



ugr | Universidad
de Granada

TRABAJO FIN DE GRADO
GRADO EN INGENIERÍA INFORMÁTICA

APP para asesoramiento financiero bursátil

Autor

David Martínez Díaz

Director

Jose Manuel Zurita López



Escuela Técnica Superior de Ingenierías Informática y de
Telecomunicación

—
Granada, Julio de 2024







ugr | Universidad
de Granada

APP para asesoramiento financiero bursátil

Autor

David Martínez Díaz

Director

Jose Manuel Zurita López



Resumen

Diseño e implementación de una APP para asesoramiento financiero bursátil

David Martínez Díaz

Palabras clave: Flutter, Flask, Programación de Apps, Análisis Bursátil, Datos en Tiempo Real, Asesoramiento Financiero

Resumen:

El propósito de este Proyecto Fin de Grado (PFG) se centra en el diseño y la implementación de una aplicación de asesoramiento financiera para el mercado bursátil, estando enfocada principalmente a usuarios con diversos niveles de conocimientos financieros, para permitirles realizar simulaciones virtuales de inversiones informadas y con recomendaciones de la bolsa de valores.

Una de las características distintivas de la aplicación es su capacidad para leer y analizar datos en tiempo real de compañías cotizadas, obtenidos de fuentes confiables como Yahoo Finance, obteniendo dichos datos a través de la base de datos implementada en el servidor y obteniendo dicha información a través de una API para un uso básico de los datos. Basándose en estos análisis, la aplicación sugiere estrategias de inversión personalizadas.

Además, la aplicación proporciona una herramienta para que los usuarios creen y gestionen una cartera de valores, facilitando el seguimiento y la evaluación de sus inversiones. La implementación del proyecto se ha llevado a cabo a través de una metodología ágil, permitiendo un desarrollo iterativo y enfocado en el usuario. Se comenzó con un análisis detallado del problema y las necesidades del mercado, seguido de la conceptualización y diseño de una interfaz de usuario intuitiva y accesible. Utilizando Flutter para el desarrollo del frontend y Flask para el backend.

Posteriormente continuaremos con el diseño de la interfaz de la aplicación y con su respectiva implementación cuyo objetivo final es poder crear una aplicación intuitiva y sencilla para el usuario. La interfaz de usuario de la aplicación está diseñada para ser intuitiva y fácil de usar, asegurando que incluso los inversores novatos puedan navegar y utilizar sus funciones sin complicaciones.



Abstract

Design and Implementation of an App for Stock Market Financial Advisory

David Martínez Díaz

Keywords: Flutter, App Programming, Stock Market Analysis, Real-Time Data, Financial Advisory

Abstract:

The purpose of this Final Degree Project (FDP) focuses on the design and implementation of a financial advisory application for the stock market. The application is aimed at users with various levels of financial knowledge, allowing them to perform informed stock market investment simulations.

One of the distinctive features of the application is its ability to read and analyze real-time data from listed companies, obtained from reliable sources such as Yahoo Finance, acquiring this data through the database implemented on the server and obtaining such information through an API for basic data usage. Based on these analyses, the application suggests personalized investment strategies.

In addition, the application provides a tool for users to create and manage a portfolio of securities, facilitating the monitoring and evaluation of their investments.

For the realization of the project, we will start from different phases or stages, where we will begin with a study of the problem or need to be addressed and then analyze it using the knowledge of software engineering. Subsequently, we will continue with the design of the application interface and with its respective implementation, whose final objective is to create an intuitive and simple application for the user.

To achieve this, the development of the app was carried out using Flutter, a popular framework for building cross-platform mobile applications. Flutter's versatility and efficiency allowed for the creation of a user-friendly interface while ensuring smooth performance across both Android and iOS platforms.

Furthermore, the stock market analysis within the app is backed by robust algorithms that process the real-time data to generate valuable insights. These algorithms consider various financial indicators, historical trends, and market news to provide users with well-informed investment recommendations.

The user interface of the application is designed to be intuitive and easy to use, ensuring that even novice investors can navigate and use its functions without complications.



Yo, **David Martínez Díaz**, alumno de la titulación *Doble Grado en Ingeniería Informática y Administración y Dirección de Empresas* de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI 44669141J, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: **David Martínez Díaz**

Granada a X de Julio de 2024





D. Jose Manuel Zurita López, Profesor de Ciencias de la Computación e Inteligencia Artificial del Departamento Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada.

Informa:

Que el presente trabajo, titulado ***Diseño e implementación de una APP para asesoramiento financiero bursátil***, ha sido realizado bajo su supervisión por **David Martínez Díaz**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a X de Julio de 2024.

Los directores:

Jose Manuel Zurita López





Agradecimientos

A mi familia y amigos, que siempre me han apoyado tanto en los momentos malos como en los buenos, permitiéndome un camino más llevadero.

A mis compañeros por aguantar junto a mi esta experiencia vivida durante estos 5 años. Y a mi tutor Jose Manuel Zurita López cuya confianza otorgada en mi junto a sus consejos han sido bastante importantes para sacar adelante este proyecto.



Índice

Resumen.....	4
Abstract.....	5
Agradecimientos	10
Índice.....	11
Índice de figuras.....	14
Índice de tablas.....	16
1. Introducción	20
1.1 Motivación	20
1.2 Descripción del Problema	20
1.3 Objetivos del proyecto	21
1.4 Estructura de la memoria.....	22
2. Planificación.....	23
2.1. Desarrollo de software	23
2.2. Planificación temporal.....	25
2.3. Presupuesto	26
2.3.1 Coste Mensual para la Empresa	26
2.3.2 Gastos de Ejecución	27
2.3.3 Resumen del Presupuesto (5 Meses).....	28
3. Análisis.....	29
3.1. Stakeholders y Usuarios	29
3.1.1. Personas.....	29
3.1.2 Escenarios.....	32
3.2. Backlogs y Requisitos	35
3.2.1. Listado inicial de Historias de Usuario	35
3.2.2. Velocidad del Equipo	37
3.2.3. Descripción de Entregas.....	38
3.2.4. Product Backlog	39
3.2.5. Tarjetas de las Historias de Usuario	41
3.2.5 Sprints Backlogs.....	52
3.3. Diagramas de secuencia	54
3.3.1 Registrar un usuario	54
3.3.2 Inicio de sesión de un usuario	55
3.3.3 Recuperar la contraseña	56
3.3.4 Obtener datos de las acciones.....	57
3.3.5 Comprar o vender una acción.....	58
3.3.6 Obtener noticias.....	59



3.3.7 Recibir notificaciones.....	60
3.3.8 Editar perfil de un usuario.....	61
4. Diseño	62
4.1. Diseño de la base de datos.....	62
4.1.1 Diagrama Entidad-Relación de la Base de Datos.....	63
4.1.2. Paso a tablas y fusión	64
4.1.3. Normalización de la base de datos	66
4.1.4 Tablas de la Base de Datos.....	67
4.1.5 Relaciones de la Base de Datos.....	69
4.2. Diseño de la arquitectura del sistema	70
4.3. Diseño de la interfaz de usuario	71
4.3.1. Diseño de la interfaz de la aplicación móvil	71
4.3.2. Diseño de la interfaz de la página web.....	82
5. Implementación.....	86
5.1. Desarrollo de aplicación.....	86
5.1.1. Entorno de desarrollo	86
5.1.2. Herramientas de Front-end.....	89
5.1.3. Herramientas de Back-end	90
5.1.4. Diagrama de clases.....	91
5.1.5. Implementación de la base de datos	101
5.1.6. API	106
5.2. Desarrollo.....	110
5.2.1 Gestión de inicio de sesión y registro.....	110
5.2.2 Sistema de recuperación de contraseña	112
5.2.3 Sistema de Simulación Virtual	115
5.2.4 Sistema de Recomendación.....	118
5.2.5 Sistema de Notificaciones	124
5.3. Despliegue del proyecto	129
5.4. Pruebas	135
5.4.1 Prueba Caja Blanca	135
5.4.2 Prueba Caja Negra.....	135
5.4.3 Pruebas Back-end.....	136
5.4.4 Pruebas Front-end	138
5.4.5 Github Actions (Workflows).....	142
6. Conclusiones	146
6.1. Temporización real.....	146
6.2. Objetivos alcanzados.....	148
6.3. Lecciones aprendidas	149
6.4. Trabajos futuros	150



7. Bibliografía	151
7.1 Referencias.....	151
8. Anexos.....	154
8.1. Manual de Usuario	154
8.1.1 Objetivo.....	154
8.1.2 Requerimientos.....	154
8.1.3 Inicio de Sesión y Registro.....	155
8.1.4 Recuperación de Contraseña	156
8.1.5 Página Principal (Home)	157
8.1.6 Página de Acciones	158
8.1.7 Página de Acción Específica	159
8.1.8 Página de Cartera	160
8.1.9 Página de Mis Acciones	161
8.1.10 Página de Noticias	162
8.1.11 Página de Aprendizaje.....	163
8.1.12 Página de Configuración	164
8.2. Vista del sitio web (Back-end).....	165
8.2.1 Página de inicio	165
8.2.2 Página de inicio de sesión Web (Back-end).....	167
8.2.3 Página de gestión de la base de datos.....	168
8.3. Tableros Scrum con Jira Software	169
8.3.1 Tablero Sprint 1.....	169
8.3.2 Tablero Sprint 2.....	169
8.3.3 Tablero Sprint 3.....	170
8.3.4 Tablero Sprint 4.....	170
8.3.4 Tablero Sprint 5.....	171
8.4. Código fuente	172
8.5. Enlace a la Web, Google Play Store y Video Explicativo	173



Índice de figuras

Ilustración 1 Diagrama de Gantt Inicial	25
Ilustración 2 Diagrama de secuencia de registro de usuario	54
Ilustración 3 Diagrama de secuencia de inicio de sesión de un usuario.....	55
Ilustración 4 Diagrama de secuencia de recuperación de contraseña.....	56
Ilustración 5 Diagrama de secuencia de obtención de datos	57
Ilustración 6 Diagrama de secuencia de comprar o vender una acción.....	58
Ilustración 7 Diagrama de secuencia de obtención de noticias	59
Ilustración 8 Diagrama de secuencia de recibir notificaciones.	60
Ilustración 9 Diagrama de secuencia de editar un perfil	61
Ilustración 10 Diagrama Entidad-Relación de la Base de Datos.....	63
Ilustración 11 Modelo-Vista-Controlador (MVC)	70
Ilustración 12 Interfaz de Inicio de Sesión y de Registro.....	72
Ilustración 13 Interfaz de Cambiar Contraseña.....	73
Ilustración 14 Interfaz de Inicio o Home.....	74
Ilustración 15 Interfaz General de Acciones	75
Ilustración 16 Interfaz de Acción Específica	76
Ilustración 17 Interfaz de Cartera y Transacciones	77
Ilustración 18 Interfaz de Mis Acciones	78
Ilustración 19 Interfaz de Noticias	79
Ilustración 20 Interfaz de Aprendizaje	80
Ilustración 21 Interfaz de Configuración.....	81
Ilustración 22 Diseño de la Página de Home (Back-end)	83
Ilustración 23 Diseño de la Página de Inicio de Sesión (Back-end).....	84
Ilustración 24 Diseño de la página de gestión de la base de datos (Back-end)	85
Ilustración 25 Diagrama de Clases (Front-end)	92
Ilustración 26 Diagrama de Clases del paquete de Modelos (Front-end)	93
Ilustración 27 Diagrama de clases del paquete de Controladores (Front-end).....	95
Ilustración 28 Diagrama de clases del paquete de Vistas (Front-end)	97
Ilustración 29 Diagrama de Clases (Back-end).....	98
Ilustración 30 Diagrama de clases del paquete de Modelos (Back-end).....	99
Ilustración 31 Diagrama de clases del paquete de Controladores (Back-end)	100
Ilustración 32 Correo con código de verificación	113
Ilustración 33 Inicio de Sesión en OCI	130
Ilustración 34 Creación de una instancia del servidor.....	130



Ilustración 35 Dashboard de nuestro servidor en OCI	131
Ilustración 36 Comando SSH para acceder a mi servidor	131
Ilustración 37 Captura de Tests pasados correctamente.....	137
Ilustración 38 Workflows de Github Actions.....	142
Ilustración 39 Diagrama de Gantt Real	146
Ilustración 40 Capturas de pantalla del Inicio de Sesión y de Registro	155
Ilustración 41 Captura de pantalla de Recuperar Contraseña.....	156
Ilustración 42 Captura de pantalla de la Página de Inicio	157
Ilustración 43 Captura de pantalla de la Página de Acciones.....	158
Ilustración 44 Capturas de pantalla de la Página de Acción Específica.....	159
Ilustración 45 Captura de pantalla de la Página de Cartera.....	160
Ilustración 46 Captura de pantalla de la Página Mis Acciones	161
Ilustración 47 Captura de pantalla de la Página Noticias	162
Ilustración 48 Captura de pantalla de la Página Aprendizaje.....	163
Ilustración 49 Captura de pantalla de la Página Configuración	164
Ilustración 50 Página de Inicio de la Web (Back-end).....	166
Ilustración 51 Página de Inicio de sesión Web (Back-end).....	167
Ilustración 52 Página de gestión de la base de datos.....	168
Ilustración 53 Tablero Sprint 1.....	169
Ilustración 54 Tablero Sprint 2.....	169
Ilustración 55 Tablero Sprint 3.....	170
Ilustración 56 Tabla Sprint 4	170
Ilustración 57 Tablero Sprint 5.....	171



Índice de tablas

Tabla 1 Definición de Objetivos	21
Tabla 2 Definición de Sprints.....	24
Tabla 3 Coste Mensual para la Empresa	26
Tabla 4 Material Inventariable	27
Tabla 5 Gastos Recurrentes y Directos	27
Tabla 6 Gastos Indirectos	28
Tabla 7 Gastos de Personal	28
Tabla 8 Presupuesto final	28
Tabla 9 Persona 1	29
Tabla 10 Persona 2	30
Tabla 11 Persona 3	31
Tabla 12 Escenario 1	32
Tabla 13 Escenario 2	32
Tabla 14 Escenario 3	33
Tabla 15 Escenario 4	33
Tabla 16 Escenario 5	34
Tabla 17 Escenario 6.....	34
Tabla 18 Historias de Usuario	36
Tabla 19 Descripción de Entrega	38
Tabla 20 Product Backlog	40
Tabla 21 Tarjeta HU.1.....	41
Tabla 22 Tarjeta HU.2.....	41
Tabla 23 Tarjeta HU.3.....	42
Tabla 24 Tarjeta HU.4.....	42
Tabla 25 Tarjeta HU.5.....	43
Tabla 26 Tarjeta HU.6.....	43
Tabla 27 Tarjeta HU.7.....	43
Tabla 28 Tarjeta HU.8.....	44
Tabla 29 Tarjeta HU.9.....	44
Tabla 30 Tarjeta HU.10.....	44
Tabla 31 Tarjeta HU.11.....	45
Tabla 32 Tarjeta HU.12.....	45
Tabla 33 Tarjeta HU.13.....	45
Tabla 34 Tarjeta HU.14.....	46



Tabla 35 Tarjeta HU.15.....	46
Tabla 36 Tarjeta HU.16.....	46
Tabla 37 Tarjeta HU.17.....	47
Tabla 38 Tarjeta HU.18.....	47
Tabla 39 Tarjeta HU.19.....	47
Tabla 40 Tarjeta HU.20.....	48
Tabla 41 Tarjeta HU.21.....	48
Tabla 42 Tarjeta HU.22.....	49
Tabla 43 Tarjeta HU.23.....	49
Tabla 44 Tarjeta HU.24.....	50
Tabla 45 Tarjeta HU.25.....	50
Tabla 46 Tarjeta HU.26.....	51
Tabla 47 Sprint Backlog 1.....	52
Tabla 48 Sprint Backlog 2.....	52
Tabla 49 Sprint Backlog 3.....	53
Tabla 50 Sprint Backlog 4.....	53
Tabla 51 Sprint Backlog 5.....	53
Tabla 52 Tabla Usuario.....	67
Tabla 53 Tabla Acción	67
Tabla 54 Tabla AccionesFavoritas.....	67
Tabla 55 Tabla Transacción	67
Tabla 56 Tabla Cartera.....	68
Tabla 57 Tabla Notificación.....	68
Tabla 58 Tabla ResetClaveToken	68
Tabla 59 Tabla ArticulosAprendizaje	68
Tabla 60 Relación Usuario.....	69
Tabla 61 Relación Accion.....	69
Tabla 62 Relación AccionFavoritas	69
Tabla 63 Relación Transaccion	69
Tabla 64 Relación Cartera.....	69
Tabla 65 Relación Notificacion.....	69
Tabla 66 Relación ResetClaveToken	69
Tabla 67 Creación de tablas en la Base de Datos.....	103
Tabla 68 Restricciones en la Base de Datos.....	105
Tabla 69 Secuencias de Inicialización en la Base de Datos	105
Tabla 70 Llamadas a la API desde ArticuloController	106
Tabla 71 Obtener artículos con el controlador ArticulosController	107



Tabla 72 Definición de Endpoints en el Backend	107
Tabla 73 Uso de la API Yahoo Finance	108
Tabla 74 Uso de la API NewsApi	109
Tabla 75 Inicio de sesión (Front-end)	110
Tabla 76 Inicio de sesión (Back-end).....	111
Tabla 77 Envío de código de verificación (Front-end)	112
Tabla 78 Envío de código de verificación (Back-end).....	113
Tabla 79 Verificar código y cambiar contraseña (Front-end)	114
Tabla 80 Verificar código y cambiar contraseña (Back-end).....	114
Tabla 81 Registrar usuario y crear cartera (Back-end).....	115
Tabla 82 Comprar acción (Front-end).....	116
Tabla 83 Comprar acción (Back-end)	117
Tabla 84 Obtener recomendaciones (Front-end).....	118
Tabla 85 Obtener recomendaciones (Back-end)	119
Tabla 86 Obtener estrategias con backtesting (Back-end)	121
Tabla 87 Obtener soportes y resistencias (Back-end)	122
Tabla 88 Obtener predicciones de los precios (Back-end).....	123
Tabla 89 Comprobaciones Notificaciones (Front-end)	124
Tabla 90 Estructura del servicio Socket (Front-end).....	126
Tabla 91 Estructura del servicio Socket (Back-end)	128
Tabla 92 Código del Dockerfile	132
Tabla 93 Código del Docker-Compose.yml.....	133
Tabla 94 Dependencias del requirements.txt	134
Tabla 95 Comandos para montar una imagen de Docker	134
Tabla 96 Código de tests para modelos de la base de datos.....	137
Tabla 97 Código del MockUsuario (Front-end).....	139
Tabla 98 Código del test de inicio de sesión.....	140
Tabla 99 Código del test Widget BalanceCard	141
Tabla 100 Código para automatización pruebas en el Front-end	143
Tabla 101 Código para automatización pruebas en el Back-end	145





1. Introducción

1.1 Motivación

Desde chico siempre ha despertado mi interés y curiosidad el mundo financiero, de hecho, este ha sido una constante en mi vida académica ya que hago el doble grado de Informática y ADE. La bolsa de valores con su complejidad y con su dinamismo siempre me ha intrigado saber los mecanismos que la mueven, motivándome a conocer profundamente como los individuos y las corporaciones interactúan entre sí en este ecosistema global.

Sin embargo, he notado que muchas personas, aunque tenga interés en invertir y participar en el mercado, se sienten abrumadas por la falta de conocimientos financieros y la incertidumbre que puedes parecer a simple vista las inversiones. George Soros, uno de los inversores más reconocidos del mundo, una vez dijo: “El problema no es lo que uno no sabe, sino lo que uno cree que sabe estando equivocado” [1].

Esta observación me dio la idea de la necesidad de desarrollar una herramienta que pueda simplificar y facilitar el acceso al mercado bursátil. Una aplicación que no solo brinde asesoramiento financiero, sino que también intente educar a los usuarios, independientemente de su nivel de experiencia.

La tecnología ha abierto nuevas vías para poder acceder a la información financiera, pero considero que todavía hay una brecha entre la teoría y la puesta en práctica de esta. Con el desarrollo de esta aplicación, mi objetivo es intentar reducir esa brecha, proporcionando no solo datos en tiempo real y análisis bursátiles, sino que también pueda orientar a los usuarios hacia decisiones de inversión informadas y estratégicas.

La intención es hacer que el mercado bursátil sea accesible y comprensible para todos, eliminando el miedo y la incertidumbre que a menudo rodean las inversiones.

1.2 Descripción del Problema

El problema central que motiva el desarrollo de esta aplicación móvil surge de la necesidad de simplificar el acceso al mundo del asesoramiento financiero bursátil. A pesar de la creciente popularidad de la inversión en bolsa, muchos potenciales inversores se ven disuadidos por la complejidad percibida del análisis de mercado y la toma de decisiones de inversión.

Esto se debe a la falta de herramientas intuitivas y accesibles que puedan guiar a los usuarios, independientemente de su nivel de experiencia, a través del complejo ecosistema de las finanzas bursátiles.

La solución que se propone consiste en el desarrollo de una aplicación móvil que brinde a los usuarios información actualizada y análisis detallados del mercado, al mismo tiempo que presenta estos datos de una manera que sea fácilmente digerible para los usuarios con poca experiencia.



1.3 Objetivos del proyecto

Para cumplir con las expectativas y necesidades identificadas para este proyecto, se establecen los siguientes objetivos. Algunos serán fundamentales para el funcionamiento básico de la aplicación, mientras que otros serán complementarios, mejorando la experiencia del usuario y enriqueciendo la funcionalidad del software. [2]

A continuación, se detallan los objetivos propuestos para el proyecto:

Objetivo 1	Desarrollo de una Interfaz Intuitiva
Tipo	Obligatorio
Descripción	Crear una interfaz de usuario sencilla y fácil de navegar, que permita a los usuarios con diferentes niveles de experiencia acceder y utilizar la aplicación sin complicaciones.
Objetivo 2	Integración de Datos en Tiempo Real
Tipo	Obligatorio
Descripción	Implementar la funcionalidad para obtener y mostrar datos financieros en tiempo real, para que los usuarios puedan acceder a información actualizada de los mercados.
Objetivo 3	Análisis Bursátil Personalizado
Tipo	Obligatorio
Descripción	Desarrollar algoritmos que analicen los datos del mercado y generen recomendaciones de inversión personalizadas basadas en el perfil y objetivos del usuario.
Objetivo 4	Gestión de Cartera de Inversiones
Tipo	Obligatorio
Descripción	Ofrecer una herramienta dentro de la aplicación que permita a los usuarios crear y gestionar su propia cartera de valores, facilitando el seguimiento de sus inversiones.
Objetivo 5	Educación y Asesoramiento Financiero
Tipo	Opcional
Descripción	Incorporar secciones educativas y asesoramiento sobre inversiones y finanzas para ayudar a los usuarios a tomar decisiones informadas.
Objetivo 6	Adaptabilidad y Escalabilidad
Tipo	Opcional
Descripción	Diseñar la aplicación de manera que sea fácilmente escalable y adaptable a futuras expansiones o cambios en los mercados financieros.
Objetivo 7	Seguridad y Privacidad de los Datos
Tipo	Opcional
Descripción	Garantizar la seguridad y la privacidad de los datos de los usuarios, implementando medidas de protección adecuadas.

Tabla 1 Definición de Objetivos



1.4 Estructura de la memoria

Esta memoria está compuesta por las siguientes partes [3]:

- **Resumen:** En este apartado se encuentra la síntesis del proyecto, destacando su finalidad, las características principales de la aplicación desarrollada, tanto en español como en inglés.
- **Introducción:** Aquí se introduce el contexto y el propósito del proyecto, este incluye:
 - Motivación que ha llevado a la creación del proyecto.
 - Descripción del problema.
 - Lista de objetivos a lograr durante la realización del proyecto.
 - Estructura básica del documento.
- **Planificación:** Se demuestra cómo es la planificación del proyecto, las varias etapas que la componen y una estimación de presupuesto del mismo:
 - Desarrollo de software: metodología utilizada.
 - Planificación temporal: programación teórica de los plazos a cumplir durante el desarrollo.
 - Presupuesto: cálculo de los costes que suponen llevar a cabo este proyecto.
- **Análisis:** En esta sección se especifica los distintos requisitos que tenga el sistema y se hará los respectivos modelos y descripciones de los casos de uso, así como una breve explicación del sistema a través de diagramas.
- **Diseño:** Aquí se crearán las estructuras y el modelado de los componentes del sistema, en nuestro caso será:
 - Base de datos.
 - Arquitectura del sistema.
 - Interfaz de usuario tanto para el móvil como para la web.
- **Implementación:** Aquí se muestran tanto las herramientas utilizadas, como los frameworks y lenguajes que se han empleado. También se comentarán las distintas fases del desarrollo, donde se explican los diversos pasos hasta la puesta en producción de nuestro proyecto.
- **Conclusiones:** Se comentan los resultados obtenidos, así como los objetivos alcanzados, las lecciones aprendidas en el transcurso además de las posibles continuaciones futuras para este.
- **Bibliografía:** Referencias de toda la documentación utilizadas para obtener los conocimientos suficientes para poder realizar el proyecto.
- **Anexos:** Por último, se referencia los posibles documentos, imágenes o distintos elementos relevantes para el trabajo, así como un posible glosario de definiciones.



2. Planificación

2.1. Desarrollo de software

Cuando se desarrolla software, suele necesitarse tiempo para analizar y planificar. Tradicionalmente, muchas veces este proceso se realizaba sin utilizar ninguna metodología definida, lo cual, en muchas ocasiones, ha provocado productos de muy mala calidad.

Para contextualizar un poco, una metodología de desarrollo consiste en un conjunto de procedimientos, técnicas, herramientas y documentos que ayudan principalmente a los desarrolladores a la hora de implementar el software. En esta última década, hay dos grandes corrientes metodológicas en el desarrollo de software, las ágiles y las tradicionales.

Las metodologías tradicionales se centran sobre todo en el proceso, controlando estrictamente las actividades y se basan en normas provenientes de estándares del entorno de desarrollo con un proceso mucho más controlado. Por otro lado, las metodologías de desarrollo ágiles, como Scrum, se centran más en el elemento humano y en la colaboración y participación activa del cliente.

Para la realización de este proyecto se ha seleccionado una metodología de desarrollo de software ágil, específicamente Scrum, debido a su naturaleza iterativa e incremental, además de su previo uso en asignaturas como Dirección y Gestión de Proyectos y Metodologías de Desarrollo Ágil [4].

Esta metodología, es ideal por su adaptabilidad a los entornos cambiantes que suelen ocurrir generalmente en el desarrollo de los proyectos, por lo que, en nuestro caso, para la aplicación de asesoramiento financiero bursátil se pueden producir desviaciones o ajustes necesarios para un correcto desarrollo. En Scrum, el desarrollo se divide en Sprints, donde al final de cada supone en un incremento funcional del software realizado, es decir, en una aplicación funcional, con lo que se consigue un desarrollo continuo y eficiente.

Los componentes de Scrum incluyen roles ya definidos, como el ‘Product Owner’, que define y valida las historias de usuario, el ‘Scrum Master’, que se encarga de que todo se realice correctamente; y el ‘Equipo de Desarrollo’, que implementa las historias de usuario. Además, hay eventos durante la ejecución de este, como son los ‘Sprint’, el ‘Sprint Planning’, los ‘Dailys’, el ‘Sprint Review’ y, por último, el ‘Sprint Retrospective’.

Debido a la complejidad y las características innovadoras de la aplicación, resulta bastante conveniente el uso de una metodología que permita adaptarse y ajustarse con rapidez por las posibles incidencias o complicaciones que se puedan producir en el transcurso del desarrollo de este. Por tanto, esta metodología se enfoca sobre todo en la colaboración, y en la autoorganización para poder entregar frecuentemente productos funcionales, adaptándose perfectamente en el contexto de la situación [5].

Habiendo explicado los componentes de Scrum y su funcionamiento en términos generales, parece evidente que está enfocado en el desarrollo en equipo, por lo que tendré que adaptarlo a mi contexto actual.



Para el desarrollo del proyecto, lo estructuraremos en varios sprints, cada uno con objetivos definidos, cada uno de estos sprints tendrán diferentes fases como conceptualización, diseño de la interfaz, implementación de funcionalidades clave, integración con fuentes de datos en tiempo real y la realización de pruebas para asegurar la calidad y eficiencia de la aplicación.

Los sprints serán los siguientes:

Sprint 1	
Conceptualización	Establecer los requisitos del proyecto, conseguir conocimientos sobre la investigación de mercado y definición de las funcionalidades claves de la aplicación y crear prototipos básicos de la interfaz de usuario para validar conceptos y flujos de usuario.
Sprint 2	
Diseño e Interfaz de Usuario	Desarrollo de la arquitectura del software y diseño de la interfaz de usuario, centrándonos principalmente en la experiencia del usuario.
Sprint 3	
Implementación Inicial	Desarrollo de las funcionalidades básicas de la aplicación, implementar la integración con la API de Yahoo Finance para la obtención de datos en tiempo real.
Sprint 4	
Ampliación y Mejora	Implementación de funcionalidades avanzadas como la gestión de cartera y herramientas de análisis bursátil, así como la implementación de la API de noticias.
Sprint 5	
Pruebas, Ajustes y Lanzamiento	Creación de pruebas para comprobar la fiabilidad de la aplicación, y realizar ajustes basados en los resultados de estos. Además de la preparación para la finalización de la aplicación y tanto a nivel de código como de la documentación.

Tabla 2 Definición de Sprints



2.2. Planificación temporal

Empezaremos con la planificación inicial para los sprints previamente definidos en este proyecto, cuya finalidad no es limitar la estructuración de los marcos temporales si no de que sirva como una guía durante el proceso de desarrollo y puntos de referencia claros y concisos.

