EASYFINANCE

E F

Autor: Héctor Santana Camacho

Fecha: 21/12/2025

Versión 1.0

Ficha Técnica

EasyFinance

DAW. Desarrollo de Aplicaciones Web
PWR. Proyecto de Desarrollo de Aplicaciones Web

Nombre del fichero:	DAW_PRW_EF_4. Ficha técnica.odt
Fecha de esta versión:	18/01/2026

Historial de revisiones

Fecha	Descripción	Autor
18/01/2026	Creación del documento	Héctor Santana Camacho

ÍNDICE

1 INTRODUCCIÓN

Este documento especifica las tecnologías, herramientas y requisitos técnicos necesarios para el desarrollo, despliegue y ejecución del proyecto EasyFinance, una aplicación web de gestión financiera personal.

Su objetivo es documentar el stack tecnológico completo del proyecto, justificar las decisiones técnicas tomadas y proporcionar una referencia clara para la configuración del entorno de desarrollo e implementación de la aplicación.

EasyFinance sigue el patrón arquitectónico MVC (Modelo-Vista-Controlador) implementado con Spring Boot en el backend, MySQL como sistema de gestión de base de datos, y un frontend basado en tecnologías web estándar (HTML5, CSS3, JavaScript ES6+) potenciado por Bootstrap 5 para diseño responsive y Chart.js para visualización de datos. El desarrollo se estructura en fases incrementales: establecimiento de la estructura frontend (maquetación y componentes visuales), integración con el backend Java para implementar la lógica de negocio, persistencia de datos mediante JPA/Hibernate, y funcionalidades de seguridad con Spring Security.

2 REQUISITOS TÉCNICOS

2.1 Plataforma y herramientas de desarrollo software

2.1.1 Requisitos front-end

A continuación se enumeran las diferentes tecnologías de desarrollo front-end, versiones y herramientas de apoyo al desarrollo:

I) Tecnologías / framework

- *HTML5, CSS3 y JavaScript ES6+*

HTML5 proporciona la estructura semántica de las vistas mediante formularios, tablas de datos y elementos de interfaz accesibles. CSS3 se utiliza para el diseño visual aplicando técnicas como Flexbox y Grid Layout, principalmente para personalización adicional de componentes Bootstrap y estilos específicos del proyecto. JavaScript ES6+ añade interactividad del lado del cliente mediante validación de formularios en tiempo real, comunicación asíncrona con el backend, y actualización dinámica de contenido sin recargar la página.

Se seleccionan tecnologías web estándar (HTML5, CSS3, JavaScript ES6+) sin frameworks complejos como React, Angular o Vue por reducir la complejidad del proyecto permitiendo enfocarse en la arquitectura backend MVC y la lógica de negocio, garantizar compatibilidad universal con todos los navegadores y minimizar la curva de aprendizaje al usar estándares web consolidados.

- *Bootstrap* | *Versión 5.3.8*

Framework CSS frontend que proporciona un sistema de grid responsive mobile-first, componentes UI predefinidos profesionales (navbars, cards, modales, formularios, botones, alerts, badges, tables) y utilidades CSS que aceleran significativamente el desarrollo garantizando consistencia visual sin necesidad de escribir CSS extenso desde cero y cuenta con amplia documentación oficial y comunidad activa con abundantes ejemplos facilitando la resolución de problemas.

Se instala localmente mediante npm en lugar de CDN para permitir trabajo offline durante el desarrollo, control preciso de versiones mediante package.json, y mejor integración con el proyecto al servir los archivos desde el mismo servidor Tomcat.

- *Chart.js | Versión 4.5.1*

Librería JavaScript para crear gráficos interactivos y visualizaciones de datos mediante Canvas API. Se utiliza para representar visualmente estadísticas financieras (gráfico de dona para distribución de gastos por categoría, gráfico de barras para comparativa de ingresos vs gastos mensuales, etc.). Los gráficos son responsive por defecto adaptándose al tamaño del contenedor, incluyen tooltips interactivos y animaciones suaves sin configuración compleja.

