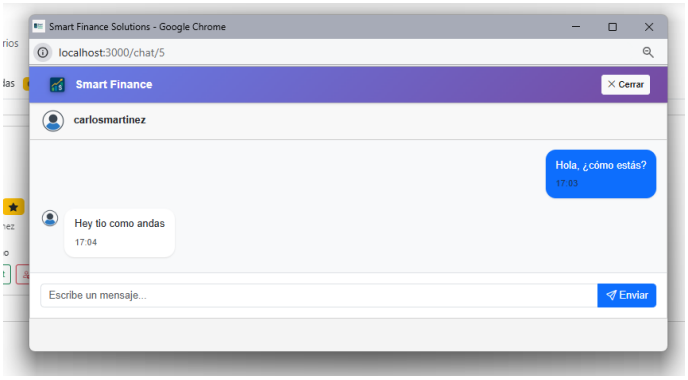

Figura 3.17: Ventana de chats

3.2.7. Comunidad y Social

La aplicación incluye funcionalidades de comunidad que facilitan la interacción entre usuarios y el intercambio de información y experiencias.

Foro de Preguntas y Respuestas

Los usuarios autenticados pueden acceder a la sección completa de preguntas frecuentes. El foro permite a los usuarios crear y responder preguntas sobre dudas y temas relacionados con la aplicación. Las preguntas se visualizan ordenadas por fecha más reciente, junto con todas sus respuestas asociadas. Cada pregunta muestra un indicador si el usuario actual le ha dado *"Me gusta"* y el sistema permite un único *"Me gusta"* por usuario y pregunta, previniendo duplicados.

Los usuarios pueden crear y editar sus propias preguntas y respuestas, y el sistema verifica que solo el autor pueda realizar modificaciones. Los cambios se reflejan automáticamente en la interfaz, manteniendo la información actualizada.

Esta funcionalidad permite que la comunidad contribuya a la documentación de la plataforma, creando un recurso colaborativo que beneficia a todos los usuarios.

Figura 3.18: Ventana de foro de preguntas

Sistema de Reseñas

Los usuarios autenticados pueden crear, visualizar y editar reseñas sobre la aplicación. Cada usuario puede escribir una única reseña, que debe incluir una puntuación con estrellas y un comentario, ambos campos obligatorios. Las reseñas se ordenan por ID descendente (más recientes primero) y el sistema identifica automáticamente si el usuario actual tiene una reseña publicada, permitiendo editarla directamente.

3.2.8. Gamificación y Logros

El sistema de logros implementa un mecanismo de gamificación diseñado para motivar el uso consistente de la aplicación. Los usuarios pueden acceder a esta sección desde la barra de navegación mediante un icono de trofeo.

La interfaz muestra la lista completa de logros disponibles, diferenciando entre aquellos que el usuario ha obtenido, los que tiene en progreso y los que aún no ha comenzado. El sistema muestra información sobre el nivel actual del usuario, los puntos totales acumulados, el número de logros desbloqueados y los puntos necesarios para alcanzar el siguiente nivel.

El sistema verifica automáticamente el cumplimiento de logros tras acciones relevantes del usuario, como crear movimientos, alcanzar metas o participar en grupos. Esta verificación automática asegura que los usuarios sean reconocidos por sus logros sin necesidad de acciones manuales.

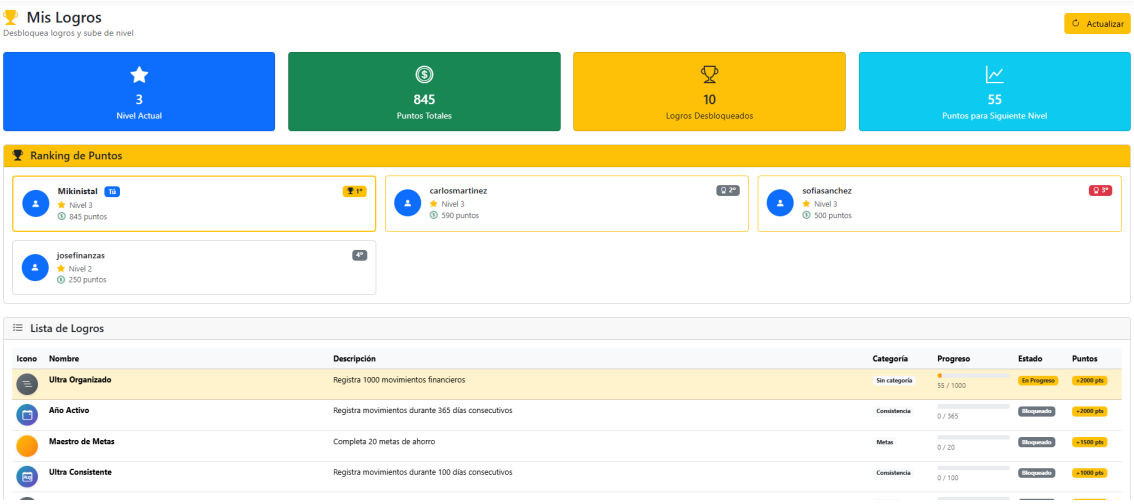

Figura 3.19: Ventana de logros

3.2.9. Sistema de Notificaciones

El módulo de notificaciones proporciona un sistema centralizado para mantener a los usuarios informados sobre eventos importantes en la aplicación. Los usuarios pueden acceder a sus notificaciones mediante un icono de campana en la barra de navegación, que muestra un contador con el número de notificaciones no leídas.

La interfaz muestra todas las notificaciones del usuario, tanto leídas como no leídas, ordenadas por fecha de creación descendente (más recientes primero). Cada notificación muestra su tipo, prioridad, estado de lectura y fecha de recepción, proporcionando contexto completo sobre cada alerta.

Los usuarios pueden marcar notificaciones individuales como leídas o marcar todas como leídas de una vez, facilitando la gestión de grandes volúmenes de notificaciones. El sistema permite eliminar notificaciones individualmente desde la lista, proporcionando control total sobre el historial de alertas.

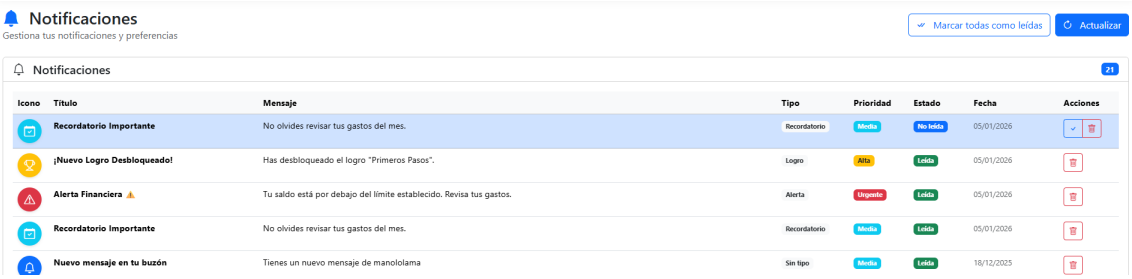

Figura 3.20: Ventana de notificaciones

3.2.10. Análisis predictivo

El módulo de Análisis Predictivo es una funcionalidad avanzada de Smart Finance Solutions que permite a los usuarios obtener información prospectiva y analítica basada en sus datos históricos. Esta característica utiliza técnicas de análisis estadístico y regresión lineal tanto para predecir tendencias futuras de ingresos, gastos y ahorro, como para identificar patrones anómalos en el comportamiento financiero del usuario.

Entre las técnicas empleadas se incluyen modelos de regresión para la predicción financiera y métodos estadísticos basados en la desviación estándar, como el z-score, orientados a la detección de anomalías en los gastos.

1. Planificación y proyección financiera

   La predicción de ingresos y gastos futuros constituye una funcionalidad central del módulo de análisis predictivo de Smart Finance Solutions. Esta característica permite a los usuarios proyectar sus finanzas personales a corto y medio plazo, facilitando la planificación financiera, la toma de decisiones informadas y la evaluación de la viabilidad de metas de ahorro establecidas.

   El objetivo principal es proporcionar proyecciones financieras precisas basadas en el historial real del usuario, utilizando técnicas estadísticas validadas para extrapolar tendencias observadas. Adicionalmente, el sistema calcula predicciones de cumplimiento para las metas de ahorro establecidas, estimando fechas de cumplimiento y evaluando su factibilidad.

   Esta funcionalidad complementa otras herramientas de análisis del sistema, proporcionando una visión integral y prospectiva de la situación financiera del usuario.

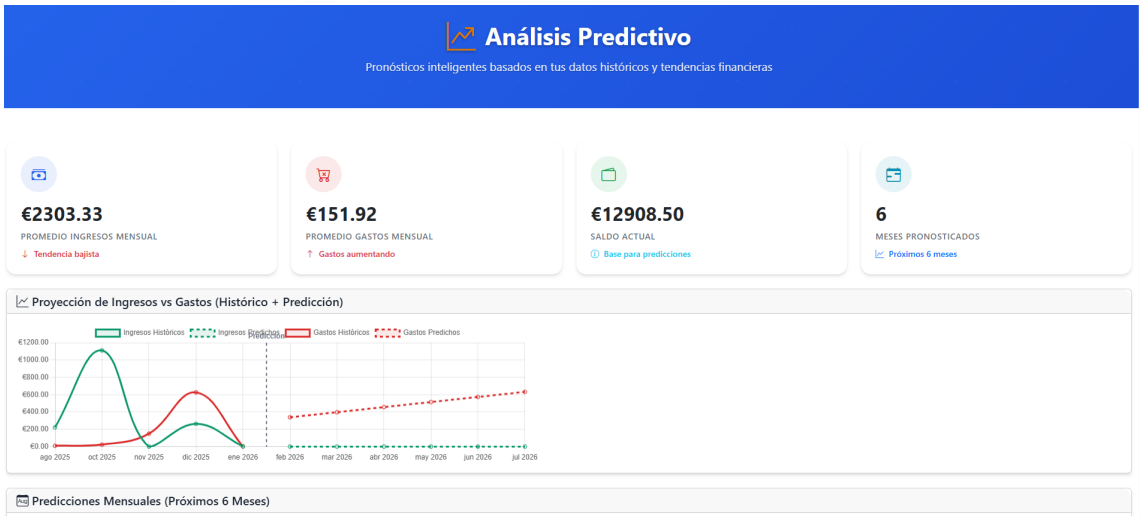

Figura 3.21: Ventana de predicción financiera

2. Detección de Anomalías de Gastos

   La detección de anomalías en gastos es una funcionalidad avanzada implementada en Smart Finance Solutions que permite identificar de forma automática

gastos inusuales en el historial financiero del usuario. Esta característica proporciona una herramienta de análisis que ayuda a los usuarios a identificar patrones atípicos en sus finanzas personales, facilitando la detección de posibles errores, fraudes, o gastos extraordinarios que requieren atención.

El objetivo principal de esta funcionalidad es dotar al usuario de un mecanismo automático de alerta que analice su historial de transacciones y detecte aquellos gastos que se desvían significativamente de sus patrones de consumo habituales. Esto permite una gestión financiera más proactiva y un mayor control sobre las finanzas personales.

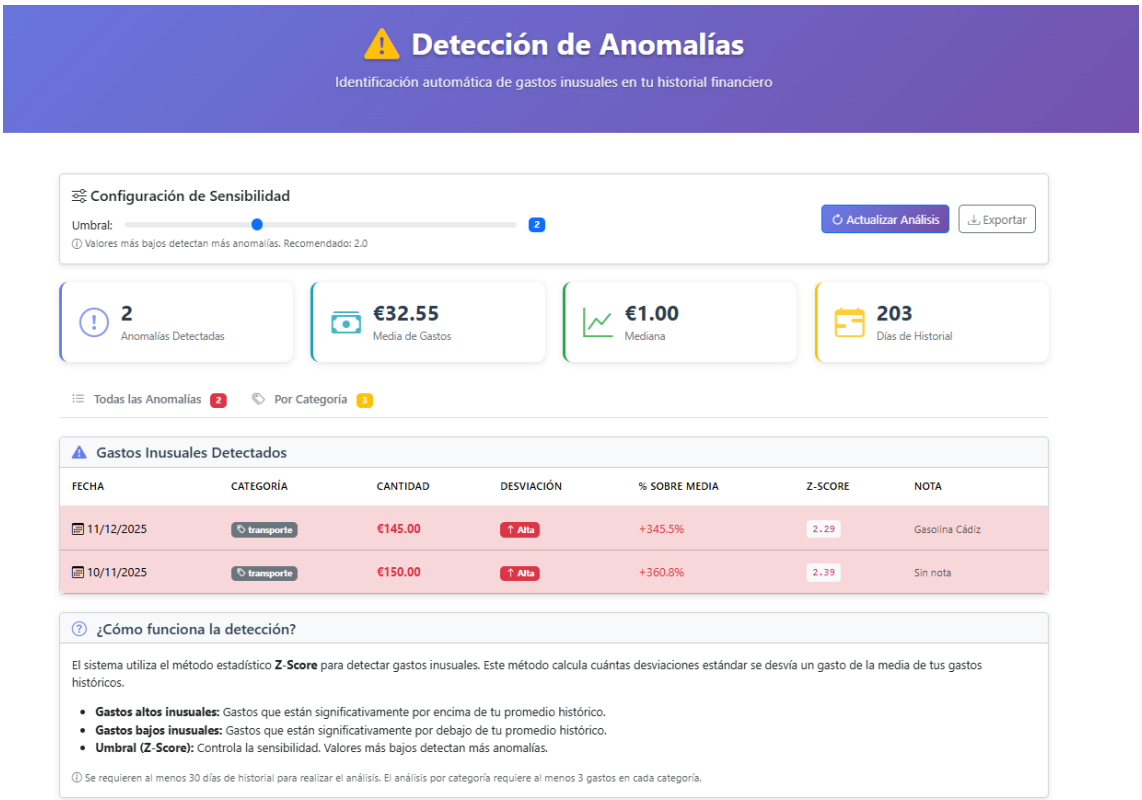

Figura 3.22: Ventana de detección de anomalías

3.2.11. Perfil y Configuración

El módulo de perfil permite a los usuarios gestionar su información personal y acceder a estadísticas detalladas de su actividad en la aplicación.

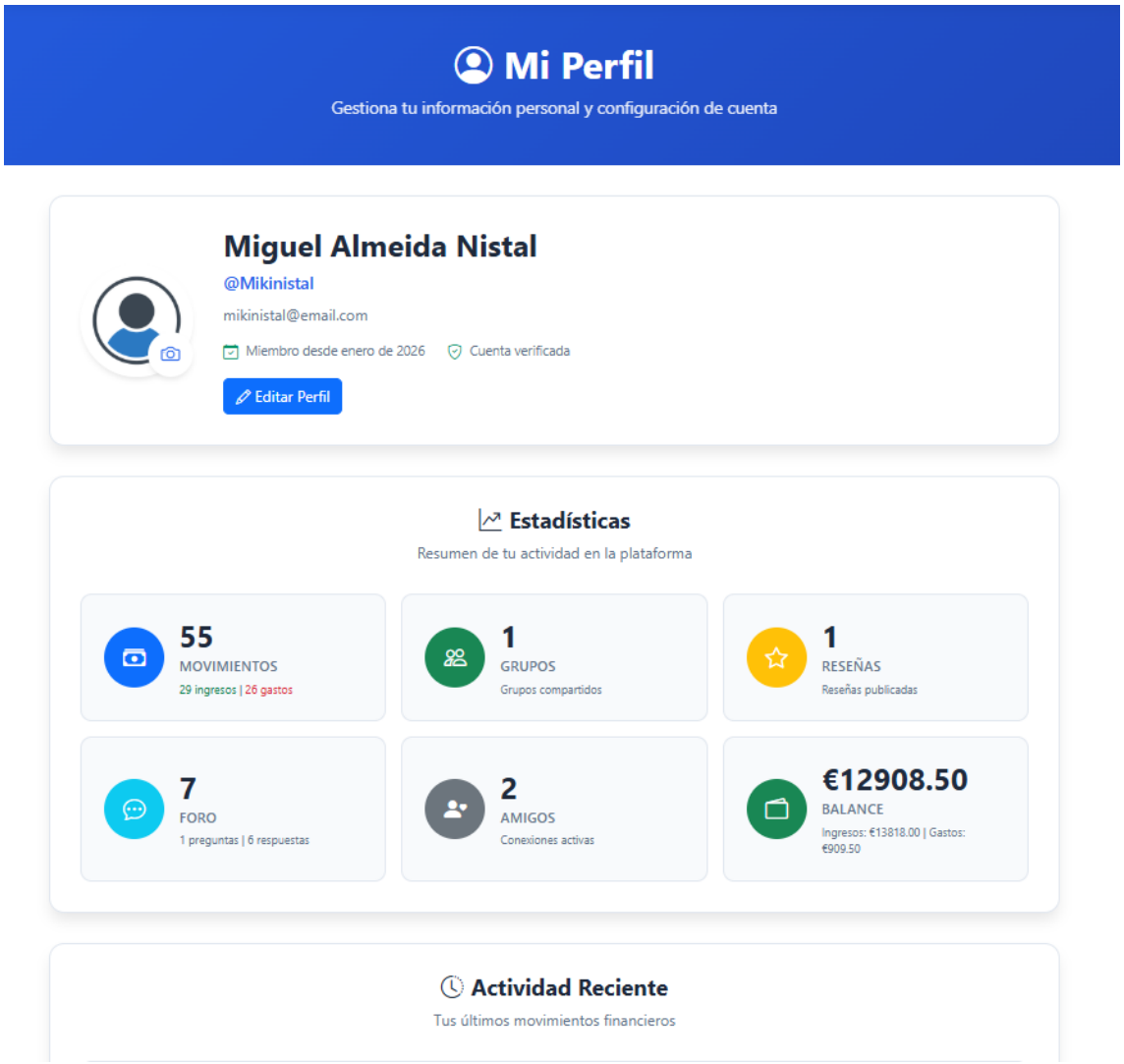

Figura 3.23: Ventana de perfil del usuario

Edición de Perfil

Los usuarios pueden editar su nombre, apellidos y correo electrónico, mientras que el nombre de usuario permanece fijo para mantener la consistencia en el sistema. El sistema realiza verificaciones para asegurar que los cambios de correo electrónico sean válidos y únicos. La foto de perfil puede actualizarse en cualquier momento, permitiendo a los usuarios personalizar su presencia en la plataforma.