Con estos puntos, seremos capaces al finalizar el proyecto de hacer una comparación entre ambas planificaciones, tanto la inicial como la real, que nos ayudará sobre todo a evaluar nuestra eficacia en el recorrido utilizado con la metodología de trabajo y conseguir sacar algunas conclusiones finales.

Para evidenciar una representación clara y visual de la planificación, haremos uso de la herramienta Canva para poder diseñar un diagrama de Gantt, el cual se trata de un recurso gráfico perfecto para mostrar las posibles previsiones temporales de las distintas secciones y actividades del proyecto. Por ello, en la ilustración 1, muestro dicho diagrama, donde se observa de manera detallada y ordenada la secuencia y duración de cada tarea dentro del proyecto [6].



Ilustración 1 Diagrama de Gantt Inicial



2.3. Presupuesto

Generalmente, para el desarrollo de una aplicación software, es necesario realizar un cálculo de los costos incurridos. Por ello, elaboraremos un presupuesto que abarcará las siguientes partes, como son los gastos de personal, que se centra primordialmente en el salario del desarrollador y sus respectivas cargas sociales y los gastos de ejecución, que abarcan tanto los materiales utilizados como los gastos indirectos.

Si hacemos una búsqueda profunda en internet sobre cuál es el salario medio para desarrolladores Junior en España, podemos encontrar una media de 25.000€ anuales en portales bastante fiables como puede ser Glassdoor [7].

Para este presupuesto, se ha tomado como referencia un salario mensual de 1.800€. Además, se han considerado las cargas sociales, que incluyen la cotización a la Seguridad Social tanto por parte de la empresa como del trabajador, y otros conceptos como el IRPF, aprendido en asignaturas como Derecho Fiscal [8] y Contabilidad Financiera [9].

2.3.1 Coste Mensual para la Empresa

Concepto	Porcentaje (%)	Cantidad
Total Mensual para la Empresa	-	1.867,60€
Seguridad Social a cargo de la empresa	23,60%	330,40€
Prestaciones por Desempleo	5,50%	77,00€
IT/IMS	3,50%	49,00€
Formación Profesional	0,60%	8,40€
FOGASA	0,20%	2,80€
Total deducciones	-	-467,60€
Salario Bruto Mensual	-	1.400€
IRPF	16,88%	236,32€
Contingencias Comunes	4,70%	65,80€
Desempleo	1,55%	21,70€
Formación Profesional	0,10%	1,40€
Total deducciones	-	-325,22€
Salario Neto Mensual	-	1.074,78€

Tabla 3 Coste Mensual para la Empresa



2.3.2 Gastos de Ejecución

También debemos de tener en cuenta los gastos de ejecución, el cual incluye tanto los costos asociados del material inventariable y otros gastos relacionados con el desarrollo de la aplicación, en nuestro caso se refiere principalmente a elementos como el portátil y el móvil, necesarios para el desarrollo y pruebas de la aplicación. Además de tener en cuenta otros costes como de alquiler, servicios, servidor... y gastos administrativos.

2.3.2.1 Material Inventariable

Descripción	Costo	Vida Útil	Amortización Anual	Base Imponible	I.V.A (21%)
Portátil	1.200€	3	400€	400€	84€
Móvil	300€	2	150€	150€	31.50€

Tabla 4 Material Inventariable

2.3.2.2 Gastos Recurrentes y Directos

Descripción	Costo	Vida Útil (meses)	Base Imponible	I.V.A (21%)
Alquiler	400€	5	2.000€	420€
Servicios (electricidad, internet, agua)	100€	5	500€	105€
Servidor	50€	5	250€	52,50€
Material oficina	50€	-	50€	10,50€
Software y Licencias	500€	-	500€	105€

Tabla 5 Gastos Recurrentes y Directos



2.3.2.3 Gastos Indirectos

En cuanto a los gastos indirectos, únicamente contamos con los gastos administrativos, ya que incluyen gastos principalmente de servicios de contabilidad y auditoría, de gestión de recursos humanos, servicios legales y otros costes, los cuales son fundamentales para el funcionamiento eficiente del proyecto.

Descripción	Estimación	Base Imponible	I.V.A (21%)
Gastos Administrativos	1.000€	1.000€	210€

Tabla 6 Gastos Indirectos

2.3.2.4 Gastos de Personal

Descripción	Uds.	Precio	Meses	Base Imponible	I.V.A (21%)
Sueldo	1	1.867,60€	5	9.338,00€	1960,98€

Tabla 7 Gastos de Personal

2.3.3 Resumen del Presupuesto (5 Meses)

Al final de los 5 meses, obtenemos un presupuesto total de 14.818,50€, incluyendo el IVA. Este total este compuesto por los siguientes elementos de la tabla:

Concepto	Subtotal	I.V.A (21%)	Total
Gastos de Personal	9.338,00€	-	9.338,00€
Material Inventariable	550€	115,50€	665,50€
Gastos Recurrentes	3.250€	693€	3.943€
Gastos Indirectos	1.000€	210€	1.210€
Total	14.138€	1.018,50€	15.156,50€

Tabla 8 Presupuesto final



3. Análisis

3.1. Stakeholders y Usuarios

En la fase inicial de la metodología Scrum, es esencial identificar a las partes interesadas clave de nuestra aplicación financiera. Para lograrlo, comenzaremos por definir detalladamente a las "Personas" [10].

3.1.1. Personas

Lo primero que debemos entender es que es una "Persona", bien esto lo podríamos definir como una persona ficticia pero concreta del grupo de usuarios al que va dirigido un producto. Y cuyo objetivo principal es realizar una compresión clara de los objetivos y necesidades de los usuarios en contextos específicos de uso.

Empezaremos por ir definiendo las distintas "Personas" potenciales para nuestro proyecto:

Nombre	Juan Pérez	
Edad	30 años	
Sexo	Masculino	
Educación	Licenciado en Ingeniería Informática	
Contexto de Uso		
¿Cuándo?	Utiliza la aplicación diariamente.	
¿Dónde?	Principalmente en casa y en su oficina	
Misión		
Objetivo	Utilizar la aplicación para gestionar su cartera de inversiones y tomar decisiones financieras informadas.	
Expectativas	Espera encontrar una interfaz intuitiva y herramientas de análisis bursátil avanzadas.	
Motivación		
Urgencia	Quiere maximizar sus inversiones y necesita acceso rápido a información financiera actualizada.	
Deseo	Aspira a obtener altos rendimientos en sus inversiones.	

Tabla 9 Persona 1



Nombre	Laura Fernández	
Edad	28 años	
Sexo	Femenino	
Educación	Estudiante de economía	
Contexto de Uso		
¿Cuándo?	Utiliza la aplicación principalmente por las tardes.	
¿Dónde?	En su departamento de estudiante y ocasionalmente en la biblioteca de la universidad.	
Misión		
Objetivo	Utiliza la aplicación para aprender sobre inversiones y practicar estrategias virtuales.	
Expectativas	Espera una interfaz educativa y simulaciones realistas del mercado.	
Motivación		
Urgencia	No tiene una urgencia específica, pero está interesada en aprender sobre inversiones.	
Deseo	Quiere mejorar sus habilidades financieras.	

Tabla 10 Persona 2



Nombre	Ana López	
Edad	45 años	
Sexo	Femenino	
Educación	Licenciada en Administración de Empresas	
Contexto de Uso		
¿Cuándo?	Utiliza la aplicación principalmente los fines de semana.	
¿Dónde?	En su casa principalmente.	
Misión		
Objetivo	Utiliza la aplicación para gestionar su cartera de inversiones a largo plazo.	
Expectativas	Espera una interfaz amigable y herramientas de seguimiento de cartera.	
Motivación		
Urgencia	Quiere mantener y hacer crecer su inversión a lo largo del tiempo.	
Deseo	Busca la estabilidad financiera y la jubilación cómoda.	

Tabla 11 Persona 3



3.1.2 Escenarios

Una vez tenemos definidos de manera correcta las “Personas” de nuestra aplicación, es ideal realizar una serie de escenarios para poner en contexto la situación que queremos abordar. Podemos definir un escenario como una descripción de una persona usando un producto para alcanzar un objetivo [11].

Es decir, describe una instancia de uso de un producto o sistema en un contexto determinado y se representa usando una narrativa que cuenta una historia. A continuación, muestro algunos posibles escenarios de nuestro proyecto:

Nombre persona	Juan Pérez
Objetivo persona	Utilizar la aplicación para gestionar su cartera de inversiones y tomar decisiones financieras informadas.
Escenario	
Un día, Juan Perez, que utiliza la aplicación casi a diario desde su casa y a veces desde la oficina, está muy contento porque la interfaz le resulta intuitiva y está muy atento pues quiere maximizar sus inversiones. Juan decide utilizar la función de análisis de la aplicación y agrega una acción a favoritos. Al día siguiente, recibe una alerta sobre la acción en la que está interesado. Despues de analizar la información proporcionada por la aplicación, decide tomar una decisión de inversión.	

Tabla 12 Escenario 1

Nombre persona	Laura Fernández
Objetivo persona	Utilizar la aplicación para aprender sobre inversiones y practicar estrategias virtuales.
Escenario	
Laura tiene 28 años y es estudiante de economía, generalmente utiliza la aplicación por la tardes en su piso y alguna vez en la biblioteca cuando está aburrida, utiliza la aplicación principalmente porque quiere aprender mas sobre inversiones y sobre el mercado financiero. Laura suele utilizar la función de simulación virtual de la aplicación, utilizando el dinero virtual de su cartera. Experimenta con distintas estrategias de inversión y observa como van fluyendo sus inversiones y el rendimiento que tienen, a través de gráficas y datos detallados.	

Tabla 13 Escenario 2



Nombre persona	Ana López
Objetivo persona	Utilizar la aplicación para gestionar su cartera de inversiones a largo plazo.
Escenario	<p>Ana es licenciada en Administración de Empresas y utiliza la aplicación solo los fines de semana en su casa. Ana suele utilizar la aplicación para ver información general de cómo van fluctuando los diferentes sectores del mercado y para vigilar su inversión a lo largo del tiempo.</p> <p>Ana suele observar los índices de las principales empresas y noticias de ultima hora para estar al tanto de sus inversiones realizadas a largo plazo. La aplicación le proporciona información sobre la evolución de su cartera.</p>

Tabla 14 Escenario 3

Nombre persona	Juan Pérez
Objetivo persona	Diversificar su cartera y tomar decisiones financieras informadas utilizando la aplicación.
Escenario	<p>Juan Pérez, ingeniero informático, utiliza la aplicación diariamente en su casa. Juan un día decide entrar en la sección de sus acciones que ha estado comprando y está en posesión en su cartera virtual.</p> <p>Se da cuenta de que solo tiene acciones de dos empresas distintas y que no tiene una cartera bien diversificada, por lo que se propone dar mas variedad a su cartera y comprar acciones de distintas empresas que hayan sido recomendadas por la aplicación. Con la información obtenida, ajusta su cartera de inversiones para maximizar sus rendimientos.</p>

Tabla 15 Escenario 4



Nombre persona	Laura Fernández
Objetivo persona	Aprender sobre inversiones y practicar estrategias virtuales.
Escenario	<p>Laura tiene como objetivo principalmente aprender sobre inversiones y como fluctúa el mercado, y como consecuencia practicar estrategias en simulaciones.</p> <p>Laura accede a la sección de aprendizaje, y empieza a leer artículos introductorios a las finanzas y a las inversiones. Una vez se ha informado y tiene una base consolidada, empieza a practicar en el simulador virtual las distintas estrategias que se le aconsejan y ve como consigue rendimientos positivos. A medida que gana experiencia, Laura se siente más segura en sus conocimientos financieros.</p>

Tabla 16 Escenario 5

Nombre persona	Juan Pérez
Objetivo persona	Mantenerse informado sobre las últimas noticias financieras y recibir notificaciones de eventos importantes en el mercado.
Escenario	<p>Juan quiere utilizar la aplicación para recibir alertas sobre sus acciones favoritas de última hora, en los que hayan surgido cambios significativos y también poder estar al tanto de las últimas noticias financieras.</p> <p>Un día, mientras que trabajaba tranquilamente, le llegan notificaciones a su móvil de que ciertas acciones que tenía guardada como favoritas han tenido una subida porcentual considerable. Juan se mete en la aplicación y se va al apartado de noticias financiera y ve una noticia donde se habla sobre información que aumenta el valor de algunas acciones que tiene en posesión.</p> <p>Y gracias a esta información recibida, Juan puede reaccionar rápidamente y vender las acciones o seguir especulando con su respectiva estrategia de inversión.</p>

Tabla 17 Escenario 6



3.2. Backlogs y Requisitos

3.2.1. Listado inicial de Historias de Usuario

Ya que nos encontramos en una metodología de desarrollo ágil, la cual es característica por seguir un desarrollo iterativo, la lista de historia de usuarios inicial se le fue añadiendo nuevas a medida que surgían en el proceso.

Aquí muestro el listado completo de historias de usuarios, que, según la terminología de Scrum, la podríamos definir como el Product Backlog, junto a su estimación y prioridad. Para la elaboración del Backlog, he utilizado el método de Planning Poker [12], que consiste en ir estimando cada Historia de Usuario con cartas con cierto valor de estimación, que suele ser más efectivo cuando el proyecto va a ser desarrollado por un grupo, pero aun así resulta efectivo.

Y después de ciertas reuniones con mi director del proyecto, y algunas encuestas a posibles usuarios de esta aplicación, he podido concluir con el siguiente listado de historias de usuarios [13]:

Identificador	Historia de Usuario	Estimación	Prioridad
HU.1	Como usuario, quiero registrarme en la aplicación para acceder a sus funciones.	3	1
HU.2	HU SPIKE - Obtener datos en tiempo real de compañías cotizadas desde fuentes confiables.	8	1
HU.3	HU SPIKE - Implementar base de datos en el servidor para almacenar información financiera.	8	1
HU.4	Como usuario, quiero recibir alertas en tiempo real sobre mis inversiones.	5	2
HU.5	Como usuario, quiero crear y gestionar una cartera de valores.	8	2
HU.6	Como usuario, quiero obtener sugerencias de estrategias de inversión personalizadas.	5	3
HU.7	Como usuario, quiero realizar simulaciones de inversiones informadas.	5	3
HU.8	HU SPIKE - Diseñar una interfaz intuitiva y fácil de usar para la aplicación.	8	3
HU.9	Como usuario, quiero analizar datos detallados y gráficos sobre mis inversiones.	6	4
HU.10	Como usuario, quiero acceder a herramientas avanzadas de análisis bursátil.	5	4
HU.11	Como usuario, quiero recibir recomendaciones basadas en mi perfil financiero y objetivos.	5	4
HU.12	Como usuario, quiero poder realizar un seguimiento detallado de mis inversiones a lo largo del tiempo.	3	3
HU.13	HU SPIKE - Implementar seguridad robusta para proteger los datos financieros de los usuarios.	8	1
HU.14	Como usuario, quiero acceder a información actualizada sobre empresas cotizadas en la bolsa.	5	2
HU.15	Como usuario, quiero recibir noticias financieras relevantes para tomar decisiones informadas.	3	3
HU.16	Como usuario, quiero tener acceso a tutoriales	5	2



	interactivos sobre inversión y uso de la aplicación.		
HU.17	Como usuario, quiero establecer metas financieras y recibir recomendaciones para alcanzarlas.	5	3
HU.18	HU SPIKE - Implementar un sistema de notificaciones push para alertas importantes.	8	2
HU.19	Como usuario, quiero tener acceso a un historial completo de mis transacciones e inversiones.	3	4
HU.20	Como usuario, quiero poder compartir información sobre mis inversiones con asesores financieros.	5	4
HU.21	Como usuario, quiero ver la evolución de mi balance a través de gráficas.	5	3
HU.22	Como administrador, quiero gestionar la base de datos desde el backend.	4	2
HU.23	Como usuario, quiero recuperar mi contraseña en caso de olvido para mantener el acceso a mi cuenta.	5	2
HU.24	Como usuario, quiero acceder a un resumen del mercado diario para tener una visión general del comportamiento del mercado.	4	3
HU.25	Como usuario, quiero acceder a una sección de "Mis Acciones" para ver el desempeño de mis inversiones individuales.	3	3
HU.26	Como usuario, quiero tener la opción de configurar mi perfil, incluyendo preferencias de notificación y objetivos financieros.	3	1
HU.27	Como usuario, quiero ver un historial detallado de las alertas recibidas para hacer un seguimiento de la información importante.	5	3

Tabla 18 Historias de Usuario



3.2.2. Velocidad del Equipo

Partimos de un equipo de desarrollo formado por 1 programador que va a dedicar un 70% de su trabajo al proyecto.

La duración de cada una de las iteraciones que vamos a realizar en el proyecto va a ser de 4 semanas. La estimación realizada del esfuerzo de cada una de las historias de usuario se ha expresado en puntos de historia.

En nuestro entorno de trabajo, establecemos que cada punto de historia se va a corresponder con una hora ideal de trabajo, y estimamos que una hora ideal de trabajo se va a corresponder con 1'5 horas reales.

Estimamos que en un día de trabajo real un integrante puede dedicar 4 horas reales al proyecto por lo que en 1 día real se realiza 2 punto de historia por integrante. Además, estimamos que el programador realizará 20 horas ideales de trabajo (unas 40 horas reales) a la semana.

La duración de una iteración va a ser:

$$1 \text{ iteración} = 4 \text{ semanas reales}$$

La velocidad del equipo de desarrollo medido en puntos de historia por iteración es:

$$\text{Velocidad: } = 4 \text{ semanas/Iteración} * 5 \text{ días reales/semana} * 1'5 \text{ PH/día real} = \mathbf{30 \text{ PH/Iteración.}}$$



3.2.3. Descripción de Entregas

Esfuerzo total del proyecto = 30 PH/iteración x 5 iteraciones = 150 PH.

Velocidad del equipo = 30 PH (por iteración)

En base al esfuerzo necesario y la velocidad estimada del equipo, para el desarrollo del proyecto se van a realizar dos entregas de dos iteraciones cada una de ellas. El desarrollo del proyecto va a comenzar el día 12 de marzo del 2024 [14].

El plan de entregas del producto es el siguiente:

Entrega	Objetivo		Fecha de la Entrega
1	Tener una aplicación de asesoramiento financiero bursátil funcional que permita a los usuarios realizar inversiones informadas en el mercado de valores.		24 de Junio del 2024
	Iteración	Objetivo	
	1	Conceptualización - Establecer requisitos, investigar el mercado y definir funcionalidades clave e implementar funcionalidades básicas.	
	2	Diseño e Interfaz de Usuario - Desarrollar la arquitectura y diseñar una interfaz centrada en la usabilidad.	
	3	Implementación Inicial - Desarrollar funcionalidades básicas e integrar la API de Yahoo Finance.	
	4	Ampliación y Mejora - Implementar funcionalidades avanzadas como gestión de cartera y análisis bursátil.	
	5	Pruebas, Ajustes y Lanzamiento - Realizar pruebas, ajustes y preparar la aplicación para su lanzamiento, incluyendo la documentación.	

Tabla 19 Descripción de Entrega



3.2.4. Product Backlog

Una vez tenemos descrito el listado inicial de Historias de Usuario y la descripción de entregas, ya podemos terminar de formular el product Backlog correspondiente [15]:

Identificador	Historia de Usuario	Estimación	Prioridad	Entrega
HU.1	Como usuario, quiero registrarme en la aplicación para acceder a sus funciones.	3	1	1
HU.2	HU SPIKE - Obtener datos en tiempo real de compañías.	8	1	1
HU.3	HU SPIKE - Implementar base de datos en el servidor.	8	1	1
HU.4	Como usuario, quiero recibir alertas en tiempo real sobre mis inversiones.	5	2	1
HU.5	Como usuario, quiero crear y gestionar una cartera de valores.	8	2	1
HU.6	Como usuario, quiero obtener sugerencias de estrategias de inversión.	5	3	1
HU.7	Como usuario, quiero realizar simulaciones de inversiones.	5	3	1
HU.8	HU SPIKE - Diseñar una interfaz intuitiva y fácil de usar.	8	3	1
HU.9	Como usuario, quiero analizar datos detallados y gráficos.	6	4	1
HU.10	Como usuario, quiero acceder a herramientas avanzadas de análisis.	5	4	1
HU.11	Como usuario, quiero recibir recomendaciones basadas en mi perfil financiero y objetivos.	5	4	1
HU.12	Como usuario, quiero poder realizar un seguimiento detallado de mis inversiones.	3	3	1
HU.13	HU SPIKE - Implementar seguridad robusta para proteger los datos financieros de los usuarios.	8	1	1
HU.14	Como usuario, quiero acceder a información actualizada sobre empresas cotizadas en la bolsa.	5	2	1
HU.15	Como usuario, quiero recibir noticias financieras relevantes para tomar decisiones informadas.	3	3	1
HU.16	Como usuario, quiero tener acceso a tutoriales interactivos sobre inversión y uso de la aplicación.	5	2	1
HU.17	Como usuario, quiero establecer metas financieras y recibir recomendaciones para alcanzarlas.	5	3	1
HU.18	HU SPIKE - Implementar un sistema de notificaciones push para alertas importantes.	8	2	1
HU.19	Como usuario, quiero tener acceso a un historial completo de mis transacciones e inversiones.	3	4	1
HU.20	Como usuario, quiero poder compartir información sobre mis inversiones con asesores financieros.	5	4	1
HU.21	Como usuario, quiero ver la evolución de mi balance a través de gráficas.	5	3	1
HU.22	Como administrador, quiero gestionar la base de datos desde el backend.	4	2	1
HU.23	Como usuario, quiero recuperar mi contraseña en caso de olvido para mantener el acceso a mi cuenta.	5	2	1



HU.24	Como usuario, quiero acceder a un resumen del mercado diario para tener una visión general del comportamiento del mercado.	4	3	1
HU.25	Como usuario, quiero acceder a una sección de "Mis Acciones" para ver el desempeño de mis inversiones individuales.	3	3	1
HU.26	Como usuario, quiero tener la opción de configurar mi perfil, incluyendo preferencias de notificación y objetivos financieros.	3	1	1
HU.27	Como usuario, quiero ver un historial detallado de las alertas recibidas para hacer un seguimiento de la información importante.	5	3	1

Tabla 20 Product Backlog



3.2.5. Tarjetas de las Historias de Usuario

Selección de Historias de Usuario del Product Backlog para desarrollar en el próximo sprint, con un plan de entrega y definición de terminado.

Identificador: HU.1	Título: Como usuario, quiero registrarme en la aplicación para acceder a sus funciones.	
Descripción: Cualquier usuario puede registrarse o iniciar sesión independientemente de si es un administrador o no.		
Estimación: 3	Prioridad: 1	Iteración: 1
Tareas relacionadas: <ul style="list-style-type: none">• Diseñar la pantalla de registro.• Implementar la lógica de registro.• Validar datos de registro.• Almacenar información de usuarios.		
Pruebas de aceptación: <ul style="list-style-type: none">• Un usuario puede registrarse con éxito.• Se valida la información de registro.• Los datos del usuario se almacenan correctamente.		
Observaciones: Garantizar una experiencia de registro fluida y segura para los usuarios.		

Tabla 21 Tarjeta HU.1

Identificador: HU.2	Implementar la obtención de datos en tiempo real de compañías cotizadas desde fuentes confiables en la aplicación.	
Descripción: Se enfoca en la integración de datos financieros para obtener y mantener actualizados los datos en tiempo real.		
Estimación: 8	Prioridad: 1	Iteración: 1
Tareas relacionadas: <ul style="list-style-type: none">• Levantar servidor con Flask.• Crear API para llamar a los datos desde la App.• Crear las funciones que llamen a la API desde la App.		
Pruebas de aceptación: <ul style="list-style-type: none">• Los datos de compañías cotizadas se obtienen correctamente en tiempo real.		
Observaciones: Garantizar la integridad de los datos.		

Tabla 22 Tarjeta HU.2



Identificador: HU.3	HU SPIKE - Implementar base de datos en el servidor.	
Descripción: Configurar y preparar la base de datos para el almacenamiento y manejo eficiente de datos en el servidor.		
Estimación: 8	Prioridad: 1	Iteración: 1
Tareas relacionadas:		
<ul style="list-style-type: none"> • Seleccionar un sistema de base de datos adecuado. • Configurar el servidor de base de datos en Flask. • Crear esquemas de base de datos para usuarios e inversiones. • Integrar la base de datos con la aplicación. 		
Pruebas de aceptación:		
<ul style="list-style-type: none"> • La base de datos se conecta y funciona correctamente con el servidor. • Los datos se almacenan y recuperan sin errores. 		
Observaciones: Asegurar que la base de datos sea escalable y segura.		

Tabla 23 Tarjeta HU.3

Identificador: HU.4	Como usuario, quiero recibir alertas en tiempo real sobre mis inversiones.	
Descripción: Permitir que los usuarios reciban notificaciones inmediatas sobre cambios significativos en sus inversiones.		
Estimación: 5	Prioridad: 2	Iteración: 1
Tareas relacionadas:		
<ul style="list-style-type: none"> • Diseñar un sistema de alertas en la aplicación. • Integrar alertas con datos en tiempo real de inversiones. • Configurar notificaciones personalizables. 		
Pruebas de aceptación:		
<ul style="list-style-type: none"> • Las alertas se reciben en tiempo real. • Los usuarios pueden personalizar las condiciones de las alertas. 		
Observaciones: Garantizar que las alertas sean precisas y rápidas.		

Tabla 24 Tarjeta HU.4



Identificador: HU.5	Como usuario, quiero crear y gestionar una cartera de valores.	
Descripción: Facilitar a los usuarios la creación y administración de una cartera de inversión personalizada.		
Estimación: 8	Prioridad: 1	Iteración: 1
Tareas relacionadas:		
<ul style="list-style-type: none"> • Diseñar interfaz para la creación de carteras. • Desarrollar funcionalidades para añadir y quitar valores. • Implementar herramientas para monitorear el rendimiento de la cartera. 		
Pruebas de aceptación:		
<ul style="list-style-type: none"> • Los usuarios pueden crear y modificar sus carteras fácilmente. • Las carteras muestran información actualizada sobre el rendimiento. 		
Observaciones: Priorizar la usabilidad y la presentación clara de la información.		

Tabla 25 Tarjeta HU.5

Identificador: HU.6	Como usuario, quiero obtener sugerencias de estrategias de inversión.	
Descripción: Proveer a los usuarios recomendaciones personalizadas de estrategias de inversión basadas en sus perfiles y preferencias.		
Estimación: 8	Prioridad: 1	Iteración: 1
Tareas relacionadas:		
<ul style="list-style-type: none"> • Desarrollar un algoritmo para generar sugerencias. • Integrar el perfil del usuario en el sistema de recomendaciones. • Crear una interfaz para mostrar sugerencias de inversión. 		
Pruebas de aceptación:		
<ul style="list-style-type: none"> • Los usuarios reciben sugerencias personalizadas. • Las sugerencias son coherentes con los perfiles y preferencias de los usuarios. 		
Observaciones: Ninguna		

Tabla 26 Tarjeta HU.6

Identificador: HU.7	Como usuario, quiero realizar simulaciones de inversiones.	
Descripción: Permitir a los usuarios simular diferentes escenarios de inversión para tomar decisiones informadas.		
Estimación: 8	Prioridad: 1	Iteración: 1
Tareas relacionadas:		
<ul style="list-style-type: none"> • Implementar una herramienta de simulación de inversiones. • Integrar datos históricos y actuales del mercado en la herramienta. • Desarrollar funcionalidades para ajustar parámetros de la simulación. 		
Pruebas de aceptación:		
<ul style="list-style-type: none"> • Las simulaciones reflejan de manera realista los resultados potenciales. • Los usuarios pueden modificar fácilmente los parámetros de las simulaciones. 		
Observaciones: Las simulaciones deben ser intuitivas y proporcionar información valiosa.		