Ofrece una API simple y clara apropiada para el nivel del proyecto, peso ligero que no afecta al rendimiento, compatibilidad con Canvas API soportada por todos los navegadores, y funcionalidad autónoma sin requerir dependencias adicionales como jQuery. La visualización gráfica es fundamental para EasyFinance ya que permite a los usuarios comprender de forma inmediata la distribución de sus gastos por categoría, la evolución de su saldo a lo largo del tiempo, y el progreso hacia sus metas de ahorro, mejorando significativamente la experiencia de usuario y el valor de la aplicación. Se instala mediante npm y se integra con el backend para recibir datos dinámicos.

II) Otras Herramientas de desarrollo, depuración, etc

- *Visual Studio Code | Versión 1.108.0*

Editor de código principal para desarrollo frontend. Se utilizan extensiones como Live Server (servidor local con hot-reload), Prettier (formateo automático de código) y ESLint (detección de errores de sintaxis JavaScript) para facilitar el desarrollo y mantener la calidad del código.

Visual Studio Code se selecciona por ser gratuito y de código abierto, ligero y rápido comparado con IDEs pesados, contar con un ecosistema extenso de extensiones específicas para desarrollo web (HTML, CSS, JavaScript), ofrecer integración nativa con Git para control de versiones, proporcionar terminal integrado que agiliza el flujo de trabajo, y ser multiplataforma (Windows, macOS, Linux) con amplia comunidad y documentación.

- *Node.js | Versión 22.09.0*

Entorno de ejecución JavaScript utilizado exclusivamente para gestión de dependencias frontend mediante npm. Aunque el backend es Java, Node.js es necesario en fase de desarrollo para instalar y actualizar librerías como Bootstrap y Chart.js.

Node.js es el estándar de la industria para gestión de paquetes frontend mediante npm. Es necesario para instalar Bootstrap, Chart.js y sus dependencias de forma automatizada, mantener control de versiones mediante package.json, y garantizar que el proyecto pueda ejecutarse de forma idéntica en cualquier momento futuro con las mismas versiones de dependencias. Sin Node.js/npm sería necesario descargar y gestionar manualmente cada librería, lo cual es propenso a errores y dificulta el mantenimiento.

- *npm | Versión 11.6.4*

Gestor de paquetes de Node.js utilizado para instalar y gestionar las dependencias frontend del proyecto. Todas las librerías frontend se declaran en el archivo package.json y se instalan en el directorio node_modules/.

Npm simplifica la gestión de dependencias frontend al centralizar todas las librerías en un archivo package.json versionable, descargar automáticamente las versiones especificadas y sus dependencias, facilitar la actualización de librerías mediante comandos simples (npm update), y garantizar reproducibilidad del entorno permitiendo reinstalar el proyecto con las mismas versiones de dependencias en cualquier momento futuro. Es el gestor de paquetes más utilizado en desarrollo web con el registro más extenso de paquetes JavaScript disponibles.

- *Google Chrome DevTools*

Herramientas de desarrollo integradas en Chrome para inspección de HTML/CSS, depuración de JavaScript, análisis de peticiones de red, monitorización de rendimiento y testing de diseño responsive.

Chrome DevTools es una de las herramientas de depuración más completas y avanzadas para desarrollo web frontend. Se selecciona por estar integrada nativamente en el navegador más utilizado, ofrecer funcionalidades profesionales sin necesidad de instalar software adicional, proporcionar inspector de elementos con modificación en vivo para experimentación rápida de CSS, incluir debugger completo de JavaScript con todas las funcionalidades necesarias, y ser el estándar de la industria con abundante documentación y tutoriales. Chrome es además el motor de renderizado más utilizado lo que garantiza que lo que funciona en Chrome funcionará en la mayoría de navegadores.