Cambio de Contraseña

El sistema incluye un formulario dedicado para cambiar la contraseña cuando el usuario la recuerda. El proceso requiere verificar que la contraseña actual sea correcta y que la nueva contraseña cumpla con los mismos requisitos de seguridad que en el registro. El sistema genera un hash de la nueva contraseña antes de almacenarla, manteniendo los estándares de seguridad.

Estadísticas y Reportes

El perfil proporciona un resumen completo de todas las acciones del usuario en la aplicación, incluyendo el número total de movimientos, sus grupos asociados, reseñas publicadas, saldo total de movimientos y otras métricas relevantes. La sección de actividad reciente muestra los últimos cinco movimientos, proporcionando un acceso rápido al historial más inmediato.

Una funcionalidad destacada es la generación de reportes en formato PDF, que permite a los usuarios descargar un documento profesional con la información que seleccionen. Este reporte puede incluir diferentes secciones según las preferencias del usuario, proporcionando un documento consolidado de su situación financiera.

Eliminación de Cuenta

El sistema permite eliminar la cuenta del usuario mediante un proceso que requiere confirmación del nombre de usuario y la contraseña para garantizar la seguridad. Al eliminar una cuenta, se genera una eliminación en cascada de todos los datos asociados al usuario, manteniendo la integridad de la base de datos. La sesión se destruye automáticamente tras la eliminación, cerrando completamente el acceso a la cuenta.

3.2.12. Dashboard Personalizable

El módulo de dashboard personalizable *(/dashboard)* permite a los usuarios configurar su propia vista principal con las funcionalidades y visualizaciones que consideren más importantes. Aunque esta funcionalidad está en desarrollo, el sistema está diseñado para soportar un sistema completo de widgets que los usuarios pueden añadir, eliminar y reorganizar según sus preferencias. Cada widget puede tener su propia configuración específica, permitiendo una personalización granular de la experiencia del usuario.

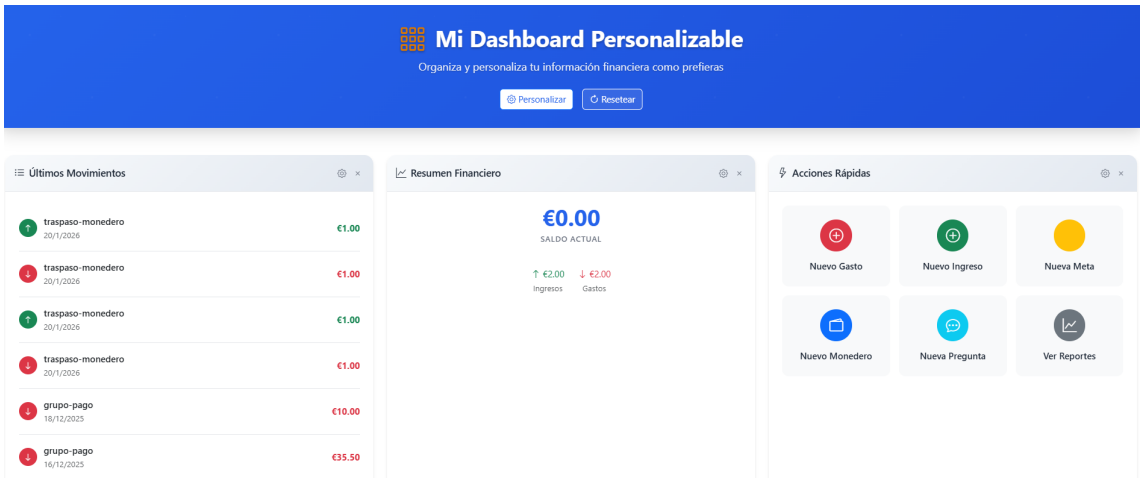

Figura 3.24: Ventana dashboard personalizable

3.2.13. Cierre de Sesión

El proceso de cierre de sesión *(/logout)* destruye completamente la sesión del usuario, limpiando todos los datos de sesión y redirigiendo automáticamente a la página principal. Este proceso garantiza que no queden datos de sesión activos tras el cierre, manteniendo la seguridad de la cuenta.

3.3. Consideraciones sobre Funcionalidades Futuras

Algunas funcionalidades documentadas están parcialmente implementadas o están diseñadas para expansión futura. El dashboard personalizable, por ejemplo, tiene la infraestructura base implementada pero está previsto para desarrollo adicional que permita una personalización completa mediante widgets arrastrables y configurables.

La arquitectura del sistema está diseñada para facilitar la adición de nuevas funcionalidades sin requerir cambios significativos en la estructura existente, permitiendo que la aplicación evolucione según las necesidades de los usuarios y las oportunidades de mejora identificadas.

Capítulo 4

Arquitectura del sistema

4.1. Visión general del sistema

La aplicación sigue una arquitectura cliente-servidor tradicional con renderizado del lado del servidor (Server-Side Rendering, SSR):

Capa de Cliente

El cliente (navegador web) se comunica con el servidor mediante peticiones HTTP estándar. La interfaz de usuario se renderiza completamente en el servidor y se envía al cliente como HTML, CSS y JavaScript. Esta aproximación proporciona:

- **Carga inicial rápida:** El HTML ya viene renderizado desde el servidor.
- **Menos carga en el cliente:** el servidor hace el trabajo pesado.

Esta decisión arquitectónica resulta especialmente adecuada para aplicaciones orientadas a la visualización de datos y a la optimización del SEO, ya que permite ofrecer contenido completamente renderizado desde la primera carga, mejorando tanto la experiencia de usuario como la indexación por parte de los motores de búsqueda.

Capa de Servidor

El servidor, implementado con Node.js y Express.js, procesa las peticiones HTTP, gestiona la lógica de negocio, interactúa con la base de datos y renderiza las vistas. El servidor mantiene el estado de las sesiones de usuario y gestiona la autenticación y autorización.

Capa de Datos

La capa de datos está implementada mediante MySQL. Todas las operaciones de persistencia se realizan a través de consultas SQL parametrizadas, garantizando la seguridad y la integridad de los datos.

4.1.1. Componentes Principales del Sistema

La aplicación está organizada en los siguientes módulos funcionales principales:

1. Módulo de Autenticación y Usuarios

   Gestiona el registro, inicio de sesión, recuperación de contraseña y gestión de perfiles de usuario. Incluye:

   - Registro de nuevos usuarios con validación de datos.
   - Autenticación mediante sesiones del lado del servidor.
   - Recuperación de contraseña mediante tokens seguros enviados por correo electrónico.
   - Gestión de perfiles.

2. Módulo de Gestión Financiera

   Núcleo de la aplicación, permite a los usuarios:

   - **Registro de movimientos:** Ingresos y gastos categorizados.
   - **Gestión de monederos:** Creación y gestión de múltiples monederos virtuales.
   - **Inversiones:** Registro de acciones e inversiones con seguimiento de precios de compra.
   - **Metas y planificación:** Creación de metas de ahorro, seguimiento del progreso y visualización del porcentaje de cumplimiento.
   - **Análisis Predictivo:** Funcionalidad avanzada que permite predecir ingresos, gastos y ahorro futuros, así como detectar anomalías en los gastos, apoyando la planificación financiera del usuario.
   - **Grupos Financieros:** Creación y gestión de grupos financieros, registro y división de gastos compartidos entre miembros, cálculo automático de balances y gestión de pagos.
   - **Gestión de Activos y Pasivos:** Funcionalidad profesional para la gestión del patrimonio del usuario.

3. Módulo de Comunidad

   Módulo orientado a fomentar la interacción social entre los usuarios de la plataforma, proporcionando diferentes herramientas de comunicación y participación:

   - Gestión de amistades entre usuarios (solicitudes, aceptación y listado de contactos).
   - Buzón y sistema de mensajería interna entre usuarios y con el sistema.
   - Foro de preguntas y respuestas.
   - Sistema de reseñas e interacciones sociales.

4. Módulo de Sistema de Incentivos y Notificaciones

   Combina gamificación y notificaciones para motivar y mantener informados a los usuarios:

   - **Gamificación:** Sistema de puntos, logros desbloqueables y rankings (general, mensual, semanal).
   - **Notificaciones:** Alertas en tiempo real sobre metas alcanzadas, eventos importantes, recordatorios y logros, con sistema de prioridades.

4.1.2. Flujo General de la Aplicación

Flujo de Usuario No Autenticado

1. El usuario accede a la página principal *(/)*.
2. Puede navegar por la información general, las características de la plataforma y consultar las reseñas públicas de otros usuarios *(/resenas)*.
3. Para acceder a las funcionalidades, debe registrarse *(/register)* o iniciar sesión *(/login)*.
4. El registro requiere validación de datos.
5. Tras completar el registro, el usuario es redirigido a la página de inicio de sesión.
6. Desde la página de inicio de sesión, el usuario dispone de la opción de recuperación de contraseña *(/forgot-password)*.

Flujo de Autenticación

1. El usuario introduce sus credenciales.
2. El servidor valida las credenciales contra la base de datos.
3. Si son correctas, se crea una sesión del lado del servidor.
4. Se almacena información del usuario en la sesión.
5. Se redirige al usuario a la página de inicio autenticada.

Flujo de Usuario Autenticado

1. El usuario accede a su dashboard personal, que actúa como página de inicio tras la autenticación *(/inicio)*.
2. Puede navegar por todas las funcionalidades de la aplicación mediante la barra de navegación.
3. Cada funcionalidad dispone de sus propias rutas y vistas:

- Registro y consulta de movimientos financieros, gestión de activos y pasivos, y seguimiento del portfolio de inversiones *(/cuentas)*.
- Creación y visualización de metas financieras *(/metas)*.
- Gestión de monederos y cuentas *(/monedero)*.
- Gestión y participación en grupos financieros *(/grupos)*.
- Gestión de amistades *(/amigos)* y comunicación mediante chat *(/chat/:amigoId)*.
- Consulta y gestión del buzón de mensajes *(/buzon)*.
- Consulta y publicación de reseñas *(/resenas)*.
- Participación en el foro de preguntas y respuestas *(/foro)*.
- Gestión del perfil de usuario *(/perfil)*.
- Consulta de análisis predictivo *(/pronosticos)* y detección de anomalías *(/anomalias)*.
- Cambio de contraseña desde el área personal *(/cambiar-password)*.

4. Las acciones del usuario pueden generar notificaciones *(/notificaciones)* y desbloquear logros dentro del sistema de gamificación *(/logros)*.
5. El sistema actualiza automáticamente las métricas y visualizaciones mostradas al usuario.

Flujo de Datos

1. El cliente envía una petición HTTP al servidor.
2. La petición atraviesa los middlewares globales, encargados del parseo de datos, la gestión de sesiones y la inyección de variables globales.
3. Express.js determina el controlador responsable de manejar la petición.
4. Si la ruta lo requiere, se verifica el estado de autenticación del usuario mediante el middleware correspondiente.
5. El controlador procesa la petición, valida los datos de entrada y delega en la capa de modelo para acceder o modificar los datos almacenados mediante las operaciones correspondientes.
6. El controlador selecciona y renderiza la vista EJS correspondiente, pasando los datos obtenidos del modelo.
7. El servidor envía el HTML renderizado al cliente como respuesta.
8. El cliente puede realizar nuevas acciones que generan nuevas peticiones, reiniciando el ciclo.

4.2. Arquitectura de capas

La aplicación sigue el patrón arquitectónico MVC (Modelo-Vista-Controlador), que separa la lógica de la aplicación en tres componentes principales.

Modelo (Model)

El modelo representa la capa de datos y la lógica de negocio de la aplicación. En esta implementación, el modelo está compuesto por los siguientes elementos:

- **Capa de acceso a datos:** El módulo *db.js* centraliza la configuración de la conexión a la base de datos mediante un pool de conexiones. Este módulo exporta una instancia única del pool que es reutilizada por todos los componentes del modelo, garantizando eficiencia en el uso de recursos.
- **Lógica de negocio y persistencia:** La lógica asociada al tratamiento de los datos y a la persistencia se encuentra encapsulada en la capa de modelo. Las operaciones de acceso a datos se realizan mediante métodos del modelo que encapsulan las consultas SQL, asegurando que los controladores actúan como intermediarios entre la vista y la lógica de negocio sin conocer los detalles de implementación de la persistencia.
- **Servicios de dominio:** Para funcionalidades que requieren lógica de negocio más compleja, la aplicación implementa servicios especializados (ubicados en la carpeta *services/*) que encapsulan operaciones específicas del dominio, como el análisis predictivo, la detección de anomalías, la gestión de notificaciones o el sistema de gamificación. Estos servicios forman parte de la capa de modelo y son utilizados por los controladores para realizar operaciones complejas que van más allá de simples consultas CRUD.
- **Entidades de dominio:** Las tablas de la base de datos (usuarios, movimientos, monederos, metas, foro, etc.) representan las entidades del dominio de la aplicación, reflejando de forma estructurada los conceptos clave del sistema.

Vista (View)

La capa de vista está implementada mediante el motor de plantillas EJS y se encuentra organizada en la carpeta *views/*. Sus principales características son:

- **Renderizado del lado del servidor (SSR):** Las vistas se renderizan en el servidor antes de ser enviadas al cliente, lo que mejora el SEO y reduce los tiempos de carga inicial.
- **Componentes reutilizables:** Se utilizan partials de EJS para elementos comunes como *navbar.ejs*, *navbarLogeado.ejs* y *footer.ejs*, favoreciendo la reutilización de código y la mantenibilidad.

- **Inyección de variables globales:** El uso de *res.locals* permite inyectar variables globales (como *username*, *foto_perfil* y *page*) en todas las vistas de forma transparente.
- **Organización modular:** Cada funcionalidad del sistema dispone de sus propias vistas (por ejemplo, *login.ejs*, *perfil.ejs*, *monedero.ejs*), manteniendo una estructura clara, organizada y coherente.

Controlador (Controller)

La capa de controlador está implementada mediante módulos de rutas de Express.js ubicados en la carpeta *routes/*. Estos controladores actúan como intermediarios entre la vista y el modelo:

- **Modularidad:** La aplicación se organiza en múltiples módulos de rutas especializados:
  - *auth.js*: Autenticación, registro, login y recuperación de contraseña.
  - *inicio.js*: Página principal del usuario autenticado.
  - *cuentas.js*: Gestión financiera.
  - *monedero.js*: Gestión de monederos virtuales.
  - Otros módulos especializados.
- **Separación de responsabilidades:** Los controladores se encargan de gestionar las peticiones HTTP, coordinar la interacción con el modelo y seleccionar la vista adecuada, manteniendo una clara separación entre lógica de presentación y lógica de negocio.
- **Gestión del flujo de la aplicación:** Cada controlador valida los datos de entrada recibidos en las peticiones HTTP, orquesta las operaciones necesarias delegando en el modelo cuando se requiere acceso a datos, y devuelve la respuesta correspondiente al cliente mediante la renderización de la vista adecuada, siguiendo el flujo establecido por el patrón MVC.

4.3. Mecanismos arquitectónicos

Sistema de Enrutamiento Modular

La aplicación implementa un sistema de enrutamiento modular que permite una organización clara y escalable de las rutas. Cada módulo funcional tiene su propio archivo de rutas en la carpeta *routes/*, que se registra en *app.js* mediante *app.use()*. Esta organización permite:

- **Separación por dominio funcional:** Cada módulo maneja un conjunto específico de rutas relacionadas (por ejemplo, todas las rutas de autenticación en *auth.js*, todas las rutas financieras en *cuentas.js*).
- **Escalabilidad:** Resulta sencillo añadir nuevos módulos funcionales sin modificar el código existente, simplemente creando un nuevo archivo de rutas y registrándolo en *app.js*.
- **Mantenibilidad:** Cada desarrollador puede trabajar en un módulo específico sin afectar a otros, facilitando el trabajo en equipo y el mantenimiento del código.

