Sistema de gestión de finanzas personales: Diseño, desarrollo e implementación

Juan Diego Lizarazo, Sergio Alejandro López
{jdiegolizarazo, sergioalejandrolopez}@ucundinamarca.edu.co

***Abstract*—Este informe documenta el proceso de diseño, desarrollo e implementación de un sistema de gestión de finanzas personales. El sistema proporciona una plataforma para el registro y seguimiento de transacciones financieras, incluyendo ingresos, egresos, inversiones y metas de ahorro. La aplicación se construyó utilizando Laravel con el panel administrativo Filament, PostgreSQL como base de datos y Redis para caché y sesiones. La arquitectura implementada se centró en la optimización del rendimiento y la escalabilidad mediante la contenerización con Docker, permitiendo un despliegue eficiente en entornos de producción. El documento detalla los requerimientos funcionales y no funcionales que guiaron el desarrollo, así como las decisiones técnicas adoptadas durante el proceso de implementación.**

***Index Terms*—Sistema financiero personal, Laravel, Filament, PostgreSQL, Redis, Docker, desarrollo web, aplicación contenerizada.**

Contents

I. Introducción

La gestión eficiente de finanzas personales representa un proceso fundamental para la planificación económica individual. La disponibilidad de herramientas digitales para administrar recursos financieros personales contribuye significativamente a la toma de decisiones informadas en materia económica. El presente trabajo documenta el desarrollo de un sistema de gestión de finanzas personales implementado como aplicación web con tecnologías actuales.

La planificación financiera personal requiere un seguimiento detallado de transacciones económicas para establecer patrones de consumo e inversión. Los sistemas tradicionales de registro manual presentan limitaciones en términos de accesibilidad, análisis y visualización de datos. El desarrollo de una plataforma digital especializada resuelve estas limitaciones mediante un diseño centrado en la usabilidad y un modelo de datos estructurado.

El sistema desarrollado implementa una arquitectura basada en Laravel como framework de desarrollo, Filament para la construcción del panel administrativo, PostgreSQL como sistema gestor de base de datos, y Redis para la optimización de caché y manejo de sesiones. La implementación se realizó adoptando el patrón Modelo-Vista-Controlador (MVC) y se desplegó mediante contenedores Docker para garantizar la portabilidad del entorno de ejecución.

La aplicación permite el registro y seguimiento de ingresos, egresos, inversiones y metas de ahorro, facilitando la categorización de transacciones financieras para un análisis posterior. El sistema incorpora módulos para la gestión de usuarios con diferentes niveles de acceso y genera visualizaciones básicas del estado financiero del usuario.

El presente documento detalla los aspectos técnicos del desarrollo, iniciando con la definición de objetivos para cada fase del proyecto, seguido por el alcance del sistema y los requerimientos funcionales, no funcionales, de hardware y de software que guiaron la implementación.

II. Objetivos

A. Objetivo general

Diseñar, desarrollar e implementar un sistema de gestión de finanzas personales basado en tecnologías web modernas que permita el registro, seguimiento y análisis de transacciones financieras individuales.

B. Objetivos específicos

- **Planificar el desarrollo del sistema** mediante la definición de alcances, requerimientos, estructuras de desglose de trabajo

Proyecto realizado como parte del CADI de Gestión de Proyectos centrada en Datos del VIII semestre de Ingeniería de Sistemas y Computación de la Universidad de Cundinamarca. Mayo de 2025.

- **Diseñar la arquitectura del sistema** mediante la creación de un modelo de datos normalizado para transacciones financieras, la implementación del patrón MVC en Laravel, y el diseño de una interfaz administrativa utilizando módulos funcionales independientes.
- **Desarrollar los componentes del sistema** siguiendo las especificaciones técnicas establecidas, incluyendo el acceso a datos, la implementación de la lógica de negocio según los requerimientos funcionales, la creación de interfaces de usuario responsivas.
- **Desplegar el sistema de gestión** mediante una estrategia de canalización para definir, construir, optimizar y ejecutar un entorno aislado con contenedores Docker, subredes y volúmenes de persistencia.

III. Alcance

El sistema de gestión de finanzas personales abarca un conjunto de funcionalidades centradas en el seguimiento de operaciones financieras individuales. La aplicación implementa un módulo de autenticación y gestión de usuarios que permite el registro de nuevos participantes, autenticación mediante correo electrónico y contraseña, y un sistema de recuperación de credenciales. Este módulo incluye la gestión de roles y permisos para controlar el acceso a las distintas funcionalidades del sistema.

El sistema incorpora un módulo de ingresos para registrar las entradas económicas de los usuarios. Este componente permite la categorización de las fuentes de ingreso, la especificación de periodicidad y montos, y proporciona una visualización tabular de los registros históricos. Las operaciones CRUD (Crear, Leer, Actualizar, Eliminar) están completamente implementadas para este módulo.

Para el control de gastos, el sistema implementa un módulo de egresos que permite registrar las salidas económicas con su correspondiente categorización. Este componente distingue entre costos fijos y variables, facilitando un análisis posterior del patrón de gastos. Al igual que el módulo de ingresos, proporciona operaciones CRUD completas y visualizaciones tabulares de datos históricos.

El sistema integra un módulo de inversiones que permite registrar distintos tipos de inversiones realizadas. Este componente incluye campos para especificar rendimientos esperados y plazos, facilitando un seguimiento básico del estado de cada inversión. Las operaciones CRUD están disponibles para la gestión completa de estos registros.

Para la planificación financiera, el sistema cuenta con un módulo de metas de ahorro que permite definir objetivos económicos con montos específicos y fechas límite. Este componente calcula y muestra el progreso porcentual hacia cada meta establecida, basándose en los ahorros registrados. El módulo implementa operaciones CRUD completas para la gestión de estos objetivos.