- *Firefox Developer Edition*

Navegador alternativo para testing de compatibilidad cross-browser y validación de accesibilidad web mediante herramientas integradas.

Aunque Chrome/Chromium domina el mercado, es fundamental verificar que la aplicación funciona correctamente en navegadores con motores de renderizado diferentes. Firefox Developer Edition incluye herramientas específicas de accesibilidad web más avanzadas que Chrome, permite detectar problemas de CSS que solo aparecen en Firefox, y garantiza que la aplicación es realmente compatible con estándares web y no depende de particularidades de un solo navegador. El testing cross-browser es una buena práctica profesional que previene problemas en producción.

2.1.2 Requisitos back-end

A continuación se enumeran las diferentes tecnologías de desarrollo back-end, versiones y herramientas de apoyo al desarrollo:

I) Plataforma de desarrollo JAVA

a) Framework, runtime y versiones

- *Java Development Kit (JDK) | Versión 17 LTS*

Plataforma de desarrollo y runtime de Java que incluye el compilador para convertir código Java en bytecode, el Java Runtime Environment (JRE) para ejecutar aplicaciones Java, y la Java Virtual Machine (JVM) que interpreta y ejecuta el bytecode.

Se utiliza la versión 17 por ser una versión LTS (Long Term Support) con soporte extendido y características modernas del lenguaje.

- *Spring Boot*

Framework principal para el desarrollo del backend que simplifica la configuración de aplicaciones Spring mediante auto-configuración. Proporciona servidor Tomcat embebido eliminando la necesidad de servidores externos, inyección de dependencias automática mediante anotaciones, gestión simplificada de la arquitectura MVC, y externalización de configuración mediante application.properties. La versión específica se definirá al iniciar el desarrollo del módulo backend.

Se selecciona Spring Boot por ser el framework empresarial Java más utilizado, reducir drásticamente el código, proporcionar un ecosistema completo de starters que integran automáticamente JPA, Security, Thymeleaf y Web, ofrecer documentación extensa y comunidad activa.

- *Spring MVC*

Módulo del framework Spring que implementa el patrón Modelo-Vista-Controlador para aplicaciones web. Gestiona las peticiones HTTP, el enrutamiento de URLs y la comunicación entre controladores y vistas.

Spring MVC se integra nativamente con Spring Boot, proporciona separación clara de responsabilidades siguiendo el patrón MVC, simplifica el manejo de peticiones HTTP mediante anotaciones declarativas, incluiye validación integrada, y es el estándar en aplicaciones web Java empresariales.

- *Thymeleaf*

Motor de plantillas del lado del servidor para generar HTML dinámico. Se integra nativamente con Spring MVC y permite renderizar datos del modelo en las vistas HTML.

Thymeleaf se integra perfectamente con Spring Boot sin configuración manual, genera HTML válido visualizable sin ejecutar el servidor, ofrece sintaxis clara mediante atributos th: que no rompe la estructura HTML y soporta fragmentos reutilizables.

- *Spring Data JPA*

Módulo que simplifica el acceso a datos mediante el estándar JPA (Java Persistence API). Proporciona repositorios automáticos para operaciones CRUD sin necesidad de escribir SQL manualmente.

Spring Data JPA elimina el código de DAOs tradicionales reduciendo líneas de código, proporciona métodos CRUD listos (save, findById, findAll, delete), genera consultas automáticamente desde nombres de métodos, gestiona transacciones, ofrece paginación y ordenación integrada, y abstrae la base de datos permitiendo cambios sin modificar código.

- *Hibernate*

Implementación de JPA y ORM (Object-Relational Mapping) más utilizado en Java. Gestiona el mapeo automático entre clases Java y tablas de base de datos y genera consultas SQL automáticamente.