Tabla 27 Tarjeta HU.7



Identificador: HU.8	HU SPIKE - Diseñar una interfaz intuitiva y fácil de usar.	
Descripción: Desarrollar una interfaz de usuario que sea simple, atractiva y fácil de navegar para mejorar la experiencia del usuario.		
Estimación: 8	Prioridad: 1	Iteración: 1
Tareas relacionadas:		
<ul style="list-style-type: none"> Realizar estudios de usuario para entender sus necesidades. Diseñar la interfaz enfocándose en la usabilidad. Probar la interfaz con usuarios reales y ajustar según el feedback. 		
Pruebas de aceptación:		
<ul style="list-style-type: none"> Los usuarios encuentran la interfaz atractiva y fácil de usar. La navegación por la app es intuitiva. 		
Observaciones: Ninguna		

Tabla 28 Tarjeta HU.8

Identificador: HU.9	Como usuario, quiero analizar datos detallados y gráficos.	
Descripción: Ofrecer a los usuarios herramientas avanzadas para analizar datos financieros y visualizarlos en gráficos detallados.		
Estimación: 6	Prioridad: 2	Iteración: 1
Tareas relacionadas:		
<ul style="list-style-type: none"> Desarrollar componentes de visualización de datos en la app. Integrar análisis de datos financieros en tiempo real. Crear opciones personalizables para los gráficos. 		
Pruebas de aceptación:		
<ul style="list-style-type: none"> Los gráficos muestran datos precisos y actualizados. Los usuarios pueden personalizar los gráficos según sus necesidades. 		
Observaciones: Ninguna		

Tabla 29 Tarjeta HU.9

Identificador: HU.10	Como usuario, quiero acceder a herramientas avanzadas de análisis.	
Descripción: Proporcionar a los usuarios acceso a herramientas de análisis financiero avanzadas para una toma de decisiones más informada.		
Estimación: 5	Prioridad: 3	Iteración: 1
Tareas relacionadas:		
<ul style="list-style-type: none"> Identificar y seleccionar herramientas de análisis financiero relevantes. Integrar estas herramientas en la aplicación. Capacitar a los usuarios en el uso de estas herramientas. 		
Pruebas de aceptación:		
<ul style="list-style-type: none"> Las herramientas son accesibles y funcionales dentro de la app. Los usuarios pueden utilizar las herramientas sin dificultades. 		
Observaciones: Ninguna		

Tabla 30 Tarjeta HU.10



Identificador: HU.11	Como usuario, quiero recibir recomendaciones basadas en mi perfil financiero y objetivos.	
Descripción: Ofrecer recomendaciones de inversión personalizadas que se ajusten a las características y objetivos financieros de cada usuario.		
Estimación: 5	Prioridad: 2	Iteración: 1
Tareas relacionadas:		
<ul style="list-style-type: none"> • Desarrollar un sistema para evaluar el perfil financiero del usuario. • Integrar un motor de recomendaciones basado en perfiles y objetivos. • Crear una interfaz para que los usuarios puedan ingresar y ajustar sus objetivos financieros. 		
Pruebas de aceptación:		
<ul style="list-style-type: none"> • Las recomendaciones se alinean con los perfiles y objetivos de los usuarios. • Los usuarios pueden modificar sus objetivos y ver cambios en las recomendaciones. 		
Observaciones: Ninguna		

Tabla 31 Tarjeta HU.11

Identificador: HU.12	Como usuario, quiero poder realizar un seguimiento detallado de mis inversiones.	
Descripción: Brindar a los usuarios herramientas para monitorear y analizar en detalle el rendimiento de sus inversiones.		
Estimación: 3	Prioridad: 3	Iteración: 1
Tareas relacionadas:		
<ul style="list-style-type: none"> • Implementar una función de seguimiento de inversiones en la aplicación. • Permitir a los usuarios visualizar el rendimiento de sus inversiones a lo largo del tiempo. 		
Pruebas de aceptación:		
<ul style="list-style-type: none"> • Los usuarios pueden acceder fácilmente a información detallada sobre sus inversiones. • Los datos mostrados son precisos y actualizados. 		
Observaciones: Ninguna		

Tabla 32 Tarjeta HU.12

Identificador: HU.13	HU SPIKE - Implementar seguridad robusta para proteger los datos financieros de los usuarios.	
Descripción: Asegurar que todos los datos financieros y personales de los usuarios estén protegidos mediante medidas de seguridad avanzadas.		
Estimación: 8	Prioridad: 1	Iteración: 1
Tareas relacionadas:		
<ul style="list-style-type: none"> • Establecer protocolos de seguridad para la base de datos y la transmisión de datos. • Implementar autenticación y encriptación de datos. 		
Pruebas de aceptación:		
<ul style="list-style-type: none"> • Los datos de los usuarios están seguros y protegidos. • Se superan las auditorías de seguridad sin incidencias. 		
Observaciones: Ninguna		

Tabla 33 Tarjeta HU.13



Identificador: HU.14	Como usuario, quiero acceder a información actualizada sobre empresas cotizadas en la bolsa.	
Descripción: Proveer a los usuarios información actualizada y detallada sobre empresas cotizadas, incluyendo precios de acciones, noticias y análisis.		
Estimación: 8	Prioridad: 2	Iteración: 1
Tareas relacionadas:		
<ul style="list-style-type: none"> • Integrar fuentes de datos financieros en tiempo real. • Desarrollar secciones en la app para mostrar información de empresas. • Implementar herramientas de búsqueda y filtros para facilitar el acceso a la información. 		
Pruebas de aceptación:		
<ul style="list-style-type: none"> • La información de empresas cotizadas es actual y precisa. • Los usuarios pueden encontrar y entender fácilmente la información. 		
Observaciones: Ninguna		

Tabla 34 Tarjeta HU.14

Identificador: HU.15	Como usuario, quiero recibir noticias financieras relevantes para tomar decisiones informadas.	
Descripción: Proporcionar a los usuarios acceso a noticias financieras actuales y relevantes para ayudarles a tomar decisiones de inversión informadas.		
Estimación: 3	Prioridad: 3	Iteración: 1
Tareas relacionadas:		
<ul style="list-style-type: none"> • Integrar un servicio de noticias financieras en la aplicación. • Permitir a los usuarios personalizar el tipo de noticias que reciben. • Crear notificaciones para noticias importantes. 		
Pruebas de aceptación:		
<ul style="list-style-type: none"> • Las noticias son relevantes y actualizadas. 		
Observaciones: Ninguna		

Tabla 35 Tarjeta HU.15

Identificador: HU.16	Como usuario, quiero tener acceso a tutoriales interactivos sobre inversión y uso de la aplicación.	
Descripción: Proporcionar a los usuarios tutoriales interactivos y educativos que les enseñen sobre conceptos de inversión y cómo usar eficientemente la aplicación.		
Estimación: 5	Prioridad: 2	Iteración: 1
Tareas relacionadas:		
<ul style="list-style-type: none"> • Identificar los temas clave que deben cubrirse en los tutoriales. • Diseñar y desarrollar tutoriales interactivos en la aplicación. • Integrar ejemplos prácticos y simulaciones en los tutoriales. 		
Pruebas de aceptación:		
<ul style="list-style-type: none"> • Los tutoriales son fáciles de seguir y entender para los usuarios. • Los usuarios pueden aplicar lo aprendido en la aplicación de manera efectiva. 		
Observaciones: Ninguna		

Tabla 36 Tarjeta HU.16



Identificador: HU.17	Como usuario, quiero establecer metas financieras y recibir recomendaciones para alcanzarlas.	
Descripción: Permitir a los usuarios establecer metas financieras personales y ofrecer recomendaciones para ayudarles a alcanzar dichas metas.		
Estimación: 5	Prioridad: 3	Iteración: 1
Tareas relacionadas:		
<ul style="list-style-type: none"> Desarrollar una funcionalidad para que los usuarios definan sus metas financieras. Integrar un sistema de recomendaciones basado en las metas establecidas. Proveer seguimiento y actualizaciones sobre el progreso hacia las metas. 		
Pruebas de aceptación:		
<ul style="list-style-type: none"> Los usuarios pueden establecer y modificar sus metas financieras. Las recomendaciones son coherentes con las metas de los usuarios. 		
Observaciones: Ninguna		

Tabla 37 Tarjeta HU.17

Identificador: HU.18	HU SPIKE - Implementar un sistema de notificaciones push para alertas importantes.	
Descripción: Desarrollar un sistema de notificaciones push para enviar alertas importantes y actualizaciones a los usuarios de forma proactiva.		
Estimación: 8	Prioridad: 2	Iteración: 1
Tareas relacionadas:		
<ul style="list-style-type: none"> Diseñar el sistema de notificaciones push. Integrar las notificaciones con eventos relevantes de la aplicación. 		
Pruebas de aceptación:		
<ul style="list-style-type: none"> Las notificaciones push se entregan de manera oportuna. Los usuarios reciben alertas relevantes según sus preferencias. 		
Observaciones: Ninguna		

Tabla 38 Tarjeta HU.18

Identificador: HU.19	Como usuario, quiero tener acceso a un historial completo de mis transacciones e inversiones.	
Descripción: Proveer a los usuarios un historial detallado de todas sus transacciones e inversiones realizadas a través de la aplicación.		
Estimación: 3	Prioridad: 4	Iteración: 1
Tareas relacionadas:		
<ul style="list-style-type: none"> Implementar una función de historial en la aplicación. Asegurar que el historial muestre información detallada y precisa. Ofrecer opciones de búsqueda y filtrado en el historial. 		
Pruebas de aceptación:		
<ul style="list-style-type: none"> El historial es completo y fácil de entender. Los usuarios pueden buscar y filtrar su historial de transacciones. 		
Observaciones: Ninguna		

Tabla 39 Tarjeta HU.19



Identificador: HU.20	Como usuario, quiero poder compartir información sobre mis inversiones con asesores financieros.	
Descripción: Habilitar una funcionalidad que permita a los usuarios compartir de forma segura detalles de sus inversiones con asesores financieros para obtener asesoramiento.		
Estimación: 3	Prioridad: 4	Iteración: 1
Tareas relacionadas: <ul style="list-style-type: none"> • Crear una opción de compartir en la aplicación. • Establecer medidas de seguridad para la compartición de información. 		
Pruebas de aceptación: <ul style="list-style-type: none"> • Los usuarios pueden compartir información de manera segura y selectiva. • Los asesores financieros reciben la información compartida correctamente. 		
Observaciones: Ninguna		

Tabla 40 Tarjeta HU.20

Identificador: HU.21	Como usuario, quiero ver la evolución de mi balance a través de gráficas.	
Descripción: Permitir a los usuarios visualizar gráficamente la evolución de su balance en un periodo de tiempo seleccionado para facilitar la comprensión de su progreso financiero.		
Estimación: 5	Prioridad: 4	Iteración: 1
Tareas relacionadas: <ul style="list-style-type: none"> • Diseñar e implementar gráficas interactivas que muestren la evolución del balance. • Integrar datos financieros reales con el sistema de gráficas. • Permitir a los usuarios seleccionar diferentes periodos de tiempo para la visualización. 		
Pruebas de aceptación: <ul style="list-style-type: none"> • Las gráficas deben reflejar con precisión la evolución del balance del usuario. • Los usuarios deben poder cambiar el periodo de tiempo de la visualización sin problemas. 		
Observaciones: Asegurarse de que las gráficas sean accesibles y comprensibles para usuarios con diferentes niveles de experiencia en finanzas.		

Tabla 41 Tarjeta HU.21



Identificador: HU.22	Como administrador, quiero gestionar la base de datos desde el backend.	
Descripción: Facilitar al administrador la gestión de la base de datos de la aplicación para asegurar el correcto acceso y funcionalidades según el nivel de usuario.		
Estimación: 4	Prioridad: 2	Iteración: 1
Tareas relacionadas:		
<ul style="list-style-type: none"> • Implementar un panel de administración en el backend para la gestión de las diferentes tablas y relaciones. • Desarrollar funcionalidades para añadir, eliminar y modificar los elementos de las tablas. 		
Pruebas de aceptación:		
<ul style="list-style-type: none"> • El administrador puede fácilmente añadir, eliminar y modificar los elementos de la base de datos. 		
Observaciones: Considerar la seguridad de los datos de usuarios al implementar las funcionalidades de gestión.		

Tabla 42 Tarjeta HU.22

Identificador: HU.23	Como usuario, quiero recuperar mi contraseña en caso de olvido para mantener el acceso a mi cuenta.	
Descripción: Permitir a los usuarios recuperar su contraseña a través de un proceso seguro que implica la verificación de su correo electrónico.		
Estimación: 5	Prioridad: 2	Iteración: 1
Tareas relacionadas:		
<ul style="list-style-type: none"> • Implementar un sistema de recuperación de contraseña. • Enviar un enlace de restablecimiento de contraseña al correo electrónico del usuario. • Diseñar la pantalla de restablecimiento de contraseña. • Asegurar el proceso de recuperación de contraseña. 		
Pruebas de aceptación:		
<ul style="list-style-type: none"> • El usuario recibe un correo con el enlace de restablecimiento de contraseña. • El usuario puede restablecer su contraseña a través del enlace enviado. • El sistema verifica la seguridad del proceso de restablecimiento. 		
Observaciones: Es crucial garantizar la seguridad en todo el proceso de recuperación de contraseña para proteger la información del usuario.		

Tabla 43 Tarjeta HU.23



Identificador: HU.24	Como usuario, quiero acceder a un resumen del mercado diario para tener una visión general del comportamiento del mercado.	
Descripción: Ofrecer a los usuarios un resumen diario del mercado que incluya información relevante sobre el comportamiento de los principales índices y acciones.		
Estimación: 4	Prioridad: 3	Iteración: 1
Tareas relacionadas:		
<ul style="list-style-type: none"> • Diseñar la interfaz del resumen del mercado. • Integrar fuentes de datos financieros para el resumen diario. • Desarrollar la lógica para actualizar el resumen cada día. • Implementar visualizaciones de datos para facilitar la comprensión. 		
Pruebas de aceptación:		
<ul style="list-style-type: none"> • El usuario puede visualizar el resumen del mercado diariamente. • Los datos mostrados son actualizados y precisos. • La interfaz es intuitiva y fácil de entender. 		
Observaciones: Ninguna.		

Tabla 44 Tarjeta HU.24

Identificador: HU.25	Como usuario, quiero acceder a una sección de "Mis Acciones" para ver el desempeño de mis inversiones individuales.	
Descripción: Habilitar una sección donde los usuarios puedan ver y gestionar el desempeño de sus acciones individuales, incluyendo ganancias, pérdidas y rendimientos.		
Estimación: 3	Prioridad: 3	Iteración: 1
Tareas relacionadas:		
<ul style="list-style-type: none"> • Crear la pantalla de "Mis Acciones" con una lista de las inversiones del usuario. • Desarrollar funcionalidades para añadir, eliminar y gestionar acciones. • Integrar con APIs financieras para obtener datos en tiempo real de las acciones. • Implementar gráficos y estadísticas para cada acción. 		
Pruebas de aceptación:		
<ul style="list-style-type: none"> • El usuario puede gestionar su portafolio de acciones fácilmente. • Los datos de las acciones se actualizan en tiempo real. • Las visualizaciones gráficas ayudan al usuario a entender el desempeño de sus inversiones. 		
Observaciones: Ninguna.		

Tabla 45 Tarjeta HU.25



Identificador: HU.26	Como usuario, quiero tener la opción de configurar mi perfil, incluyendo preferencias de notificación y objetivos financieros.	
Descripción: Permitir a los usuarios personalizar su experiencia en la aplicación a través de la configuración de su perfil, preferencias y objetivos financieros.		
Estimación: 3	Prioridad: 1	Iteración: 1
Tareas relacionadas:		
<ul style="list-style-type: none"> • Diseñar la interfaz de configuración del perfil de usuario. • Implementar la funcionalidad para editar preferencias de notificación. • Desarrollar la opción para que los usuarios establezcan y editen sus objetivos financieros. • Asegurar que las configuraciones del perfil se guarden y apliquen correctamente. 		
Pruebas de aceptación:		
<ul style="list-style-type: none"> • El usuario puede modificar su perfil y preferencias sin problemas. • Los cambios se reflejan inmediatamente y son persistentes. • La personalización mejora la experiencia del usuario en la aplicación. 		
Observaciones: Ninguna.		

Tabla 46 Tarjeta HU.26



3.2.5 Sprints Backlogs

Una vez tenemos todo definido correctamente, podemos asignar las Historias de Usuario en sprints, por tanto, según la velocidad y la estimación de las HU, lo desglosaremos en 5 sprints.

Cada sprint tendrá una duración de 4 semanas, al final de las cuales se llevarán a cabo las reuniones de revisión del sprint (Sprint Review) [16], así como la planificación del siguiente sprint (Sprint Planning) [17]. Es importante señalar que, al concluir la tercera iteración, se dedicará a la retrospectiva del sprint (Sprint Retrospective) [18].

Por tanto, mostrare como se abordarán las HU en cada sprint, realizando primero las de mayor prioridad, estimando que cada sprint tenga un total de 30 PH (Puntos de Historia), aunque puede variar ligeramente debido a las posibles dependencias entre las HU [19].

Sprint 1		Inicio: 12/02/2024
Identificador	Título de la historia de usuario	Estimación
HU.1	Como usuario, quiero registrarme en la aplicación para acceder a sus funciones.	3
HU.2	HU SPIKE - Obtener datos en tiempo real de compañías.	8
HU.3	HU SPIKE - Implementar base de datos en el servidor.	8
HU.13	HU SPIKE - Implementar seguridad robusta para proteger los datos financieros de los usuarios.	8
HU.26	Como usuario, quiero tener la opción de configurar mi perfil, incluyendo preferencias de notificación y objetivos financieros.	3

Tabla 47 Sprint Backlog 1

Sprint 2		Inicio: 12/03/2024
Identificador	Título de la historia de usuario	Estimación
HU.22	Como administrador, quiero gestionar la base de datos desde el backend.	4
HU.5	Como usuario, quiero crear y gestionar una cartera de valores.	8
HU.14	Como usuario, quiero acceder a información actualizada sobre empresas cotizadas en la bolsa.	5
HU.16	Como usuario, quiero tener acceso a tutoriales interactivos sobre inversión y uso de la aplicación.	5
HU.23	Como usuario, quiero recuperar mi contraseña en caso de olvido para mantener el acceso a mi cuenta.	5

Tabla 48 Sprint Backlog 2



Sprint 3		Inicio: 12/04/2024
Identificador	Título de la historia de usuario	Estimación
HU.6	Como usuario, quiero obtener sugerencias de estrategias de inversión.	5
HU.7	Como usuario, quiero realizar simulaciones de inversiones.	5
HU.8	HU SPIKE - Diseñar una interfaz intuitiva y fácil de usar.	8
HU.15	Como usuario, quiero recibir noticias financieras relevantes para tomar decisiones informadas.	3
HU.17	Como usuario, quiero establecer metas financieras y recibir recomendaciones para alcanzarlas.	5
HU.24	Como usuario, quiero acceder a un resumen del mercado diario para tener una visión general del comportamiento del mercado.	4

Tabla 49 Sprint Backlog 3

Sprint 4		Inicio: 12/05/2024
Identificador	Título de la historia de usuario	Estimación
HU.9	Como usuario, quiero analizar datos detallados y gráficos.	6
HU.10	Como usuario, quiero acceder a herramientas avanzadas de análisis	5
HU.11	Como usuario, quiero recibir recomendaciones basadas en mi perfil financiero y objetivos.	5
HU.19	Como usuario, quiero tener acceso a un historial completo de mis transacciones e inversiones.	3
HU.21	Como usuario, quiero ver la evolución de mi balance a través de gráficas.	5
HU.25	Como usuario, quiero acceder a una sección de "Mis Acciones" para ver el desempeño de mis inversiones individuales.	3

Tabla 50 Sprint Backlog 4

Sprint 5		Inicio: 12/06/2024
Identificador	Título de la historia de usuario	Estimación
HU.12	Como usuario, quiero poder realizar un seguimiento detallado de mis inversiones a lo largo del tiempo.	3
HU.18	HU SPIKE - Implementar un sistema de notificaciones push para alertas importantes.	8
HU.20	Como usuario, quiero poder compartir información sobre mis inversiones con asesores financieros.	5
HU.4	Como usuario, quiero recibir alertas en tiempo real sobre mis inversiones.	5
HU.27	Como usuario, quiero ver un historial detallado de las alertas recibidas para hacer un seguimiento de la información importante.	5

Tabla 51 Sprint Backlog 5



3.3. Diagramas de secuencia

Ya lo último que nos toca hacer en cuanto a la parte del análisis, es diseñar los diagramas de secuencia, estos son herramientas visuales derivadas de UML que ilustran cómo diferentes objetos en un sistema interactúan entre sí buscando describir el “qué” ocurre.

Estos diagramas destacan por su habilidad para demostrar la secuencia temporal de los mensajes, organizados de arriba hacia abajo en el eje vertical, permitiendo así una interpretación clara de la secuencia de eventos.

Los ejemplos presentados a continuación buscan ilustrar cómo se visualiza la interacción entre componentes del sistema en diferentes escenarios, basándose en uno o más casos de uso específicos.

3.3.1 Registrar un usuario

En primer lugar, tenemos la funcionalidad básica de casi cualquier aplicación o sitio web, el registro de un usuario en el sistema, donde en primer lugar el usuario envía los datos del respectivo formulario al controlador, el cual comprueba el correo del usuario en el sistema.

Si no lo encuentra quiere decir que no está registrado en el sistema y por tanto se crea el usuario con los datos recibidos, pero si no se devuelve un error comentando que el usuario ya existe.

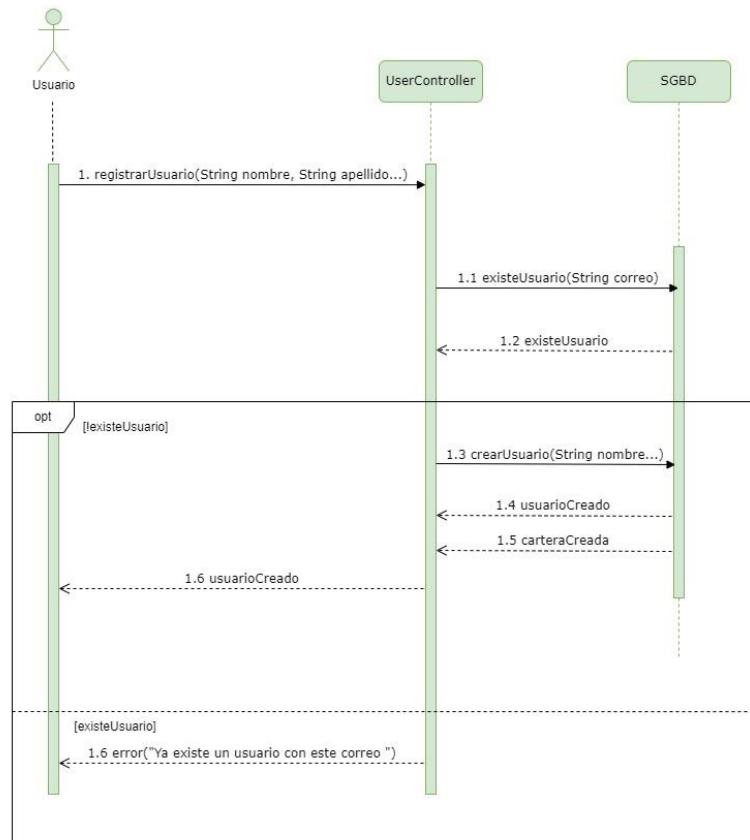


Ilustración 2 Diagrama de secuencia de registro de usuario



3.3.2 Inicio de sesión de un usuario

Por otro lado, tenemos el correspondiente inicio de sesión, una vez que el usuario ha sido registrado en el sistema. El primer paso que hace el usuario es introducir su correo y la contraseña en la aplicación, donde el controlador llama al sistema para que compruebe la base de datos para ver si el correo introducido existe en el sistema y de ser así, comprueba la contraseña.

Si todos los datos introducidos están correctos, el sistema devuelve un booleano a “true” con el nombre de “success” y en caso contrario, devolverá false y también adjuntará un mensaje notificando del fallo correspondiente.

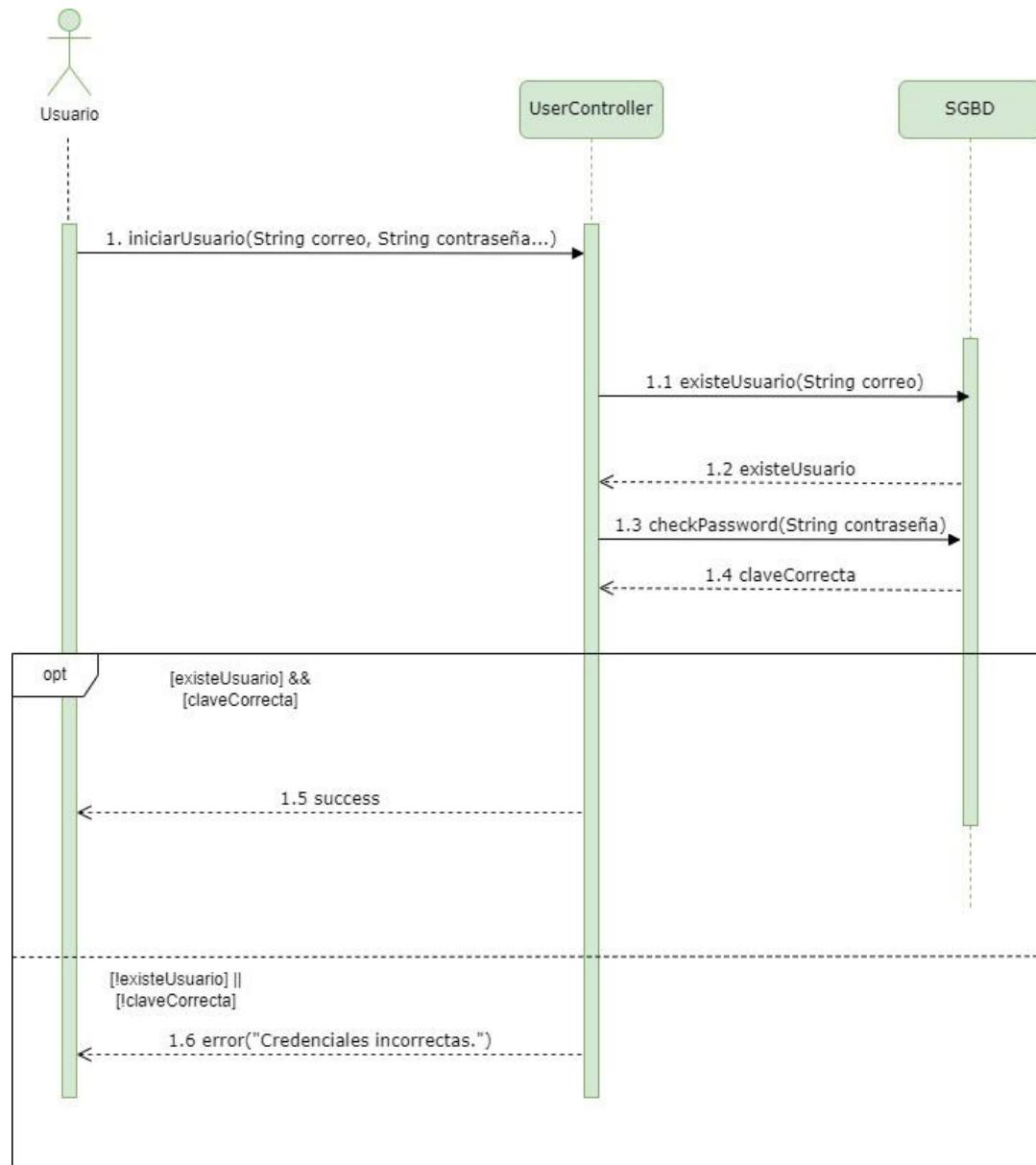


Ilustración 3 Diagrama de secuencia de inicio de sesión de un usuario.



3.3.3 Recuperar la contraseña

Para cuando un usuario olvida su contraseña, esta implementado una funcionalidad donde el usuario puede introducir su correo en la aplicación y el respectivo controlador se encargará de conectarse con el sistema y comprobar que el correo existe.

En caso de afirmativo, el sistema a través de FlaskMail enviará un correo de 6 dígitos al usuario, el cual deberá de introducir en la aplicación con su nueva contraseña, si todos los campos han sido introducidos correctamente, se estable la nueva contraseña y el sistema devuelve el resultado de la operación a la vez que se actualiza la base de datos.

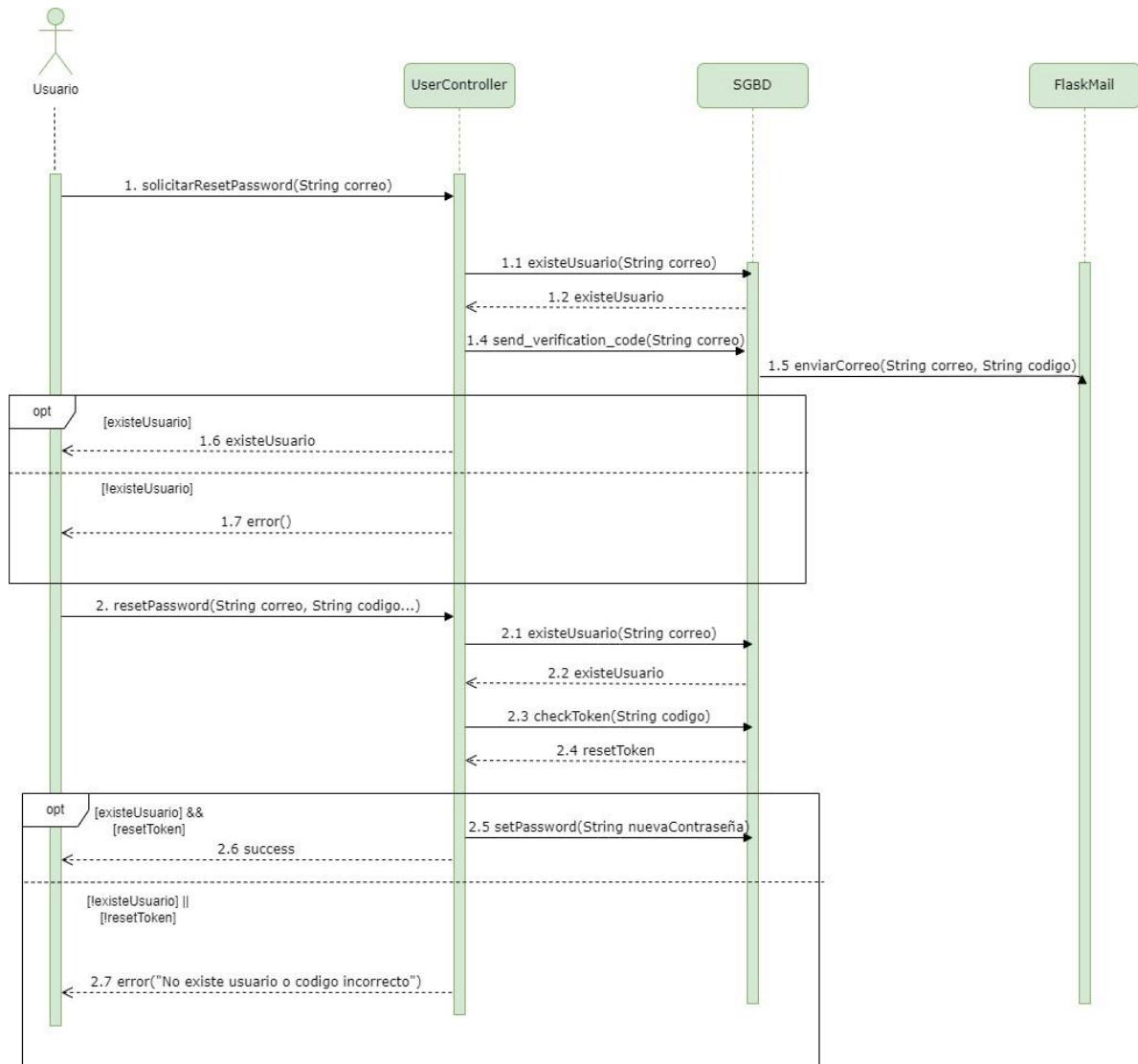


Ilustración 4 Diagrama de secuencia de recuperación de contraseña.



