**¡Vámonos Ahorrando!**

CFGS DAW

CURSO 2024/2025

Nombres:

* Pablo Castro Morato  
  - Gonzalo Campos Vicente
* Hugo Blanco Cuadrado

COLEGIO CALASANZ SALAMANCA

**Índice**

[1. Introducción 2](#_Toc1692939057)

[2. Descripción de la aplicación. 3](#_Toc1804015182)

[3. Plan de empresa 3](#_Toc2082966567)

[4. Tecnologías escogidas y justificación 3](#_Toc125002693)

[5. Diseño de la aplicación 3](#_Toc1708989672)

[5.1. Diagramas y definición de casos de uso 3](#_Toc392223112)

[5.2. Diagramas de clase 3](#_Toc1677372707)

[5.3. Modelo entidad relación 3](#_Toc1288488794)

[6. Arquitectura de la aplicación 3](#_Toc743674766)

[6.1. Estructura del proyecto 4](#_Toc868295700)

[6.2. Librerías externas utilizadas 4](#_Toc1371776852)

[7. Manual de despliegue 4](#_Toc1064151562)

# Introducción

Aplicación de Ahorro en la cual intentamos que sea una experiencia interactiva donde los usuarios pueden registrar y visualizar transacciones diarias en un calendario, y tener el acceso de diferentes secciones en la barra de navegación, incluyendo estadísticas de ahorro, opciones para configurar y modificar categorías de gasto, y la posibilidad de establecer objetivos financieros semanales o mensuales.   
Creemos que, en la sociedad actual, el ahorro y la buena administración de los recursos económicos son cada vez mas importantes, pero a muchas personas se encuentran con dificultad para hacer el seguimiento de sus gastos e ingresos. Con el objetico de gestionar esta necesidad nuestra aplicación ¡Vámonos,ahorrando! , intentamos ofrecer una plataforma donde los usuarios podrán registra sus transacciones diarias de forma atractiva.   
Esta aplicación no solo busca ayudar a los usuarios a conocer su situación financiera, sino también a detectar los malos gatos.

# Descripción de la aplicación.

Tiene como objetivo principal ayudar a los usuarios a gestionar sus finanzas de manera sencilla y visual, para registrar y analizar sus ingresos y gastos. A través de una interfaz intuitiva, el usuario puede acceder  
 Calendario interactivo: Que facilita el registro diario de transacciones y permite visualizar a los patrones de ingresos con un símbolo verde y gastos con un símbolo rojo a tiempo real. Así permitiendo ver de una vez como se distribuye los ingresos y los gastos a lo largo del mes. Al hacer clic en cualquier día, el usuario puede ver en detalle de las todas las transacciones registrada para ese día. Llevando así un control diario, permitiendo detectar picos de gasto o periodos de ahorro de una forma visual.

Personalización y Modificación de Categorías: Ofrece una sección de estadísticas en la que los usuarios pueden ver un gráfico detallado sobre sus hábitos de ahorro y gasto, pudiendo visualizarlo mensualmente.

Además, permite al usuario establecer metas financieras específicas, como ahorrar un poco de su dinero a final de mes o reducir sus gastos en una categoría particular.

Administración de Usuarios: La aplicación estará diseñada para tener un usuario cada persona, lo cual permite llevar un registro individual para cada miembro. Es ideal para personas que deseen gestionar sus financias de forma diferenciada.

* *Registro de Usuarios*: Cada uno puede registrarse en la aplicación y gestionar sus finanzas de manera independiente. Se le asignará un perfil único en el que puede almacenar y revisar su historial financiero.
* *Roles y Permisos*: Permitirá asignar distintos roles, como “Administrador” y “Usuario”. El administrador puede gestionar la información de otros usuarios, mientras que los usuarios tienen acceso únicamente a sus propias transacciones y estadísticas.

CRUD : (Crear, Leer, Actualizar y Eliminar)  
Permitiendo a los usuarios un manejo completo y flexible de su información financiera.

* CRUD de transacciones: Los usuarios puede añadir nuevas transacciones al calendario, visualizar las transacciones existentes, modificar detalles de transacciones pasadas, y eliminar aquellas que consideren innecesarias.

Funcionalidad necesaria para mantener el registro actualizado y ajustado a la realidad de cada usuario.

* CRUD de categorías: Cada usuario tiene la opción de crear, editar o eliminar categorías de ingresos y gastos según sus necesidades.

Permitirá adaptar la aplicación a su estilo de vida y patrones de gastos específicos.