Hibernate es la implementación JPA más madura y estable, elimina la necesidad de SQL manual para CRUD, genera SQL optimizado específico para cada base de datos, gestiona automáticamente relaciones y es el estándar industrial con integración nativa en Spring Boot.

- *Spring Security*

Framework de seguridad para implementar autenticación, autorización basada en roles, protección contra CSRF (Cross-Site Request Forgery), gestión de sesiones HTTP, y protección de URLs.

Spring Security se integra nativamente con Spring Boot activando protección automática, proporciona autenticación con múltiples estrategias, incluye protección automática contra vulnerabilidades y gestiona sesiones con estrategias configurables.

b) Esquema de desarrollo

La aplicación sigue la arquitectura MVC de Spring Boot con persistencia mediante JPA/Hibernate, separando claramente las responsabilidades en capas independientes que facilitan el mantenimiento, testing y escalabilidad del proyecto.

Estructura de paquetes del proyecto:

- es.easyfinance.models

  Entidades JPA que representan el modelo de datos. Son clases Java anotadas con @Entity que Hibernate mapea automáticamente a tablas de la base de datos.

- es.easyfinance.repositories

  Interfaces que extienden JpaRepository<T, ID> de Spring Data JPA. Spring genera automáticamente la implementación de métodos CRUD (save, findById, findAll, delete) y consultas personalizadas mediante convención de nombres (findByUsuarioId, findByFechaBetween) sin necesidad de escribir código.

- es.easyfinance.services

  Capa de lógica de negocio intermedia entre Controllers y Repositories. Contiene operaciones complejas de procesamiento: cálculo de saldos totales, cálculo de progreso de metas de ahorro, generación de estadísticas para gráficos, validaciones de negocio y orquestación de múltiples repositorios en una misma operación transaccional.

- es.easyfinance.controllers

  Controladores anotados con @Controller que gestionan las peticiones HTTP. Reciben datos de formularios mediante @ModelAttribute o @RequestParam, invocan Services o Repositories para operaciones de base de datos, procesan resultados, añaden datos al Model, y retornan el nombre de la vista Thymeleaf correspondiente.

- es.easyfinance.config

  Clases de configuración del sistema.

- src/main/resources/templates/

  Vistas HTML con motor de plantillas Thymeleaf para renderizar datos dinámicos del lado del servidor.

- src/main/resources/static/

  Recursos estáticos servidos directamente por Tomcat: archivos CSS (Bootstrap personalizado, estilos propios), JavaScript (Chart.js, scripts de validación, peticiones AJAX), imágenes y favicon.

- src/main/resources/application.properties

  Archivo de configuración de la aplicación.

La arquitectura MVC proporciona separación clara de responsabilidades. Models definen la estructura de datos y reglas de persistencia mediante anotaciones JPA, Repositories abstraen completamente el acceso a datos eliminando SQL manual y gestión de conexiones, Services encapsulan la lógica de negocio compleja manteniéndola separada de la presentación, Controllers gestionan el flujo de peticiones HTTP y la lógica de presentación sin conocer detalles de persistencia, y Views se encargan exclusivamente de la presentación visual sin lógica de negocio.

c) IDE

- *Spring Tools Suite for Eclipse* | *Versión 4.32.0*

Entorno de desarrollo integrado especializado para desarrollo Spring Boot basado en Eclipse. Proporciona soporte nativo para proyectos Spring con asistencia inteligente de código para anotaciones Spring, visualización gráfica de beans y dependencias del contexto Spring, integración completa con Maven para gestión automática de dependencias, servidor Tomcat embebido integrado para ejecutar la aplicación directamente desde el IDE sin configuración externa, depurador avanzado de Java y consola integrada para logs de aplicación en tiempo real.