3.3.4 Obtener datos de las acciones

En cuanto a la manera de obtener datos sobre las acciones, ya sea para distintas causas en la aplicación, como, por ejemplo, los datos históricos, los indicadores económicos o fundamentales, la aplicación a través de su controlador llama al sistema para procesar las solicitudes. Sin embargo, en este caso nos vamos a centrar en obtener la lista de acciones con sus respectivos datos relevantes para mostrar.

Donde en cada iteración del bucle, el controlador hace una llamada a la API de YahooFinance para recuperar los datos de una acción en específico. Para cada una de ellas, el controlador recoge la información relevante y los añade a una lista dedicada a almacenar los datos, este proceso se repite hasta que se hayan procesado todas las acciones.

En caso de que se genere algún error durante la recogida de datos sobre una acción, el sistema registra la notificación de error para informar al usuario donde se encuentra y que se puedan realizar las correcciones pertinentes. Sin embargo, si no hay ningún problema y se completado el bucle, el controlador devuelve al usuario la lista completa acciones con su información correspondiente.

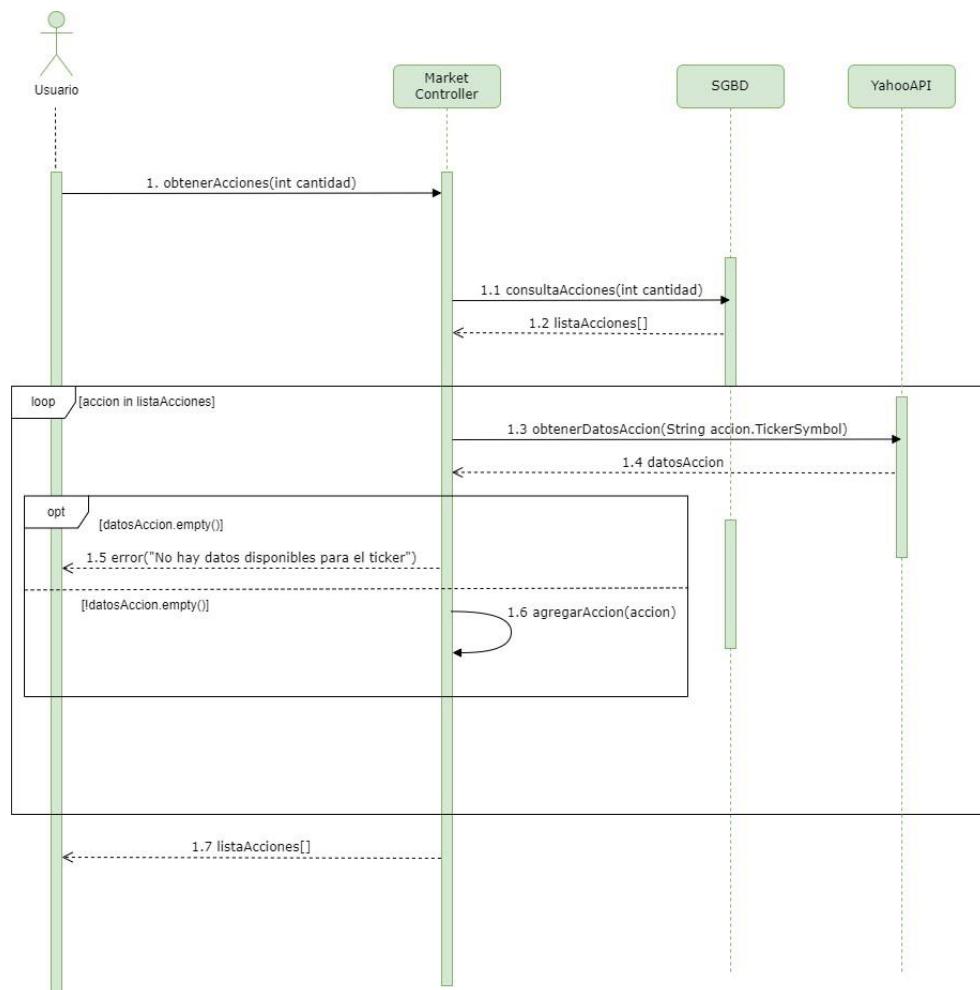


Ilustración 5 Diagrama de secuencia de obtención de datos



3.3.5 Comprar o vender una acción

El proceso que se realiza cuando un usuario compra o vende unas acciones, empieza con introducir un número válido de acciones y se hace una llamada al sistema donde se verifica la existencia del usuario y de su cartera en la base de datos.

Si todo procede correctamente, si hace una serie de comprobaciones financieras en la transacción, por ejemplo, si hay suficiente saldo en la cartera para realizar la operación y si se dispone de las suficientes acciones disponibles.

Si se cumplen las condiciones, se realiza la transacción, se actualiza la base de datos y se devuelve al usuario un mensaje sobre el resultado de la operación y el estado actualizado de su cartera.

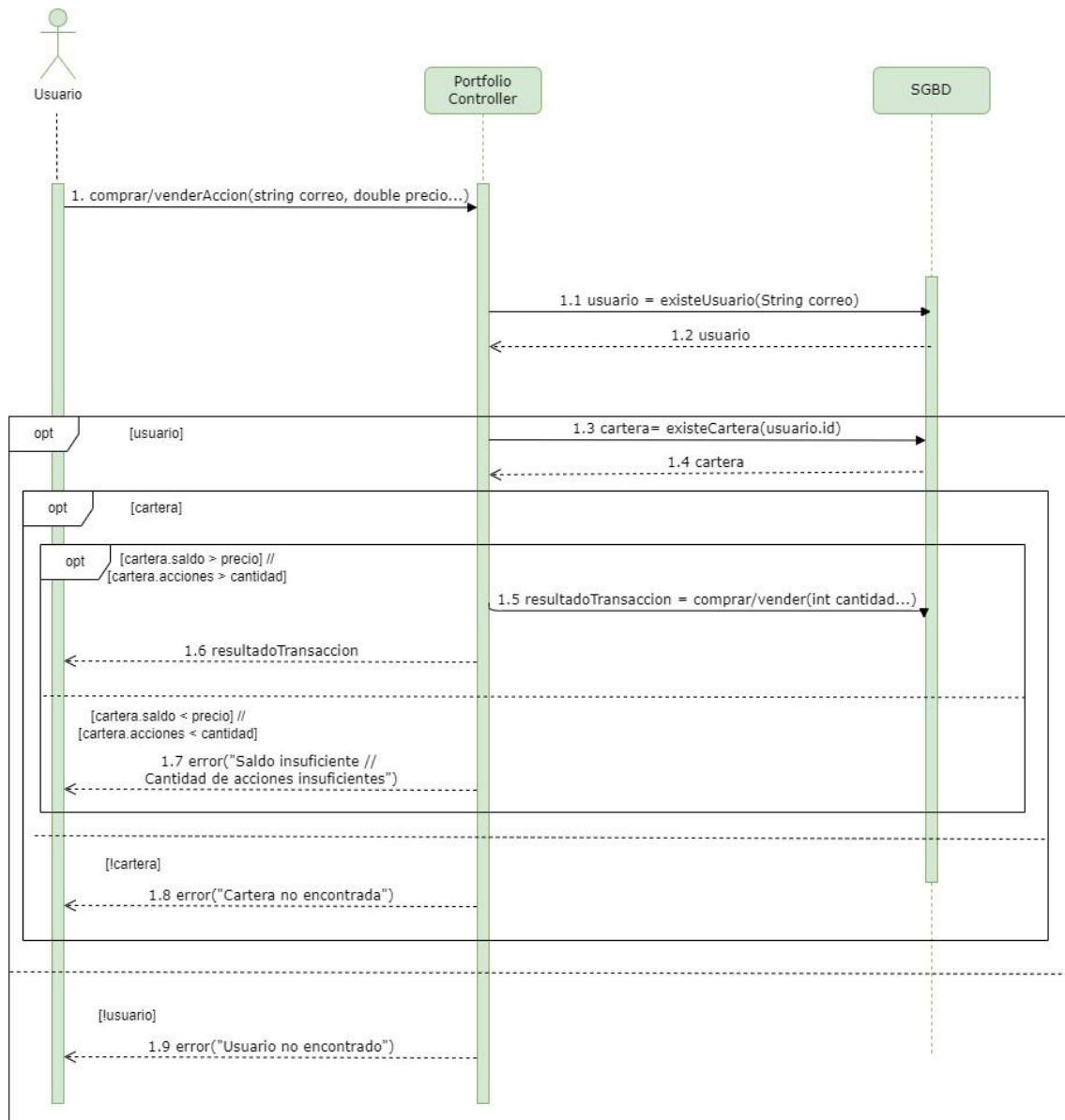


Ilustración 6 Diagrama de secuencia de comprar o vender una acción.



3.3.6 Obtener noticias

Cuando un usuario solicita noticias de una temática específica, el controlador correspondiente inicia una sesión con la API de noticias, enviando los criterios de búsqueda del usuario.

Si la respuesta de la API tiene un código de estado 200, significa que la solicitud fue exitosa y el controlador procesa y devuelve las noticias relevantes al usuario. En caso contrario, si la respuesta no es exitosa, el controlador devuelve un mensaje de error, informando al usuario que no se pudieron obtener las noticias solicitadas.

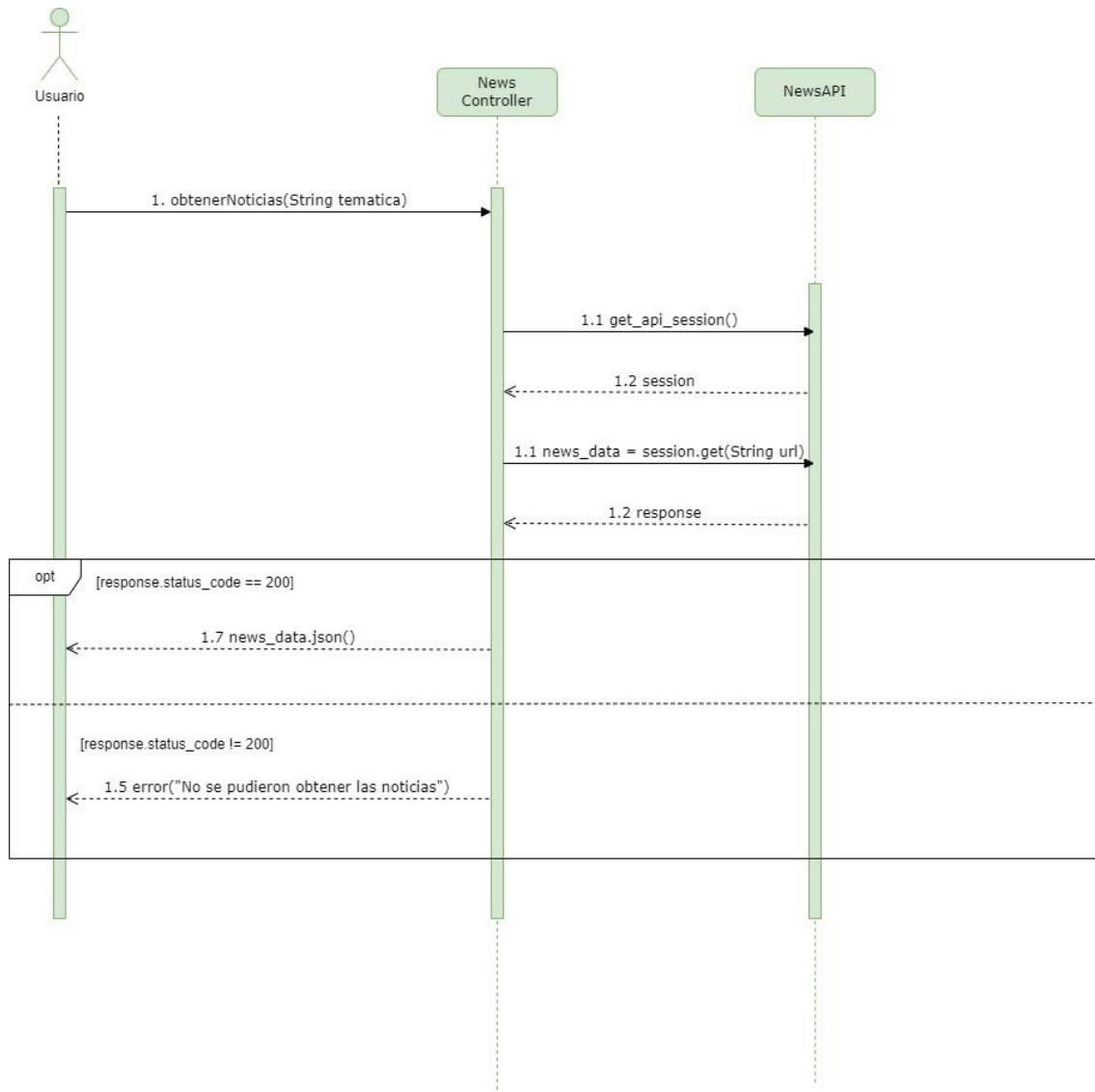


Ilustración 7 Diagrama de secuencia de obtención de noticias



3.3.7 Recibir notificaciones

El sistema de notificaciones funciona a través de una serie de interacciones entre el usuario, un servicio de planificación de tareas (Scheduler), una base de datos (SGBD) y un servicio de comunicación en tiempo real (SocketService).

Cuando un usuario se suscribe a las notificaciones, el Scheduler periódicamente verifica cambios significativos, como la fluctuación en los precios de las acciones favoritas.

Si se detecta un cambio relevante, el Scheduler consulta al SGBD para obtener los detalles del usuario y las acciones implicadas, y luego procede a crear una nueva notificación en la base de datos. Paralelamente, emite una alerta a través del SocketService, que notifica al usuario en tiempo real sobre el evento de interés.

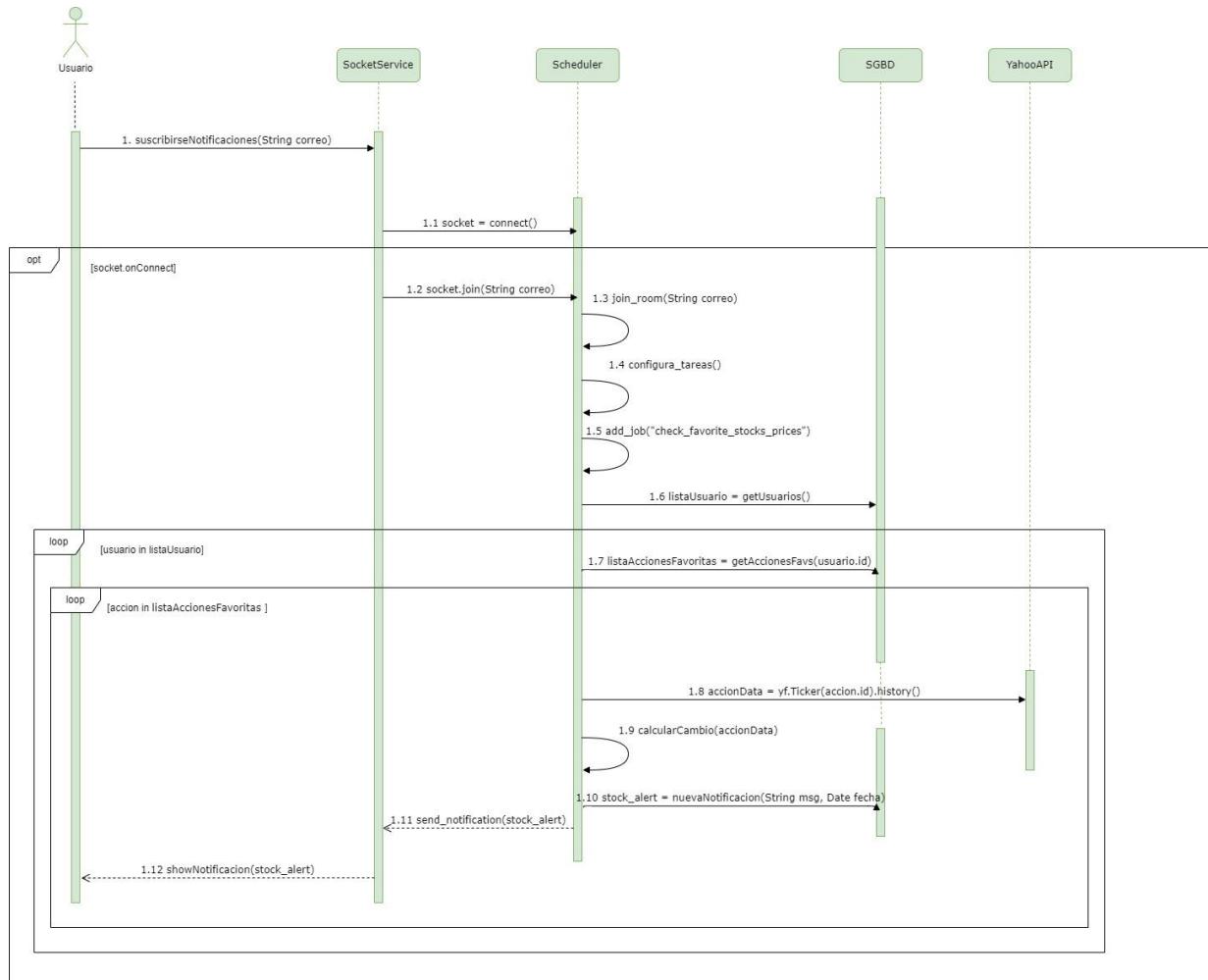


Ilustración 8 Diagrama de secuencia de recibir notificaciones.



3.3.8 Editar perfil de un usuario

Si un usuario quiere modificar algún campo de su perfil, debe completar o editar unos formularios con la información deseada y enviarlo al sistema. El controlador al recibir estos datos, consulta al sistema de gestión de base de datos (SGBD) si el usuario existe.

Una vez se consigue pasar esta condición con éxito, el sistema procede a actualizar la información del perfil del usuario con los nuevos datos proporcionados. Una vez termina la operación se le informa al usuario del éxito o fracaso de la operación, con su respectivo mensaje informando de la situación.

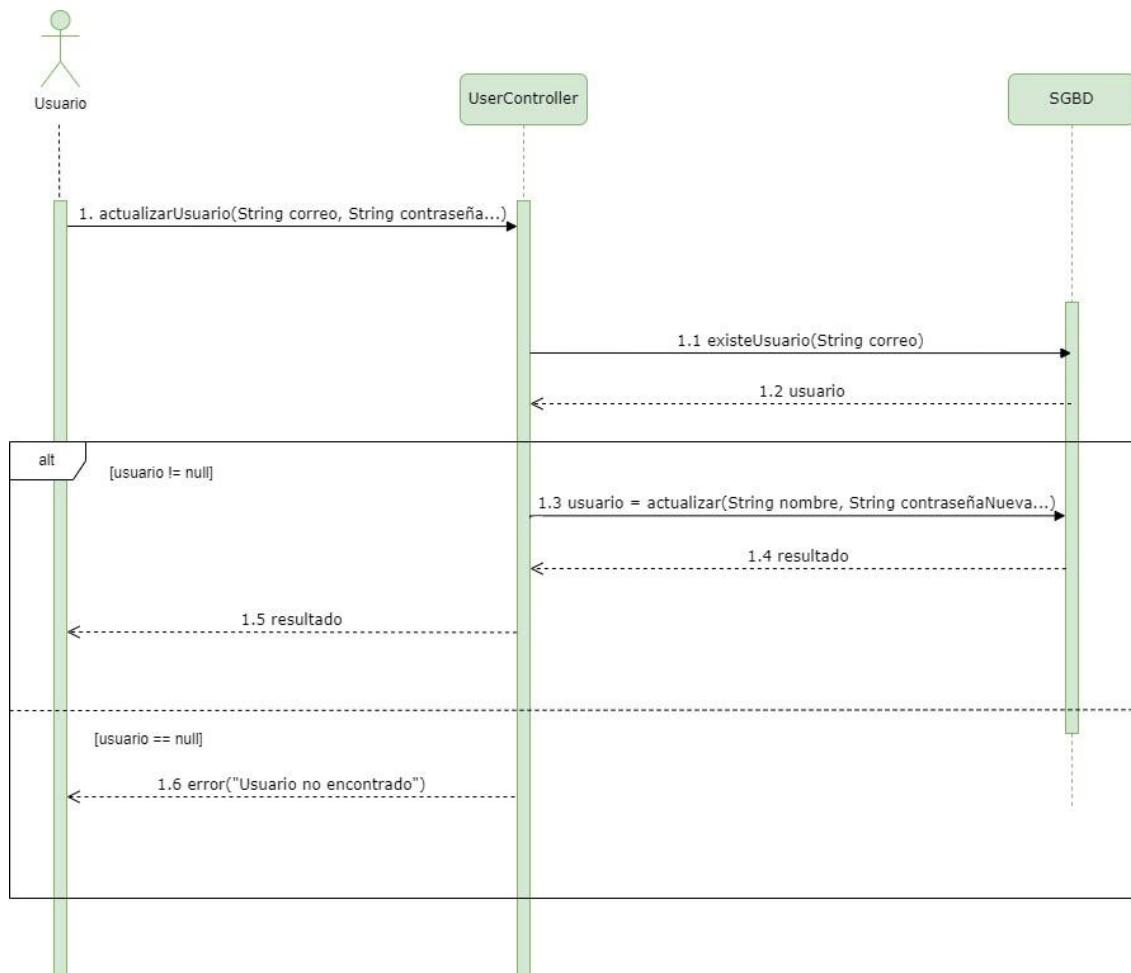


Ilustración 9 Diagrama de secuencia de editar un perfil



4. Diseño

4.1. Diseño de la base de datos

Una vez hemos hecho el análisis de la aplicación y sabemos todas las funcionalidades que esta debe tener, comenzamos con la etapa de diseño, donde tendremos que ver que componentes serán necesarios para un correcto funcionamiento de la aplicación, de la manera más eficiente y sencilla posible.

Para eso, empezaremos por diseñar la base de datos de nuestra aplicación, la cual será la base para poder ir desarrollando el proyecto, sirve para gestionar información relacionada con usuarios, acciones, operaciones, carteras y alertas en un sistema financiero.

Nuestro diseño de base de datos inicia con la tabla Usuario, que almacena información personal y credenciales de acceso. Siguiendo con la tabla Accion, registramos datos de acciones financieras, permitiendo a los usuarios marcar ciertas acciones como favoritas a través de la tabla AccionesFavoritas, la cual establece una relación entre usuarios y acciones de interés.

Para representar la cartera de inversiones de cada usuario, se introduce la tabla Cartera, donde se registra el saldo y las transacciones realizadas, detalladas en la tabla Transaccion. Estas transacciones reflejan compras y ventas de acciones, vinculando cada operación con una acción específica y una cartera, lo que permite un seguimiento detallado de la actividad financiera del usuario.

Además, se considera el aprendizaje y la educación financiera mediante la tabla ArticulosAprendizaje, que alberga contenidos educativos segmentados por secciones, y la tabla Notificacion, encargada de almacenar mensajes y alertas dirigidos a los usuarios, asegurando que estén informados sobre eventos significativos y cambios en el mercado o en su cartera.

Finalmente, para la recuperación de contraseñas olvidadas, implementamos la tabla ResetClaveToken, que guarda tokens de un solo uso para verificar la identidad del usuario y permitir un cambio seguro de contraseña.



4.1.1 Diagrama Entidad-Relación de la Base de Datos

Para poder representar nuestros datos, utilizaremos el modelo Entidad-Relación, estudiado en asignaturas como Fundamento de Base de Datos o Diseño y Desarrollo de Sistemas de Información, ya que es especialmente útil para visualizar la estructura y las relaciones entre las diferentes entidades que componen nuestro sistema [20].

Todo lo descrito anteriormente, lo podemos ver de una manera más visual en el siguiente diagrama Entidad-Relación, utilizando la herramienta online “Drawio” [21].

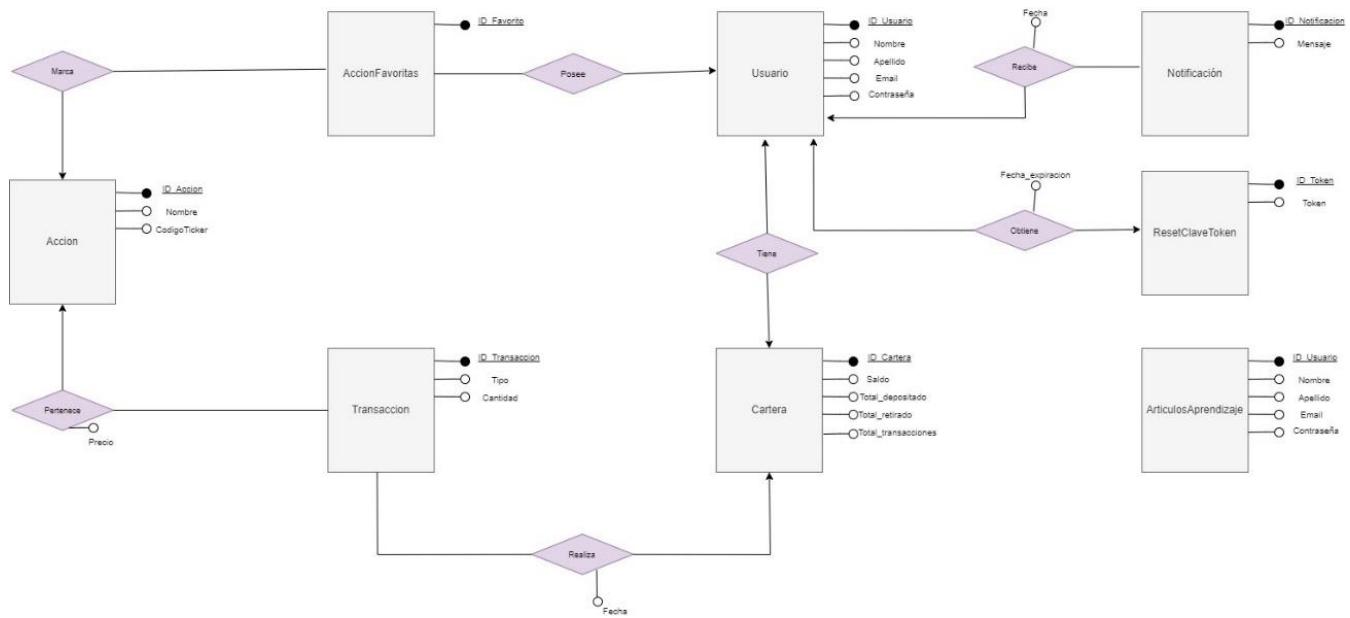


Ilustración 10 Diagrama Entidad-Relación de la Base de Datos

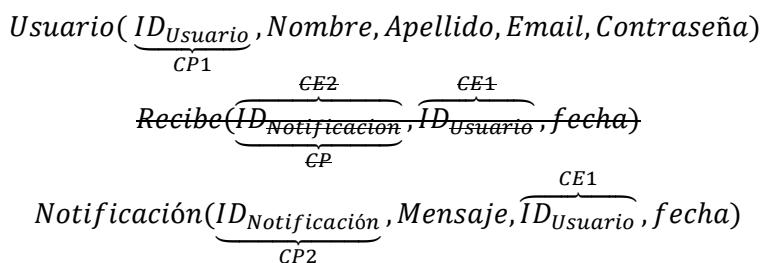


4.1.2. Paso a tablas y fusión

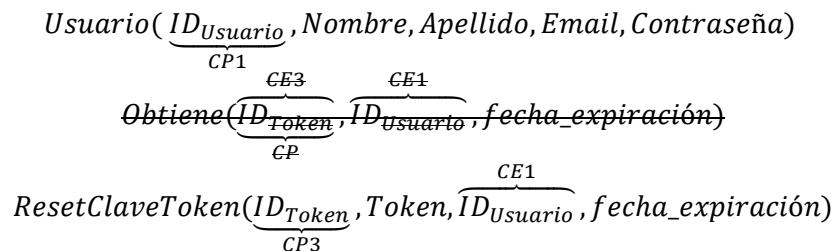
Una vez establecidas las estructuras y relaciones en nuestro diagrama Entidad-Relación, el siguiente paso es la transformación de este modelo conceptual en un modelo físico, es decir, la creación de las tablas en la base de datos.

Este proceso implica la definición precisa de las tablas y sus campos, así como la implementación de las relaciones entre ellas mediante claves primarias y foráneas. Durante este proceso, también evaluamos la posibilidad de fusionar tablas.