* CRUD de usuarios: En el caso que haya administradores, podrán gestionar la información de otros usuarios, incluyendo la posibilidad de crear, modificar o eliminar perfiles de usuario.
* CRUD de recordatorios: El usuario puede crear todos los recordatorios que vea conveniente, poder borrarlos en cualquier momento.

# Plan de empresa

Objetivo: Convertirse en una app de ahorro visual, proporcionando una herramienta sencilla para alcanzar una vida financiera saludable y equilibrada.

Análisis del Mercado: Publico: Jóvenes, adultos y familias que desean mejorar su salud financiera de forma accesible y sin complicaciones.

Esta aplicación no es la primera, pero se diferencia por su enfoque en el ahorro visual interactivo y una navegación simplificada.

Tendencias de Mercado: El interés en la educación financiera y la popularidad de las aplicaciones de finanzas personales están en crecimiento. Cada vez más personas buscan herramientas para ahorrar y gestionar sus finanzas en un formato sencillo.

Modelo de negocio: La versión gratuita permite el registro básico de transacciones y visualización en el calendario.

La versión premium incluye funcionalidades avanzadas, como recomendaciones de ahorro y estadísticas detalladas.

Publicidad: Anuncios en la versión gratuita de la app.

*Marketing*

Redes Sociales: Instagram, TikTok dirigidas a usuarios jóvenes interesados en finanzas personales.

Colaboraciones: Con influencers de finanzas y blogs especializados para aumentar la credibilidad y atraer.

# Tecnologías escogidas y justificación

Hemos seleccionado un conjunto de tecnologías que nos permiten crear una aplicación web sólida, interactiva y fácil de mantener. Cada tecnología ha sido elegida por sus ventajas específicas para satisfacer los requerimientos del proyecto.

*Front-end:*

* Vue.js: Hemos optado por Vue para el desarrollo de la interfaz de usuario debido a su simplicidad, flexibilidad y rendimiento. Vue es un framework progresivo de JavaScript que permite crear componentes reutilizables, lo que facilita la gestión de una interfaz interactiva como el calendario de transacciones. Su enfoque en la reactividad facilita la visualización en tiempo real de los cambios en los ingresos y gastos.
* HTML, CSS y JavaScript: Estos lenguajes fundamentales permiten diseñar una interfaz visualmente atractiva y completamente personalizable. HTML estructura el contenido, CSS proporciona estilo y diseño, y JavaScript se encarga de la funcionalidad interactiva, asegurando una experiencia de usuario fluida.

*Back-end:*

* Java: Para el desarrollo del servidor y la lógica de negocio, utilizamos Java, un lenguaje de programación robusto, seguro y ampliamente usado en aplicaciones empresariales. Java facilita la gestión de operaciones complejas como el sistema CRUD para transacciones, categorías, usuarios y recordatorios, permite una conexión segura y estable con la base de datos.

*Base de datos:*

* MySQL: La elección de MySQL como sistema de gestión de base de datos se debe a su fiabilidad, facilidad de integración con Java y su capacidad para manejar grandes volúmenes de datos de manera eficiente. MySQL nos permite almacenar y organizar la información de usuarios, transacciones, recordatorios, categorias y estadísticas de manera estructurada, facilitando su consulta y análisis.