Se selecciona Spring Tools Suite por estar específicamente diseñado para desarrollo Spring Boot con herramientas optimizadas que no incluye Eclipse estándar, proporcionar autocompletado inteligente de anotaciones Spring y validación en tiempo real de configuraciones, ofrecer plantillas predefinidas para crear rápidamente Controllers, Services, Repositories y Entities, incluir todas las funcionalidades de Eclipse IDE, facilitar la configuración inicial de proyectos Spring Boot, permitir ejecutar y depurar aplicaciones Spring directamente sin configuración de servidores externos, ser gratuito y de código abierto sin restricciones de licencia, contar con actualizaciones regulares sincronizadas con nuevas versiones de Spring Boot, y reducir la curva de aprendizaje al proporcionar herramientas visuales para entender la estructura del proyecto Spring.

d) Otras herramientas o gestores de proyecto

- *Apache Maven*

Herramienta de gestión de dependencias y construcción de proyectos Java basada en el concepto de Project Object Model (POM). Gestiona automáticamente las librerías del proyecto y sus dependencias, compila el código fuente Java, ejecuta tests unitarios, empaqueta la aplicación en formato JAR o WAR, y gestiona el ciclo de vida del proyecto con fases bien definidas.

Maven es el estándar en proyectos Java empresariales con adopción masiva en la industria, simplifica radicalmente la gestión de dependencias al resolver automáticamente versiones compatibles y dependencias evitando conflictos, facilita la compilación del proyecto en cualquier entorno, ya que Maven gestiona automáticamente todas las dependencias sin configuración manual, y la integración continua al ser compatible con todas las plataformas de automatización y utiliza una estructura de proyecto convencional ampliamente reconocida que facilita la navegación del código.

- *MySQL Workbench | Versión 8.0*

Herramienta gráfica oficial de MySQL para diseño, administración y desarrollo de bases de datos.

MySQL Workbench es la herramienta oficial desarrollada por Oracle garantizando compatibilidad completa con MySQL 8.0, integra todas las funcionalidades necesarias en una sola aplicación sin necesidad de instalar múltiples herramientas, es completamente gratuito y de código abierto, cuenta con documentación oficial extensa y comunidad activa, funciona en Windows, macOS y Linux, está ampliamente utilizado en entornos educativos con curva de aprendizaje suave y permite verificar en tiempo real que Hibernate/JPA genera correctamente las tablas facilitando la depuración del mapeo objeto-relacional.

d) Otras dependencias

- *MySQL Connector/J*

Driver JDBC oficial de MySQL que permite la comunicación entre aplicaciones Java y el servidor de base de datos MySQL. Hibernate lo utiliza internamente para establecer conexiones, ejecutar consultas SQL generadas automáticamente por JPA, y recuperar resultados convirtiéndolos en objetos Java.

Es el driver oficial desarrollado por Oracle garantizando compatibilidad completa con MySQL 8.0, está optimizado específicamente para MySQL siendo el más eficiente y es un requisito obligatorio ya que Hibernate necesita un driver JDBC específico del motor de base de datos utilizado.

- *Jackson Databind*

Librería Java que proporciona funcionalidades de enlace de datos para convertir objetos Java en JSON y viceversa.

Es el estándar para procesamiento JSON en Java con rendimiento superior a alternativas, se integra automáticamente con Spring Boot sin configuración adicional y es fundamental para APIs REST que intercambian datos JSON con el frontend.

2.2 Plataforma de Ejecución

2.2.1 Servidor

A continuación se indican las características técnicas de mínimos para la ejecución de la aplicación descrita en el proyecto:

Servidor web y versión: Apache Tomcat (embebido en Spring Boot). La versión específica de Tomcat será determinada automáticamente por la versión de Spring Boot seleccionada al iniciar el desarrollo del backend.

Sistema operativo y versión: Compatible con Windows 10/11, macOS 10.15 o superior, y distribuciones Linux (Ubuntu 20.04+, Fedora, Debian). La aplicación es independiente del sistema operativo gracias a la Java Virtual Machine (JVM).