Sistema de Middlewares

- **Middlewares Globales:** Se configuran en *app.js* y se ejecutan para todas las peticiones:
  - *express.urlencoded* y *express.json*: Parsean los datos del cuerpo de las peticiones HTTP, convirtiendo los datos de formularios y JSON en objetos JavaScript accesibles mediante *req.body*.
  - *express-session*: Gestiona las sesiones de usuario, creando y manteniendo el estado de sesión entre peticiones.
  - **Middleware personalizado de variables globales:** Inyecta automáticamente variables como *username*, *foto_perfil*, *user* y *page* a todas las vistas mediante *res.locals*, evitando la necesidad de pasarlas manualmente en cada renderizado.
  - *express.static*: Sirve archivos estáticos (CSS, JavaScript e imágenes) desde la carpeta *public/*.
- **Middlewares de Aplicación:** Se aplican a rutas específicas mediante *router.use()* o como segundo parámetro en la definición de rutas:
  - *isAuthenticated*: Middleware de autenticación que verifica si el usuario tiene una sesión activa. Si no está autenticado, redirige al login o devuelve un error JSON en caso de peticiones AJAX. Existe una implementación centralizada en *middlewares/authMiddleware.js* que maneja tanto peticiones HTTP tradicionales como peticiones AJAX/API. Algunos archivos de rutas (como *routes/inicio.js* y *routes/cuentas.js*) definen

variantes locales cuando la lógica de autenticación es específica de ese módulo.

- *upload.single(foto_perfil)*: Middleware de Multer que procesa la subida de archivos, específicamente para imágenes de perfil. Valida el tipo y el tamaño del archivo antes de que llegue al controlador.

- **Middlewares de Ruta:** Algunos archivos de rutas definen sus propios middlewares locales cuando la lógica es específica de ese módulo. Por ejemplo, *routes/inicio.js* define sus propias funciones *isAuthenticated* adaptadas a sus necesidades concretas, aunque la mayoría de rutas utilizan la implementación centralizada.

Patrón de Servicios

Para funcionalidades complejas que requieren una lógica de negocio más elaborada, la aplicación implementa el patrón de servicios:

- **Servicios especializados:** Funcionalidades como el dashboard personalizable, el sistema de notificaciones y el sistema de logros utilizan servicios dedicados (por ejemplo, *DashboardService.js*, *NotificationService.js* y *AchievementService.js*), ubicados en la carpeta *services/*.

- **Separación de lógica:** Los servicios encapsulan la lógica de negocio compleja, permitiendo que los controladores se mantengan ligeros y enfocados en el manejo de peticiones HTTP y la coordinación entre modelo y vista.

- **Reutilización:** Los servicios pueden ser reutilizados por múltiples controladores o incluso por otros servicios, promoviendo el principio DRY (Don't Repeat Yourself).

Gestión de Estado y Sesiones

La aplicación utiliza sesiones del lado del servidor para mantener el estado del usuario:

- **Almacenamiento en memoria:** Las sesiones se almacenan en la memoria del servidor mediante el almacenamiento por defecto de *express-session*. En un entorno de producción, este almacenamiento podría externalizarse a un sistema persistente para mejorar la escalabilidad y la tolerancia a fallos.

- **Persistencia entre ventanas:** Las sesiones se mantienen al abrir nuevas ventanas o pestañas del navegador gracias al uso de cookies, que son compartidas entre todas las ventanas del mismo dominio, independientemente del tipo de almacenamiento utilizado.

- **Datos de sesión:** La sesión almacena información del usuario autenticado (*req.session.user*), incluyendo *id*, *username*, *nombre*, *apellidos*, *email* y *foto_perfil*.

- **Seguridad:** Las cookies de sesión están configuradas con *httpOnly: true* para prevenir ataques XSS, y pueden configurarse con *secure: true* en entornos de producción con HTTPS para garantizar que las cookies solo se transmitan a través de conexiones seguras.
- **Expiración:** Las sesiones expiran después de 24 horas de inactividad.

Organización de Archivos y Carpetas

La estructura del proyecto sigue convenciones claras que facilitan la navegación y el mantenimiento del sistema. Esta organización promueve:

- **Separación de responsabilidades:** Cada tipo de archivo tiene su ubicación lógica dentro del proyecto.
  - *routes/*: Controladores (módulos de rutas).
  - *views/*: Vistas (plantillas EJS).
  - *business/*: Lógica de negocio (capa de negocio que coordina operaciones entre DAOs).
  - *dao/*: Objetos de acceso a datos (Data Access Objects) que encapsulan las consultas SQL.
  - *services/*: Lógica de negocio compleja (servicios transversales).
  - *middlewares/*: Middlewares reutilizables.
  - *public/*: Archivos estáticos (CSS, JavaScript, imágenes).
- **Escalabilidad:** Resulta sencillo añadir nuevas funcionalidades sin afectar a las existentes, simplemente creando nuevos archivos en las carpetas correspondientes.
- **Mantenibilidad y preparación para producción:** La estructura modular y la clara separación de responsabilidades facilitan el mantenimiento del código y su adaptación a entornos de producción.

Manejo de Errores y Validación

La aplicación implementa un manejo de errores consistente:

- **Bloques try-catch en operaciones asíncronas:** Todas las operaciones de base de datos y tareas asíncronas están encapsuladas en bloques try-catch para capturar y gestionar errores de forma controlada.
- **Validación de entrada:** Los datos introducidos por el usuario se validan tanto en el cliente (mediante HTML5 y JavaScript) como en el servidor antes de ser procesados.

- **Mensajes de error comprensibles:** Los errores se presentan al usuario mediante mensajes claros y comprensibles, sin exponer detalles técnicos internos que podrían comprometer la seguridad o confundir al usuario.
- **Registro de errores:** Los errores se registran en la consola del servidor mediante `console.error()` para facilitar el proceso de depuración durante el desarrollo. En un entorno de producción, estos errores podrían ser registrados en un sistema de logging más robusto.

4.4. Gestión de Dependencias

Todas las dependencias del proyecto se gestionan mediante el archivo `package.json`, utilizando npm como gestor de paquetes. Esto permite una instalación sencilla de todas las dependencias mediante el comando `npm install` y garantiza la reproducibilidad del entorno de desarrollo.

4.5. Bases de datos

La base de datos está implementada utilizando MySQL y ha sido diseñada siguiendo principios de normalización, estableciendo relaciones adecuadas entre las distintas entidades del sistema con el objetivo de garantizar la integridad referencial, minimizar la redundancia de información y optimizar el rendimiento de las consultas.

Se ha utilizado el entorno XAMPP como plataforma local para la ejecución del servidor web y del sistema gestor de bases de datos. Este entorno incluye la herramienta phpMyAdmin, que ha sido empleada para su administración. El diseño se inició con la elaboración de un modelo entidad–relación en el que se identificaron las principales entidades del sistema, junto con sus atributos y relaciones.

Para optimizar el rendimiento de las consultas, se han definido índices estratégicos en claves foráneas y en campos utilizados con frecuencia en operaciones de búsqueda, optimizando así las consultas más habituales. Estos índices son los que empiezan por `idx` en las tablas.

La base de datos está compuesta por un total de 41 tablas, organizadas en distintos módulos funcionales que reflejan la estructura lógica de la aplicación. A continuación, se describe la estructura completa de la base de datos:

Módulo de Autenticación y Gestión de Usuarios

El módulo de Autenticación y Gestión de Usuarios es el encargado de administrar la identidad de los usuarios dentro de la aplicación. Este módulo constituye la base del sistema, ya que todas las demás funcionalidades dependen de la identificación y autenticación de los usuarios.

La tabla principal es *usuarios*. Esta tabla implementa restricciones de unicidad tanto en el nombre de usuario como en el correo electrónico, garantizando que no puedan existir duplicados. Desde el punto de vista de seguridad, las contraseñas se almacenan utilizando técnicas de hashing, nunca en texto plano, lo que protege la información sensible incluso en caso de acceso no autorizado a la base de datos.

La tabla *usuarios* actúa como núcleo central del sistema, siendo referenciada por prácticamente todas las demás tablas mediante claves foráneas. Esta estructura garantiza la integridad referencial: cuando un usuario es eliminado del sistema, todas sus relaciones con otras entidades se eliminan automáticamente gracias a la configuración de eliminación en cascada.

Para gestionar la recuperación de contraseñas, el sistema cuenta con la tabla *password_resets*, que almacena tokens temporales y seguros. Estos tokens permiten a los usuarios recuperar sus contraseñas mediante enlaces enviados por correo electrónico, y están diseñados para expirar automáticamente después de un período determinado, garantizando la seguridad del proceso de recuperación.

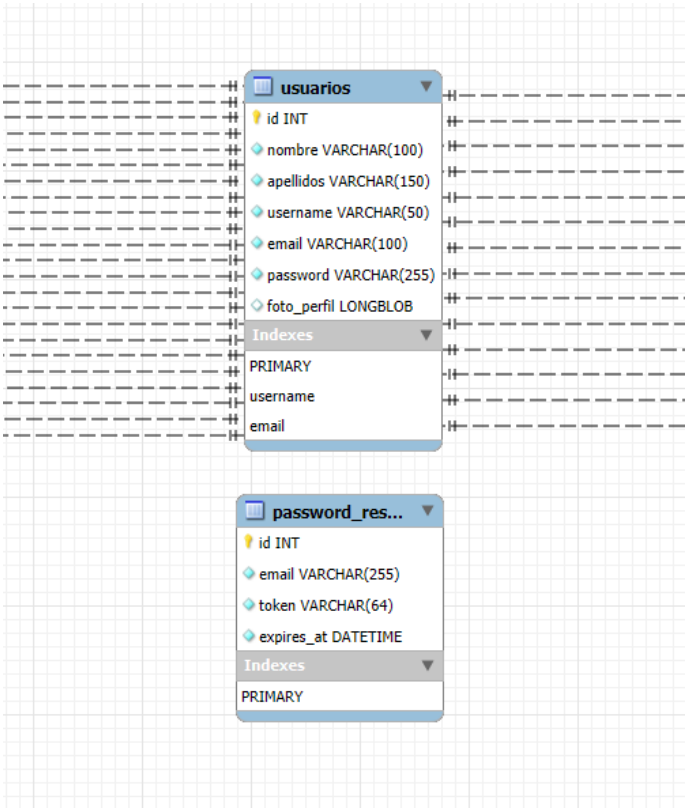

Figura 4.1: Modelo Entidad Relación del módulo de autenticación y gestión financiera

Módulo de Gestión Financiera

El módulo de gestión financiera representa el núcleo funcional de la aplicación, permitiendo a los usuarios registrar, organizar y gestionar todos sus movimientos económicos.

La tabla *movimientos* constituye el registro central de todas las transacciones financieras del usuario, tanto ingresos como gastos. La arquitectura permite categorías especiales que facilitan funcionalidades avanzadas, como la identificación de pagos entre usuarios en grupos financieros o transferencias entre monederos virtuales.

Para ofrecer una organización más granular de las finanzas, el sistema implementa el concepto de monederos virtuales a través de la tabla *monederos*.

Los movimientos dentro de cada monedero se registran en la tabla *movimientos_monedero*, que mantiene una relación tanto con el usuario como con el monedero específico. Adicionalmente, la tabla *transferencias_monedero* registra las transferencias entre monederos del mismo usuario, proporcionando un historial completo de los movimientos entre diferentes contenedores.

La tabla *metas* permite a los usuarios establecer objetivos financieros específicos y hacer seguimiento de su progreso. Las metas pueden estar asociadas a monederos

específicos o a la cuenta de ahorros general.

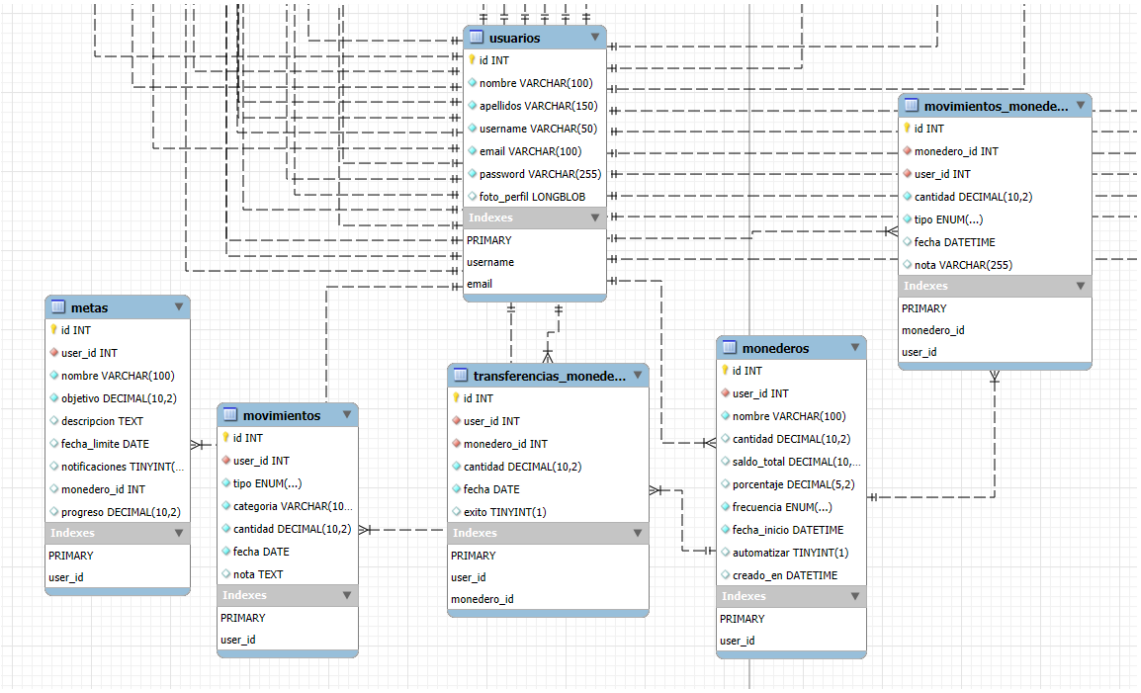

Figura 4.2: Modelo Entidad Relación del módulo de gestión financiera

Módulo de Inversiones

El módulo de inversiones permite a los usuarios registrar y hacer seguimiento de sus inversiones en diferentes tipos de activos financieros. La arquitectura está diseñada para soportar múltiples tipos de inversiones, cada una con sus características específicas.

La tabla *acciones* almacena las inversiones en acciones de los usuarios, registrando información como el símbolo de la acción, la cantidad adquirida, el precio de compra y la fecha de la transacción. Esta estructura permite a los usuarios mantener un registro preciso de su cartera de acciones y realizar un seguimiento de sus inversiones a lo largo del tiempo.

La tabla *criptomonedas* almacena información similar pero adaptada a las características específicas de estos activos digitales. Se han implementado índices adicionales en esta tabla para optimizar las consultas frecuentes, mejorando significativamente el rendimiento de las operaciones de consulta.

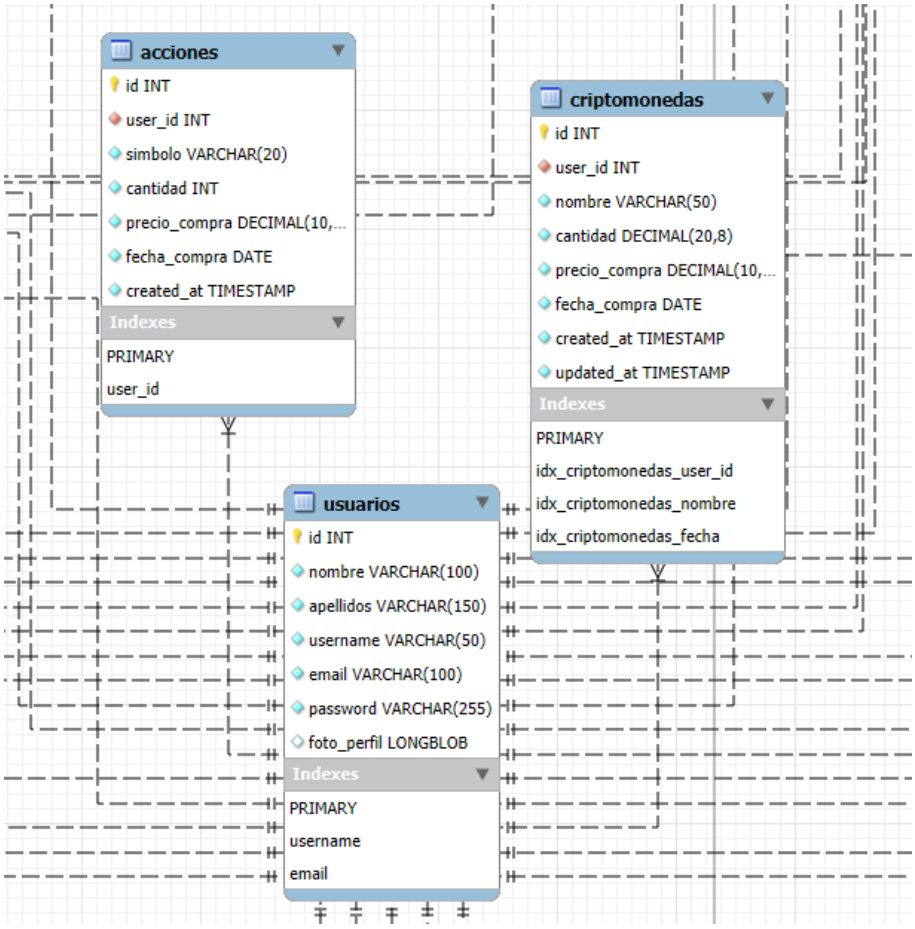

Figura 4.3: Modelo Entidad Relación del módulo de inversiones

Módulo de Grupos Financieros

Este módulo implementa funcionalidades de colaboración financiera similares a las de la aplicación TriCount.