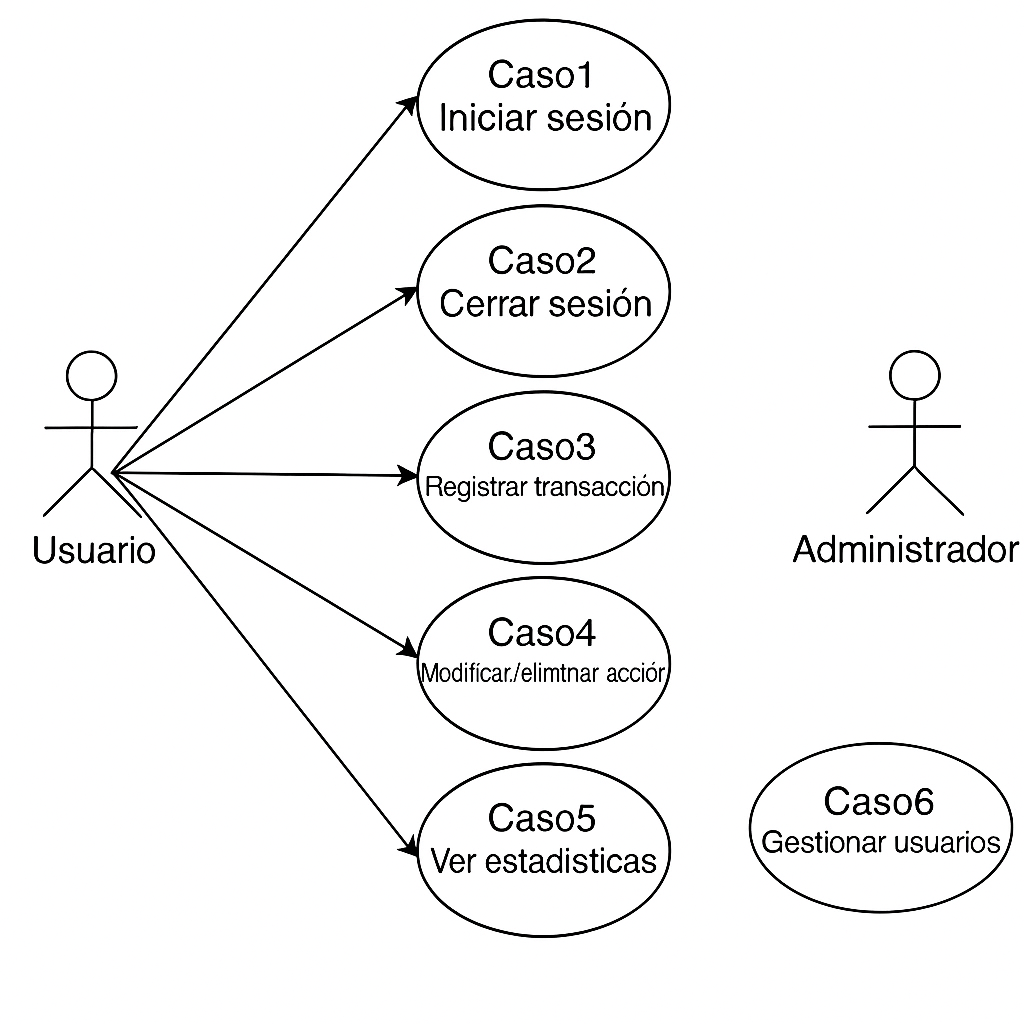
Este conjunto de tecnologías asegura una aplicación escalable y de alto rendimiento. Vue.js permite una experiencia de usuario reactiva y dinámica en el front-end, mientras que Java y MySQL en el back-end proporcionan una infraestructura sólida para la gestión de datos y la ejecución de la lógica de negocio. La combinación de estas herramientas crea una base confiable para desarrollar funcionalidades avanzadas como estadísticas de ahorro, recomendaciones y gestión de múltiples usuarios de manera efectiva.

# Diseño de la aplicación

En este apartado, describiremos cómo se ha diseñado la aplicación, incluyendo los diagramas necesarios para representar los casos de uso, las clases y el modelo de datos.

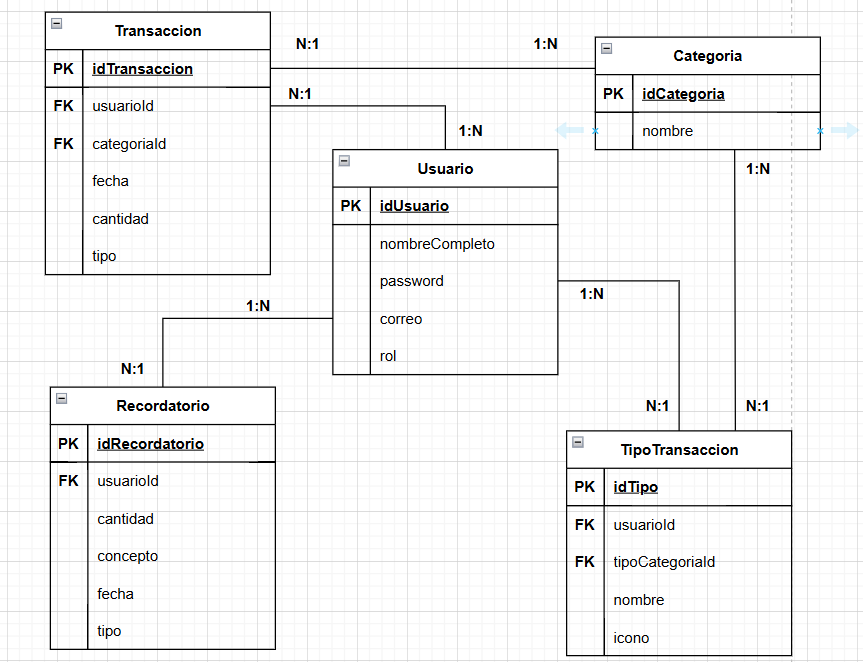
## Diagramas y definición de casos de uso

Los casos de uso representan las principales interacciones entre los usuarios y la aplicación, abordando las funcionalidades esenciales.