Runtime/Interprete y versión: Java Runtime Environment (JRE) 17 LTS o superior. Adicionalmente requiere servidor MySQL 8.0 o superior instalado y ejecutándose para la gestión de la base de datos.

Observaciones a la configuración: El servidor Tomcat está embebido en el archivo JAR generado por Spring Boot, por lo que no requiere instalación separada. La aplicación se ejecuta con java -jar easyfinance.jar y escucha en puerto 8080 por defecto (configurable en application.properties). Requiere servidor MySQL 8.0 ejecutándose en puerto 3306 con base de datos easyfinance creada. Ambos puertos deben estar disponibles.

2.2.2 Alojamiento físico/ Entornos de ejecución

A continuación, se indican los directorios donde se ubicarán los ficheros de la aplicación desplegada:

Desarrollo: C:\Users\[usuario]\workspace-spring\easyfinance\target\easyfinance-1.0.jar (Windows) o /home/[usuario]/workspace-spring/easyfinance/target/easyfinance-1.0.jar (Linux/macOS).

Pre-Producción/Pruebas: Por definir.

Producción: Por definir.

2.3 Puntos de acceso a la aplicación

A continuación se indican los puntos de acceso url a la aplicación:

Desarrollo: http://localhost:8080 o http://127.0.0.1:8080

Pre-Producción: Por definir.

Producción: Por definir.

3 Bases de Datos

Base de datos: MySQL

Versión: 8.0

Esquema: easyfinance_db

MySQL es un sistema de gestión de bases de datos relacional de código abierto ampliamente utilizado en aplicaciones web. Es completamente gratuito bajo licencia GPL eliminando costes, ofrece excelente rendimiento y escalabilidad para aplicaciones web de tamaño medio, proporciona compatibilidad completa con el estándar SQL y se integra nativamente con JPA/Hibernate, cuenta con documentación oficial extensa y comunidad activa con soluciones a prácticamente cualquier problema y funciona en múltiples plataformas (Windows, Linux, macOS) facilitando portabilidad del proyecto.

Su modelo relacional se adapta perfectamente a la estructura de datos del proyecto donde las entidades principales tienen relaciones claramente definidas que requieren integridad referencial para evitar inconsistencias.

4 Interfaces externas

La aplicación no consume interfaces externas, APIs de terceros ni WebServices externos. Toda la lógica de negocio y gestión de datos se realiza internamente mediante controladores REST del backend Spring Boot que se comunican con la base de datos MySQL.

5 Seguridad

5.1 Autenticación y Autorización

- *Spring Security*

Framework de seguridad que gestiona toda la autenticación y autorización del sistema. Implementa un sistema basado en sesiones HTTP con cookies seguras y protección CSRF automática. Las contraseñas se almacenan hasheadas utilizando el algoritmo BCrypt proporcionado por Spring Security, que genera un hash único para cada contraseña imposibilitando su recuperación incluso con acceso a la base de datos.

Spring Security valida automáticamente roles antes de permitir el acceso a cada URL.

- *Roles de usuario*

USER: Usuario estándar con acceso a gestión de sus propias finanzas (transacciones, categorías personales, metas de ahorro, dashboard).

ADMIN: Administrador con acceso a gestión de usuarios y categorías globales. Los administradores NO tienen acceso a datos financieros privados de usuarios para garantizar privacidad.

- *Control de acceso basado en roles*

URLs públicas (sin autenticación): /login, /register, /css/**, /js/**, /images/**
URLs de usuario (requiere rol USER): /dashboard, /transactions/**, /goals/**, /categories/**, /profile/**
URLs de administrador (requiere rol ADMIN): /admin/users/**, /admin/categories/**

5.2 Gestión de Sesiones

Spring Security gestiona automáticamente las sesiones de usuario mediante cookies de sesión para prevenir acceso mediante JavaScript. Las sesiones tienen un tiempo de expiración configurable, tras el cual el usuario debe autenticarse nuevamente. Al cerrar sesión mediante /logout, Spring Security invalida completamente la sesión del servidor y elimina la cookie del navegador, previniendo reutilización de sesiones cerradas.