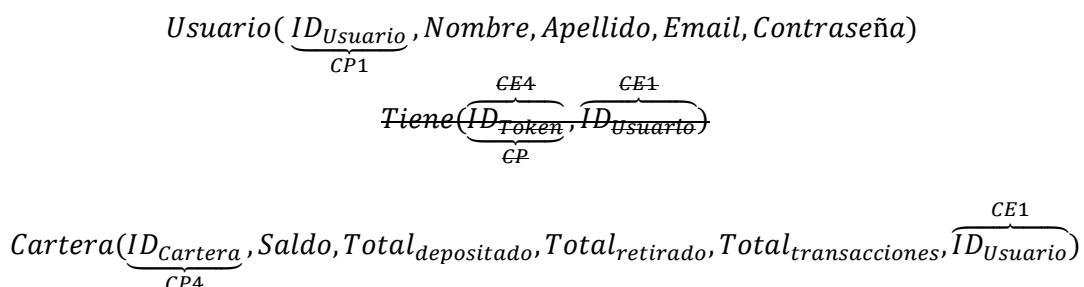
En primer lugar, la relación "Recibe" entre 'Usuario' y 'Notificación' se traduce en una incorporación directa del atributo 'fecha' en la tabla 'Notificación', debido a la naturaleza de 1:N de esta asociación. La entidad 'Usuario' proporciona su clave primaria 'ID_Usuario' como una clave foránea en 'Notificación', lo cual permite identificar al receptor de cada notificación:



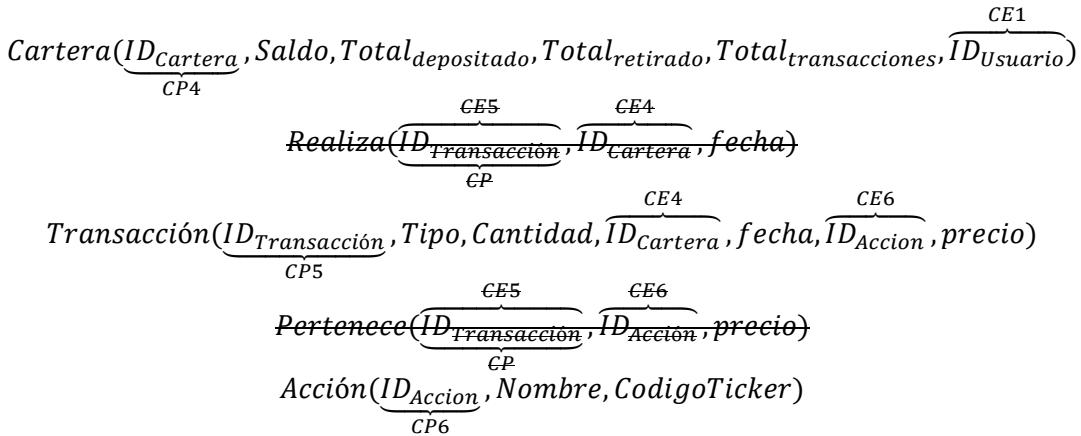
En cuanto a 'Usuario' y 'ResetClaveToken', hay una fusión de la relación, donde 'usuario_id' actúa como una clave foránea en 'ResetClaveToken' para relacionar cada token con un usuario específico. La 'fecha_expiracion' es un atributo esencial que determina la validez del token que se incorpora en la tabla del token ya que cuando la relación es 1:1, tenemos potestad para elegir donde nos conviene más.



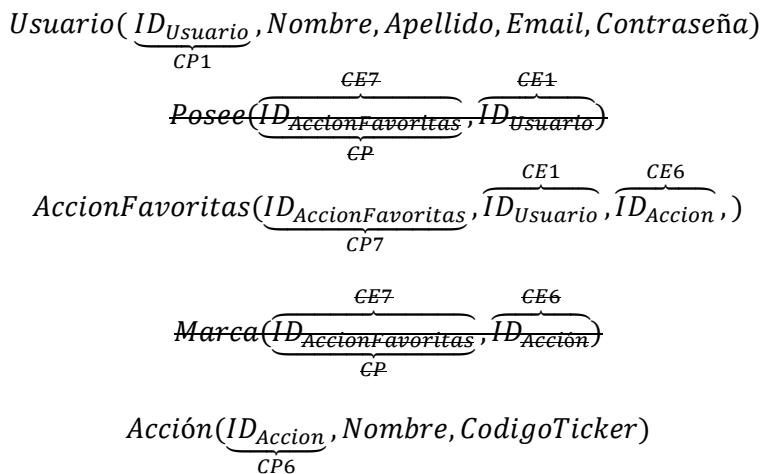
La entidad 'Cartera' incorpora 'id_usuario' para asociar cada cartera con su respectivo usuario, así podremos rastrear, gestionar y reportar la actividad financiera de manera personalizada para cada usuario:



En cuanto a la entidad 'Transacción' asimila 'id_cartera' e 'id_accion' además de atributos propios como 'precio' y 'fecha', provenientes de 'Accion' y 'Cartera' respectivamente. Donde la relación 'Pertenece' vincula cada transacción a una acción específica, que servirá para poder realizar análisis de rendimientos mientras que la relación "Realiza" conecta las transacciones a la cartera para ver el rendimiento de estas a nivel financiero:



Y, por último, en la entidad 'AccionesFavoritas', se enlazan 'id_usuario' y 'id_accion' para reflejar las preferencias de inversión de los usuarios. Esto permite a nuestra app ofrecer recomendaciones y alertas personalizadas basadas en las preferencias del usuario, además, la relación 'Posee' entre las tablas 'Usuario' y 'AccionesFavoritas' permite consultas más rápidas y apoyan a los algoritmos de recomendación:



Una vez realizado nuestro paso a tablas y hechas las respectivas fusiones, en nuestro diseño actual, cada tabla tiene un propósito claro y distinto y procedemos con la normalización de la base de datos [22].



4.1.3. Normalización de la base de datos

La normalización de la base de datos es un proceso crucial para minimizar la redundancia de datos y mejorar la integridad. Nuestro diseño sigue los principios de normalización para asegurar que cada dato se almacene en un solo lugar. Esto se logra a través del uso adecuado de claves primarias y foráneas, garantizando que las relaciones entre las tablas sean lógicas y eficientes [23]. Por ello, vamos a ir pasando por cada forma normal hasta llegar a su formal final:

La **primera forma normal**, se logra asegurando que los valores en cada columna de nuestras tablas sean atómicos y que cada registro sea único. Y nuestro caso, no hay grupos repetitivos de campos, y cada campo contiene solo datos de una sola parte de la información, cumpliendo así con esta forma normal.

La **segunda forma normal**, trata sobre eliminar la redundancia funcional parcial, es decir, que cada atributo no clave de la base de datos dependa completamente de la clave primaria. Esto se observa en la tabla 'Transaccion', donde todos los atributos dependen del 'id_transaccion', y no hay dependencias parciales en una clave primaria compuesta.

En cuanto a la **tercera forma normal**, esta quiere que ningún atributo no clave deba depender de otro atributo no clave. Por ejemplo, en la tabla 'Cartera', el 'total_depositado' y el 'total_retirado' dependen directamente del 'id_cartera' y no de otros atributos no clave.

Por último, mencionar que el modelo puede ajustarse a la **forma normal de Boyce-Codd**, una versión más estricta de la 3NF, que requiere que no haya atributos no clave que determinen otros atributos no clave. Nuestras tablas están diseñadas para que cualquier atributo que determine otro sea una clave primaria o única.

Nuestro diseño de base de datos refleja una organización lógica de la información que apoya la coherencia y el rendimiento óptimo del sistema. La implementación de este modelo normalizado se traduce en un sistema capaz de manejar transacciones complejas y consultas de datos de manera eficiente.



4.1.4 Tablas de la Base de Datos

Usuario	
Campo	Tipo
ID_Usuario	INT AUTO INCREMENT PRIMARY KEY
Nombre	VARCHAR (255)
Apellido	VARCHAR (255)
Contraseña	VARCHAR (255)
Email	VARCHAR (255) UNIQUE

Tabla 52 Tabla Usuario

Acción	
Campo	Tipo
ID_Accion	INT AUTO INCREMENT PRIMARY KEY
Nombre	VARCHAR (255)
CodigoTicker	VARCHAR (10)

Tabla 53 Tabla Acción

AccionesFavoritas	
Campo	Tipo
ID_Favorito	INT AUTO INCREMENT PRIMARY KEY
ID_Usuario	INT
ID_Accion	INT
FOREIGN KEY (ID_Usuario) REFERENCES Usuario(ID_Usuario)	
FOREIGN KEY (ID_Accion) REFERENCES Accion(ID_Accion)	

Tabla 54 Tabla AccionesFavoritas

Transacción	
Campo	Tipo
ID_Transaccion	INT AUTO INCREMENT PRIMARY KEY
ID_Cartera	INT
ID_Accion	INT
Tipo	VARCHAR(10)
Cantidad	INT
Precio	DECIMAL(10, 2),
FechaOperación	DATETIME DEFAULT CURRENT_TIMESTAMP
FOREIGN KEY (ID_Cartera) REFERENCES Cartera(ID_Cartera)	
FOREIGN KEY (ID_Accion) REFERENCES Accion(ID_Accion)	

Tabla 55 Tabla Transacción



Cartera	
Campo	Tipo
ID_Cartera	INT AUTO INCREMENT PRIMARY KEY
ID_Usuario	INT
Saldo	DECIMAL(10, 2),
Total_depositado	DECIMAL(10, 2),
Total_retirado	DECIMAL(10, 2),
Total_transacciones	INT
FOREIGN KEY (ID_Usuario) REFERENCES Usuario(ID_Usuario),	

Tabla 56 Tabla Cartera

Notificación	
Campo	Tipo
ID_Noticacion	INT AUTO INCREMENT PRIMARY KEY
ID_Usuario	INT
Mensaje	TEXT
Fecha	DATETIME DEFAULT CURRENT_TIMESTAMP
FOREIGN KEY (ID_Usuario) REFERENCES Usuario(ID_Usuario)	

Tabla 57 Tabla Notificación

ResetClaveToken	
Campo	Tipo
ID_Token	INT AUTO INCREMENT PRIMARY KEY
ID_Usuario	INT
Token	VARCHAR(100) UNIQUE
Fecha_expiracion	DATETIME
FOREIGN KEY (ID_Usuario) REFERENCES Usuario(ID_Usuario)	

Tabla 58 Tabla ResetClaveToken

ArticulosAprendizaje	
Campo	Tipo
ID_Articulo	INT AUTO INCREMENT PRIMARY KEY
Seccion	VARCHAR(255)
Titulo	VARCHAR(255)
ImageUrl	TEXT
Resumen	TEXT
Contenido	TEXT

Tabla 59 Tabla ArticulosAprendizaje



4.1.5 Relaciones de la Base de Datos

Usuario	
Nombre	Relacion
Posee	Usuario (1) -> AccionFavoritas (N)
Obtiene	Usuario (1) -> ResetClaveToken (1)
Tiene	Usuario (1) -> Cartera (1)
Recibe	Usuario (1) -> Notificacion (N)

Tabla 60 Relación Usuario

Acción	
Nombre	Relacion
Pertenece	Accion (1) -> Transaccion (N)
Marca	Accion (1) -> AccionFavoritas (N)

Tabla 61 Relación Accion

AccionFavoritas	
Nombre	Relacion
Marca	AccionFavoritas (N) -> Accion (1)
Posee	AccionFavoritas (N) -> Usuario (1)

Tabla 62 Relación AccionFavoritas

Transaccion	
Nombre	Relacion
Realiza	Transaccion (N) -> Cartera (1)
Pertenece	Transaccion (N) -> Accion (1)

Tabla 63 Relación Transaccion

Cartera	
Nombre	Relacion
Tiene	Cartera (1) -> Usuario (1)
Realiza	Cartera (1) -> Transaccion (N)

Tabla 64 Relación Cartera

Notificacion	
Nombre	Relacion
Recibe	Notificacion (N) -> Usuario (1)

Tabla 65 Relación Notificacion

ResetClaveToken	
Nombre	Relacion
Obtiene	ResetClaveToken (1) -> Usuario (1)

Tabla 66 Relación ResetClaveToken



4.2. Diseño de la arquitectura del sistema

Cuando ya tenemos formalizado nuestra base de datos, es importante saber cómo vamos a comunicarnos entre los distintos componentes del proyecto, por ello, es muy común el uso de patrones arquitectónicos para la mejora de la eficiencia y escalabilidad de nuestra aplicación.

Uno de los más comunes, es la arquitectura Modelo-Vista-Controlador, ya que se suele utilizar comúnmente cuando el componente visual es el más importante, y donde es crucial mantener una cierta separación clara entre la lógica de negocio y la interfaz de usuario [24].

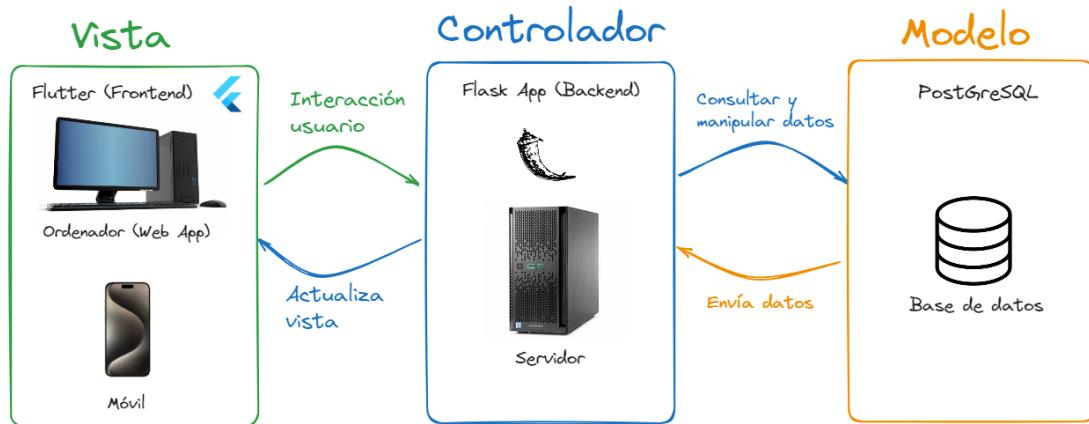


Ilustración 11 Modelo-Vista-Controlador (MVC)

Componentes del Sistema:

- **Flutter App (Frontend):**
 - Vista en MVC: Se encarga de la interfaz de usuario, y teniendo especial hincapié en la sencillez visual y en la interactividad del usuario.
 - Interacción con el Backend: La vista se comunica con la aplicación Flask a través de controladores para enviar y recibir datos.
- **Flask App (Backend):**
 - Controlador en MVC: Se encarga de gestionar la lógica de la aplicación, desde las solicitudes recibidas del frontend como de las operaciones y cálculos financieros.
 - Comunicación con la Base de Datos: Interactúa con la base de datos PostGreSQL para consultar y manipular datos.
- **PostgreSQL Database:**
 - Modelo en MVC: Representa la estructura de datos y almacena la información relativa a usuarios, acciones, operaciones, carteras, noticias financieras y alertas.
 - Integración con el Backend: Se comunica directamente con la aplicación Flask para realizar operaciones de datos.

Implementar el patrón MVC ofrece múltiples ventajas, sobre todo en términos de escalabilidad y la capacidad de dividir el trabajo. Gracias a dividir el proyecto en distintos módulos nos aseguramos que cualquier modificación que se realice sea fácil y manejable, además de ser menos propensas a generar errores en cualquier otra área [25].



4.3. Diseño de la interfaz de usuario

Una vez hemos diseñado como va a funcionar la lógica de nuestra aplicación, sería conveniente hacer una lluvia de ideas o bocetos de cómo va a ser la UI de nuestro proyecto, ya que el diseño de la interfaz podría ser la parte más importante en el desarrollo de una aplicación, principalmente porque va a establecer contacto directo con el usuario.

La clave del éxito reside en una experiencia del usuario intuitiva, cómoda para los ojos y que facilite la navegación y el acceso a las distintas funcionalidades, por ello, esencial hacer bocetos representativos de cómo queremos montar la aplicación, ya que nos servirán para tener una imagen mental de hacia dónde queremos dirigirnos, además de que nos ayudara a planificarnos mejor.

El tiempo que invertimos en hacer estos diseños, se verá recompensando para la comunicación entre los desarrolladores y los stakeholders, ya que, al tener una referencia clara, se reducen los riesgos de que se produzcan malentendidos o cambios de dirección inesperados durante el desarrollo. Para dicho propósito, haremos uso de la herramienta online Figma y Canva [26], el cual nos ofrecen una serie de herramientas necesarias para crear diseños atractivos y funcionales [27].

Para ver estos diseños o mockups, dejo adjunto el enlace que redirija a la página de Figma para poder verlos con una mejor calidad:

<https://www.figma.com/file/TLfeVLNdZUun75nwyY8gLQ/MockUps-Grownomics?type=design&node-id=0%3A1&mode=design&t=sdCj2pVX5ZY4vxKM-1>

4.3.1. Diseño de la interfaz de la aplicación móvil

Dado que en nuestro proyecto optamos por utilizar Flutter, una plataforma versátil que facilita el desarrollo simultáneo de aplicaciones móviles y web, nuestra atención se centrará principalmente en el diseño para dispositivos móviles, que constituye el enfoque principal de este proyecto.



4.3.1.1 Inicio de Sesión y Registrarse

Para el diseño de la pantalla de inicio de sesión y de registro, podemos ver campos de texto donde se solicita tanto el nombre, el correo y la contraseña del usuario, necesarios para la identificación y autenticación y acceder a su área personal.

Estos campos están etiquetados y hay suficiente espacio para que la entrada de datos sea cómoda para el usuario, con tonos ligeramente distintivos del fondo para conseguir crear un contraste y que sean fácilmente visibles.

En el inicio de sesión, hay opciones adicionales como "Recordarme", para mantener la sesión activa, y un enlace para recuperar la contraseña en caso de olvido, lo cual es importante para mantener la accesibilidad y la experiencia del usuario lo más sencilla y fácil posible.

El botón de "Iniciar sesión" sigue los mismos principios de diseño que el botón "Registrarse" en términos de visibilidad y accesibilidad y con colores propios del contexto de la aplicación con un verde que entona con los colores del logo. Por último, mencionar los últimos enlaces en cada página, que permiten la navegación entre ambas páginas, para cuando un usuario no tiene una cuenta creada o para cuando si la tiene.

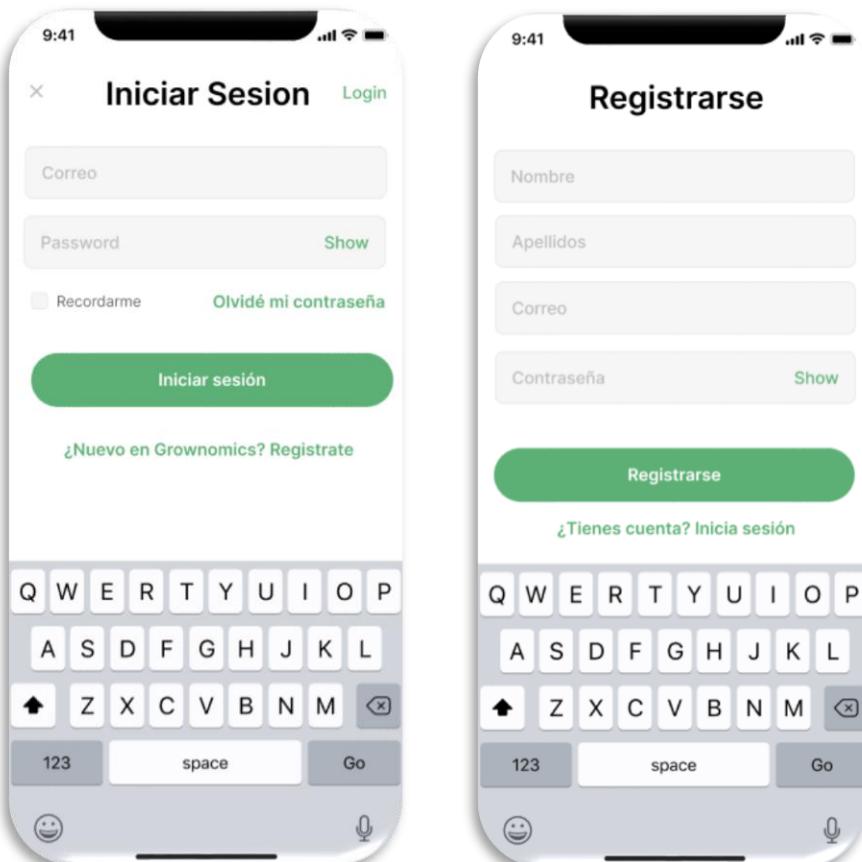


Ilustración 12 Interfaz de Inicio de Sesión y de Registro



4.3.1.2 Cambiar contraseña

Como hemos mencionado anteriormente, disponemos de una pantalla diseñada para la cambiar la contraseña, de una manera intuitiva y simple, siguiendo el mismo patrón para los campos de texto, etiquetas correctamente para ingresar el correo electrónico, la clave y la nueva contraseña.

Después de ingresar el correo electrónico, se presenta un botón destacado, en tono verde para mantener la coherencia con la identidad visual de nuestro logo que permite al usuario solicitar el código de verificación necesario para completar el proceso de cambio de contraseña.

Una vez solicitado el código de verificación, la pantalla revela tres nuevos campos. Estos campos permiten al usuario ingresar el código recibido, así como establecer y confirmar una nueva contraseña. Este enfoque paso a paso guía al usuario a través del proceso, asegurando que se proporcionen todos los datos necesarios de manera clara y precisa.

Finalmente, para completar el proceso, se presenta un botón con el diseño del botón anterior que permite al usuario confirmar y enviar la solicitud de cambio de contraseña.



Ilustración 13 Interfaz de Cambiar Contraseña



4.3.1.3 Página de Inicio o “Home”

Una vez nos hemos registrado o iniciado sesión, accederemos a la página principal, donde se encontrarán los elementos más importantes y los que tendrán que estar perfectamente pulidos para causar un buen impacto al usuario. En la parte superior de esta página, los usuarios encontrarán una barra de navegación intuitiva que les permitirá acceder al menú de navegación de la aplicación y a la derecha un ícono de una “campana” para poder acceder a las notificaciones recibidas de la aplicación. En él, se mostrará información relevante al perfil del usuario, como su nombre y su correo. Debajo estarán los botones que servirán para navegar entre las distintas secciones de la aplicación.

Pero centrándonos en el diseño de la página de inicio, en primer lugar, encontramos un elemento donde se mostrarán las últimas noticias financieras para mantener al usuario informado de las novedades más interesantes. Debajo de este encontrará un resumen de cómo están fluctuando los sectores en el mercado, así como diferentes datos relevantes como los índices de las empresas como puede ser el SP500 o el IBEX35 y que el usuario pueda estar al tanto de la situación del mercado a nivel global.

Justo debajo de esta sección, habrá una serie de elementos para cuando el usuario se encuentre registrado, que será un resumen de sus simulaciones, para ver su saldo virtual, el total retirado y el total depositado, así como el número de transacciones realizadas. Además de otro componente que mostrará información sobre las acciones favoritas del usuario, permitiéndole un acceso rápido y sencillo a sus inversiones preferidas.

Por último, como una lista de las últimas transacciones realizadas por el usuario para que pueda recordar sus últimos movimientos realizados, junto con un botón que les redirigirá para poder ver toda la lista completa.



Ilustración 14 Interfaz de Inicio o Home



4.3.1.4 Pagina de Análisis

En cuanto al diseño de la página de análisis, nos encontramos primero con la barra de búsqueda en la parte superior, que permite a los usuarios filtrar las acciones por nombre, lo cual es bastante importante para cuando los usuarios que siguen una amplia gama de acciones necesiten acceder rápidamente a la información específica de una acción.

Justo debajo hallamos los botones de "Todos" y "Favoritas" que ofrecen un método de filtración rápido y simple para alternar entre ver todas las acciones disponibles y aquellas que el usuario ha marcado como favoritas.

Cada acción se presenta en un formato de tarjeta, lo que facilita su visualización y poder ver la información claramente. Las tarjetas incluyen el nombre de la compañía, el ticker de la acción, el precio actual y el cambio porcentual.

Junto a cada acción, hay un icono distintivo en forma de estrella, que los usuarios podrán seleccionar para añadir o eliminar una acción de su lista de favoritos. Además, cada acción contará con un botón dedicado que permitirá acceder a información más detallada sobre ella.

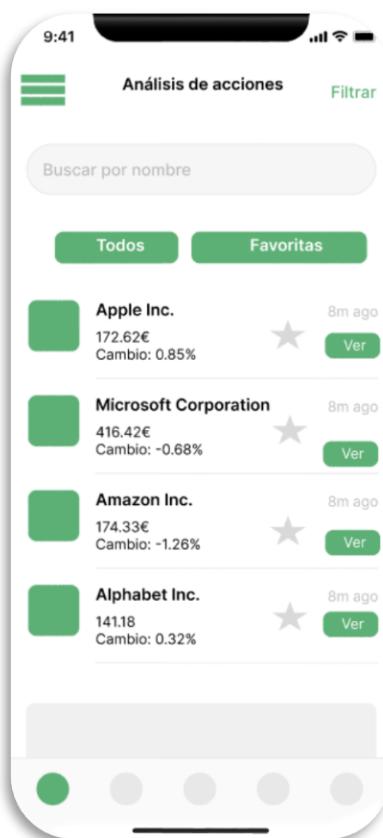


Ilustración 15 Interfaz General de Acciones



Una vez dentro de la página específica de una acción, podremos observar información mucho más precisa y completa, un gráfico para poder ver a nivel visual como ha ido evolucionando el valor de la acción. Donde podremos darle distintos modos, según queramos ver respecto a 1 semana, 1 mes o 1 año.

Además, habrá una barra de navegación para poder ver diferentes aspectos de la acción, como los análisis técnicos, análisis fundamentales, estrategias e indicadores. Luego podremos encontrar un análisis de la acción, utilizando algoritmos, indicadores económicos y unas recomendaciones utilizando diferentes estrategias para poder darle ciertas recomendaciones al inversor sobre cuál sería la mejor opción a nivel teórico.

Y, por último, unos botones de comprar o vender, para dar paso a una simulación virtual y ver cómo nos iría al realizar ciertas compras o ventas en el mercado.



Ilustración 16 Interfaz de Acción Específica



4.3.1.5 Pagina de Cartera Virtual

En esta página podremos acceder a una sección de la plataforma de simulación financiera, brindando a los usuarios información detallada de su desempeño financiero virtual, donde se puede obtener una visión general de su situación financiera a través de una tarjeta, viendo valores como su saldo actual, el beneficio, el total depositado... etc. y las opciones para depositar y retirar dinero virtual, lo que les permite continuar con su experiencia de simulación de inversión.

Además, habrá un historial detallado del rendimiento financiero del usuario, el cual incluye información sobre cómo ha evolucionado el saldo o balance con el tiempo, permitiendo que se pueda evaluar la efectividad de las estrategias de inversión y las recomendaciones implementadas. Por último, se podrá navegar a una página que les permitirá acceder a información detallada y relevante sobre cada transacción realizada.

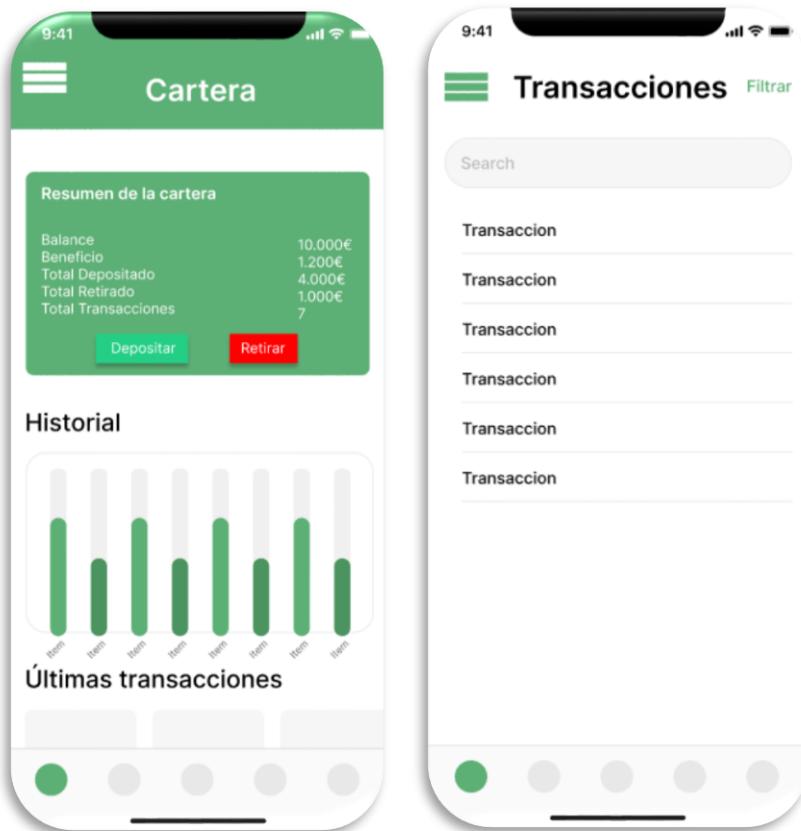


Ilustración 17 Interfaz de Cartera y Transacciones



4.3.1.6 Página de mis Acciones

Sobre la página “Mis Acciones”, la idea fundamental es proporcionar a los usuarios una visión clara y sencilla de cómo está compuesta su cartera de inversiones, enfocándose principalmente en la diversificación de esta, junto a información relevante sobre las acciones y las transacciones realizadas.