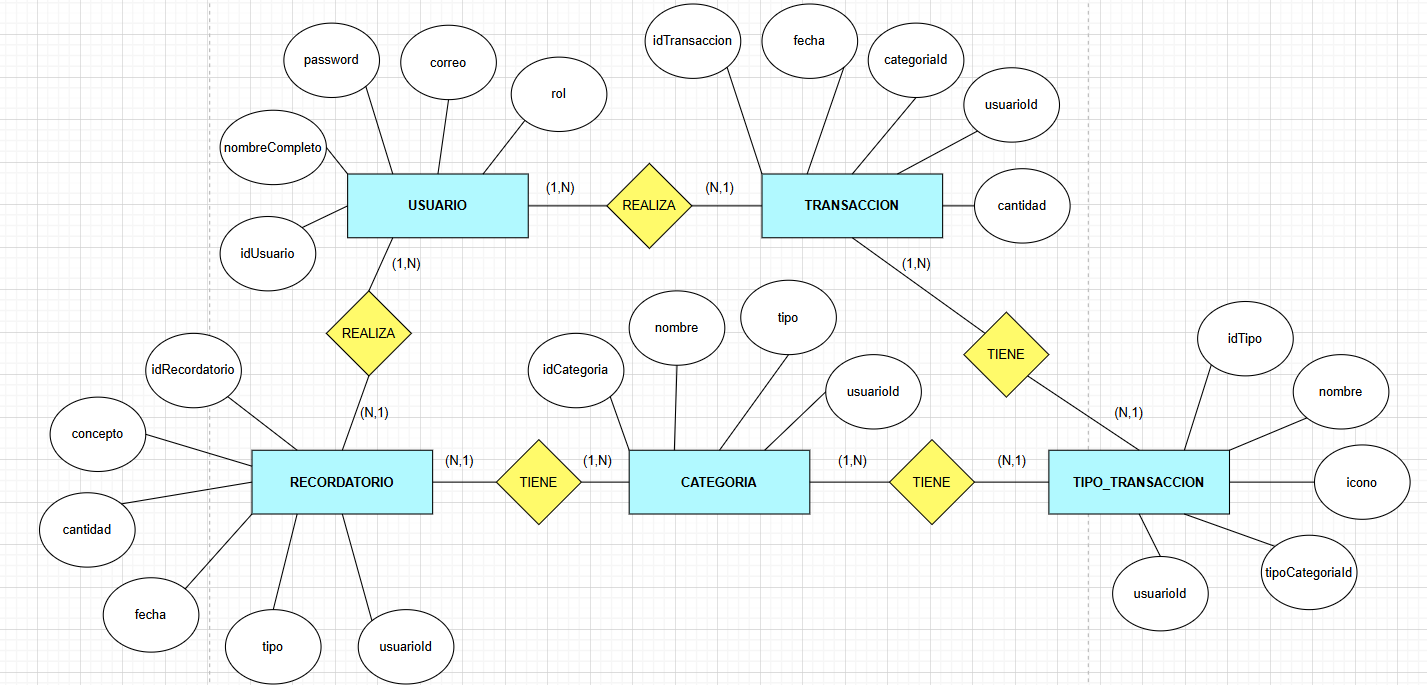


## Diagramas de clase

Un diagrama de clase muestra las entidades principales de la aplicación, sus atributos y relaciones.



## Modelo entidad relación



# Arquitectura de la aplicación

En este apartado, se describe cómo se organiza el proyecto y las tecnologías empleadas.

## Estructura del proyecto

La arquitectura seguirá el patrón **MVC (Modelo-Vista-Controlador)** para separar las responsabilidades de la aplicación:

1. **Modelo**:

* Gestiona la lógica de negocio y la interacción con la base de datos.
* Ejemplo: Clases como Usuario, Transacción, Categoría.

1. **Vista**:

* Gestiona la interfaz gráfica que el usuario interactúa.
* Componentes Vue.js para cada sección: calendario, estadísticas, configuración de categorías.

1. **Controlador**:

* Intermediario entre la vista y el modelo.
* Ejemplo: Controladores en Java que manejan las solicitudes del cliente y actualizan la base de datos.