5.3 Almacenamiento Seguro de Contraseñas

Las contraseñas nunca se almacenan en texto plano en la base de datos. Spring Security utiliza BCrypt, un algoritmo de hashing diseñado específicamente para contraseñas que genera un hash único para cada contraseña incluso si dos usuarios usan la misma. Esto garantiza que las contraseñas no puedan recuperarse incluso si un atacante obtiene acceso a la base de datos. Durante el login, Spring Security compara el hash almacenado con el hash de la contraseña introducida sin necesidad de descifrarla. El algoritmo BCrypt es resistente a ataques de fuerza bruta debido a su naturaleza computacionalmente costosa.

5.4 Protección contra Ataques Comunes

- *CSRF (Cross-Site Request Forgery)*

Protección habilitada automáticamente por Spring Security mediante tokens CSRF. Cada formulario HTML generado por Thymeleaf incluye automáticamente un token CSRF oculto que se valida en el servidor al procesar peticiones POST/PUT/DELETE. Esto previene que sitios maliciosos externos ejecuten acciones no autorizadas en nombre del usuario autenticado.

- *SQL Injection*

Protección automática mediante JPA/Hibernate que utiliza consultas parametrizadas (prepared statements). Todos los métodos de repositorio (findByUsername, save, delete) generan consultas SQL con parámetros separados del código SQL, previniendo inyección de código malicioso. La aplicación no ejecuta SQL dinámico mediante concatenación de strings, eliminando el vector de ataque más común de SQL injection.

- *XSS (Cross-Site Scripting)*

Protección proporcionada automáticamente por Thymeleaf que escapa todos los valores del modelo renderizados en las vistas mediante th:text y atributos similares. Los caracteres especiales HTML (<, >, &, ", ') se convierten automáticamente en entidades HTML (<, >, &, etc.), previniendo la inyección de código JavaScript malicioso. Adicionalmente, Spring Security configura headers de seguridad HTTP como X-XSS-Protection y Content-Security-Policy para protección adicional en el navegador.

6 Control de versiones

Sistema de control de versiones: Git | Versión 2.52.0

Plataforma de hosting: GitHub

Repositorio del proyecto: https://github.com/HectorSantanaC/EasyFinance

El proyecto utiliza una estrategia de branching simplificada con dos ramas principales:

- main: Rama principal de producción que contiene únicamente código estable, probado y listo para desplegar. Solo se actualiza cuando se completan hitos importantes del proyecto o versiones funcionales completas que pueden ser presentadas o evaluadas. Representa el estado "oficial" del proyecto en cada momento.

- develop: Rama de desarrollo activo donde se realizan todos los commits diarios, se implementan nuevas funcionalidades y se corrigen errores. Es la rama de trabajo principal del desarrollador. Una vez que el código en develop es estable y funcional tras completar un módulo o funcionalidad significativa, se fusiona con main mediante pull request o merge directo.

Para un proyecto académico individual, una estructura con dos ramas es suficiente y ofrece las ventajas de simplicidad de gestión siendo fácil de entender sin experiencia previa avanzada en Git, separación clara donde main contiene versiones estables, mientras develop sirve para trabajo diario sin afectar la rama estable, menor complejidad evitando la sobrecarga de gestionar múltiples ramas innecesarias para un solo desarrollador que ralentizarían el desarrollo y escalabilidad ya que si el proyecto crece o se incorporan colaboradores se pueden añadir ramas para funcionalidades específicas o para correcciones urgentes sin reestructurar el flujo existente.

El repositorio en GitHub actúa como respaldo remoto del código local, previniendo pérdida de trabajo por fallos de hardware. Adicionalmente, GitHub permite compartir el proyecto para revisión y evaluación mediante acceso al repositorio público.