En cuanto al tema de la diversificación, para poder representarla correctamente se ha utilizado un gráfico circular donde se muestra el porcentaje de cada acción en la cartera del usuario. Y también tendremos disponible un componente con forma de tarjeta con los colores corporativos de la aplicación, donde se muestra principalmente el valor actual de la cartera según el precio de las acciones que posee, para que el usuario pueda tener una idea clara de cómo está funcionando su cartera en tiempo real.

Para finalizar, se proporciona una lista de las acciones con su respectiva cantidad que posee la cartera junto a sus transacciones asociadas, para poder ver información detallada como el precio de compra, la cantidad de compra y la fecha de la compra o venta de estas.



Ilustración 18 Interfaz de Mis Acciones



4.3.1.7 Página de Noticias

Por otro lado, tendríamos el apartado de noticias cuyo objetivo principal es mantener al día con las últimas novedades y eventos que haya ocurrido en el ámbito financiero. Para ello se utilizarán componentes para presentar las noticias financieras actuales, seleccionadas cuidadosamente según su importancia y relevancia en el mercado.

Para comentar el diseño, empezaremos por la parte superior, donde dispondremos de una barra de búsqueda, que permite a los usuarios filtrar las noticias según sus intereses consiguiendo crear una experiencia más personalizada.

Justo debajo de la barra de búsqueda, encontraremos una serie de secciones como “Finanzas”, “Economía”, “Inversiones” o “Bolsa de Valores”, que facilitan sobre todo a las filtraciones del contenido y que se pueda acceder más rápido a las áreas que le interesen al usuario.

En relación a como se muestra las noticias, vemos que el contenido se organiza en una serie de tarjetas, compuestas por una imagen en grande, un titular y un breve resumen sobre lo que trata en sí. Si pulsamos la noticia, podremos navegar directamente al sitio web de dichas noticias para poder leerla más detenidamente.



Ilustración 19 Interfaz de Noticias



4.3.1.8 Página de Aprendizaje

Además, la aplicación tendrá un apartado de aprendizaje diseñado especialmente para enriquecer los conocimientos financieros y económicos de los usuarios, independientemente de su nivel de experiencia que tengan en estos ámbitos.

Por ello, los elementos de esta sección se mostrarán como pestañas en la parte superior en el menú de navegación, lo que permite desplazarse de una ventana a otra de manera fluida e intuitiva. Cada artículo de las distintas secciones se presentará en un formato de tarjeta que incluirán una imagen principal, un título y un breve resumen sobre lo que tratan.

Y para complementar todas estas lecciones, también habrá una pestaña de tutorial, donde estará destinado principalmente a guiar a los usuarios sobre como utilizar correctamente las distintas funcionalidades de la aplicación.



Ilustración 20 Interfaz de Aprendizaje



4.3.1.9 Página de Configuración

Para finalizar con la etapa de diseño, comentaremos brevemente la página de configuración, enfocada en personar y ajustar la experiencia del usuario según sus gustos y preferencias.

Aquí, los usuarios serán capaces de editar los detalles de su perfil, como el nombre de usuario, la contraseña... etc. También podrán activar o desactivar la opción de que se muestren las notificaciones sobre las acciones favoritas, lo que le permitirá tener un cierto control sobre como recibir las actualizaciones.

Y, por último, comentar que habrá una sección para los usuarios que deseen saber un poco mas sobre la aplicación, donde se proporcionará diferentes pantallas sobre nosotros, nuestra política de privacidad y los términos y condiciones de uso.



Ilustración 21 Interfaz de Configuración



4.3.2. Diseño de la interfaz de la página web

Aunque la atención principal de nuestro proyecto se centre en el diseño de una aplicación móvil tampoco podemos pasar por alto la importancia de diseñar la página web orientado al administrador de nuestro back-end. Es verdad que, aunque no se encuentre destinada al uso directo de los clientes, es algo necesario para poder garantizar una gestión cómoda y eficiente desde la parte administrativa.

Por ello, también realizaremos algunos bocetos con un diseño intuitivo y fácilmente accesible a las diferentes funcionalidades que los administradores utilizarán a diario. Para realizar dichos diseños, haremos uso de Figma ya que ofrece una gran cantidad de opciones para adaptar los “mockups” a las necesidades del back-end.



4.3.2.1 Página de Inicio (Back-end)

La página de Inicio del back-end tiene como objetivo proporcionar una visión general sobre lo que es “Grownomics” y de sus servicios. En primer lugar, tenemos una cabecera con una barra de navegación que tiene enlaces rápidos a las distintas secciones del sitio como “Inicio”, “Acerca de”, “Servicios”, “Contacto” e “Iniciar sesión”, fundamental para que tanto los usuarios como los administradores puedan navegar con facilidad mejorando así la experiencia de usuario.

En la parte central de la página, podemos ver el contenido de cada sección, como la visión general de plataforma, información con más detalle sobre los valores y la propuesta del proyecto, buscando establecer un marco de confianza y transparencia con el usuario. Además, podemos observar en formato de tarjetas la descripción de los servicios que ofrece la aplicación para una comprensión rápida de cómo se deben de utilizar.

En la parte más baja de la página encontramos una sección de contacto para facilitar una comunicación con los usuarios y la administración de Grownomics, teniendo una descripción del desarrollado junto a su correo que redirigirá a un “mailto” correspondiente, para tener una comunicación eficiente. Y, por último, tenemos el “footer” con los derechos de autor y cualquier otra información legal para asegurar el cumplimiento de las normativas.

The screenshot displays the Grownomics Back-end Home Page. At the top is a green header bar with the logo "Grownomics" on the left and navigation links "Inicio", "Acerca de", "Servicios", "Contacto", and "Iniciar sesión" on the right. Below the header, the main content area features three service cards under the heading "Bienvenido a Grownomics". Each card has a title "Gestion de Cartera" and a short description. The "Contacto" section follows, featuring a placeholder for a user profile picture, input fields for "Nombre" and "Apellido", and a "correo@gmail.com" email field. The page concludes with a dark green footer bar containing the copyright notice "© 2024 - Grownomics. Todos los derechos reservados."

Ilustración 22 Diseño de la Página de Home (Back-end)



4.3.2.3 Página de Inicio de Sesión Web (Back-end)

Si se quiere acceder a la sección administrativa del back-end, se re conducirá a la página de inicio de sesión, que está diseñada para poder acceder rápidamente a la plataforma. Esta sección se compone por la misma cabecera con su respectiva barra de navegación para poder acudir a las diferentes secciones del sitio web.

En el contenido principal observamos dos campos principales para ingresar la información, uno para el correo electrónico y otro para la contraseña, junto a un botón con la misma estética visual que los “tabs” de navegación para iniciar sesión.

En general podemos ver que la estructura sigue un esquema minimalista, utilizando los colores corporativos del proyecto con tonos verdes.

Grownomics

Inicio Acerca de Servicios Contacto Iniciar sesión

Inicio de sesión a plataforma administrativa

Email:

Contraseña:

Iniciar sesión

© 2024 - Grownomics, Todos los derechos reservados.

Ilustración 23 Diseño de la Página de Inicio de Sesión (Back-end)

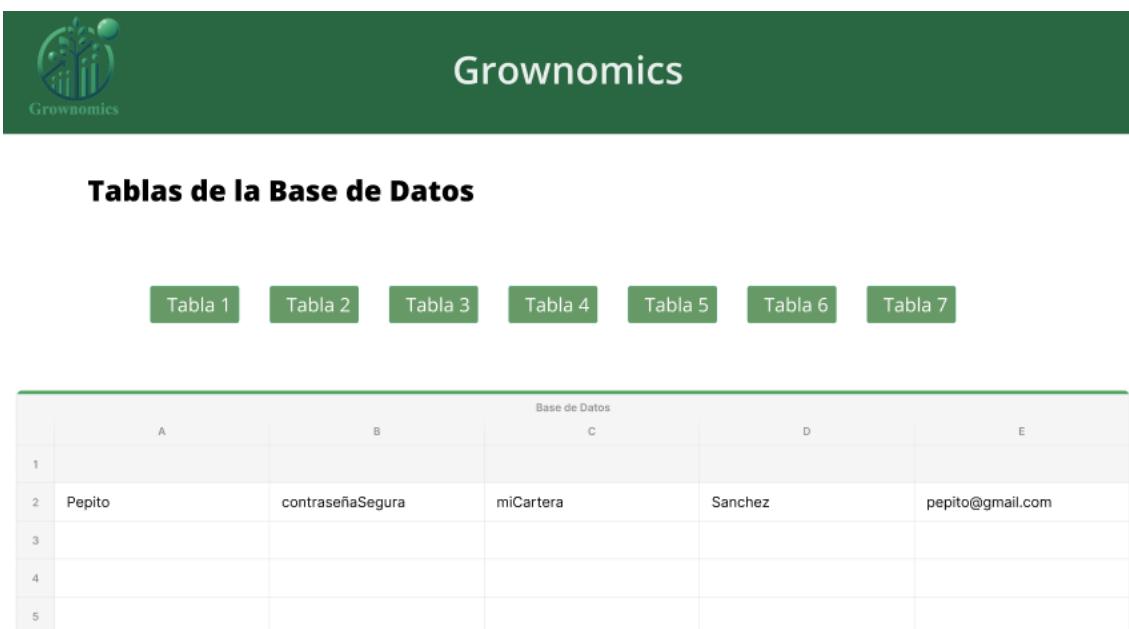


4.3.2.3 Página de Gestión de la Base de Datos

Al igual que en la página de inicio del back-end, esta sección contiene una cabecera con el nombre de la página y su logo. En la parte superior de la página, encontramos una sección de botones o pestañas numeradas con el nombre de “Tabla 1”, “Tabla 2”, “Tabla 3” ... etc.

Cada una representando las diferentes tablas que componen la base de datos de Grownomics. Esto permite a los administradores cambiar fácilmente entre las diferentes vistas de los datos, optimizando así su gestión como el análisis de la información.

Y justo debajo, se presenta una tabla dinámica que muestra los datos correspondientes a la pestaña seleccionada, incluyendo atributos clave como el nombre de usuario, la contraseña y el correo electrónico, tomados como ejemplo de la tabla de Usuarios.



The screenshot shows the Grownomics database management interface. At the top, there is a dark green header bar with the Grownomics logo on the left and the word "Grownomics" in white on the right. Below the header, the title "Tablas de la Base de Datos" is centered in bold black font. Underneath the title, there is a row of seven green buttons labeled "Tabla 1" through "Tabla 7". A horizontal line separates this from the main content area. The main area contains a table titled "Base de Datos" with columns labeled A, B, C, D, and E. The table has five rows, numbered 1 to 5. Row 2 contains the data: Pepito in column A, contraseñaSegura in column B, miCartera in column C, Sanchez in column D, and pepito@gmail.com in column E.

Base de Datos				
	A	B	C	E
1				
2	Pepito	contraseñaSegura	miCartera	Sanchez
3				
4				
5				

Ilustración 24 Diseño de la página de gestión de la base de datos (Back-end)



5. Implementación

5.1. Desarrollo de aplicación

5.1.1. Entorno de desarrollo

Para adentrarnos sobre cómo se ha realizado el proyecto, debemos comentar como es el entorno de desarrollo, donde se han utilizado diversas herramientas especializadas cada una para un determinado aspecto del trabajo.

Por ejemplo, para la gestión y organización del proyecto, se ha utilizado la plataforma **Jira** [28], para poder seguir correctamente la metodología Scrum, que sirve principalmente para la administración de tareas y el seguimiento iterativo del progreso del proyecto. Y por eso adjunto en el anexo, muestro el proceso evolutivo de los tableros de sprint.

Por otro lado, para la planificación de las actividades y de las distintas etapas del proyecto se ha utilizado el software **GanttPro** [29], útil para la asignación de recursos, sin embargo, al ser una plataforma de pago y la limitación del período de prueba, he tenido que complementarlo con **Canva** para poder diseñar los respectivos diagramas.

Entrando ya en herramientas propias del desarrollo, en el lado del cliente, he utilizado como software principal para el front-end **Flutter** [30], se trata de una framework para interfaces de usuario de código abierto que fue creado por Google y es un entorno que ya he tenido cierta práctica durante la carrera en asignaturas como Desarrollo Software o Dirección y Gestión de Proyectos.

Gracias a este framework, se pueden crear aplicaciones nativas compiladas para móvil, web y escritorio a partir del mismo código universal, aparte de la facilidad para crear interfaces fluidas y atractivas y de poseer una amplia comunidad y soporte.

Ventajas:

- **Consistencia en múltiples plataformas:** Con Flutter, podemos garantizar que la aplicación mantenga una apariencia y funcionalidad consistente en todas las plataformas sin tener un código específico para cada una de ellas.
- **Desarrollo rápido:** Flutter tiene una funcionalidad que permite realizar “Hot Reloads”, que permite ver los cambios realizados en el código al instante sin tener que reiniciar y montar de nuevo la aplicación, lo que agiliza el proceso de desarrollo.
- **Gran variedad de widgets:** Hay una gran cantidad de widgets preconstruidos y que son fácilmente personalizables para poder crear interfaces completas y profesionales de manera rápida y eficiente.

Desventajas:

- **Curva de aprendizaje:** Bien es cierto que, aunque tenga mucho potencial, puede que tenga cierto grado de dificultad para aprenderse debido a su dependencia al lenguaje Dart, que es menos común en la comunidad de desarrollo.



- **Tamaño del archivo de la aplicación:** Las aplicaciones desarrolladas tienden a tener un tamaño de archivo inicial más grande comparado con otras tecnologías, lo que perjudica sobre todo para los dispositivos con almacenamiento limitado.

Alternativas:

- **React Native:** Utiliza un lenguaje bastante común y sencillo como es JavaScript, lo que lo hace accesible a una gran base de desarrolladores, sin embargo, puede requerir más código para adaptarse a todas las especificaciones de cada plataforma.
- **Xamarin:** Su lenguaje es C# y .NET para permitir el desarrollo de aplicaciones móviles nativas para múltiples plataformas desde un único código base, ofreciendo un alto rendimiento comparable al de las soluciones nativas. Sin embargo, el tamaño de las aplicaciones resultantes puede ser grande y las actualizaciones dependen en gran medida de Microsoft, lo que podría limitar la flexibilidad en algunos proyectos.

Y cambiando hacia el back-end, he optado por crear el servidor con una imagen de **Flask** [31], que es un microframework para Python que es ligero y fácil de usar, y que es muy usado en la comunidad del desarrollo software. La justificación de su uso es principalmente para descubrir nuevas tecnologías potentes, ya que es ideal por su simplicidad y por la rapidez para poder desarrollar APIs RESTful.

Además, tiene una amplia biblioteca funcional para analizar datos y que son compatibles para poder utilizar la API de YahooFinance y tampoco quería quedarme en la zona de confort, utilizando, por ejemplo, NodeJS, con el que ya he utilizado en diferentes prácticas.

Ventajas:

- **Flexibilidad:** Proporciona el esqueleto básico para la aplicación, permitiendo al desarrollador usar extensiones sin imponer una estructura rígida.
- **Facilidad de aprendizaje:** Por ser un microframework, Flask es fácil de aprender y utilizar, lo que es perfecto para desarrolladores principiante o que se están adentrando por primera vez en el desarrollo web.

Desventajas:

- **Funcionalidades limitadas:** A diferencia de frameworks más completos como puede ser Django, Flask no tiene muchas características incorporadas, lo que me puede requerir la integración de múltiples extensiones para funcionalidades comunes.
- **Escalabilidad:** Mientras que Flask puede manejar aplicaciones de pequeña a mediana escala con facilidad, no es la mejor opción para aplicaciones a gran escala sin una cuidadosa planificación de la arquitectura.

Alternativas

- **Node.js con Express:** Tiene un enfoque similar en términos de simplicidad y flexibilidad, pero en el entorno de JavaScript, lo que puede ser beneficioso para proyecto que quieren mantener consistencia entre el front-end y el back-end.



- **Django:** Este framework es ideal para proyectos grandes y complejos, ya que viene equipado con una amplia gama de funcionalidades y componentes de manera predeterminada. Por ejemplo, Django incluye un sistema de administración, un sistema de autenticación, un ORM (Object-Relational Mapping) ... etc.

La base de datos elegida es **PostgreSQL** [32], básicamente es un sistema de gestión de bases de datos relacional que es ampliamente utilizado, debido a su robustez para manejar grandes volúmenes de datos y que se adapta muy bien a mis necesidades para el manejo de datos financieros.

Ventajas:

- **Integridad de datos:** PostgreSQL es conocido por su conformidad con los estándares SQL y su enfoque en la integridad de los datos.
- **Extensibilidad:** Ofrece extensas capacidades de extensión y personalización, incluyendo soporte para tipos de datos personalizados y procedimientos almacenados.

Desventajas:

- **Complejidad:** Puede que sea más complejo de configurar y administrar en comparación con otras bases de datos más simples, como puede ser MySQL.
- **Recursos:** También es más exigente en términos de recursos del sistema en configuraciones de gran escala, aunque esto generalmente se compensa con su alto rendimiento.

Alternativas:

- **MySQL:** Aunque es menos potente, es ampliamente utilizado en muchas aplicaciones web y puede ser suficiente para proyectos que no requieren avanzadas características.
- **MongoDB:** Para proyectos que se benefician de datos más flexibles, ofrece esquemas dinámicos que pueden adaptarse más fácilmente a cambios.

Para finalizar, he utilizado **Docker** [33], para la orquestación de servicios, ya que es una plataforma de contenedores que permite empaquetar aplicaciones y sus dependencias en un contenedor virtual, lo que facilita el despliegue y la escalabilidad.



5.1.2. Herramientas de Front-end

Especificando más adentro del front-end, como ya se ha dicho anteriormente se ha seleccionado **Flutter** como el framework principal, basado en el lenguaje de programación **Dart** [34], que ha ganado bastante popularidad dentro del diseño de interfaces de usuario, y con el que podemos aprovechar una gran cantidad de widgets y diseños de material con el que conseguiremos una experiencia de usuario moderna en todas las plataformas.

Nuestra aplicación hace uso de varios paquetes y dependencias para ampliar la funcionalidad y conseguir una mejor interacción con el usuario, entre ellos, tenemos el paquete **http** [35] que facilita el procedimiento de hacer solicitudes HTTP para comunicarse con la API implementada en nuestro back-end.

Si seguimos con la representación de datos financieros y estadísticas, podemos comentar los paquetes **fl_chart** [36] y **k_chart** [37], que me ofrecen una gran cantidad de opciones para gráficos interactivos, como pueden ser de líneas, barras y de velas.

Para asegurar una experiencia de usuario agradable y adaptable a diferentes tamaños de pantalla, se utiliza el paquete **auto_size_text** [38], que ajusta automáticamente el tamaño del texto para que encaje dentro de los límites asignados. La interacción con datos locales, como las preferencias del usuario, se gestiona a través de **shared_preferences** [39] que me permite guardar en cookies locales diferentes variables para personalizar la experiencia del usuario y el paquete **webview_flutter** [40] es crucial para mostrar contenido web dentro de la aplicación, permitiendo a los usuarios acceder a noticias y análisis financieros sin salir de la aplicación.

La aplicación también integra **socket_io_client** [41], una dependencia especializada en la comunicación en tiempo real con el servidor, que he utilizado para la funcionalidad de alertas y notificaciones. Esto, junto con **flutter_local_notifications** [42], permite notificar a los usuarios sobre las posibles varianzas que ha sufrido sus acciones favoritas en el mercado financiero.

Y para finalizar con las librerías, mencionar **intl** [43], que facilita la localización y formateo de fechas según la localización del usuario y de **markdown_viewer** [44] que permite mostrar contenido en formato Markdown, utilizado en la sección de aprendizaje para mostrar los artículos y tutoriales con un aspecto visual simple pero bonito.

Adicionalmente, para la etapa de diseño de las interfaces de usuario, se han utilizado las siguientes herramientas como **Canva** [45] y **Figma** [46], que han sido fundamentales para visualizar y crear “mockups”, que me han ido permitiendo tener una idea más clara y detallada de los elementos visuales antes de su implementación en el código.



5.1.3. Herramientas de Back-end

Y si hablamos del back-end, como ya sabemos, nuestra aplicación se construye sobre **Flask**, que ofrece una base sólida para servicios web con una implementación sencilla y rápida. Introduciéndonos en las librerías y paquetes utilizados, empezaremos por cómo hacemos para conseguir administrar y facilitar la gestión de los datos, en este caso utilizamos **Flask-Admin**, que construye una interfaz para administrar la base de datos de manera automática.

Además, utilizamos los modelos que se han definido con **SQLAlchemy** para interactuar con nuestra base de **PostGreSQL** que garantiza realizar operaciones complejas, pero de manera eficiente y segura.

Adentrándonos en el aspecto de la seguridad, hemos implementado el uso de **Flask-Login** para gestión las sesiones de los usuarios y **Flask-CORS**, que ayuda a controlar el intercambio de recursos entre diferentes orígenes.

Si nos metemos en librerías relacionadas con el análisis y el procesamiento de datos financieros, hemos implementado una gran variedad de ellas, empezando con **Yfinance** [47], el cual se utiliza para acceder a datos financieros como índices o acciones de Yahoo Finance, proporcionando dichos datos en tiempo real.

En cuanto al paquete utilizado para realizar la gran mayoría de cálculos y la manipulación de los datos en el análisis técnico, utilizamos la librería **TA-Lib** [48], una biblioteca que ofrece entre otras cosas, cálculos de indicadores técnicos como el RSI, SMA... etc.

Esto lo complementamos con **numpy** [49], utilizado especialmente para formatear las variables numéricas y manejar vectores y matrices. Y para probar y evaluar las estrategias de trading implementadas, utilizamos la biblioteca **backtesting** [50], con el que los usuarios pueden comprobar el resultado de las estrategias con datos históricos y analizar su rendimiento, lo que resulta en una herramienta muy potente para la planificación estratégica.

Además, el uso de **pandas** [51] facilita la manipulación de los datos como numpy, sin embargo, en nuestro proyecto lo enfocamos para gestionar las series de tiempos de precios de cierre de acciones, adaptando los datos a los días laborables para conseguir mayor precisión en el contexto del mercado. Y gracias a que ajustamos y preparamos estos datos, podemos crearnos modelos estadísticos con **ARIMA**, permitiendo realizar predicciones sobre los futuros precios de las acciones.

Flask-Mail se utiliza para enviar correos electrónicos desde la aplicación del back-end, cuya funcionalidad es importante para la obtención de códigos para la recuperación de contraseña, así como para las notificaciones recibidas tanto en la app sobre las acciones favoritas. Por otro lado, **Flask-APScheduler** es lo que nos permite programar las tareas como son el envío de alertas cada cierto tiempo y que vayan comprobando cada cierto tiempo como se va moviendo el mercado.

Por último, mencionar **Flask-SocketIO** [52], ya que facilita la comunicación en tiempo real entre el cliente y el servidor, siendo un componente importante e imprescindible para notificar de actualizaciones de mercado siguiendo la misma línea de las anteriores librerías.



5.1.4. Diagrama de clases

Para poder representar el conjunto de entidades de mi proyecto utilizaré los **diagramas UML**, estos consisten en generar una visión conceptual y detallada del sistema a implementar, por ello obtendremos una representación de las estructuras y las relaciones entre las distintas entidades que componen el proyecto.

Dado que este proyecto se desarrollará bajo la **arquitectura software MVC** (Modelo-Vista-Controlador), como ya se ha mencionado anteriormente, dentro de esta arquitectura, se distinguen dos categorías fundamentales de clases: modelos y controladores.

Los primeros son los encargados de interactuar con la base de datos, realizando operaciones de lectura y escritura de datos cuando sea necesario y se comunican con los controladores, actuando como el enlace que permite el flujo de información entre la base de datos y la lógica de la aplicación.

Y los segundos, desempeñan una función muy importante al gestionar la lógica operativa de la aplicación, facilitando así la interacción entre el usuario y el sistema a través de la presentación adecuada de datos en la interfaz de usuario.

Finalmente, el componente de "Vista" en nuestra arquitectura MVC tiene el objetivo de a través de los archivos definidos en dart, crear un lienzo donde se dibuja y origina la experiencia del usuario, donde paralelamente se consumen los datos y servicios proporcionados por los controladores a través de la API RESTful implementada.

Aunque es importante tener en cuenta que los diagramas presentados han sido simplificados para proporcionar una visión general del proyecto y sus componentes. Esta simplificación se realiza con el objetivo de facilitar la comprensión y la visualización de las relaciones entre las entidades del sistema.



5.1.4.1 Diagrama de clases (Front-end)

El diagrama de clases presentado establece una estructura clara de cómo las distintas páginas en la aplicación Flutter interactúan con los controladores para gestionar y presentar los datos. Cada página es representada como una clase en Flutter, y es responsable de mostrar la información en la interfaz de usuario y del usuario.

Las clases como **PaginaInicio**, **PaginaNoticias**, **PaginaAnalisis**, entre otras, representan diferentes pantallas en la aplicación, los cuales poseen métodos como `initState()` y `build()`, que son fundamentales en el ciclo de vida de los widgets en Flutter.

Los **controladores**, tales como **MercadoController**, **NoticiasController**, **CarteraController**, actúan como intermediarios entre las vistas y los modelos de datos. Su función principal es recibir solicitudes de las vistas, interactuar con los modelos para obtener o modificar los datos y devolver los resultados a las vistas para su presentación. Y los **modelos** como Accion, NewsArticle, Cartera, etc., definen el esqueleto de los datos que se manejan en la aplicación.

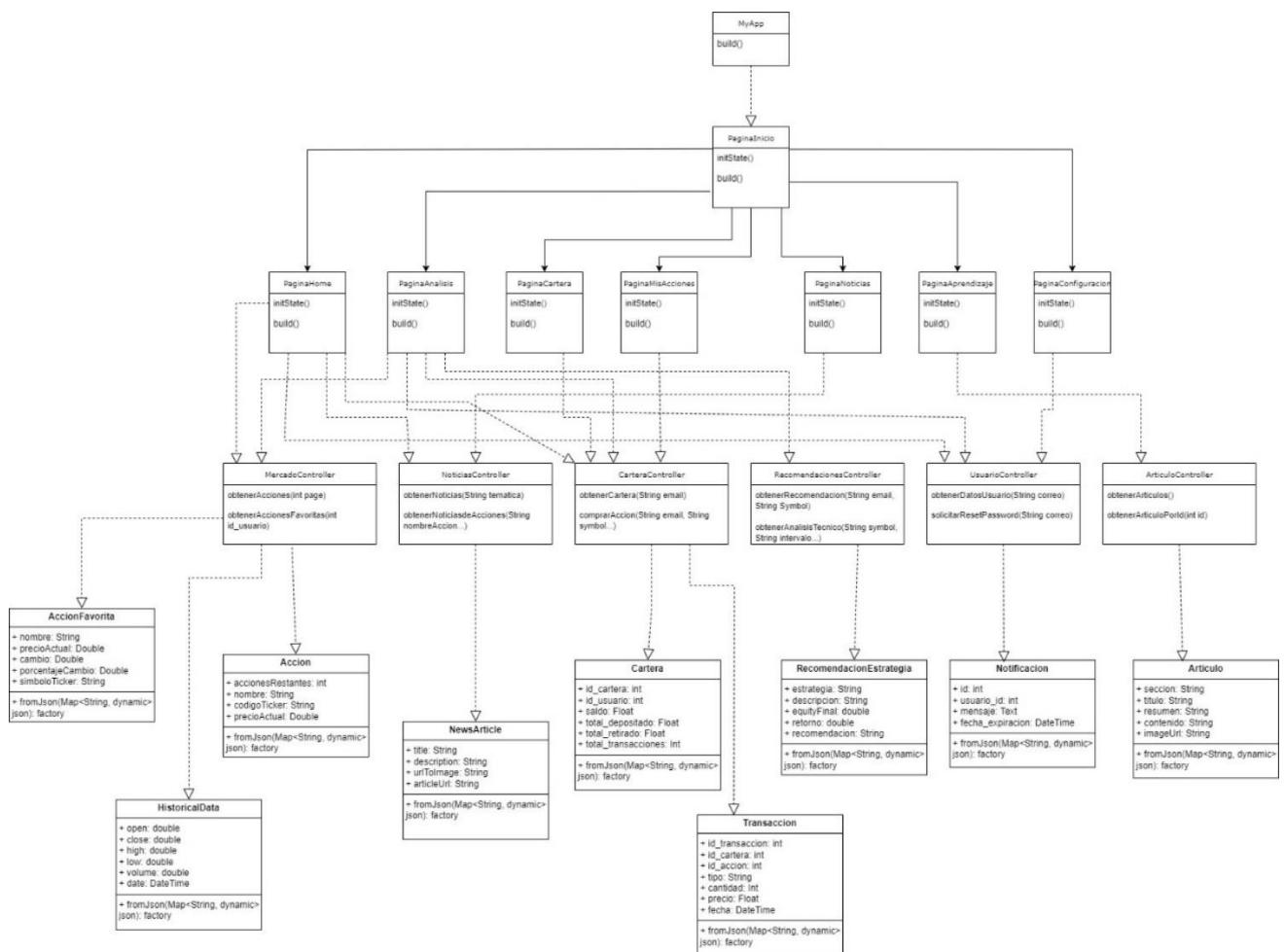


Ilustración 25 Diagrama de Clases (Front-end)



Para conseguir lograr una comprensión más sencilla y menos sobrecargada, he decidido crearme tres diagramas de clases separados para representar cada “paquete” de manera más detallada: uno para los Modelos, otro para las Vistas, y finalmente para los Controladores.

Con esto, podemos comentar y definir claramente cada componente del diagrama global, asegurándonos que se entiendo el papel y la función específica de cada elemento dentro de la estructura de la aplicación.