La interfaz del sistema se implementa mediante un panel administrativo desarrollado con Filament, proporcionando componentes reutilizables para formularios, tablas y widgets. El panel incluye validaciones en el lado del cliente y servidor para garantizar la integridad de los datos ingresados. La interfaz es responsiva, adaptándose a diferentes tamaños de pantalla para facilitar el acceso desde dispositivos móviles.

El sistema no incluye funcionalidades avanzadas como programación automática de egresos basada en patrones de costos, integración directa con servicios bancarios externos, sistemas de notificaciones programadas, o aplicaciones móviles nativas. Estas capacidades quedan fuera del alcance inicial del proyecto.

IV. Requerimientos

A. Requerimientos funcionales

- Registro y autenticación de usuarios mediante correo electrónico y contraseña, con recuperación de credenciales y validación de cuentas.
- Control de acceso basado en roles con permisos específicos para cada tipo de usuario, incluyendo roles predeterminados de administrador y usuario regular.
- Registro de ingresos económicos con atributos de monto, fecha, fuente y descripción, con soporte para categorización y recurrencia.
- Control de gastos mediante registro de egresos categorizados como costos fijos o variables, incluyendo campos para fecha, monto y descripción.
- Módulo de inversiones para registrar colocaciones económicas con atributos de monto, tipo, rendimiento esperado y fecha.
- Gestión de metas de ahorro con objetivos monetarios, fechas límite y cálculo automático del progreso.
- Panel principal con resumen visual del estado financiero mediante widgets, mostrando balances actuales, tendencias y progreso hacia metas.
- Consultas históricas de transacciones con filtrado por fecha, categoría y monto, visualizables en formato tabular.
- Validaciones en formularios para garantizar la integridad y consistencia de la información ingresada.

B. Requerimientos no funcionales

- **Rendimiento:** Respuesta a operaciones CRUD en menos de 2 segundos y tiempo de respuesta de interfaz inferior a 1 segundo.
- **Escalabilidad:** Soporte para un mínimo de 100 usuarios concurrentes sin degradación significativa del rendimiento.
- **Seguridad:** Cifrado de datos sensibles, hashing de contraseñas con bcrypt y comunicaciones protegidas mediante HTTPS.
- **Disponibilidad:** Tiempo de actividad superior al 99.9% con monitoreo continuo y mecanismos de respaldo y recuperación.
- **Compatibilidad:** Funcionamiento en navegadores web modernos como Chrome, Firefox, Safari y Edge con diseño responsivo.

- **Protección:** Validación de entradas para prevenir inyecciones SQL, XSS y otros vectores de ataque comunes.
- **Usabilidad:** Mensajes de error informativos y claros sin exponer detalles técnicos internos.
- **Respaldo:** Copias de seguridad automáticas de la base de datos con políticas de retención definidas.

C. Requerimientos de hardware

- **Servidor de producción:**
  - Mínimo 2 núcleos de CPU y 4GB de RAM.
  - Almacenamiento mínimo de 20GB para aplicación y base de datos.
  - Conexión a internet con ancho de banda mínimo de 10Mbps.
  - Latencia de red inferior a 100ms.
- **Dispositivos cliente:**
  - 2GB de RAM y procesador de doble núcleo como mínimo.
  - Resolución de pantalla mínima de 1280x720 píxeles.
  - Conexión a internet de al menos 1Mbps.

D. Requerimientos de software

- Docker Engine instalado[1].
- PHP 8.4 o superior con extensiones esenciales para desarrollo web: PostgreSQL, Redis, JSON, XML, Mbstring, Fileinfo, OpenSSL, PDO, Tokenizer, BCMath e intl.
- PostgreSQL 14 o superior como sistema gestor de base de datos.
- Redis 6.2 o superior para gestión de caché y sesiones.
- NGINX como servidor web con soporte para conexiones SSL.
- Docker y Docker Compose para construcción y gestión de entornos contenerizados.
- Node.js 18 o versión LTS superior con NPM para procesamiento de activos frontend.
- Composer para la gestión de dependencias PHP.

References

[1] Laravel, “Laravel - The PHP Framework For Web Artisans,” [Online]. Available: https://laravel.com/. [Accessed: 24-May-2025].
[2] Filament, “Filament - The Laravel Admin Panel Builder,” [Online]. Available: https://filamentphp.com/. [Accessed: 24-May-2025].
[3] Docker Inc., “Docker: Accelerated, Containerized Application Development,” [Online]. Available: https://www.docker.com/. [Accessed: 24-May-2025].
[4] PostgreSQL Global Development Group, “PostgreSQL: The World’s Most Advanced Open Source Relational Database,” [Online]. Available: https://www.postgresql.org/. [Accessed: 24-May-2025].
[5] Redis Ltd., “Redis: The open source, in-memory data store used by millions of developers,” [Online]. Available: https://redis.io/. [Accessed: 24-May-2025].

[1]En sistemas Linux se requiere únicamente Docker Engine, mientras que en Windows y macOS es necesario instalar Docker Desktop.

Juan Diego Lizarazo Desarrollador de software con experiencia en aplicaciones web y sistemas de gestión de datos. Especialista en tecnologías PHP, con enfoque en el ecosistema Laravel y bases de datos PostgreSQL. Su trabajo se centra en el desarrollo de soluciones escalables mediante arquitecturas basadas en contenedores y despliegue en entornos cloud.

Sergio Alejandro López Ingeniero de software especializado en desarrollo web y diseño de interfaces de usuario. Con experiencia en frameworks frontend modernos y tecnologías de backend. Su trabajo se enfoca en la optimización del rendimiento de aplicaciones web y la implementación de patrones de diseño para mejorar la experiencia del usuario.