La tabla *grupos* almacena la información fundamental de cada grupo financiero. Cada grupo está asociado a un usuario creador del grupo (administrador) y la relación está configurada para eliminar el grupo automáticamente si el administrador es eliminado del sistema. Se han implementado índices adicionales para optimizar las búsquedas por administrador y los filtros por estado.

La tabla *grupo_miembros* establece las relaciones entre usuarios y grupos, definiendo quiénes son miembros de cada grupo. Esta tabla implementa una restricción de clave única compuesta que previene que un usuario sea miembro del mismo grupo múltiples veces. Los índices en esta tabla están optimizados para consultas tanto por grupo como por usuario, facilitando operaciones como listar todos los miembros de un grupo o todos los grupos a los que pertenece un usuario.

Los gastos compartidos se registran en la tabla *grupo_gastos*. La arquitectura

utiliza un campo JSON para almacenar la estructura de división del gasto, lo que proporciona flexibilidad para diferentes métodos de distribución (por partes iguales, porcentajes personalizados, etc.).

Para gestionar los pagos entre miembros que equilibran las deudas, el sistema cuenta con la tabla *grupo_pagos*, que registra los pagos pendientes entre usuarios. Una vez completados, estos pagos se mueven a la tabla *grupo_pagos_historial*, manteniendo un registro histórico completo de todas las transacciones realizadas. Esta separación entre pagos pendientes e históricos facilita la gestión y consulta de información, permitiendo distinguir rápidamente entre deudas activas y transacciones completadas.

El sistema de invitaciones está gestionado por la tabla *grupo_invitaciones*, que almacena tokens únicos para cada invitación. Estos tokens permiten a los usuarios unirse a grupos mediante enlaces seguros y están diseñados para expirar después de un período determinado. Los índices en esta tabla optimizan las búsquedas por grupo y por token, facilitando la validación de invitaciones.

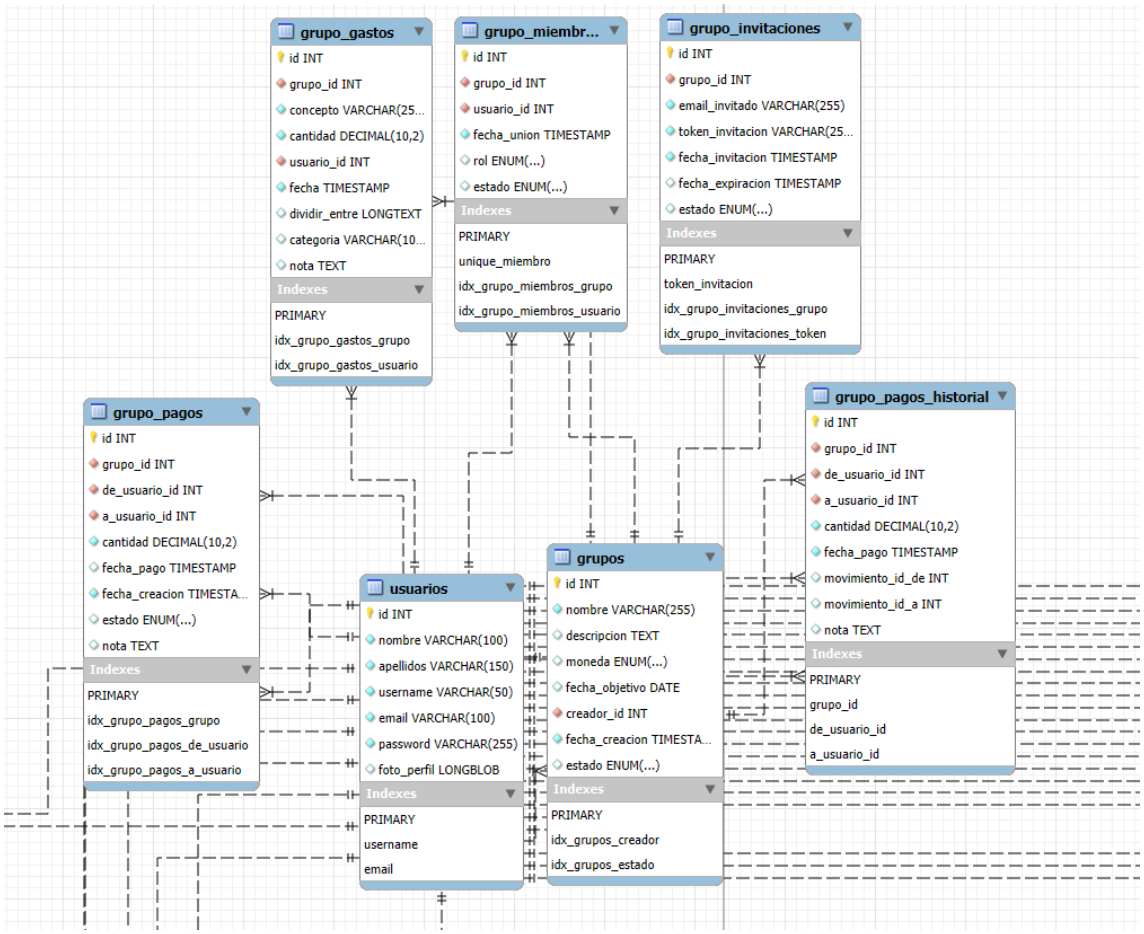

Figura 4.4: Modelo Entidad Relación del módulo de grupos financieros

Módulo de Notificaciones

El módulo de notificaciones implementa un sistema inteligente y personalizable para mantener a los usuarios informados sobre eventos importantes en la aplicación. La arquitectura está diseñada para ser flexible y eficiente, permitiendo diferentes tipos de notificaciones con distintos niveles de prioridad.

La tabla *notificaciones* constituye el núcleo del módulo, almacenando todas las notificaciones generadas para cada usuario. La estructura permite diferentes tipos de notificaciones, desde alertas financieras hasta recordatorios y notificaciones de logros desbloqueados. Se han implementado múltiples índices para optimizar las consultas más frecuentes, incluyendo búsquedas por usuario, filtros por tipo y estado de lectura, y ordenación por fecha. Un índice compuesto adicional optimiza las consultas que combinan usuario y fecha, que son muy comunes en la aplicación.

Para facilitar la reutilización y mantener consistencia en los mensajes, el sistema cuenta con la tabla *plantillas_notificaciones*, que almacena plantillas predefinidas para diferentes tipos de notificaciones. Esta aproximación permite que el sistema genere notificaciones consistentes y bien formateadas sin necesidad de duplicar código o mensajes.

La personalización del sistema de notificaciones se gestiona mediante la tabla *preferencias_notificaciones*, que permite a cada usuario configurar sus preferencias para cada tipo de notificación. Esta tabla implementa una clave única compuesta que garantiza que cada usuario tenga una única configuración por tipo de notificación, evitando duplicados y manteniendo la integridad de los datos.

Los recordatorios personalizables se almacenan en la tabla *recordatorios*, que permite a los usuarios crear recordatorios para eventos o tareas específicas.

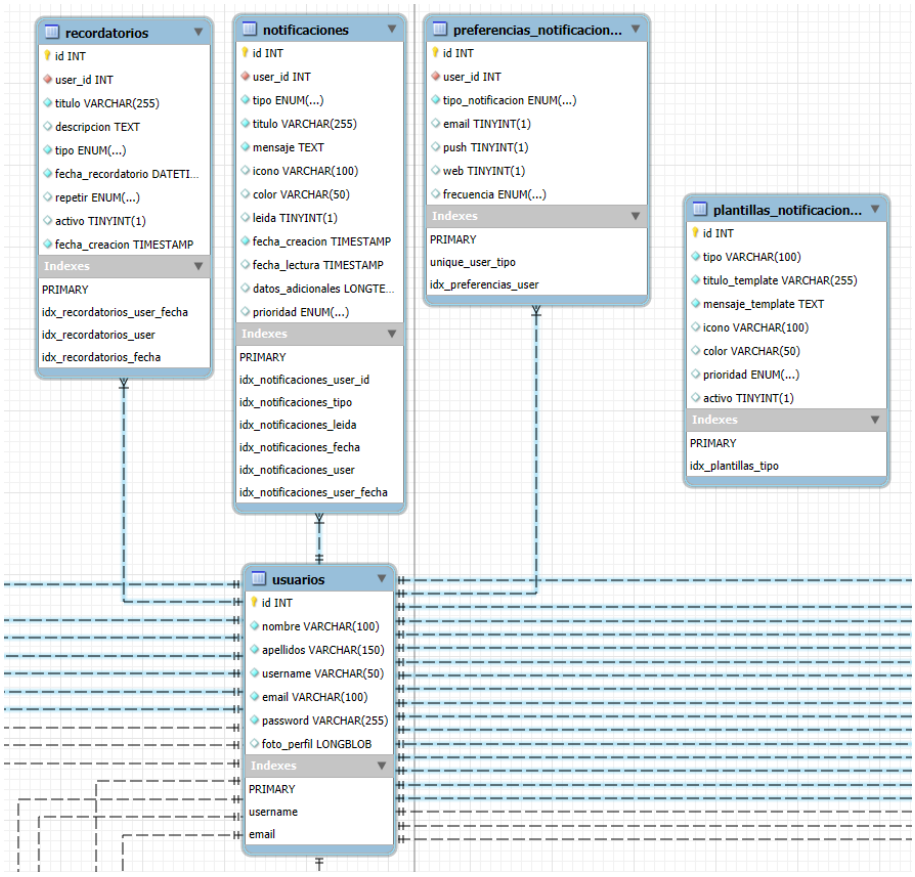

Figura 4.5: Modelo Entidad Relación del módulo de notificaciones

Módulo de Gamificación y Logros

La tabla *logros* define todos los logros disponibles en el sistema, incluyendo información sobre sus requisitos, categorías y estados. Esta tabla actúa como catálogo de logros, permitiendo que el sistema verifique automáticamente si un usuario ha cumplido los requisitos para desbloquear un logro específico.

La relación entre usuarios y logros desbloqueados se gestiona mediante la tabla *logros_usuario*, que registra qué logros ha obtenido cada usuario y cuándo los desbloqueó. Esta tabla implementa una clave única compuesta que previene que un usuario tenga el mismo logro registrado múltiples veces. Los índices están diseñados para optimizar consultas tanto por usuario como por logro, y un índice compuesto adicional facilita las consultas que combinan usuario y estado de completado, que son frecuentes en la aplicación.

El sistema de puntos y niveles se gestiona mediante la tabla *puntos_usuario*, que mantiene el estado actual de puntos y nivel de cada usuario. La restricción de unicidad en el identificador de usuario garantiza que cada usuario tenga un único registro de puntos. Para mantener un historial completo de cómo se obtuvieron los puntos, el sistema cuenta con la tabla *historial_puntos*, que registra cada evento que generó puntos, incluyendo la fecha y el logro asociado si aplica.

Los rankings se almacenan en la tabla *rankings*, que permite mantener clasificaciones en diferentes períodos. La estructura implementa una clave única compuesta que garantiza que cada usuario tenga una única posición en cada categoría y período, evitando duplicados.

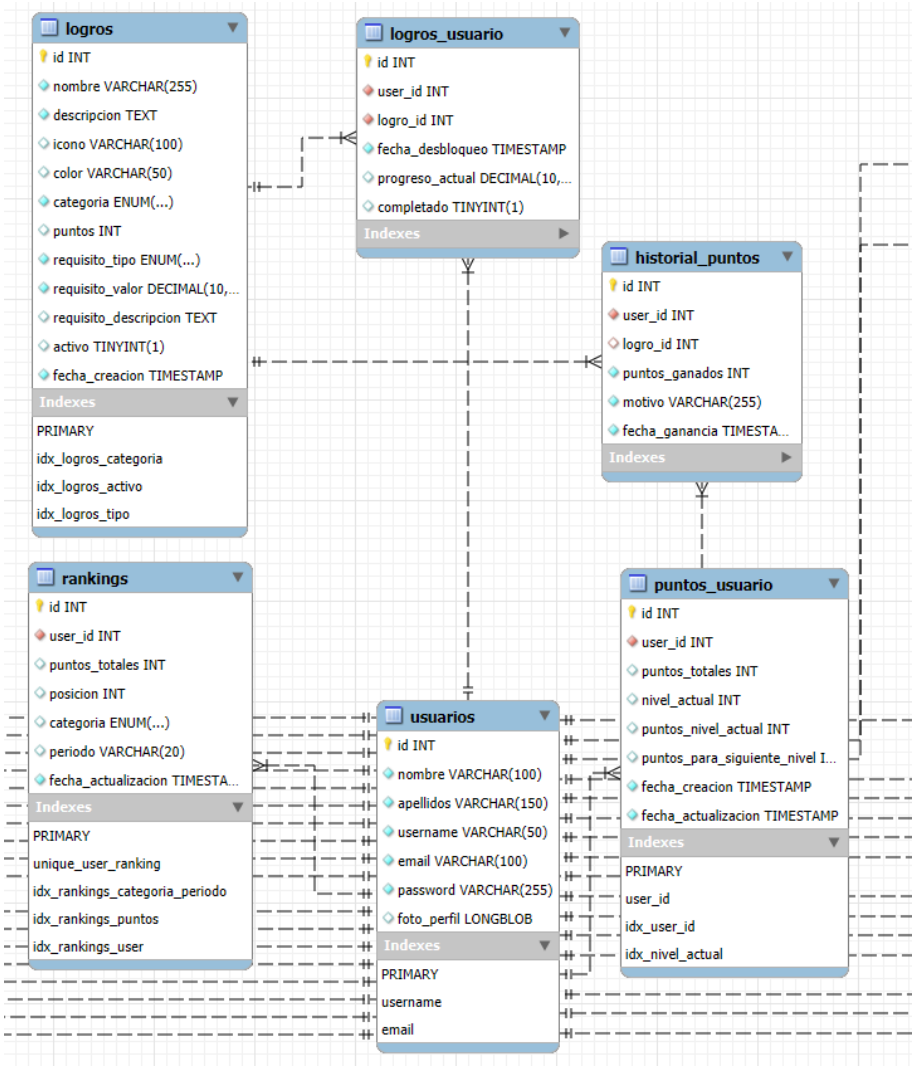

Figura 4.6: Modelo Entidad Relación del módulo de gamificación y logros

Módulo de Dashboard Personalizable

El módulo de Dashboard Personalizable permite personalizar su experiencia visual mediante la configuración de widgets y temas.

La tabla *widgets* define todos los widgets disponibles en el sistema, actuando como catálogo de componentes que los usuarios pueden agregar a su dashboard.

La personalización individual de cada usuario se gestiona mediante la tabla *widgets_usuario*, que almacena la configuración específica de cada widget para cada usuario. Esta tabla mantiene relaciones tanto con la tabla de usuarios como con la tabla de widgets, e implementa una clave única compuesta que garantiza que cada usuario tenga una única configuración por widget.

Figura 4.7: Modelo Entidad Relación del módulo del Dashboard personalizable

Módulo de Comunidad y Social

La arquitectura de este módulo está diseñada para soportar funcionalidades de comunidad como foros, reseñas y preguntas frecuentes.

El sistema de foro se estructura mediante tres tablas principales. La tabla *foro_preguntas* almacena las preguntas realizadas por los usuarios. La relación con la tabla de usuarios está configurada con eliminación en cascada, y los índices están optimizados para consultas por autor, facilitando la recuperación de todas las preguntas de un usuario específico.

Las respuestas a las preguntas se almacenan en la tabla *foro_respuestas*, que mantiene relaciones tanto con la pregunta original como con el usuario que respondió. Esta estructura permite un seguimiento completo de las conversaciones en el foro. Los índices están optimizados para consultas tanto por pregunta como por autor, mejorando el rendimiento cuando se recuperan todas las respuestas de una pregunta o todas las respuestas de un usuario.

Para permitir que los usuarios expresen su apreciación por las preguntas, el sistema cuenta con la tabla *foro_likes*, que registra los "me gusta" en las preguntas. La estructura implementa una clave única compuesta que previene que un usuario pueda dar múltiples "me gusta.ª la misma pregunta, manteniendo la integridad de los datos.

El sistema de reseñas permite a los usuarios compartir sus opiniones sobre la aplicación mediante la tabla *resenas*, que almacena calificaciones y comentarios. Las preguntas frecuentes se gestionan mediante las tablas *faqs* y *secciones*, donde las secciones organizan las FAQs en categorías lógicas, facilitando la navegación y búsqueda de información.