5.1.4.1.1 Diagrama de clases del paquete de Modelos (Front-end)

El paquete de modelos en el front-end de nuestra aplicación es fundamental para estructurar y gestionar la información que maneja y presenta el sistema. Este paquete incluye modelos diseñados para cubrir desde las operaciones financieras básicas hasta el suministro de contenidos educativos y noticias actualizadas.

Cada clase vemos que tiene implementado un método ‘fromJson()’ para permitir a la aplicación que pueda procesar y mostrar los datos que le llegan en formato JSON. La conexión indicada hacia el paquete de ‘Controladores’ muestra cómo se plantea el patrón MVC para poder facilitar la interacción entre los modelos y la lógica de negocio.

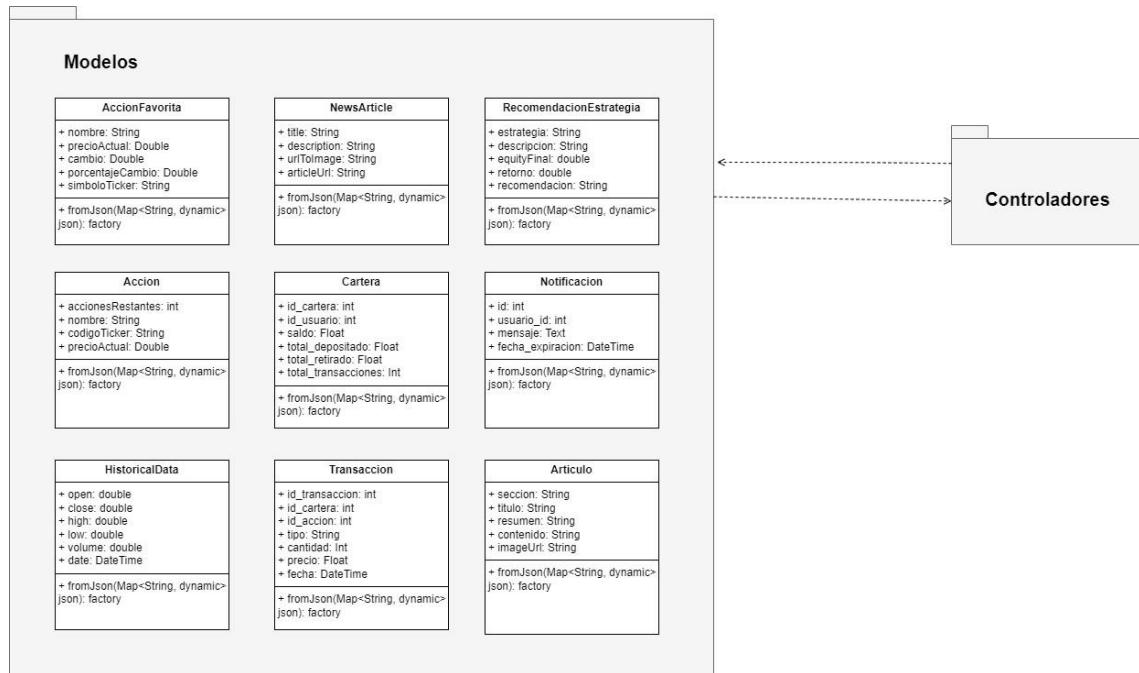


Ilustración 26 Diagrama de Clases del paquete de Modelos (Front-end)

Vamos a comentar un poco sobre cada uno de los modelos y para qué sirve cada uno:

- **AccionFavorita:** Se utiliza para marcar las preferencias del usuario, para saber en qué acciones tiene más interés, lo que permite tener un acceso rápido a estas y también habilita la recepción de notificaciones específicas sobre las mismas.
- **NewsArticle:** Estructura la información proveniente de las noticias que llegan desde el back-end a través de la API de noticias, para que puedan ser mostradas de manera sencilla y clara.



- **RecomendacionEstrategia:** Almacena información detallada sobre las estrategias de inversión, incluyendo una descripción de esta, el retorno y la equidad.
- **Accion:** Guarda datos sobre acciones específicas en el mercado, como es su código Ticker, el precio actual, siendo uno de los modelos principales de la aplicación para poder tener un seguimiento en tiempo real de las fluctuaciones del mercado.
- **Cartera:** Representa la cartera virtual del usuario, con el que se registra toda la información relacionada con las operaciones realizadas por este, con atributos como el saldo actual, el total depositado, el total retirado y el número total de transacciones.
- **Notificacion:** Almacena la estructura de los mensajes que se envían a los usuarios que tienen activadas esta funcionalidad y con alguna acción marcada como favorita para poder tener un seguimiento de estas cuando ocurra cualquier evento o fluctuación en el mercado.
- **HistoricalData:** Conserva los datos históricos de las acciones, contiene atributos como el precio de apertura y de cierre, máximos y mínimos del día, el volumen de las transacciones y la fecha, para poder mostrar correctamente los datos en los gráficos de velas entre otros.
- **Transaccion:** Registra las operaciones realizadas por los usuarios, ya sean compras o ventas de acciones, especificando el numero de acciones y el precio de la transacción.
- **Articulo:** Almacena información para los artículos de aprendizaje que se ofrecen a los usuarios, como parte de una formación educativa o de enriquecimiento de conocimientos financieros en la aplicación.



5.1.4.1.2 Diagrama de clases del paquete de Controladores (Front-end)

El diagrama de clases del paquete de Controladores refleja el modelo MVC (Modelo-Vista-Controlador) y la importancia de la intermediación entre las vistas y los modelos de la aplicación. Este diseño permite que cada controlador maneje interacciones específicas y funcionalidades detalladas dentro de la aplicación, asegurando una clara separación de responsabilidades y mejorando la organización del código.

Cada controlador está especializado en un aspecto particular de la aplicación, por ejemplo, ‘MercadoController’ se encarga de manejar las operaciones relacionadas con el mercado, como es el obtener acciones, datos históricos o resúmenes del mercado. Por otro lado, ‘CarteraController’ se centra en las operaciones referentes a la cartera de inversiones que tiene el usuario, incluyendo el depósito, retiro y el cálculo de beneficios.

Además, otros controladores como ‘UsuarioController’ se encarga de manejar todo lo relación a la información y gestión de usuarios, como es el registro, inicio de sesión y actualización de sus datos, ‘NoticiasController’ en cambio, se encarga de proveer acceso a noticias actualizadas, asegurando que los usuarios permanezcan informados sobre los últimos acontecimientos del mercado.

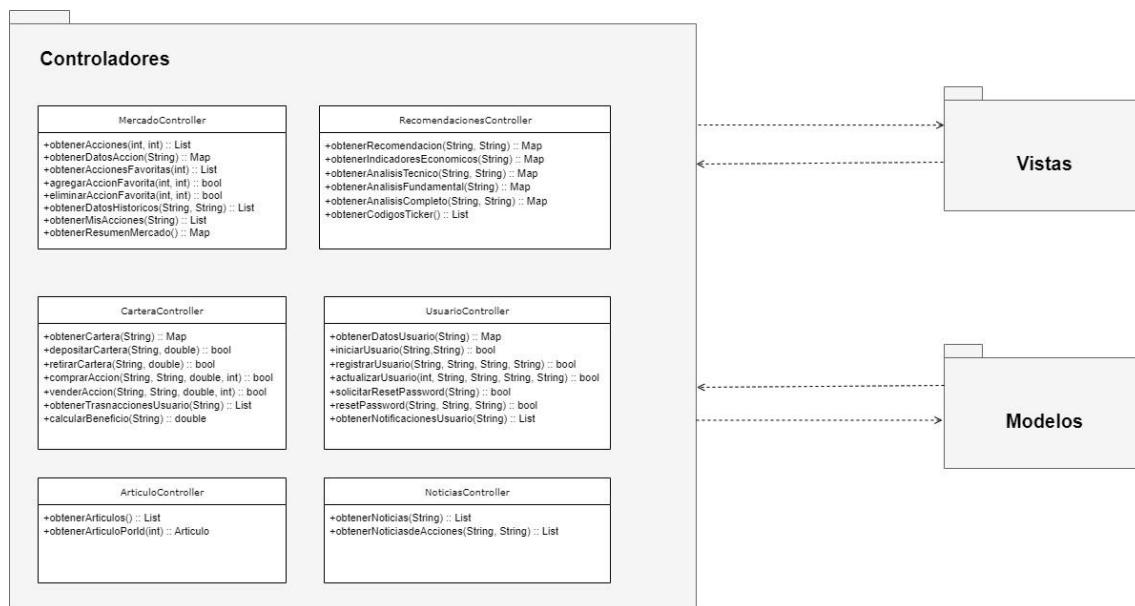


Ilustración 27 Diagrama de clases del paquete de Controladores (Front-end)

Si hacemos una breve descripción de cada uno de los controladores:

- **MercadoController:** Se encarga de gestionar la lista de acciones, tanto por criterios específicos como por popularidad, permite agregar y eliminar acciones favoritas de un usuario y ofrece métodos para ver resúmenes del mercado.
- **CarteraController:** Proporciona funcionalidades para obtener los detalles de la cartera de un usuario, facilita la compra y venta de acciones y calcula el beneficio de las inversiones de la cartera del usuario.



- **RecomendacionesController:** Realiza recomendaciones personalizadas y análisis sobre acciones específicas, así como provee de análisis técnicos y fundamentales para informar correctamente al inversor ante de realizar cualquier operación.
- **UsuarioController:** Maneja todas las operaciones relacionadas con los datos del usuario, como el registro, inicio y cierre de sesión, además de poder actualizar sus datos y generar solicitudes de cambio de contraseña.
- **NoticiasController:** Obtiene listas de noticias relacionadas con el mercado financiera, además permite también obtener noticias sobre acciones financieras para que los usuarios estén al tanto de lo que ocurre actualmente con dicha acción.
- **ArticuloController:** Recupera los artículos para educación y aprendizaje de usuarios del back-end, además puede devolver artículos específicos según su id.



5.1.4.1.3 Diagrama de clases del paquete de Vistas (Front-end)

En cuanto a las clases del paquete de Vistas, estas proporcionan una visión propiamente dicha, sobre como cada pantalla esta diseñada para interactuar con los usuarios. Entre las diferentes pantallas como ‘PantallaInicio’, ‘PaginaAnalisis’, ‘Pagina Cartera’, entre otras, tienen responsabilidades específicas para gestionar la interacción usuario-aplicación.

Si entramos en detalles, vemos por ejemplo que la pantalla de inicio, es el punto de partida para los usuarios, donde se genera la navegación entre las distintas páginas y la conexión inicial con el servicio de WebSockets para las notificaciones.

A partir de dicho inicio, podemos acceder a otras clases como ‘PaginaNoticias’ o ‘PaginaAprendizaje’ que ofrecen funcionalidades orientadas a noticias y recursos educativos, respectivamente, o la clase ‘PaginaCartera’ que permite a los usuarios ver y gestionar sus inversiones, reflejando la integración directa con el modelo de datos de la cartera y el controlador correspondiente para realizar operaciones como consultar o modificar inversiones.

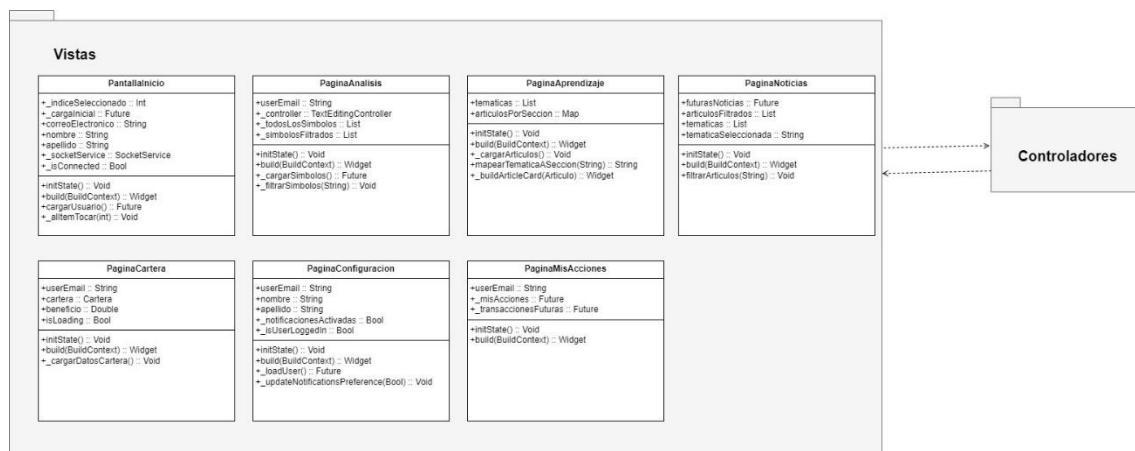


Ilustración 28 Diagrama de clases del paquete de Vistas (Front-end)



5.1.4.2 Diagrama de clases (Back-end)

Y pasándonos a la parte de nuestro servidor, inicialmente podemos encontrar el corazón del back-end, el **FlaskApp**, siendo nuestra clase central cuya función es inicializar y configurar la aplicación Flask. Este controlador es el responsable de registrar los “**blueprints**”, que son una especie de módulos para poder separar las distintas lógicas de la aplicación en espacios de nombres distintos, facilitando la mantenibilidad y la escalabilidad del código.

MarketController, **NewsController**, **PortfolioController**... etc. son representados como clases en el diagrama, donde cada uno de estos **Blueprints** gestiona rutas específicas y lógica asociada a distintas áreas funcionales de la aplicación. Por ejemplo, **MarketController** maneja las rutas relacionadas con los datos del mercado y **NewsController** gestiona la obtención de noticias financieras.

Accion, **Notificacion**, **Cartera**... etc. representan las entidades y su estructura en la base de datos. Estos **modelos** definen los campos, las relaciones y los métodos para interactuar con la base de datos, lo cual permite realizar operaciones CRUD en la aplicación.

Aparte de los controladores y modelos, el diagrama muestra el **Scheduler**, un componente responsable de ejecutar tareas programadas, como actualizar precios de acciones y enviar notificaciones de alerta.

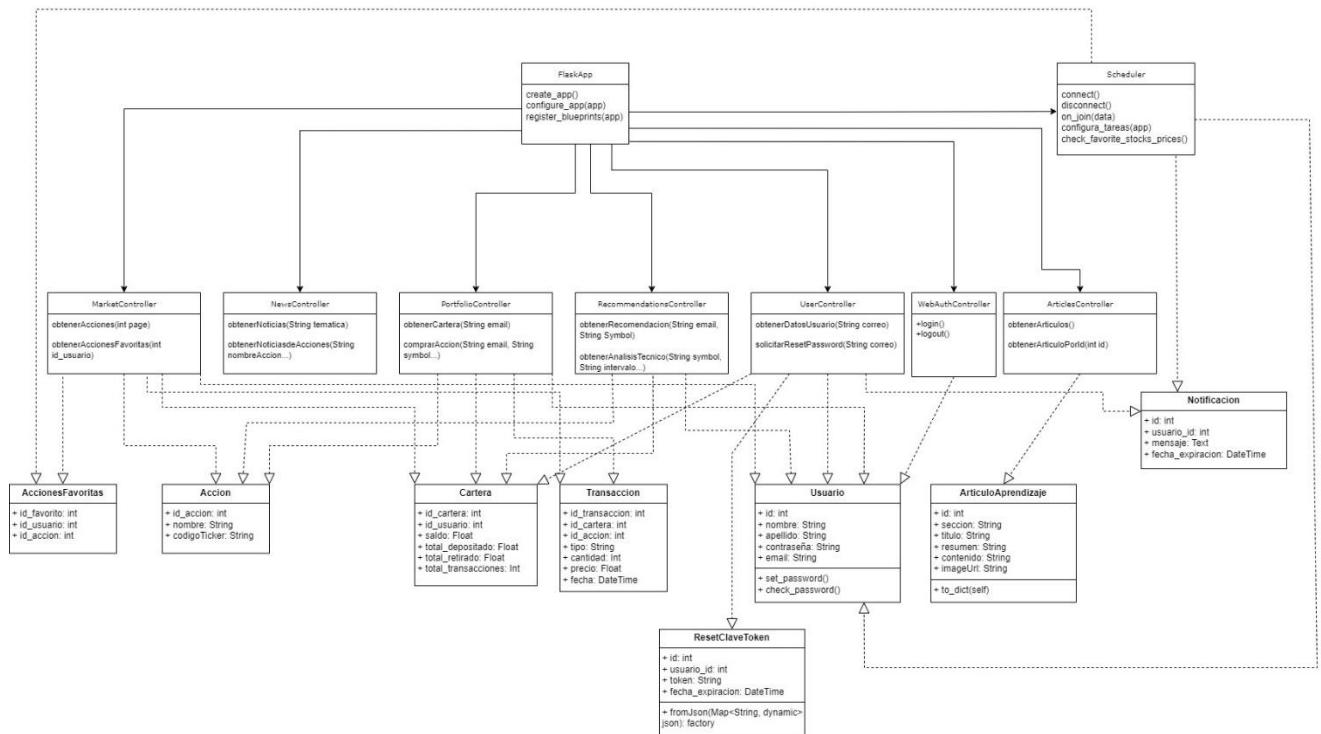


Ilustración 29 Diagrama de Clases (Back-end)



5.1.4.2.1 Diagrama de clases del paquete de Modelos (Back-end)

Si nos adentramos dentro de la estructura, podemos ver que los modelos representan una tabla en la base de datos y están diseñadas para poder guardar toda la información de las entidades relevantes de la aplicación.

Para estos modelos, no aportaremos una descripción para cada uno de ellos ya que previamente los hemos definido en el front-end, sin embargo, podemos comentar los que no han sido descritos, por ejemplo, tenemos el modelo ‘**Usuario**’ que incluye atributos como id, nombre, apellido, contraseña y email para poder gestionar su autenticación, así como métodos para poder establecer y verificar las contraseñas.

Por último, ‘**ResetClaveToken**’ sirve para la recuperación de contraseña, gracias a asignarle un token único a cada usuario para poder verificar la veracidad en el restablecimiento de la contraseña.

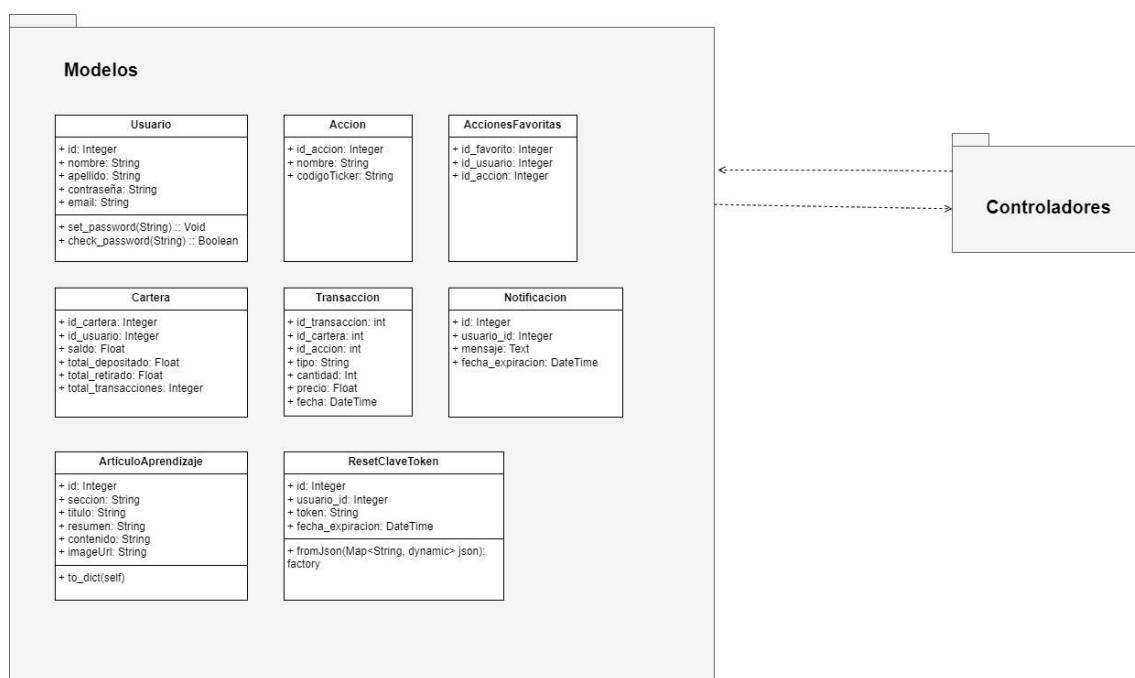


Ilustración 30 Diagrama de clases del paquete de Modelos (Back-end)



5.1.4.2.2 Diagrama de clases del paquete de Controladores (Back-end)

En el back-end de nuestra aplicación, los controladores son el eje principal del mecanismo de gestión de las funcionalidades del sistema, y a través de los ‘blueprints’, que ayudan a organizar el código en modulo claramente definidos, ayudan a mejorar la escalabilidad y la mantenibilidad de este.

Y como ya he hemos definido en el front-end los controladores, solo cabe mencionar el controlador ‘WebAuthController’, cuya funcionalidad se centra principalmente en la autenticación web, manejando las sesiones de los usuarios mediante operaciones de ‘login’ y ‘logout’.

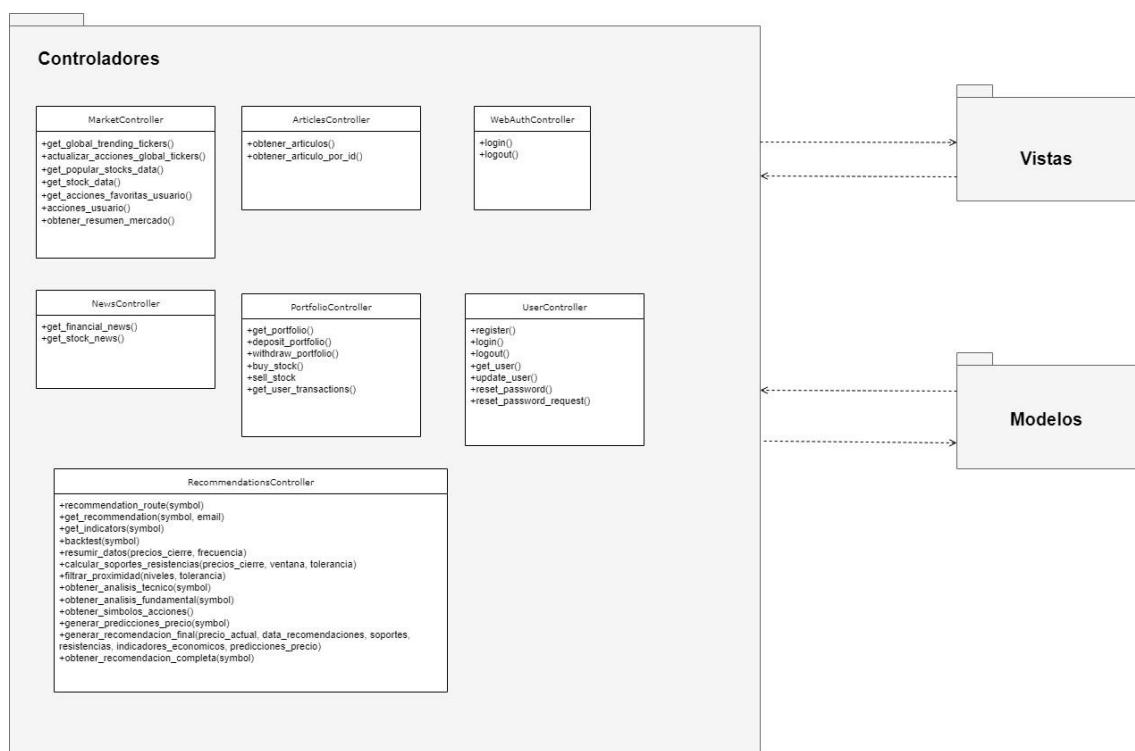


Ilustración 31 Diagrama de clases del paquete de Controladores (Back-end)



5.1.5. Implementación de la base de datos

Para poder implementar o crear la base de datos, he adoptado por utilizar Docker, conocido por ser un sistema de contenedores que me permite encapsular el servidor de la base de datos PostGreSQL y el servidor Flask en contenedores separados pero conectados entre sí.

Con esta arquitectura, que ha ido cogiendo fama en estos últimos años, ayuda notablemente en la creación de un entorno de desarrollo que pueda ser replicable y aislado, lo que facilita significativamente el proceso de despliegue y la configuración de las distintas máquinas o entornos en la nube, manteniendo siempre la coherencia entre los entornos de desarrollos, pruebas y producción.

La base de datos PostgreSQL se configura y gestiona a través de un contenedor Docker, el cual es definido y configurado mediante un docker-compose.yml, permitiendo especificar la versión de PostgreSQL, las variables de entorno para la configuración inicial y los volúmenes para persistir los datos más allá del ciclo de vida del contenedor.

Además, mediante scripts SQL o “backups”, se definen estructuras de tablas, relaciones, índices y restricciones, lo que nos permite una inicialización y configuración precisa de la base de datos al momento de desplegar el contenedor.



5.1.5.1 Estructura de Tablas:

Para ilustrar la implementación específica de la base de datos y tener una visión más detallada de cómo se estructura nuestra base de datos, proporcionamos las sentencias SQL que definen las tablas principales de nuestro esquema.

Estas sentencias son esenciales para establecer la arquitectura subyacente de nuestra aplicación y permiten definir la forma en que los datos se organizan y almacenan en la base de datos. Además, cada una de ellas incluye la definición de las columnas de la tabla, especificando el tipo de datos, las restricciones de integridad y, en algunos casos, los valores por defecto.

```
1 CREATE TABLE public.accion (
2     id_accion integer NOT NULL,
3     nombre character varying(255),
4     codigoticker character varying(10)
5 );
6
7 CREATE TABLE public.accionesfavoritas (
8     id_favorito integer NOT NULL,
9     id_usuario integer,
10    id_accion integer
11 );
12
13 CREATE TABLE public.articulosaprendizaje (
14     id integer NOT NULL,
15     seccion character varying(255) NOT NULL,
16     titulo character varying(255) NOT NULL,
17     imageurl text NOT NULL,
18     resumen text NOT NULL,
19     contenido text NOT NULL
20 );
21
22 CREATE TABLE public.cartera (
23     id_cartera integer NOT NULL,
24     id_usuario integer,
25     saldo double precision DEFAULT 10000 NOT NULL,
26     total_depositado double precision DEFAULT 0 NOT NULL,
27     total_retirado double precision DEFAULT 0 NOT NULL,
28     total_transacciones integer DEFAULT 0 NOT NULL
29 );
30
31 CREATE TABLE public.notificaciones (
32     id integer NOT NULL,
33     usuario_id integer NOT NULL,
34     mensaje text NOT NULL,
35     fecha timestamp without time zone DEFAULT CURRENT_TIMESTAMP
36 );
37
38 CREATE TABLE public.reset_clave_token (
39     id integer NOT NULL,
40     usuario_id integer NOT NULL,
41     token character varying(100) NOT NULL,
42     fecha_expiracion timestamp without time zone NOT NULL
43 );
```



```
44
45 CREATE TABLE public.transaccion (
46     id_transaccion integer NOT NULL,
47     id_cartera integer,
48     id_accion integer,
49     tipo character varying(10) NOT NULL,
50     cantidad integer NOT NULL,
51     precio double precision NOT NULL,
52     fecha timestamp without time zone DEFAULT CURRENT_TIMESTAMP NOT NULL
53 );
54
55 CREATE TABLE public.usuario (
56     id integer NOT NULL,
57     nombre character varying(255),
58     apellido character varying(255),
59     "contraseña" character varying(255),
60     email character varying(255)
61 );
```

Tabla 67 Creación de tablas en la Base de Datos



5.1.5.2 Relaciones y Restricciones

En el diseño de bases de datos relacionales, las relaciones y restricciones juegan un papel importante ya que son los que garantizan la coherencia, la integridad y el correcto acceso a los datos. Para lograr esto, hacemos uso de las claves primarias y foráneas para ver como se estructuran las relaciones de las tablas y como se mantienen los datos organizados y accesibles.

Utilizando dichas claves foráneas conseguimos implementar la integridad referencial, tratándose de un mecanismo que asegura que las relaciones entre tablas se mantengan coherentes. Por ejemplo, las restricciones FOREIGN KEY en las tablas accionesfavoritas, cartera, notificaciones, y transaccion establecen una relación directa con las claves primarias de otras tablas, asegurando que no existan referencias a registros que no existen.