7 Observaciones

7.1 Escalabilidad

Aunque el proyecto está diseñado como aplicación monolítica para simplificar el desarrollo académico y cumplir con los requisitos del ciclo formativo, se han aplicado principios de diseño que facilitan la escalabilidad y mantenibilidad futura si el proyecto evoluciona a un entorno de producción real:

- Arquitectura por capas: Separación clara entre capa de presentación (controladores Spring MVC), capa de lógica de negocio (services con anotación @Service), y capa de persistencia (repositories JPA). Esta separación permite modificar o reemplazar componentes de forma independiente sin afectar al resto del sistema siguiendo principios de arquitectura limpia y Domain-Driven Design.

- Inversión de dependencias: Uso de interfaces Java para los servicios y repositorios permitiendo inyección de dependencias mediante @Autowired de Spring. Esto facilita testing mediante mocks y permite cambiar implementaciones sin modificar código cliente, cumpliendo con el principio de inversión de dependencias (SOLID).

- Configuración externalizada: Todas las configuraciones sensibles (credenciales de base de datos, puerto del servidor, timeout de sesiones, configuración de Hibernate) están externalizadas en application.properties permitiendo modificar configuraciones sin recompilar el código. Esto facilita despliegue en diferentes entornos (desarrollo, testing, producción) mediante profiles de Spring Boot (application-dev.properties, application-prod.properties).

- Optimización de base de datos: Uso de índices en columnas de búsqueda frecuente (username único, email único, foreign keys automáticas) para mejorar rendimiento de consultas. Las relaciones JPA están configuradas con estrategias de carga apropiadas (LAZY para colecciones, EAGER solo cuando es necesario) para evitar problemas de rendimiento N+1.

- Potencial de microservicios: Aunque actualmente es monolítico, los módulos del sistema (usuarios, transacciones, categorías, metas) están suficientemente desacoplados para extraerse como microservicios independientes en el futuro si el proyecto crece significativamente.

- APIs REST internas: Los controladores siguen convenciones REST (@GetMapping, @PostMapping, @PutMapping, @DeleteMapping) facilitando la creación de una API REST pública en el futuro para integración con aplicaciones móviles o servicios externos.

7.2 Licencia

El proyecto EasyFinance se desarrolla con fines académicos como Proyecto de Fin de Ciclo del Ciclo Formativo de Grado Superior en Desarrollo de Aplicaciones Web (DAW). El código fuente es propiedad del autor y está publicado en GitHub bajo licencia MIT, permitiendo uso, modificación y distribución libre con atribución.

Las tecnologías y dependencias utilizadas (Spring Boot, MySQL, Thymeleaf, Chart.js, Bootstrap) son de código abierto con licencias permisivas (Apache License 2.0, GPL, MIT) que permiten uso libre en proyectos académicos y comerciales sin restricciones significativas.

7.3 Dependencias de Terceros

Todas las dependencias del backend se gestionan automáticamente mediante Apache Maven y están declaradas explícitamente en el archivo pom.xml con sus versiones correspondientes. Maven descarga automáticamente las librerías desde Maven Central Repository durante la fase de compilación, garantizando reproducibilidad del entorno de desarrollo.

7.4 Entorno de desarrollo recomendado

El proyecto ha sido desarrollado y probado en el siguiente entorno:

- Sistema operativo: Windows 11
- IDE: Spring Tool Suite 4 / Eclipse con plugin Spring Tools
- Java: OpenJDK 17 LTS
- Base de datos: MySQL Workbench 8.0
- Navegadores probados: Google Chrome, Mozilla Firefox.
- Control de versiones: Git 2.52.0 con GitHub como repositorio remoto

El proyecto es compatible con otros entornos de desarrollo (IntelliJ IDEA, VS Code con extensiones Java) y sistemas operativos (Linux nativo, macOS) siempre que cumplan los requisitos mínimos especificados en la sección 2.2.