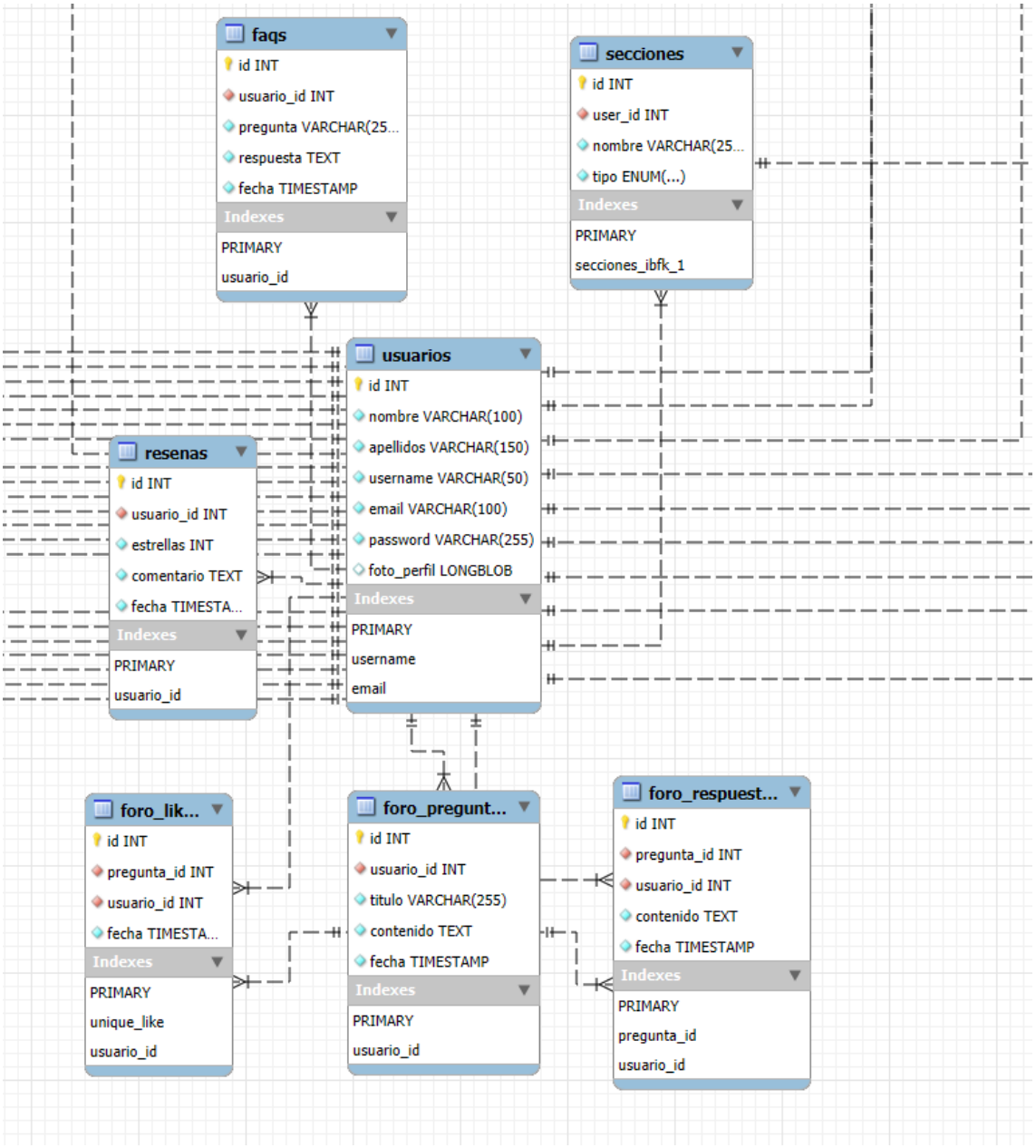

Figura 4.8: Modelo Entidad Relación del módulo de comunidad y social

Módulo de Mensajería y Comunicación

Este módulo implementa sistemas de comunicación tanto síncronos como asíncronos entre usuarios, permitiendo la interacción directa y el intercambio de mensajes.

Las relaciones de amistad se gestionan mediante la tabla *amistades*, que establece conexiones bidireccionales entre usuarios. La estructura implementa una clave única compuesta que previene relaciones duplicadas, y los índices están optimizados para consultas en ambas direcciones, facilitando la recuperación tanto de los amigos de un usuario como la verificación de si dos usuarios son amigos.

Para la comunicación en tiempo real, el sistema cuenta con las tablas *chats* y *mensajes_chat*. La tabla *chats* establece las conversaciones entre pares de usuarios, almacenando información sobre el último mensaje para facilitar la recuperación rápida de conversaciones activas. La estructura previene conversaciones duplicadas mediante una clave única compuesta, y los índices están optimizados para búsquedas bidireccionales. Los mensajes individuales se almacenan en la tabla *mensajes_chat*, que mantiene relaciones con la conversación y el remitente. Los índices están optimizados para consultas por conversación, remitente y fecha, facilitando la recuperación eficiente de mensajes ordenados cronológicamente.

El sistema de buzón implementa mensajería asíncrona mediante la tabla *mensajes_buzon*, que permite a los usuarios enviar mensajes que pueden ser leídos posteriormente. Esta tabla mantiene relaciones con la tabla de usuarios tanto para el remitente como para el destinatario, y almacena información sobre el estado de lectura, tipo de mensaje y fechas.

Para soportar el envío de archivos adjuntos, el sistema cuenta con la tabla *adjuntos_buzon*, que almacena información sobre los archivos asociados a cada mensaje. La relación con la tabla de mensajes está configurada con eliminación en cascada, asegurando que cuando un mensaje es eliminado, todos sus adjuntos también se eliminan.

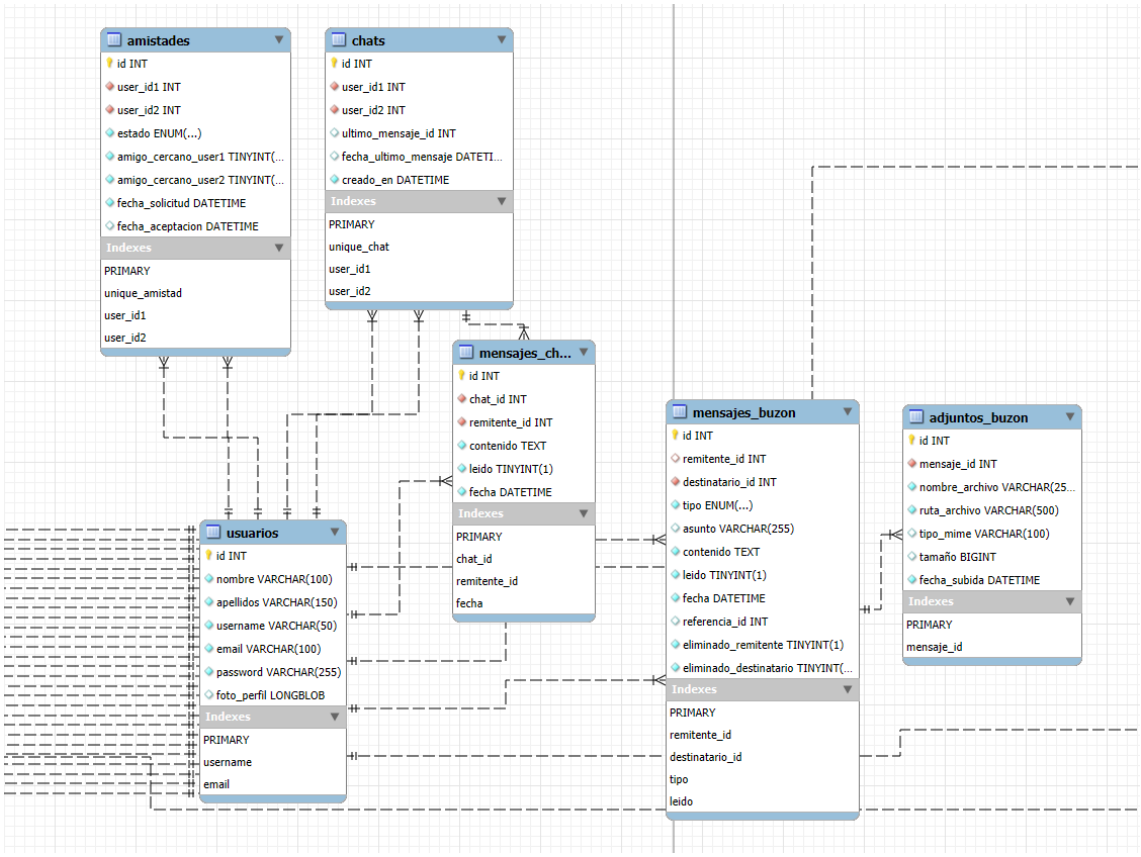

Figura 4.9: Modelo Entidad Relación del módulo de mensajería y comunicación

Módulo de Configuración y Preferencias

Este módulo gestiona las preferencias de configuración y personalización visual de cada usuario.

La tabla *preferencias_usuario* almacena todas las configuraciones personalizadas de cada usuario, incluyendo preferencias de visualización, notificaciones y otras opciones del sistema. La estructura implementa una restricción de unicidad que garantiza que cada usuario tenga un único registro de preferencias, manteniendo la integridad de los datos.

Los temas visuales disponibles para el dashboard se definen en la tabla *temas_visuales*, que actúa como catálogo de opciones de personalización. Esta tabla almacena información sobre cada tema, incluyendo su tipo (claro, oscuro, personalizado) y los colores asociados, permitiendo que los usuarios elijan entre diferentes esquemas de color para personalizar su experiencia visual.

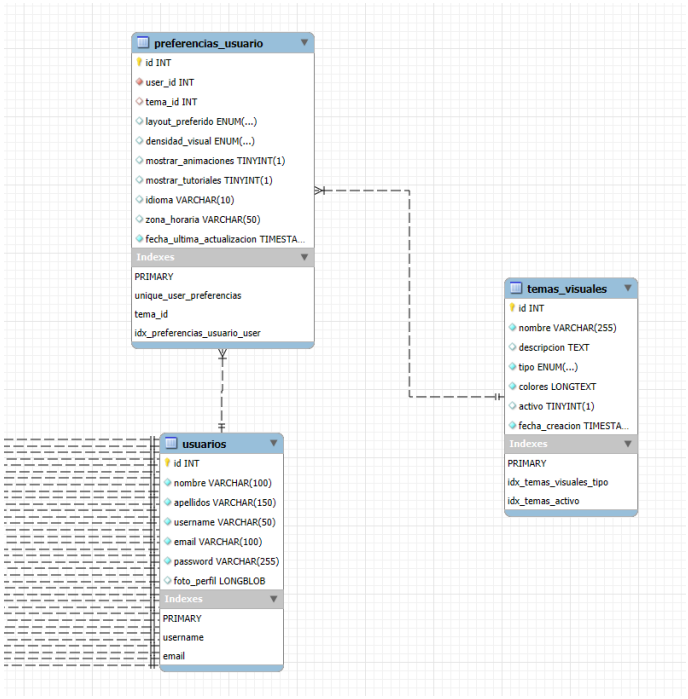

Figura 4.10: Modelo Entidad Relación del módulo de configuración y preferencias

Tabla de Sesiones

Tabla disponible en la base de datos para almacenar las sesiones de usuario mediante *express-mysql-session*. Las sesiones se almacenan en memoria del servidor mediante el almacenamiento por defecto de *express-session*.

Tablas Auxiliares

El sistema cuenta con tablas auxiliares que soportan funcionalidades específicas del sistema. La tabla *gestion_items* proporciona una estructura genérica para la gestión de elementos del sistema que requieren una organización flexible y extensible.

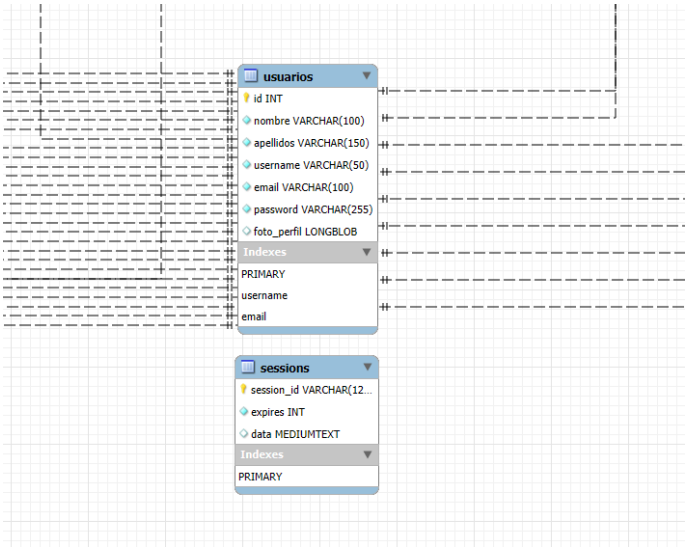

Figura 4.11: Modelo Entidad Relación del módulo de sesiones

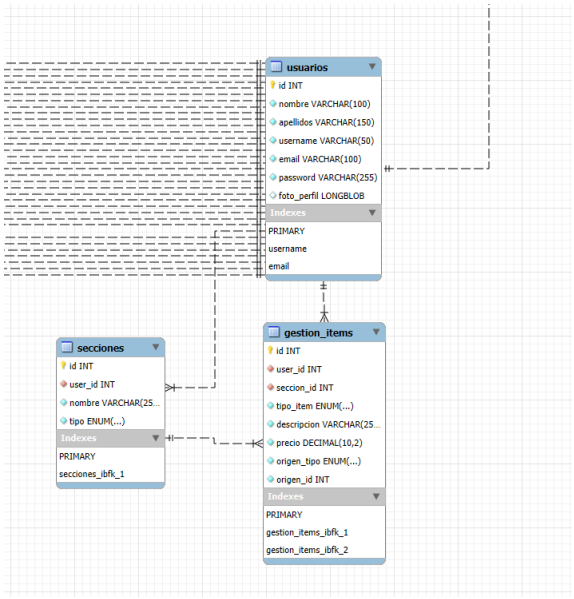

Figura 4.12: Modelo Entidad Relación del módulo de tablas auxiliares

4.6. Despliegue de la aplicación

En este apartado se describen los pasos necesarios para desplegar y ejecutar la aplicación en un entorno local, incluyendo los requisitos previos, la configuración de la base de datos y la puesta en marcha del servidor.

4.6.1. Requisitos previos

Antes de proceder con la instalación, es necesario contar con los siguientes elementos:

- **XAMPP** (o software similar) con **MySQL** instalado.
- **Node.js** y **npm** correctamente instalados.
- **Visual Studio Code** o cualquier otro editor de código.

4.6.2. Instrucciones de instalación

4.6.2.1. Configuración de la base de datos

1. Abrir **XAMPP** e iniciar el servicio de **MySQL**.
2. Acceder a **phpMyAdmin**, normalmente disponible en:

   ```
   http://localhost/phpmyadmin
   ```

3. Crear una nueva base de datos llamada `sfs`:
   - Pulsar en "Nueva" en el menú lateral.
   - Introducir el nombre `sfs`.
   - Seleccionar la intercalación `utf8mb4_general_ci`.
   - Hacer clic en "Crear".
4. Importar el archivo SQL:
   - Seleccionar la base de datos `sfs`.
   - Acceder a la pestaña "Importar".
   - Seleccionar el archivo `sfs.sql` ubicado en la raíz del proyecto.
   - Pulsar "Continuar" para ejecutar la importación.

Como alternativa, el contenido del archivo `sfs.sql` puede copiarse y ejecutarse directamente desde la pestaña "SQL" de phpMyAdmin.

4.6.2.2. Configuración del proyecto

1. Clonar o extraer la carpeta del proyecto en la máquina local.
2. Abrir el proyecto con **Visual Studio Code** o el editor preferido.
3. Abrir una terminal en la carpeta raíz del proyecto.

4.6.2.3. Instalación de dependencias

Para instalar todas las dependencias necesarias, se debe ejecutar el siguiente comando en la terminal:

```
npm install
```

Este comando descargará todos los módulos definidos en el archivo `package.json` dentro de la carpeta `node_modules`.

4.6.2.4. Inicio de la aplicación

Una vez instaladas las dependencias, la aplicación puede iniciarse de dos formas:
Opción 1: Usando Node.js directamente

```
node app.js
```

Opción 2: Usando Nodemon (recomendado para desarrollo)

```
nodemon app.js
```

En caso de no disponer de Nodemon instalado de forma global, puede instalarse mediante:

```
npm install -g nodemon
```

Alternativamente, puede utilizarse el script definido en npm:

```
npm start
```

4.6.2.5. Acceso a la aplicación

Cuando el servidor se encuentre en ejecución, se mostrará en la consola el siguiente mensaje:

```
Servidor escuchando en http://localhost:3000
```

La aplicación estará disponible desde el navegador web en la siguiente dirección:

```
http://localhost:3000
```

4.6.3. Estructura del proyecto

La estructura principal del proyecto es la siguiente:

- `app.js`: archivo principal de la aplicación.
- `sfs.sql`: script SQL para la creación de la base de datos.
- `routes/`: definición de las rutas de la aplicación.
- `views/`: plantillas EJS.
- `public/`: archivos estáticos (CSS, JavaScript e imágenes).
- `business/`: lógica de negocio.
- `dao/`: acceso a datos.
- `middlewares/`: middlewares personalizados.
- `services/`: servicios de la aplicación.

4.6.4. Solución de problemas

- **Error de conexión a MySQL**: verificar que XAMPP esté en ejecución y que el servicio MySQL esté iniciado.
- **Puerto en uso**: si el puerto 3000 está ocupado, puede modificarse en el archivo `app.js`.
- **Módulos no encontrados**: ejecutar nuevamente `npm install` para asegurar la correcta instalación de las dependencias.

4.6.5. Notas adicionales

- La aplicación utiliza sesiones para mantener el estado del usuario.
- Los archivos subidos se almacenan en la carpeta `public/uploads/`.
- Es necesario disponer de permisos de escritura en dicha carpeta para la correcta subida de archivos.