## Librerías externas utilizadas

* **Front-end:**
  + **Vue.js**: Framework progresivo para construir la interfaz de usuario de forma dinámica y modular.
  + **SweetAlert2**: Librería de alertas personalizadas para mostrar mensajes visuales y feedback al usuario.
  + **Chart.js**: Librería para representar datos estadísticos mediante gráficos visuales e interactivos.
  + **Fetch API** *(en lugar de Axios)*: Se utiliza para realizar peticiones HTTP al backend de manera nativa, sin depender de librerías externas.
* **Back-end:**
  + **Spring Boot**: Framework para simplificar el desarrollo de aplicaciones Java basadas en Spring.
  + **Spring Data JPA**: Abstracción de acceso a datos que facilita el uso de repositorios y operaciones CRUD.
  + **Hibernate**: Implementación de JPA para el mapeo Objeto-Relacional y gestión de entidades.
  + **Springdoc OpenAPI (Swagger)**: Generación automática de documentación interactiva de la API REST.
  + **JUnit**: Librería de pruebas unitarias para verificar la lógica del backend.
* **Base de datos:**
  + **MySQL Connector/J**: Driver JDBC oficial para conectar la aplicación Java con la base de datos MySQL.
* **Otras herramientas:**
  + **Maven**: Herramienta de gestión de dependencias y automatización del ciclo de vida del proyecto.

# Manual de despliegue

## 7.1. Requisitos del sistema

- Java 17 o superior   
- Node.js 18.x o superior   
- MySQL 8.x   
- Apache Maven 3.9.x   
- Navegador web moderno (Google Chrome, Firefox, etc.)   
- Editor recomendado: Eclipse (para el backend) y Visual Studio Code (para el frontend)

## 7.2. Instalación y configuración del backend (Spring Boot)

1. Clonar el repositorio del backend o importar el proyecto como "Existing Maven Project" en Eclipse.  
2. Configurar el archivo src/main/resources/application.properties con los siguientes parámetros:

spring.datasource.url=jdbc:mysql://localhost:3306/vamonos\_ahorrando  
spring.datasource.username=tu\_usuario\_mysql  
spring.datasource.password=tu\_contraseña\_mysql  
spring.jpa.hibernate.ddl-auto=update  
spring.jpa.show-sql=true

3. Ejecutar la clase VamonosAhorrandoApplication.java como una aplicación Spring Boot.  
4. Si se ha incluido Swagger para documentación de la API, se puede acceder en:   
http://localhost:8080/swagger-ui.html

## 7.3. Instalación y configuración del frontend (Vue 3)

1. Abrir una terminal y situarse en la carpeta raíz del frontend.  
2. Ejecutar el siguiente comando para instalar las dependencias:

npm install

3. Instalar también las librerías externas necesarias:

npm install pinia chart.js sweetalert2 html2canvas jspdf

4. Crear un archivo .env en la raíz del proyecto con la siguiente línea para apuntar al backend:

VITE\_API\_URL=http://localhost:8080

5. Levantar el servidor de desarrollo con el comando:

npm run serve

6. Acceder a la aplicación desde el navegador en la dirección:   
http://localhost:5173

## 7.4. Configuración de la base de datos MySQL

1. Crear la base de datos ejecutando el siguiente comando en el cliente MySQL:

CREATE DATABASE vamonos\_ahorrando;

2. No es necesario crear las tablas manualmente, ya que se generarán automáticamente al iniciar la aplicación gracias a la configuración:

spring.jpa.hibernate.ddl-auto=update

## 7.5. Librerías externas utilizadas

Backend (Java – Spring Boot):  
- Spring Boot: framework principal para el desarrollo del backend.  
- Spring Data JPA: para simplificar el acceso a datos y operaciones CRUD.  
- Hibernate: proveedor JPA para el mapeo objeto-relacional.  
- Swagger (springdoc-openapi-ui): para documentar y probar la API REST.  
- MySQL Connector/J: driver JDBC oficial para conectar con MySQL.  
- JUnit: librería para pruebas unitarias en Java.  
  
Frontend (Vue.js):  
- Vue.js 3: framework principal para construir la interfaz de usuario.  
- Pinia: sistema de gestión del estado en aplicaciones Vue 3.  
- Chart.js: librería de gráficos para representar estadísticas visuales (ingresos, gastos, ahorro...).  
- SweetAlert2: para mostrar alertas y mensajes visuales personalizados.  
- html2canvas + jsPDF: para capturar secciones visuales (como gráficos y estadísticas) y exportarlas en formato PDF.  
- Fetch API: para realizar peticiones HTTP al backend sin necesidad de librerías externas como Axios.