Capítulo 5

Tecnologías empleadas

La selección de las tecnologías empleadas en este proyecto no sólo responde a criterios técnicos, sino también a factores como la adecuación a los objetivos planteados, la escalabilidad, la mantenibilidad y la experiencia previa del equipo de desarrollo.

Dado el alcance del Trabajo de Fin de Grado y el tiempo disponible para su realización, se ha optado por un conjunto de tecnologías ampliamente consolidadas en el desarrollo de aplicaciones web, que permiten construir una solución completa, funcional y coherente. La familiaridad previa con gran parte de estas herramientas ha permitido centrar los esfuerzos en el diseño de la arquitectura, la implementación de la lógica de negocio y la calidad de la experiencia de usuario.

A lo largo del desarrollo se han incorporado librerías y herramientas que no formaban parte de la experiencia inicial del equipo, como *Chart.js* para la visualización de datos, *PDFKit* para la generación de documentos o la integración de APIs externas para la obtención de información financiera en tiempo real. Esto ha permitido ampliar las competencias técnicas y enriquecer el proyecto con funcionalidades avanzadas, manteniendo un equilibrio entre aprendizaje y viabilidad.

Existen alternativas tecnológicas igualmente válidas para la implementación de este tipo de aplicaciones, como frameworks frontend más complejos (*Angular*, *React* o *Vue*), otros sistemas de bases de datos (*MongoDB* o *PostgreSQL*) o APIs financieras de carácter comercial. Con algunas de ellas también existe una familiaridad por parte del equipo de desarrollo, que ha sido adquirido durante la realización del grado en Ingeniería de Software. Sin embargo, se ha priorizado el uso de tecnologías ligeras, bien documentadas y con una amplia comunidad, que también garantizan un desarrollo estable y sostenible.

A continuación, se describen de forma detallada las tecnologías seleccionadas, así como los criterios técnicos que han fundamentado su elección en cada una de las áreas del sistema, abarcando el desarrollo del backend, la construcción del frontend y la implementación de los mecanismos de seguridad orientados a la protección de los datos del usuario.

5.1. Backend

El stack de backend seleccionado se basa en Node.js como entorno de ejecución, Express.js como framework web, y MySQL como sistema de gestión de bases de datos.

Node.js

Node.js es el entorno de ejecución de JavaScript en el servidor que permite ejecutar código JavaScript fuera del navegador.

Se ha elegido esta tecnología por su eficiencia en el manejo de operaciones asíncronas y su amplio ecosistema de paquetes disponibles a través de npm (Node Package Manager).

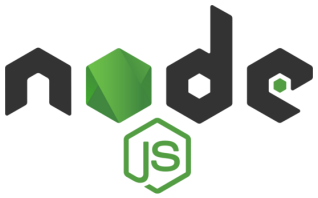

Figura 5.1: Logo Node.js

Express.js

Express.js es un framework web minimalista y flexible para Node.js que simplifica el desarrollo de aplicaciones web y APIs REST.

Se ha utilizado Express.js para definir las rutas de la aplicación, gestionar las peticiones HTTP y configurar los middlewares necesarios para el procesamiento de datos, autenticación y gestión de sesiones. Express.js permite una estructura de código organizada mediante el uso de routers modulares, facilitando la mantenibilidad del proyecto.

Figura 5.2: Logo Express.js

MySQL

MySQL es el sistema de gestión de bases de datos relacional (SGBDR) utilizado para almacenar toda la información de la aplicación.

Figura 5.3: Logo MySQL

Express Session

Para la gestión de las sesiones de usuario se ha utilizado *express-session*, un middleware que permite mantener información asociada a cada usuario entre distintas peticiones HTTP. Este mecanismo resulta fundamental para funcionalidades como la autenticación, el mantenimiento de la sesión iniciada y la protección de rutas privadas dentro de la aplicación.

Las sesiones se almacenan actualmente en la memoria del servidor mediante el almacenamiento por defecto de *express-session*. Esta configuración es adecuada para el desarrollo y entornos de prueba, ya que proporciona un rendimiento óptimo y una implementación sencilla. Para entornos de producción, la aplicación dispone de la dependencia *express-mysql-session*, que permite migrar el almacenamiento de sesiones a la base de datos MySQL, evitando así la pérdida de sesiones en caso de reinicio del servidor y permitiendo una gestión más robusta y escalable del sistema de sesiones en entornos distribuidos.

5.2. Frontend

Para el frontend se ha utilizado un enfoque basado en plantillas del lado del servidor (EJS) combinado con tecnologías web estándar y frameworks modernos para proporcionar una interfaz responsive e interactiva.

EJS (Embedded JavaScript)

EJS es un motor de plantillas que permite generar HTML dinámico mediante la inserción de código JavaScript en el servidor.

Se ha elegido EJS por su simplicidad y su sintaxis similar a HTML, lo que facilita la integración con el código existente. EJS permite pasar variables desde el servidor a las vistas, facilitando la renderización de datos dinámicos y la reutilización de componentes mediante partials (como la barra de navegación y el pie de página).

Figura 5.4: Logo EJS

HTML y CSS

Se ha utilizado HTML5 para la estructura semántica de las páginas web y CSS3 para el diseño y la presentación visual. Se ha implementado una arquitectura de estilos modular, con archivos CSS específicos para cada sección de la aplicación (login, registro, perfil, monedero, etc.), lo que facilita el mantenimiento y la organización del código de estilos.

Figura 5.5: Logo HTML y CSS

Bootstrap

Bootstrap es un framework CSS de código abierto que proporciona un sistema de diseño responsivo basado en un grid system de 12 columnas.

Se ha utilizado Bootstrap para garantizar que la aplicación sea completamente responsive y se adapte correctamente a diferentes tamaños de pantalla (móviles, tablets y escritorio). Bootstrap también proporciona componentes predefinidos como botones, formularios, modales y navegación, acelerando el desarrollo y asegurando una interfaz de usuario consistente.

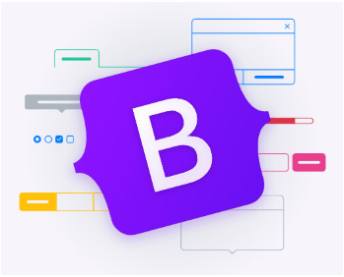

Figura 5.6: Logo Bootstrap

jQuery

jQuery es una biblioteca JavaScript que facilita la manipulación del DOM y el manejo de eventos en el lado del cliente.

En este proyecto se ha utilizado para implementar interacciones dinámicas y realizar peticiones AJAX, complementando el uso de JavaScript estándar.

Figura 5.7: Logo jQuery

Chart.js

Chart.js es una librería JavaScript de código abierto orientada a la visualización de datos mediante gráficos interactivos.

Se ha empleado en la aplicación para representar información financiera de forma visual e intuitiva, facilitando la interpretación de datos y tendencias por parte del usuario.

Figura 5.8: Logo Chart.js

5.3. Seguridad

Se han implementado medidas de seguridad orientadas a la protección de la información sensible de los usuarios, incluyendo el uso de algoritmos de hash para el almacenamiento de contraseñas y la generación segura de tokens para funcionalidades críticas como la recuperación de contraseña.

bcrypt

Para garantizar la seguridad de las contraseñas de los usuarios, se ha implementado el algoritmo de hash `bcrypt` mediante el módulo `bcrypt` de Node.js. Las

contraseñas se almacenan en la base de datos únicamente en su versión hasheada, nunca en texto plano.

Figura 5.9: Logo hash bcrypt

Crypto (Node.js)

El módulo nativo *crypto* de Node.js se ha utilizado para la generación de tokens criptográficos seguros en funcionalidades como la recuperación de contraseña. Estos tokens se generan utilizando *crypto.randomBytes()*, que produce bytes aleatorios criptográficamente seguros, garantizando la unicidad y seguridad de los tokens generados.

Figura 5.10: Logo Crypto

5.4. APIs Externas

Para proporcionar información financiera en tiempo real a los usuarios, se han integrado dos APIs externas especializadas en datos de mercado.

La integración de estas APIs se realiza mediante el patrón Gateway, donde el backend actúa como intermediario entre el frontend y las APIs externas, proporcionando una capa de abstracción que facilita el mantenimiento y permite la posible sustitución por otras APIs en el futuro.

CoinGecko API

CoinGecko es una plataforma de agregación de datos de criptomonedas que proporciona una API gratuita y sin necesidad de autenticación para obtener información sobre precios y estadísticas de mercado.

En este proyecto se ha integrado *CoinGecko API* en el módulo de gestión de inversiones para actualizar automáticamente el valor de las criptomonedas del portfolio del usuario.

La comunicación con la API se realiza mediante peticiones HTTP asíncronas, permitiendo la actualización de los precios sin necesidad de recargar la página.

Figura 5.11: Logo CoinGecko

Yahoo Finance API

Para la obtención de precios en tiempo real de acciones del mercado se ha utilizado la API no oficial de *Yahoo Finance*, que permite acceder de forma gratuita a datos de mercado.

Dado que los precios se obtienen en dólares estadounidenses, se ha implementado una conversión de moneda en el backend para mantener la coherencia con el resto del sistema, que opera en euros.

Al tratarse de una API no oficial, su uso está orientado a entornos de desarrollo y pruebas. Para un entorno de producción, podrían considerarse alternativas oficiales como *Alpha Vantage* o *IEX Cloud*.

Figura 5.12: Logo Yahoo Finance

5.5. Funcionalidades adicionales

Además de las tecnologías principales del backend y frontend, se han integrado librerías y herramientas para funcionalidades específicas que complementan las capacidades principales del sistema, como el envío de correos electrónicos, la gestión de archivos y la generación de documentos.

Nodemailer

Nodemailer es un módulo de Node.js diseñado para el envío de correos electrónicos.

Se ha implementado para funcionalidades como la recuperación de contraseña, permitiendo enviar correos electrónicos con enlaces de restablecimiento de forma segura. La configuración utiliza el servicio Gmail mediante autenticación con contraseña de aplicación, que es una alternativa segura a las contraseñas normales para servicios de terceros, garantizando la entrega confiable de los mensajes.

Figura 5.13: Logo Nodemailer

Multer

Multer es un middleware de Node.js para el manejo de datos *multipart/form-data*, utilizado principalmente para la subida de archivos.

En este proyecto se ha utilizado para permitir la carga de imágenes de perfil de usuario, aplicando validaciones sobre el tipo de archivo (JPEG, JPG y PNG) y estableciendo límites de tamaño para garantizar un uso seguro y controlado.

Las imágenes se almacenan en el servidor de forma organizada, asegurando la correcta gestión de los archivos y evitando conflictos en los nombres de almacenamiento.

Figura 5.14: Logo Multer Node.js

PDFKit

PDFKit es una librería para Node.js que permite la generación programática de documentos PDF.

Se ha implementado para generar reportes financieros personalizados, que incluyen información relevante sobre los movimientos, inversiones y metas del usuario.

Los documentos se generan dinámicamente con un formato estructurado y adaptado a los datos del usuario, facilitando la consulta y exportación de la información financiera.

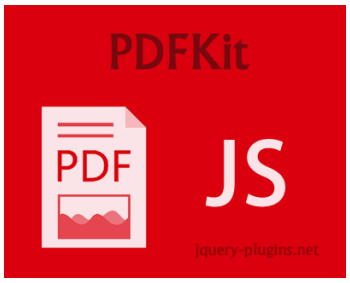

Figura 5.15: Logo PDFKit js

Capítulo 6

Conclusiones y Trabajo Futuro

6.1. Conclusiones generales

El desarrollo de la aplicación propuesta ha permitido cumplir los objetivos planteados al inicio de este Trabajo de Fin de Grado, dando lugar a una solución funcional orientada a la gestión de finanzas personales. A lo largo del proyecto se ha diseñado e implementado una aplicación capaz de registrar ingresos y gastos, establecer presupuestos, analizar el comportamiento financiero del usuario y ofrecer herramientas de apoyo a la toma de decisiones económicas.

Uno de los aspectos más relevantes del trabajo ha sido la incorporación de funcionalidades avanzadas, como el análisis predictivo, la visualización de los datos mediante las gráficas empleadas y la inclusión de módulos relacionados con inversiones y criptomonedas. Estas características diferencian la aplicación de soluciones más tradicionales, aportando un valor añadido basado en la automatización y el análisis inteligente de la información.

Asimismo, el proyecto ha permitido aplicar conocimientos adquiridos durante la titulación en áreas como el desarrollo web, la gestión de bases de datos y el análisis de datos, integrándolos en un entorno realista y orientado a un usuario final.

En conjunto, el trabajo realizado demuestra la viabilidad técnica y funcional de la aplicación, así como su potencial para evolucionar hacia una herramienta más completa y escalable en el ámbito de las finanzas personales.

6.2. Retos del desarrollo y aprendizajes adquiridos

Durante el desarrollo del proyecto se han presentado diversos retos que han influido tanto en la planificación como en la implementación de la aplicación.

- Uno de los principales condicionantes ha sido el **tiempo disponible** para el desarrollo, ya que ambos integrantes del proyecto compaginan este trabajo con

otras responsabilidades académicas y laborales. Esta limitación ha obligado a priorizar funcionalidades y a gestionar el tiempo de manera eficiente, poniendo en práctica habilidades de organización y planificación.

- Desde el punto de vista técnico, uno de los principales desafíos ha sido la **integración de APIs externas**. En particular, se ha trabajado con APIs que presentan limitaciones en el número de peticiones y en la frecuencia de actualización de los datos. Como consecuencia, la información financiera no siempre se obtiene en tiempo real, siendo necesario que el usuario realice actualizaciones manuales para disponer de los datos más recientes. Este aspecto ha supuesto un aprendizaje importante sobre las restricciones habituales de los servicios externos y la necesidad de diseñar soluciones que se adapten a dichas limitaciones.

- Además, el desarrollo del proyecto ha permitido adquirir experiencia en la resolución de problemas técnicos, la búsqueda de documentación especializada y la toma de decisiones sobre el diseño y la arquitectura de la aplicación.

- A nivel personal, este proceso ha contribuido al desarrollo de competencias como el trabajo en equipo, la comunicación y la capacidad de adaptación ante imprevistos.

6.3. Posibles mejoras futuras

Aunque la aplicación cumple con los objetivos definidos, existen diversas mejoras que podrían implementarse en el futuro para ampliar sus funcionalidades y mejorar la experiencia de usuario.

- Una de las principales líneas de mejora consistiría en alojar tanto la base de datos como la aplicación web en **servidores externos**. El traslado de la base de datos a un servidor de este tipo permitiría mejorar la escalabilidad, garantizando que el sistema pueda manejar un número creciente de usuarios y transacciones sin comprometer su rendimiento. Además, aumentaría la seguridad y la disponibilidad de la información, asegurando que los datos del usuario estén protegidos y accesibles en todo momento. Por su parte, el despliegue de la aplicación web en un entorno de servidor externo facilitaría su acceso desde cualquier ubicación y dispositivo, acercando la solución a un escenario de producción real y ofreciendo una experiencia de uso más flexible y profesional.

- Otra mejora relevante estaría relacionada con la **integración directa con entidades bancarias**, permitiendo la sincronización automática de movimientos financieros y reduciendo la necesidad de intervención manual por parte del usuario. La internacionalización de la aplicación también constituye una oportunidad de mejora, incorporando soporte para múltiples idiomas y monedas, así como la posibilidad de realizar conversiones de divisas de forma automática.

- Asimismo, se podría implementar un **usuario administrador** con funciones de control sobre la comunidad, como la gestión de foros, reseñas y contenido generado por los propios usuarios, contribuyendo a mantener un entorno organizado y seguro.
- Desde el punto de vista de la **experiencia de usuario**, podrían aplicarse mejoras centradas en la accesibilidad y la apariencia visual, como permitir elegir entre tema claro u oscuro o desarrollar una versión optimizada para dispositivos móviles. Igualmente, se podrían incluir funcionalidades de asistencia directa, como un chatbot que resolviera dudas de manera inmediata, ofreciendo soporte contextual y personalizado durante el uso de la aplicación. Complementariamente, sería beneficioso implementar la opción de gestionar múltiples cuentas dentro de un mismo perfil, facilitando al usuario el control de diferentes fuentes de ingresos o cuentas bancarias, y permitiendo consolidar toda su información financiera en un único espacio.
- En paralelo, el **análisis predictivo** ofrece nuevas oportunidades para enriquecer el valor de la aplicación. La incorporación de técnicas de *Machine Learning* permitiría desarrollar modelos capaces de aprender los patrones financieros específicos de cada usuario, mientras que el uso de análisis de series temporales facilitaría la detección de tendencias y variaciones estacionales en los hábitos de consumo, considerando factores como pagos recurrentes, periodos vacacionales o cambios en los ingresos. Asimismo, la aplicación de técnicas de *Clustering* permitiría agrupar gastos similares y descubrir patrones de comportamiento financiero, contribuyendo a una clasificación más precisa y a una detección de anomalías más robusta. De manera complementaria, la implementación de un sistema de alertas automáticas permitiría notificar al usuario en tiempo real cuando se detectaran anomalías relevantes o cuando las predicciones indicaran posibles problemas financieros, reforzando el carácter preventivo de la aplicación.

En conjunto, estas mejoras ofrecen un camino claro para evolucionar la aplicación hacia una solución más completa, robusta y adaptada a un mayor número de usuarios y necesidades, potenciando tanto su funcionalidad como su valor diferencial en el ámbito de las finanzas personales.

Webgrafía

- https://bootsnipp.com/
- https://getbootstrap.com/docs/5.3/getting-started/introduction/
- https://www.w3schools.com/Css/
- https://www.ionos.es/startupguide/gestion/el-activo-y-el-pasivo-en-el-balance/
- https://www.coingecko.com/en/api
- https://www.alphavantage.co/
- https://finance.yahoo.com/
- https://desarrolloweb.com/manuales/manual-jquery.html
- https://nodejs.org/
- https://expressjs.com/
- https://ejs.co/
- https://dev.mysql.com/doc/
- https://www.apachefriends.org/es/index.html
- https://developer.mozilla.org/es/docs/Web/JavaScript
- https://www.npmjs.com/
- https://github.com/
- https://www.chartjs.org/
- https://sortablejs.github.io/Sortable/
- https://es.wikipedia.org/wiki/Coeficiente_de_correlaci%C3%B3n_de_Pearson#Criterios_para_la_interpretaci%C3%B3n
- https://www.khanacademy.org/math/statistics-probability/modeling-distributions z-scores/a/z-scores-review

- https://www.khanacademy.org/math/statistics-probability
- https://en.wikipedia.org/wiki/Standard_score
- https://towardsdatascience.com/anomaly-detection-cheat-sheet-5502fc4f6bea/
- https://en.wikipedia.org/wiki/Outlier
- https://es.wikipedia.org/wiki/Regresi%C3%B3n_lineal
- https://es.wikipedia.org/wiki/M%C3%ADnimos_cuadrados
- https://en.wikipedia.org/wiki/Linear_regression

Introduction

"Do not save what is left after spending,
spend what is left after saving"
— Warren Buffett

The increasing digitalization of society has transformed the way people manage their personal finances. The widespread use of electronic payments, subscriptions, and digital financial services, among others, generates a large amount of information that, if not properly organized, can hinder medium- and long-term financial control and planning.

In this context, there is a need for tools that not only allow recording financial information but also help interpret it clearly and usefully. Personal financial management ceases to be a merely administrative process and becomes a fundamental support in everyday decision-making, such as controlling expenses, planning economic goals, or detecting inefficient consumption habits.

This Final Degree Project addresses this issue by developing a user-oriented solution, with special emphasis on simplicity, clarity of information, and support for decision-making.

Throughout this report, the development of a solution aimed at personal finance management is presented, starting from an analysis of the current context and the needs arising in the area of individual financial planning. This initial approach allows establishing the foundations of the project, defining the pursued objectives, and justifying the decisions made during its development, both at a functional and technical level.

6.4. Motivation

Nowadays, personal finance management has become a fundamental necessity for a large part of the population. The increase in the cost of living, the variety of recurring expenses, and the ease of making digital payments cause many people to lose control over their income and expenses, making economic planning and responsible financial decision-making difficult. Despite the existence of multiple financial tools, many of them are complex, unintuitive, or do not adapt to the real needs of users. This situation highlights the importance of having accessible applications that allow for clear and organized personal finance management.

The main motivation for this Final Degree Project arises from the need to develop a solution that simplifies financial management, facilitating the monitoring of income, expenses, and budgets in a visual, understandable, and efficient way.

The incorporation of elements such as gamification, user interaction, and clear visualization of financial progress helps increase user motivation and improves engagement in the continuous monitoring of their financial situation, turning financial management into a more dynamic and accessible process.

Moreover, the inclusion of analytical tools responds to the motivation to go beyond mere data recording, providing added value to the user through useful information that supports economic decision-making. In this way, the application acts not only as a financial record but also as a tool to encourage more conscious, responsible, and sustainable financial planning.

The development of Smart Finance Solutions allows for applying and consolidating knowledge acquired throughout the degree, such as web application design, backend and frontend development, database management, and the implementation of secure user authentication systems. It also offers the opportunity to explore new areas and acquire new skills, learning techniques, tools, and methodologies that enrich academic and professional training, such as using specialized libraries or implementing predictive analysis techniques, among others.

In conclusion, the motivation of this Final Degree Project is based both on the social relevance of the problem addressed and on personal and academic interest in developing a complete, practical, and user-oriented web application aimed at improving personal finance management through the use of current technologies.

6.5. Objectives

This section presents the objectives for the development of Smart Finance Solutions, aimed at providing a complete and flexible tool for personal finance management and the analysis of the user's financial information.

- Ensure data security and confidentiality through a secure registration and login system.
- Design visual and intuitive interfaces that facilitate understanding and monitoring of the user's financial situation.
- Develop budget, income, and expense management functionalities, including classification, filtering, and efficient analysis of financial information.
- Incorporate predictive analysis tools based on consumption patterns to support informed financial planning.
- Implement a dashboard that centralizes financial information, showing visually appealing summaries for the user.
- Include advanced functionalities such as portfolio and wallet management, financial goal setting, and a shared expenses management system.
- Provide real-time access to market values of cryptocurrencies and stocks, enabling users to make better-informed financial decisions.
- Encourage user engagement and consistency through gamification and social elements, such as forums, chat with friends, reviews, achievements, and notifications.
- Ensure system scalability and stability to support growth in the number of users and data volume.
- Validate the application through functional and usability testing, ensuring it meets user requirements and provides an attractive and accessible experience.

6.6. Work Methodology

For the development of Smart Finance Solutions, a methodology based on Scrum has been followed, one of the most widely used agile software development methods, which allows organizing work in iterative cycles, promoting continuous delivery of functionalities, and adapting quickly and efficiently to changes.

Agile Methodology

- Focuses on partial and frequent deliveries, where the product is developed in short iterations (sprints).
- Allows adapting requirements and functionalities as user feedback is obtained or new needs are identified.

Iterative and Incremental Development

- **Iterative:** the cycle of analysis, design, implementation, and testing is repeated multiple times. Each iteration improves and refines the previous work.
- **Incremental:** new functionalities are added gradually, forming increasingly complete versions of the system.

This methodology is ideal when requirements are not fully clear at the start or may change. It guarantees organized, flexible development focused on user needs, ensuring that the application is functional, secure, and easy to use.

During the project, task management and communication tools were used to facilitate team coordination and track development progress.

Initially, **Trello** was used to organize and prioritize tasks, and later the project migrated to **Jira**, connected to a **Discord** server to foster communication. This integration allowed automatic notifications whenever a new card was created in Jira, keeping the team informed in real time about progress, incidents, and new tasks.

Furthermore, this combination of tools allowed responsibilities to be clearly assigned, sprints tracked, and decisions and changes documented, ensuring the entire team was synchronized and that iterations met established objectives, reinforcing transparency and efficiency in the development process.

6.7. Work Plan

6.7.1. Requirements Specification

1. Requirements Analysis

   The project development began with an analysis phase aimed at identifying the general user needs and defining the functional scope of the system.

   To this end, a study of similar financial management applications was carried out, allowing the identification of common patterns, good practices, and opportunities for improvement that served as a reference for the initial design of Smart Finance Solutions.

   - **Functional requirements:** definition of the capabilities the application should offer to enable users to manage, analyze, and consult their financial information in a structured way, as well as interact with the system efficiently and personally.
   - **Non-functional requirements:** identification of criteria that ensure reliable and smooth operation of the application, safeguarding user data, providing an agile user experience, and allowing future evolution of the system without compromising its structure.

   This analysis allowed establishing a flexible foundation for designing and implementing the various system functionalities, maintaining adaptability to new needs identified during development.

2. System Design

   Once the requirements were defined, the system design was addressed to establish a clear and coherent structure that would facilitate both development and application maintenance.

   - Definition of the general architecture of the web application and the database structure, ensuring correct organization of information.
   - Design of interfaces oriented to user experience (UX/UI), prioritizing visual clarity and ease of navigation.
   - Development of flow diagrams and data models to guarantee correct system structure.

3. Implementation

   The implementation phase was carried out progressively, following an iterative and incremental approach in line with the adopted Scrum methodology.

   - Iterative development of functionalities, organizing work into tasks and sprints that allowed continuous system evolution.

- Progressive implementation of the different functionalities, prioritizing those of greatest value to the user and allowing improvements to be incorporated throughout development.

4. Testing and Validation

   Parallel to development, testing and validation tasks were performed to ensure application quality and reliability.

   - Execution of unit and integration tests to verify the correct functioning of the different system modules.
   - Usability tests with potential users to detect possible improvements in the interface and user experience.
   - Correction of detected errors and optimization of overall application performance.

5. Documentation and Presentation

   The final phase of the work plan focused on project documentation and preparation for the final presentation.

   - Preparation of the Final Degree Project report using LaTeX, a tool particularly suitable for writing extensive technical documents. The use of LaTeX allowed maintaining a clear and coherent structure, ensuring high typographical quality, and separating content from formatting, favoring scalability and maintainability of the documentation.
   - Preparation of the final presentation and supporting materials necessary for project defense. This phase included the creation of a structured and clear presentation summarizing the most relevant aspects of the work. Visual support elements were also designed to facilitate project understanding during the presentation, with special attention to clarity, visual coherence, and time management, aiming to effectively communicate the work carried out to the examination committee.

Work Tools

Various tools were used during project development to facilitate implementation, code management, and team coordination:

- **Development environment: Visual Studio Code** was used as the main editor due to its versatility, lightweight nature, wide availability of extensions, and integration with version control systems. Familiarity with the tool was also a benefit.
- **Version control: Git** and **GitHub** were used to manage the project source code, allowing tracking of changes, recovery of previous versions, and maintenance of a complete development history.
- **Management and communication:** previously mentioned tools such as **Trello**, **Jira**, and **Discord** allowed efficient work organization and smooth communication during development.
- **Testing and debugging:** browser development tools were used to detect errors and validate correct application functionality.

Figure 6.1: Visual Studio Code application logo

Figure 6.2: GitHub application logo

6.7.2. Planning

Project planning was structured following Scrum principles, organizing work into variable-length sprints to enable continuous delivery of functionalities.

Each sprint included defining partial objectives, task assignment, and time estimation, ensuring that development remained aligned with project requirements and priorities.

Although weekly sprints were initially planned, they were adapted based on workload, functionality complexity, and availability, allowing for a more realistic and flexible project management.

At the end of each sprint, Sprint Review and Retrospective meetings were held, evaluating progress, collecting user feedback, and adjusting objectives for the following sprints.

This iterative planning allowed flexible responses to requirement changes, optimized resource use, and continuously improved application quality.

6.7.3. Development

During the development phase, Smart Finance Solutions functionalities were implemented incrementally, prioritizing in early iterations those that allowed users to use the application effectively from the start, such as basic financial information management. Each iteration resulted in functional system versions that were continuously evaluated and improved.

Next, functionalities that offered market differentiation and added user value were implemented, ensuring the application provided relevant and competitive features. At the same time, adjustments and improvements were made to already implemented functionalities, refining their behavior and optimizing the user experience.

Regular testing was conducted to detect errors early and ensure stable application operation. Additionally, usability tests with potential users allowed interface adjustments and navigation experience improvements.

Iterative and incremental development ensured that each system version was functional and tested, guaranteeing that the application evolved consistently with the initial project objectives, incorporating improvements and refining requirements as development progressed.

Chapter 7

Conclusions and Future Work

7.1. General Conclusions

The development of the proposed application has made it possible to achieve the objectives set at the beginning of this Final Degree Project, resulting in a functional solution focused on personal finance management. Throughout the project, an application was designed and implemented that can record income and expenses, establish budgets, analyze the user's financial behavior, and provide tools to support economic decision-making.

One of the most relevant aspects of the work has been the incorporation of advanced functionalities, such as predictive analysis, data visualization through the employed charts, and the inclusion of modules related to investments and cryptocurrencies. These features differentiate the application from more traditional solutions, providing added value based on automation and intelligent information analysis.

Additionally, the project has allowed the application of knowledge acquired during the degree in areas such as web development, database management, and data analysis, integrating them into a realistic environment oriented towards the end user.

Overall, the work carried out demonstrates the technical and functional feasibility of the application, as well as its potential to evolve into a more complete and scalable tool in the field of personal finance.

7.2. Development Challenges and Lessons Learned

During the development of the project, various challenges arose that influenced both the planning and implementation of the application.

- One of the main constraints has been the **available time** for development, as both project members balance this work with other academic and professional responsibilities. This limitation forced prioritization of functionalities

and efficient time management, putting organizational and planning skills into practice.

- From a technical perspective, one of the main challenges was the **integration of external APIs**. In particular, APIs with limitations on the number of requests and the frequency of data updates were used. As a result, financial information is not always obtained in real time, requiring the user to perform manual updates to access the most recent data. This aspect provided an important learning experience regarding the usual constraints of external services and the need to design solutions that adapt to these limitations.

- Furthermore, the project development allowed gaining experience in problem-solving, searching for specialized documentation, and making decisions regarding application design and architecture.

- On a personal level, this process contributed to the development of competencies such as teamwork, communication, and the ability to adapt to unforeseen circumstances.

7.3. Potential Future Improvements

Although the application meets the defined objectives, there are several improvements that could be implemented in the future to expand its functionalities and enhance the user experience.

- One of the main lines of improvement would be hosting both the database and the web application on **external servers**. Moving the database to such a server would improve scalability, ensuring the system can handle a growing number of users and transactions without compromising performance. Additionally, it would increase security and data availability, ensuring that user information is protected and accessible at all times. Deploying the web application on an external server environment would also facilitate access from any location or device, bringing the solution closer to a real production scenario and providing a more flexible and professional user experience.

- Another relevant improvement would be **direct integration with banking institutions**, allowing automatic synchronization of financial transactions and reducing the need for manual user intervention. The internationalization of the application also represents an opportunity for enhancement, incorporating support for multiple languages and currencies, as well as the ability to perform automatic currency conversions.

- Additionally, a **administrator user** could be implemented with control functions over the community, such as managing forums, reviews, and user-generated content, contributing to maintaining an organized and safe environment.

- From a **user experience** perspective, improvements could focus on accessibility and visual appearance, such as allowing users to choose between light and dark themes or developing a version optimized for mobile devices. Furthermore, direct assistance functionalities could be included, such as a chatbot that resolves queries immediately, providing contextual and personalized support during application use. Complementarily, it would be beneficial to implement the option to manage multiple accounts within a single profile, allowing the user to control different income sources or bank accounts and consolidating all financial information in one place.
- In parallel, **predictive analysis** offers new opportunities to enhance the application's value. Incorporating *Machine Learning* techniques would allow developing models capable of learning each user's specific financial patterns, while using time series analysis would facilitate the detection of trends and seasonal variations in consumption habits, considering factors such as recurring payments, holiday periods, or changes in income. Additionally, applying *Clustering* techniques would enable grouping similar expenses and discovering financial behavior patterns, contributing to more accurate classification and robust anomaly detection. Complementarily, implementing an automatic alert system would notify the user in real time when relevant anomalies are detected or when predictions indicate potential financial issues, reinforcing the preventive nature of the application.

Overall, these improvements provide a clear path to evolving the application into a more complete, robust solution, adapted to a larger number of users and needs, enhancing both its functionality and its distinctive value in the field of personal finance.

Contribuciones Personales

Miguel Martínez-Almeida Nistal

Diseño de la arquitectura del sistema

La contribución de Miguel Martínez-Almeida Nistal al desarrollo de este Trabajo de Fin de Grado se ha centrado principalmente en el diseño e implementación de la arquitectura del sistema, así como en el desarrollo del backend y de los principales módulos de gestión financiera de la aplicación. Este trabajo ha sido clave para garantizar la solidez técnica, la escalabilidad y la coherencia estructural de la solución desarrollada.

Configuración del servidor y backend

Entre sus responsabilidades, destaca el diseño de la arquitectura software basada en el patrón Modelo-Vista-Controlador (MVC), complementada con una capa de negocio intermedia. Configuró el servidor con Express.js, definiendo la estructura general del proyecto, la gestión de rutas, la integración de middlewares y la coordinación de los distintos componentes del backend, incluyendo el sistema de autenticación y la gestión de sesiones de usuario.

Gestión de acceso a datos y capa de negocio

En cuanto al acceso a datos, Miguel implementó el patrón DAO mediante clases específicas para las entidades principales, abstrayendo las operaciones sobre la base de datos y normalizando los procesos CRUD. Complementariamente, desarrolló la capa de negocio, aplicando validaciones, reglas funcionales y coordinación de operaciones complejas que involucran múltiples entidades.

Desarrollo de módulos financieros

Dentro de los módulos funcionales, Miguel fue el principal responsable del desarrollo de los componentes relacionados con la gestión financiera. Implementó el sistema de registro de ingresos y gastos, incorporando la categorización de movimientos, el cálculo de balances y la gestión de secciones personalizadas como activos

y pasivos. Asimismo, diseñó y desarrolló el sistema de monederos inteligentes, permitiendo al usuario gestionar fondos con objetivos específicos mediante transferencias automáticas periódicas y cálculos dinámicos de saldos, así como el módulo de metas de ahorro con seguimiento de progreso, control de fechas límite y notificaciones asociadas a hitos relevantes.

Análisis predictivo

En el ámbito del análisis predictivo, Miguel implementó algoritmos que permiten estudiar tendencias financieras a partir de datos históricos y generar proyecciones de ingresos y gastos.

Diseño de la base de datos

Por último, fue responsable del diseño del esquema relacional de la base de datos, definiendo tablas, relaciones, claves foráneas e índices, y optimizando consultas SQL complejas para mejorar la eficiencia en operaciones que involucran múltiples uniones y agregaciones.

Daniela Valentina Valera Fuentes

Módulos sociales y colaborativos

Daniela Valentina Valera Fuentes ha centrado sus contribuciones en el desarrollo de los módulos sociales y colaborativos de la aplicación, así como en el diseño e implementación de la interfaz de usuario y la experiencia de uso. Su trabajo ha sido clave para dotar a la plataforma de un enfoque intuitivo y orientado al usuario final, complementando la robustez del backend con una experiencia visual y funcional coherente.

Entre sus principales aportaciones se encuentra el desarrollo del sistema de foro, concebido como un espacio de interacción entre los usuarios, con funcionalidades de publicación de preguntas y respuestas, categorización de contenidos y mecanismos de votación y destacados. También implementó el módulo de gestión de grupos para proyectos financieros compartidos, junto con el sistema de amistades que permite establecer y gestionar conexiones entre usuarios. Gracias a este módulo, la aplicación amplía su alcance más allá de la gestión financiera individual, facilitando el control de economías colaborativas.

Mensajería, perfil y configuración

Daniela desarrolló igualmente el sistema de mensajería y chat en tiempo real, así como el perfil de usuario y los módulos de configuración, incorporando métricas, estadísticas personales y la gestión de privacidad, notificaciones y seguridad.

Frontend y experiencia de usuario

En el frontend, se encargó de la implementación de vistas EJS reutilizables, layouts responsivos, formularios interactivos y componentes visuales como tarjetas y gráficos, asegurando una coherencia estética y mejorando la usabilidad y la experiencia de navegación. Asimismo, implementó la lógica del lado del cliente mediante JavaScript y AJAX, permitiendo interacciones dinámicas, validaciones y actualizaciones de contenido sin recarga de página, lo que generó una experiencia más fluida y moderna para el usuario final.

Servicios adicionales y análisis predictivo

Desarrolló servicios adicionales como el sistema de logros, orientado a recompensar al usuario mediante *badges* y reconocimientos por su actividad, y el servicio de dashboard, encargado de agregar y mostrar métricas relevantes de distintos módulos del sistema. Dentro del ámbito del análisis predictivo, implementó el sistema de detección de anomalías financieras, orientado a identificar comportamientos inusuales y generar alertas preventivas.

Contribuciones comunes

Ambos autores han participado conjuntamente en la elaboración, revisión y ajuste de la memoria del TFG, así como en la preparación de la presentación final, garantizando la coherencia, claridad y calidad del documento y de la exposición.

De manera adicional, colaboraron en tareas de migración y refactorización, adaptando código inicial de carácter más monolítico a la arquitectura por capas definitiva, asegurando la consistencia de los datos y el correcto funcionamiento de los distintos módulos del sistema.

Asimismo, ambos revisaron continuamente el trabajo del otro mediante comunicación constante, intercambio de opiniones y sugerencias, lo que permitió ajustar y mejorar la implementación de los distintos módulos y asegurar una integración armoniosa de todas las funcionalidades.

Apéndice A

Análisis Predictivo

Este apéndice documenta los detalles técnicos y matemáticos del módulo de análisis predictivo, incluyendo las fórmulas utilizadas, los algoritmos implementados y las limitaciones del modelo.

A.1. Predicciones financieras

El análisis predictivo implementado utiliza el método de **Regresión Lineal Simple**, que es una técnica estadística que permite modelar la relación entre una variable dependiente (y) y una variable independiente (x) mediante una línea recta.

La ecuación general es:

$$y = mx + b$$

Donde:

- y: Variable dependiente (valor predicho: ingresos, gastos, etc.)
- x: Variable independiente (tiempo: mes 1, mes 2, ...)
- m: Pendiente de la recta (tendencia)
- b: Intercepto (valor inicial)

La pendiente de la recta de regresión se calcula mediante el método de **mínimos cuadrados**, que minimiza la suma de los cuadrados de las diferencias entre los valores observados y los valores predichos. Esta técnica garantiza que la línea ajustada sea la que mejor representa la tendencia general de los datos.

Fórmula de la pendiente (m):

$$m = \frac{n \cdot \sum(xy) - \sum(x) \cdot \sum(y)}{n \cdot \sum(x^2) - (\sum(x))^2}$$

Donde:

- n: Número de puntos de datos históricos
- $\sum(xy)$: Suma de productos $x \cdot y$
- $\sum(x)$: Suma de valores x (índices temporales)
- $\sum(y)$: Suma de valores y (valores observados)
- $\sum(x^2)$: Suma de cuadrados de x

Una vez calculada la pendiente, el intercepto (b) se determina mediante:

$$b = \bar{y} - m \cdot \bar{x}$$

O utilizando el último valor real conocido:

$$b = y_{\text{último}} - m \cdot x_{\text{último}}$$

El intercepto representa el valor estimado de la variable dependiente en el punto inicial de la serie temporal (ajustado al contexto de los datos).

Proceso de Cálculo Detallado

- **Obtención de Datos Históricos**

  El sistema obtiene datos históricos de movimientos financieros del usuario desde la base de datos, realizando las siguientes operaciones:

  1. **Datos históricos extensos (12 meses):** Se obtienen los últimos 12 meses de datos para calcular promedios representativos de ingresos y gastos mensuales.
  2. **Datos históricos recientes (6 meses):** Se obtienen los últimos 6 meses de datos agregados mensualmente para realizar el análisis de regresión lineal. Este período proporciona un balance óptimo entre representatividad y relevancia temporal.
  3. **Agregación mensual:** Los datos se agrupan por mes, sumando los ingresos y gastos de cada mes para obtener valores mensuales totales.
  4. **Cálculo de balance:** Para cada mes, se calcula el balance (ingresos - gastos), que representa el ahorro o desahorro mensual.

- **Cálculo de Promedios**

  Se calculan los promedios aritméticos simples de ingresos y gastos basados en los últimos 12 meses:

$$\bar{I} = \frac{1}{n} \sum_{i=1}^{n} I_i$$

$$\bar{G} = \frac{1}{n} \sum_{i=1}^{n} G_i$$

Donde I_i y G_i representan los ingresos y gastos del mes i, respectivamente, y n es el número de meses disponibles (máximo 12).

Estos promedios proporcionan una línea base conservadora para predicciones cuando no hay suficientes datos para realizar regresión lineal.

- **Cálculo de la Regresión Lineal**

  Para series temporales con al menos 3 meses de datos, se realiza un análisis de regresión lineal:

  - **Asignación de índices temporales:** Se asignan valores numéricos secuenciales a cada mes (1, 2, 3, ..., n).
  - **Cálculo de sumatorias:** Se calculan las siguientes sumatorias necesarias para la regresión:
    - Σx: Suma de índices temporales
    - Σy: Suma de valores (ingresos, gastos o balance)
    - Σxy: Suma de productos $x \cdot y$
    - Σx^2: Suma de cuadrados de índices
  - **Cálculo de pendiente:** Se aplica la fórmula de mínimos cuadrados para cada variable (ingresos, gastos, balance).
  - **Cálculo de intercepto:** Se calcula el intercepto utilizando la media de valores y la media de índices.
  - **Interpretación de tendencias:**
    - **Pendiente positiva:** Tendencia creciente (ingresos/gastos aumentando)
    - **Pendiente negativa:** Tendencia decreciente (ingresos/gastos disminuyendo)
    - **Pendiente cercana a cero:** Tendencia estable

1. Predicción Financiera de Ingresos y Gastos

   El sistema genera predicciones para los próximos 6 meses mediante el siguiente proceso:

   - Selección del Método de Predicción
     - **Regresión Lineal:** Se utiliza cuando hay 3 o más meses de datos históricos disponibles. Este método captura las tendencias y permite proyecciones más precisas cuando existe una tendencia clara.
     - **Promedio Simple:** Se utiliza cuando hay menos de 3 meses de datos. Este método conservador evita predicciones exageradas basadas en datos insuficientes.
   - Cálculo de Predicciones Mensuales

     Para cada mes futuro ($i = 1, 2, ..., 6$):

     a) **Predicción de ingresos:** - Si se usa regresión lineal: $I_{predicho} = b_I + m_I \cdot x_i$ - Si se usa promedio: $I_{predicho} = \bar{I}$

b) **Predicción de gastos:** - Si se usa regresión lineal: $G_{predicho} = b_G + m_G \cdot x_i$ - Si se usa promedio: $G_{predicho} = \bar{G}$

c) **Restricción de valores negativos:** Se aplica la función $max(0, valor)$ para evitar predicciones negativas, ya que los ingresos y gastos son siempre valores positivos o cero.

d) **Cálculo de balance predicho:** $B_{predicho} = I_{predicho} - G_{predicho}$

e) **Cálculo de ahorro acumulado:** Se suma el balance predicho al ahorro acumulado hasta ese momento:

$$Ahorro_{acumulado} = Saldo_{actual} + \sum_{j=1}^{i} B_{predicho,j}$$

2. Seguimiento de Objetivos

El sistema calcula predicciones específicas para cada meta de ahorro establecida por el usuario. Para cada meta, se obtiene la cantidad objetivo que se desea alcanzar, cantidad ya ahorrada en el monedero asociado (o en la cuenta principal si no hay monedero asociado), fecha objetivo para alcanzar la meta (si existe) y monedero específico donde se está ahorrando para esa meta (opcional).

Cálculo de Tasa de Ahorro Mensual

La tasa de ahorro mensual se calcula como la diferencia entre el promedio de ingresos y el promedio de gastos:

$$T_{ahorro} = \bar{I} - \bar{G}$$

Esta tasa representa la cantidad promedio que el usuario puede ahorrar mensualmente según su historial financiero.

Predicción de Meses para Alcanzar la Meta

Si la tasa de ahorro es positiva (los ingresos superan a los gastos en promedio), se calcula cuántos meses se necesitarán para alcanzar la meta:

$$Meses = \lceil \frac{Objetivo - Saldo_{actual}}{T_{ahorro}} \rceil$$

Donde $\lceil \rceil$ representa la función techo (redondeo hacia arriba).

Si la tasa de ahorro es negativa o cero, se considera que la meta no es alcanzable con las tendencias actuales y no se proporciona una estimación de fecha.

Fecha Estimada de Cumplimiento

La fecha estimada se calcula sumando los meses predichos a la fecha actual:

$$Fecha_{estimada} = Fecha_{actual} + Meses_{predichos}$$

Evaluación de Factibilidad

El sistema evalúa si la meta es factible comparando la fecha estimada con la fecha límite (si existe):

- Meta factible: $Fecha_{estimada} \leq Fecha_{limite}$ (o si no hay fecha límite)
- Meta no factible: $Fecha_{estimada} > Fecha_{limite}$

Esta evaluación permite al usuario identificar metas que pueden requerir ajustes en sus hábitos financieros o en los objetivos establecidos.

Progreso Actual

Se calcula el porcentaje de progreso actual hacia la meta:

$$Progreso(\%) = \frac{Saldo_{actual}}{Objetivo} \times 100$$

A.2. Detección de Anomalías

La detección de anomalías implementada se basa en el método estadístico de **Z-Score** (también conocido como puntuación estándar o desviación estándar normalizada). Este método es ampliamente utilizado en estadística y análisis de datos para identificar valores atípicos en un conjunto de datos.

El Z-Score cuantifica cuántas desviaciones estándar se aleja un valor individual de la media de la distribución. La fórmula matemática es la siguiente:

$$Z = \frac{x - \mu}{\sigma}$$

Donde:

- Z: Z-Score (puntuación estándar)
- x: Valor individual a evaluar (gasto)
- μ: Media aritmética del conjunto de datos
- σ: Desviación estándar del conjunto de datos

Interpretación del Z-Score

El valor del Z-Score permite clasificar la desviación de un gasto respecto a la media:

- |Z| <1: Valores normales, dentro de una desviación estándar de la media
- 1 <= |Z| <2: Valores ligeramente inusuales
- 2 <= |Z| <3: Valores inusuales (anomalías moderadas)
- |Z| >= 3: Valores muy inusuales (anomalías significativas)

Un Z-Score positivo indica que el gasto está por encima de la media, mientras que un Z-Score negativo indica que está por debajo de la media.

Para determinar qué gastos se consideran anomalías, se utiliza un **umbral configurable** por el usuario. Este umbral define el valor mínimo del |Z-Score| para que un gasto sea clasificado como anómalo.

- **Umbral bajo (1.5-1.8):** Alta sensibilidad, detecta más anomalías pero puede incluir falsos positivos
- **Umbral medio (2.0-2.5):** Balance óptimo entre precisión y detección (recomendado)
- **Umbral alto (2.5-3.0):** Baja sensibilidad, detecta solo anomalías muy significativas

La aplicación permite ajustar este umbral mediante un control deslizante en el rango de 1.5 a 3.0, con un valor por defecto de 2.0, que representa un equilibrio óptimo según la literatura estadística.

Proceso de Análisis

El proceso de detección de anomalías sigue los siguientes pasos:

1. Obtención de Datos Históricos

   El sistema recupera todos los gastos registrados del usuario desde la tabla 'movimientos' de la base de datos. Se valida que exista un historial mínimo de 30 días para garantizar la fiabilidad estadística del análisis.

2. Cálculo de Estadísticas Descriptivas

   Para todos los gastos del usuario, se calculan las siguientes métricas estadísticas:

   - **Media aritmética (μ):** Promedio de todos los gastos
   - **Mediana:** Valor que divide la distribución en dos mitades iguales
   - **Desviación estándar (σ):** Medida de dispersión de los datos
   - **Valores mínimo y máximo:** Extremos de la distribución
   - **Percentiles 25 y 75:** Cuartiles de la distribución

3. Cálculo del Z-Score

   Para cada gasto individual, se calcula su Z-Score utilizando la fórmula mencionada anteriormente, comparándolo con la media y desviación estándar globales.

4. Identificación de Anomalías

   Se identifican como anomalías aquellos gastos cuyo |Z-Score| supera el umbral configurado por el usuario. Se clasifican en dos tipos:

- **Anomalías altas:** Gastos significativamente por encima de la media (Z-Score positivo y alto)
- **Anomalías bajas:** Gastos significativamente por debajo de la media (Z-Score negativo y alto en valor absoluto)

5. Análisis por Categoría

   Adicionalmente, se realiza un análisis independiente agrupando los gastos por categoría. Para cada categoría con al menos 3 gastos registrados:

   - Se calculan estadísticas independientes (media y desviación estándar específicas de esa categoría)
   - Se calcula el Z-Score de cada gasto comparándolo con la media de su propia categoría
   - Se detectan anomalías dentro del contexto específico de cada categoría

   Este análisis complementario permite identificar gastos que, aunque no son anómalos en el contexto general, sí lo son dentro de su categoría específica (por ejemplo, un gasto muy alto en transporte cuando normalmente los gastos en esa categoría son bajos).

Validaciones y Requisitos

- **Historial mínimo:** Se requiere al menos 30 días de historial para realizar el análisis general. Si no se cumple este requisito, se muestra un mensaje informativo al usuario.
- **Mínimo de gastos por categoría:** Para el análisis por categoría, se requiere al menos 3 gastos en cada categoría. Las categorías con menos gastos no se analizan, ya que no se puede calcular una desviación estándar fiable con menos de 3 puntos de datos.
- **Desviación estándar no nula:** Si todos los gastos de una categoría tienen el mismo valor (desviación estándar = 0), no se puede realizar el análisis estadístico y esa categoría se omite.

A.3. Limitaciones del Modelo

A.3.1. Limitaciones de la Regresión Lineal

- **Asunción de linealidad:** La regresión lineal asume una tendencia lineal, que puede no reflejar cambios abruptos, ciclos estacionales o eventos extraordinarios.
- **Extrapolación temporal:** Las predicciones se realizan mediante extrapolación, y la precisión disminuye a medida que se proyecta más hacia el futuro. El sistema limita las predicciones a 6 meses para mantener un nivel razonable de precisión.

- **Simplificación de metas:** El cálculo de cumplimiento de metas asume una tasa de ahorro constante, que puede no reflejar variaciones reales en la capacidad de ahorro.
- **Dependencia del historial:** La precisión del análisis depende de la cantidad y calidad de los datos históricos disponibles. Usuarios con historiales cortos pueden obtener resultados menos precisos.

A.3.2. Limitaciones de la Detección de Anomalías

- **Variabilidad estacional:** El método no considera automáticamente variaciones estacionales (por ejemplo, gastos navideños) que podrían ser normales en ciertos períodos del año pero anómalos en otros.
- **Cambios de comportamiento:** Si el usuario modifica significativamente sus hábitos de gasto, el método puede identificar como anomalías gastos que representan una nueva normalidad.
- **Categorías con pocos datos:** Las categorías con menos de 3 gastos no pueden ser analizadas mediante el método estadístico por categoría.
- **Gastos periódicos grandes:** Gastos periódicos legítimos pero de alto valor (como seguros anuales) pueden ser marcados como anomalías si no se tienen suficientes datos históricos para reconocer el patrón.