Además, las restricciones de unicidad, como las implementadas en las tablas reset_clave_token y usuario, previenen la duplicación de información en campos específicos que deben ser únicos, como el email del usuario o el token de restablecimiento de contraseña.

```
1 ALTER TABLE ONLY public.accion
2   ADD CONSTRAINT accion_pkey PRIMARY KEY (id_accion);
3
4 ALTER TABLE ONLY public.accionesfavoritas
5   ADD CONSTRAINT accionesfavoritas_pkey PRIMARY KEY (id_favorito);
6
7 ALTER TABLE ONLY public.articulosaprendizaje
8   ADD CONSTRAINT articulosaprendizaje_pkey PRIMARY KEY (id);
9
10 ALTER TABLE ONLY public.cartera
11   ADD CONSTRAINT cartera_pkey PRIMARY KEY (id_cartera);
12
13 ALTER TABLE ONLY public.notificaciones
14   ADD CONSTRAINT notificaciones_pkey PRIMARY KEY (id);
15
16 ALTER TABLE ONLY public.reset_clave_token
17   ADD CONSTRAINT reset_clave_token_pkey PRIMARY KEY (id);
18
19 ALTER TABLE ONLY public.reset_clave_token
20   ADD CONSTRAINT reset_clave_token_token_key UNIQUE (token);
21
22 ALTER TABLE ONLY public.transaccion
23   ADD CONSTRAINT transaccion_pkey PRIMARY KEY (id_transaccion);
24
25 ALTER TABLE ONLY public.usuario
26   ADD CONSTRAINT usuario_email_key UNIQUE (email);
27
28 ALTER TABLE ONLY public.usuario
29   ADD CONSTRAINT usuario_pkey PRIMARY KEY (id);
30
31 ALTER TABLE ONLY public.accionesfavoritas
32   ADD CONSTRAINT accionesfavoritas_id_accion_fkey FOREIGN KEY (id_accion)
33 REFERENCES public.accion(id_accion);
34
35 ALTER TABLE ONLY public.accionesfavoritas
36   ADD CONSTRAINT accionesfavoritas_id_usuario_fkey FOREIGN KEY (id_usuario)
37 REFERENCES public.usuario(id);
38
39 ALTER TABLE ONLY public.cartera
```



```

40    ADD CONSTRAINT cartera_id_usuario_fkey FOREIGN KEY (id_usuario) REFERENCES
41 public.usuario(id);
42
43 ALTER TABLE ONLY public.notificaciones
44     ADD CONSTRAINT notificaciones_usuario_id_fkey FOREIGN KEY (usuario_id)
45 REFERENCES public.usuario(id);
46
47 ALTER TABLE ONLY public.reset_clave_token
48     ADD CONSTRAINT reset_clave_token_usuario_id_fkey FOREIGN KEY (usuario_id)
49 REFERENCES public.usuario(id);
50
51 ALTER TABLE ONLY public.transaccion
52     ADD CONSTRAINT transaccion_id_accion_fkey FOREIGN KEY (id_accion) REFERENCES
53 public.accion(id_accion);
54
55 ALTER TABLE ONLY public.transaccion
56     ADD CONSTRAINT transaccion_id_cartera_fkey FOREIGN KEY (id_cartera) REFERENCES
57 public.cartera(id_cartera);
58

```

Tabla 68 Restricciones en la Base de Datos

5.1.5.3 Secuencias de Inicialización

Además, la correcta inicialización de las secuencias es fundamental para mantener la consistencia de los identificadores únicos en nuestra base de datos. Mediante el uso de las siguientes sentencias:

```

1 SELECT pg_catalog.setval('public.accion_id_accion_seq', valor, true);
2
3 SELECT pg_catalog.setval('public.accionesfavoritas_id_favorito_seq', valor, true);
4
5 SELECT pg_catalog.setval('public.articulosaprendizaje_id_seq', valor, true);
6
7 SELECT pg_catalog.setval('public.cartera_id_cartera_seq', valor, true);
8
9 SELECT pg_catalog.setval('public.notificaciones_id_seq', valor, true);
10
11 SELECT pg_catalog.setval('public.reset_clave_token_id_seq', valor, true);

```

Tabla 69 Secuencias de Inicialización en la Base de Datos

Además, garantizamos la integridad y consistencia de nuestros datos al inicializar adecuadamente las secuencias en la base de datos. Mediante las sentencias mostradas, configuramos los identificadores únicos autoincrementales para que continúen desde el último valor insertado, lo que evita conflictos y mantiene la integridad de los datos al insertar nuevos registros.



5.1.6. API

En la actualidad, cuando se intenta desarrollar aplicaciones móviles modernas, generalmente implica la creación de APIs (Interfaces de programación de aplicaciones) que sean robustas y escalables que faciliten la comunicación entre el front-end y el back-end, independientemente de la plataforma.

Por ello, en este contexto, se ha implementado una API REST, para garantizar las características previamente explicadas, así como garantizar la integración fluida y eficiente entre los componentes de la aplicación móvil desarrollada en Flutter con los controladores del back-end.

En cuanto al lado del front-end, los controladores de este escenario, contienen toda la lógica de las operaciones de la red, como son las solicitudes HTTP a la API y de utilizar los modelos definidos para estructurar los datos recibidos de la API, como se muestra a continuación:

```
1 // Importaciones necesarias
2 import 'dart:convert'; // Importo el paquete 'dart:convert' para manejar la
3 codificación y decodificación JSON
4 import 'package:grownomics/modelos/Articulo.dart'; // Importo el modelo de datos de
5 Articulo.dart
6 import 'package:http/http.dart' as http; // Importo el paquete 'http' para realizar
7 solicitudes HTTP
8
9 class ArticuloController {
10   // URL base para las solicitudes HTTP
11   static const String _baseUrl = 'http://10.0.2.2:5000';
12
13   // Función asincrónica para obtener todos los artículos
14   static Future<List<Articulo>> obtenerArticulos() async {
15     // Construyo la URL para la solicitud GET
16     final url = Uri.parse('$_baseUrl/articles/get_articles');
17     // Realizo la solicitud GET y espera la respuesta
18     final respuesta = await http.get(url);
19
20     // Verifico si la solicitud fue exitosa (código de estado 200)
21     if (respuesta.statusCode == 200) {
22       // Decodifico el cuerpo de la respuesta JSON en una lista de objetos dinámicos
23       List<dynamic> body = jsonDecode(respuesta.body);
24       // Mapeo cada objeto dinámico a un objeto Articulo y crea una lista de
25       Articulo
26       List<Articulo> articulos = body
27         .map((dynamic item) => Articulo.fromJson(item))
28         .toList();
29       // Devuelvo la lista de artículos
30       return articulos;
31     } else {
32       // Si la solicitud no fue exitosa, lanzo una excepción
33       throw Exception('No se pudieron cargar los artículos');
34     }
35 }
```

Tabla 70 Llamadas a la API desde ArticuloController



Los modelos representan entidades de datos, como los artículos en este ejemplo, y definen la estructura de estos datos, como pueden ser los títulos, contenidos, fechas de publicación.

En la práctica, cuando una vista necesita presentar, por ejemplo, una lista de artículos, llama al controlador correspondiente, como ArticuloController. Este controlador, a su vez, realiza la solicitud a la API REST para obtener los datos, veamos como:

```
1 void cargarArticulos() async {
2     List<Articulo> articulosObtenidos =
3         await ArticuloController.obtenerArticulos(); // Obtengo los artículos del
4 API
5
6     // Agrupo los artículos por sección
7     for (var articulo in articulosObtenidos) {
8         articulosPorSección.update(
9             articulo.sección, (list) => list..add(articulo),
10            ifAbsent: () => [articulo]);
11    }
12
13    setState(() {});
14 }
```

Tabla 71 Obtener artículos con el controlador ArticulosController

Por otro lado, el back-end, desarrollado con Flask en Python, gestiona la lógica de negocio y el acceso a los datos. Los controladores y rutas definidas manejan las solicitudes entrantes, interactuando con la base de datos para servir los recursos solicitados en formato JSON:

```
1 from flask import Blueprint, jsonify, request
2 from ..models import ArticulosAprendizaje # Importa el modelo de la tabla de
3 artículos de aprendizaje
4 from ..extensions import db
5
6
7 # Crea un blueprint para los artículos de aprendizaje
8 article_bp = Blueprint('articles', __name__)
9
10 # Ruta para obtener todos los artículos de aprendizaje
11 @article_bp.route('/get_articles', methods=['GET'])
12 def obtener_articulos():
13     artículos = ArticulosAprendizaje.query.all()
14     # Convierte los objetos de los artículos en un formato JSON y devuelve la lista
15     de artículos
16     return jsonify([artículo.to_dict() for artículo in artículos]), 200
```

Tabla 72 Definición de Endpoints en el Backend



Además, tengo que hablar sobre dos API externas a mi servidor, en primer lugar, la API de Yahoo Finance, que he utilizado principalmente para obtener datos financieros en tiempo real, así como información histórica de acciones e índices. Un ejemplo de ello:

```
1 from flask import jsonify, request, Blueprint, current_app as app
2 from datetime import datetime, timedelta
3 import yfinance as yf
4
5 from app.models import Accion, AccionesFavoritas, Transaccion, Usuario, Cartera
6 from ..extensions import db
7
8 import numpy as np
9
10 finance_bp = Blueprint('finance', __name__)
11
12 @finance_bp.route('/popular_stocks_data')
13 def get_popular_stocks_data():
14     page = request.args.get('page', default=1, type=int)
15     per_page = request.args.get('per_page', default=10, type=int)
16
17     start_index = (page - 1) * per_page
18     acciones = Accion.query.offset(start_index).limit(per_page).all()
19
20     end_date = datetime.now()
21     start_date = end_date - timedelta(days=2)
22     popular_stocks_data = {}
23
24     for accion in acciones:
25         ticker_symbol = accion.codigoticker
26         ticker = yf.Ticker(ticker_symbol)
27         data = ticker.history(start=start_date, end=end_date)
28         if not data.empty:
29             last_row = data.iloc[-1]
30             stock_info = {
31                 'id': accion.id_accion, # Asume que tu modelo tiene este campo como
32                 ID
33                 'name': ticker.info.get('longName', 'Unknown'),
34                 'ticker_symbol': ticker_symbol,
35                 'current_price': last_row['Close'],
36                 'change': last_row['Close'] - last_row['Open'],
37                 'change_percent': ((last_row['Close'] - last_row['Open']) /
38 last_row['Open']) * 100,
39             }
40             popular_stocks_data[accion.id_accion] = stock_info
41         else:
42             print(f"No hay datos disponibles para el ticker: {ticker_symbol}")
43
44 return jsonify(list(popular_stocks_data.values()))
```

Tabla 73 Uso de la API Yahoo Finance

En el código proporcionado, se utiliza esta API para obtener los datos de las acciones más populares, para que los usuarios puedan tener acceso a información actualizada que puede influir en sus decisiones de inversión. Este proceso comienza con la obtención de los símbolos ticker de las acciones de interés y posteriormente, mediante la biblioteca yfinance, se consultan los datos recientes de dichas acciones, incluyendo el precio de cierre, el cambio en el precio, y el porcentaje de cambio.



Y en cuanto a la API de noticias financieras NewsAPI, lo he utilizado principalmente para mostrar noticias financieras en tiempo real es esencial para inversores y personas interesadas en el mercado, ya que les permite estar informados sobre eventos que podrían afectar sus decisiones de inversión. Una muestra de código sobre esta API sería lo siguiente:

```
1 from flask import Blueprint, jsonify, request
2 import requests
3 from requests.adapters import HTTPAdapter
4 from requests.packages.urllib3.util.retry import Retry
5 from dotenv import load_dotenv
6 import os
7
8 # Crear un Blueprint para las rutas de noticias
9 load_dotenv()
10
11 # Crear un Blueprint para las rutas de noticias
12 news_bp = Blueprint('news', __name__)
13
14 # Ruta para obtener noticias financieras
15 @news_bp.route('/financial_news')
16 def get_financial_news():
17     api_key = os.getenv('NEWS_API_KEY')
18     # Extraer el tema de las noticias financieras de los parámetros de la solicitud,
19     # o usar 'finanzas' como valor predeterminado
20     tematica = request.args.get('tematica', 'finanzas')
21     # Construir la URL de la API de noticias financieras con el tema proporcionado y
22     # la clave de API
23     url =
24 f'https://newsapi.org/v2/everything?q={tematica}&language=es&apiKey={api_key}'
25
26     # Obtener una sesión de requests configurada con reintento
27     session = get_api_session()
28     try:
29         # Realizar una solicitud GET a la URL de la API
30         response = session.get(url)
31         if response.status_code == 200: # Si la solicitud es exitosa
32             # Convertir los datos de respuesta JSON en un formato JSON válido y
33             # devolver los artículos de noticias
34             news_data = response.json()
35             return jsonify(news_data['articles'])
36         else: # Si la solicitud no es exitosa
37             # Devolver un mensaje de error junto con el código de estado
38             return jsonify({"error": "No se pudieron obtener las noticias"}),
39         response.status_code
40     except requests.exceptions.RequestException as e: # Capturar excepciones de
41     # solicitud
42         # Devolver un mensaje de error junto con el código de estado
43         return jsonify({"error": str(e)}), 500
44
```

Tabla 74 Uso de la API NewsApi

En esta implementación, se realiza una solicitud HTTP a la API de noticias, filtrando los resultados por temática financiera, a la par de que manejo las solicitudes y errores en este código con lo que consigo que los usuarios finales reciban datos confiables y pertinentes, mejorando así la experiencia del usuario al proporcionar un contexto más amplio sobre el mercado financiero.



5.2. Desarrollo

5.2.1 Gestión de inicio de sesión y registro

En primer lugar, vamos a hablar del sistema de gestión de inicio de sesión y registro, ya que es una funcionalidad bastante importante que asegura tanto la seguridad del acceso de los usuarios a la aplicación como una experiencia de usuario fluida y personalizada.

Este sistema permite que los usuarios se creen cuentas personales, accedan a ellas mediante autenticación y mantengan sus sesiones activas o las cierren según sea necesario, por lo que podríamos decir que es la base de cualquier aplicación a día de hoy.

En el lado del cliente, el proceso de inicio de sesión se maneja a través de una interfaz de usuario intuitiva y responsive. A continuación, muestro un ejemplo de dicha funcionalidad de inicio de sesión en el front-end:

```
1 // Función para manejar el inicio de sesión
2 Future<void> _iniciarSesion() async {
3     final correo = _controladorCorreo.text; // Obtener el correo electrónico ingresado
4     final contraseña = _controladorContraseña.text; // Obtener la contraseña ingresada
5
6     final inicioSesionExitoso = await UsuarioController.iniciarUsuario(
7         correo,
8         contraseña,
9     ); // Intentar iniciar sesión con las credenciales proporcionadas
10
11    if (inicioSesionExitoso) {
12        final preferencias = await SharedPreferences.getInstance(); // Obtener las
13        preferencias del usuario
14        await preferencias.setString('userEmail', correo); // Guardar el correo
15        electrónico del usuario
16        await preferencias.setBool('isUserLoggedIn', true); // Guardar la preferencia de
17        recordar al usuario
18
19        if (_estaRecordado) {
20            // Si se selecciona recordar al usuario
21            await preferencias.setBool('isUserRemember', true); // Guardar la preferencia
22            de recordar al usuario
23        }
24
25        // Inicio de sesión exitoso, redirigir a la página de inicio
26        Navigator.of(context).pushReplacementNamed('/home');
27    } else {
28        // Inicio de sesión fallido, mostrar un mensaje de error
29        ScaffoldMessenger.of(context).showSnackBar(
30            SnackBar(
31                content: Text('Inicio de sesión fallido. Verifica tus credenciales.'),
32                duration: Duration(seconds: 3),
33            ),
34        );
35    }
36}
```

Tabla 75 Inicio de sesión (Front-end)



El funcionamiento del código es bastante intuitivo, primero lo que hacemos es obtener el correo y la contraseña del usuario. Después, utilizamos un controlador para intentar iniciar sesión con esa información. Si todo va bien y el inicio de sesión es exitoso, guardamos algunas preferencias del usuario, como su correo electrónico y si quiere recordar su sesión o no.

Y ya en el backend, el manejo de la autenticación y el registro se realiza verificando las credenciales y gestionando los datos de los usuarios en nuestra base de datos tal que así:

```
1 @auth_bp.route('/login', methods=['POST'])
2 def login():
3     if request.method == 'POST':
4         email = request.form['email']
5         password = request.form['password']
6
7         # Verificar las credenciales del usuario
8         user = Usuario.query.filter_by(email=email).first()
9         if user and user.check_password(password):
10             login_user(user)
11             return jsonify({'success': True, 'message': 'Inicio de sesión exitoso.'}), 200
12         else:
13             return jsonify({'success': False, 'message': 'Credenciales incorrectas. Inténtalo de nuevo.'}), 401
```

Tabla 76 Inicio de sesión (Back-end)

Aquí se puede ver como se verifica si el usuario existe en el sistema y luego si la contraseña proporcionada coincide con la almacenada en la base de datos. Este proceso asegura que solo los usuarios con credenciales válidas puedan acceder a sus cuentas.



5.2.2 Sistema de recuperación de contraseña

Además, he implementado el sistema de recuperación de contraseña, siendo una función esencial en la gestión de seguridad de cualquier aplicación moderna, proporcionando a los usuarios la capacidad de recuperar el acceso a sus cuentas de manera segura y eficiente en caso de olvido de sus credenciales.

En el lado del cliente, el proceso comienza cuando el usuario indica que ha olvidado su contraseña. A continuación, se solicita al usuario que ingrese su dirección de correo electrónico, la cual es utilizada para verificar si existe una cuenta asociada en el sistema. Si es así, se procede a enviar un código de verificación al correo electrónico proporcionado:

```
1 // Método para enviar el código de verificación al correo electrónico
2 void _enviarCodigoVerificacion() async {
3     bool enviado = await UsuarioController.solicitarResetPassword(_email);
4     if (enviado) {
5         // Código enviado con éxito, mostrar formulario para ingresar código
6         setState(() {
7             _mostrarFormulario = true;
8         });
9     } else {
10        // Mostrar mensaje de error
11        showDialog(
12            context: context,
13            builder: (context) => AlertDialog(
14                title: Text('Error'),
15                content:
16                    Text('Hubo un problema al enviar el código de verificación.'),
17                actions: [
18                    TextButton(
19                        onPressed: () => Navigator.pop(context),
20                        child: Text('Aceptar'),
21                    ),
22                ],
23            ),
24        );
25    }
26}
```

Tabla 77 Envío de código de verificación (Front-end)

Una vez que el front-end solicita la recuperación de contraseña, el back-end asume la responsabilidad de generar un código de verificación único y de tiempo limitado. Este código se envía al correo electrónico del usuario para confirmar su identidad:

```
1 @auth_bp.route('/reset_password_request', methods=['POST'])
2 def reset_password_request():
3     data = request.get_json()
4     email = data.get('email')
5     usuario = Usuario.query.filter_by(email=email).first()
6
7     if usuario:
8         token = ''.join([str(random.randint(0, 9)) for _ in range(6)])
9         fecha_expiracion = datetime.utcnow() + timedelta(minutes=5)
10
```



```

11     # Crear el token de reseteo
12     reset_token = ResetClaveToken(usuario_id=usuario.id, token=token,
13 fecha_expiracion=fecha_expiracion)
14     db.session.add(reset_token)
15     db.session.commit()
16
17     send_verification_code(email, token) # Asume que esta función ya está
18 implementada
19
20     return jsonify({'message': 'Si el correo electrónico está registrado,
21 recibirás un código de verificación.'}), 200
22
23     return jsonify({'message': 'Si el correo electrónico está registrado, recibirás
24 un código de verificación.'}), 200
25
26 #-----
27
28 def send_verification_code(email_to, verification_code):
29
30     mail = current_app.extensions['mail']
31     subject = "Tu Código de Verificación"
32     sender = current_app.config['MAIL_DEFAULT_SENDER']
33     recipients = [email_to]
34     body = f"Tu código de verificación es: {verification_code}"
35
36     msg = Message(subject, sender=sender, recipients=recipients, body=body)
37
38     try:
39         mail.send(msg)
40         return True
41     except Exception as e:
42         print(e)
43         return False
44

```

Tabla 78 Envío de código de verificación (Back-end)

Posteriormente, el usuario deberá recibir un correo electrónico similar a este, el cual contendrá un código de verificación de seis dígitos:

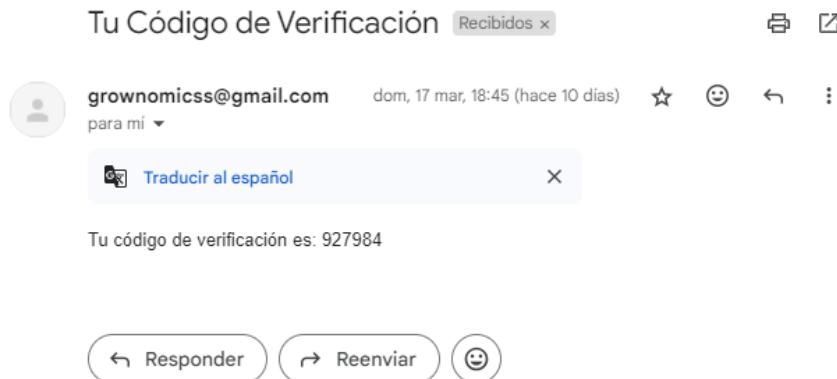


Ilustración 32 Correo con código de verificación



Después de recibir el código de verificación, el usuario lo ingresa en la aplicación. El front-end luego solicita el cambio de contraseña, proporcionando el código de verificación, el correo electrónico del usuario y la nueva contraseña deseada:

```

1 // Método para cambiar la contraseña
2 void _verificarYCambiarClave() async {
3     if (_nuevaClave != _confirmarClave) {
4         // Mostrar mensaje de que las contraseñas no coinciden
5         showDialog(
6             context,
7             builder: (context) => AlertDialog(
8                 title: Text('Error'),
9                 content: Text('Las contraseñas no coinciden.'),
10                actions: [
11                    TextButton(
12                        onPressed: () => Navigator.pop(context),
13                        child: Text('Aceptar'),
14                    ),
15                ],
16            );
17        );
18    }
19
20    bool cambiada =
21        await UsuarioController.resetPassword(_email, _codigoVerificacion,
22        nuevaClave);
23 }
```

Tabla 79 Verificar código y cambiar contraseña (Front-end)

Y ya en el back-end se valida el token proporcionado por el usuario, asegurándose de que coincida con el enviado y que no haya expirado. Si la validación es exitosa, se procede a actualizar la contraseña del usuario en la base de datos, asegurando así que el usuario pueda acceder a su cuenta con las nuevas credenciales:

```

@auth_bp.route('/reset_password', methods=['POST'])
def reset_password():
1     data = request.get_json()
2     email = data.get('email')
3     token_provided = data.get('code')
4     new_password = data.get('new_password')
5
6     usuario = Usuario.query.filter_by(email=email).first()
7     reset_token = ResetClaveToken.query.filter_by(usuario_id=usuario.id,
8 token=token_provided).first()
9
10    if usuario and reset_token and reset_token.fecha_expiracion >=
11 datetime.utcnow():
12        usuario.set_password(new_password)
13        db.session.delete(reset_token) # Eliminar el token ya que ya no será
14 necesario
15        db.session.commit()
16        return jsonify({'message': 'Contraseña actualizada con éxito.'}), 200
17    else:
18        return jsonify({'error': 'Código de verificación incorrecto, expirado o
19 usuario no encontrado.'}), 400
```

Tabla 80 Verificar código y cambiar contraseña (Back-end)



5.2.3 Sistema de Simulación Virtual

Una de las novedades que se salen de lo tradicional en esta aplicación, es el sistema de simulación virtual que permite a los usuarios experimentar con inversiones en el mercado de valores sin el riesgo de perder dinero real.

Esta funcionalidad no solo es educativa sino también una manera de fomentar la participación del usuario en la aplicación. Al registrarse en la aplicación, a cada usuario se le asigna una cartera virtual con un saldo base de 10.000€, lo que les permite comenzar a operar inmediatamente, como se puede ver en este código:

```
1 @auth_bp.route('/register', methods=['POST'])
2 def register():
3     if request.method == 'POST':
4         # Obtener los datos del formulario de registro
5         username = request.form.get('nombre')
6         password = request.form.get('password')
7         email = request.form.get('email')
8         apellido = request.form.get('apellido')
9
10        # Verificar si ya existe un usuario con el mismo correo electrónico
11        existeUsuario = Usuario.query.filter_by(email=email).first()
12        if existeUsuario:
13            return jsonify({'success': False, 'message': 'Ya existe un usuario con
14 este correo electrónico.'}), 409
15
16        # Crear un nuevo usuario
17        new_user = Usuario(nombre=username, apellido=apellido, email=email)
18        new_user.set_password(password) # Hashear la contraseña
19
20        db.session.add(new_user)
21        db.session.flush() # Obtener el ID del nuevo usuario antes de hacer commit
22
23        # Crear una nueva cartera para el usuario
24        new_cartera = Cartera(id_usuario=new_user.id)
25        db.session.add(new_cartera)
26
27        db.session.commit()
28
29        return jsonify({'success': True, 'message': 'Registro exitoso. Ahora puedes
30 iniciar sesión y tienes una cartera creada.'}), 201
```

Tabla 81 Registrar usuario y crear cartera (Back-end)

Este código gestiona la creación de un nuevo usuario y su cartera asociada tras completar el proceso de registro, asegurando que cada usuario tenga los recursos necesarios para comenzar a explorar las funcionalidades de inversión de la aplicación.



Una vez tenemos nuestra cartera, podemos realizar varias funciones, como depositar dinero virtual o retirarlo, ver también nuestras transacciones realizadas entre otras cosas, pero nos vamos a centrar sobre todo en como compramos o vendemos acciones, en el front-end tenemos lo siguiente:

```
1 ElevatedButton(
2   onPressed: cantidad <= 0 ? null
3     : () async {
4       bool resultado =
5         await CarteraController.comprarAccion(
6           widget.correoElectronico,
7           widget.simboloAccion,
8           precio,
9           cantidad,
10          );
11       if (resultado) {
12         ScaffoldMessenger.of(context).showSnackBar(
13           SnackBar(
14             content: Text("Compra realizada con éxito"),
15           ),
16         );
17         setState(() {
18           cantidad = 0; // Restablece la cantidad a cero
19         });
20         _textFieldController
21           .clear(); // Restablece el valor del TextField a cero
22       } else {
23         ScaffoldMessenger.of(context).showSnackBar(
24           SnackBar(
25             content: Text("Error al realizar la compra"),
26           ),
27         );
28       }
29 },
```

Tabla 82 Comprar acción (Front-end)

Lo que hace esto, es que cuando se presiona el botón, primero verifica si la cantidad de acciones a comprar es mayor que cero. Si la cantidad es mayor a cero, entonces intenta realizar una compra de acciones utilizando el controlador de la cartera. La función espera un resultado booleano que indica si la compra fue exitosa o no.

Si la compra fue exitosa, muestra un mensaje emergente indicando que la compra se realizó con éxito, restablece la cantidad de acciones a cero y limpia el campo de texto utilizado para ingresar la cantidad de acciones. En caso de que ocurra un error durante la compra, muestra un mensaje emergente indicando que hubo un error al realizar la compra.



Cuando el back-end recibe una solicitud de compra de acciones en el sistema de simulación virtual, comienza por verificar el saldo disponible en la cartera del usuario para asegurar que cuenta con fondos suficientes para realizar la operación. Tras esta verificación, el sistema procede a actualizar el saldo de la cartera, deduciendo el monto correspondiente al costo de las acciones compradas.

Además, el paso crítico en este proceso es la creación de una transacción detallada asociada directamente a la cartera del usuario, la cual registra específicamente la compra realizada, incluyendo datos como el tipo de acción, la cantidad, y el precio pagado:

```
1 # Funcion para comprar una accion
2 @portfolio_bp.route('/buy_stock', methods=['POST'])
3 def buy_stock():
4     data = request.get_json()
5     print("Datos recibidos en la solicitud POST:", data) # Imprimir los datos
6 recibidos en la solicitud
7
8     user_email = data['email']
9     stock_symbol = data['symbol']
10    cantidad = int(data['cantidad'])
11    precio = float(data['precio'])
12
13    # Comprobaciones
14    # ...
15
16    total_cost = precio * cantidad # Calcular el costo total de la compra
17    if cartera.saldo < total_cost:
18        print("Saldo insuficiente en la cartera del usuario:", usuario.id)
19        return jsonify({'error': 'Saldo insuficiente'}), 400 # Devolver un error
20 si el saldo es insuficiente
21
22    cartera.saldo -= total_cost # Restar el costo total de la compra al saldo de
23 la cartera
24    cartera.total_transacciones += 1 # Incrementar el contador de transacciones de
25 la cartera
26
27    transaccion = Transaccion(id_cartera=cartera.id_cartera,
28 id_accion=accion.id_accion, tipo='compra', cantidad=cantidad, precio=precio) # # Crear una nueva transacción de compra
29 db.session.add(transaccion) # Agregar la transacción a la sesión de la base de
30 datos
31 db.session.commit() # Confirmar la transacción en la base de datos
32
33    print("Compra realizada con éxito para el usuario:", usuario.id)
34    return jsonify({'success': True, 'message': 'Compra realizada con éxito'}), 200
35 # Devolver una respuesta exitosa
```

Tabla 83 Comprar acción (Back-end)



5.2.4 Sistema de Recomendación

Ahora entramos en la parte más crucial de la aplicación, donde realmente se decide su valor. Se trata de la implementación del sistema de recomendación, un componente que orienta al usuario sobre qué acciones tomar basándose en distintos análisis de la acción en cuestión.

Para empezar, tengo un método en el front-end que se encarga de obtener las recomendaciones. Lo que hago es solicitar un análisis completo de una acción específica, basándome en su símbolo y el correo electrónico del usuario, como se puede ver aquí:

```
1 void obtenerRecomendaciones() async {
2     try {
3         final resultado = await RecomendacionesController.obtenerAnalisisCompleto(
4             widget.simboloAccion,
5             widget
6                 .correoElectronico); // Obtener recomendaciones según el símbolo
7         setState(() {
8             recomendaciones = resultado;
9             _isLoading = false;
10        });
11    } catch (e) {
12        print("Error al obtener recomendaciones: $e");
13        setState(() {
14            _isLoading = false;
15        });
16    }
17 }
```

Tabla 84 Obtener recomendaciones (Front-end)

En el back-end, genero la recomendación final analizando una combinación de factores como estrategias de inversión, soportes, resistencias, indicadores económicos y predicciones de precios.

Aquí está cómo lo hago:

```
1 def generar_recomendacion_final(precio_actual, data_recomendaciones, soportes,
2 resistencias, indicadores_economicos, predicciones_precio):
3     recomendaciones_estrategias = data_recomendaciones.get('recommendations', [])
4
5     recomendacion_final = "Mantener"
6     precio_compra_recomendado = None
7     precio_venta_recomendado = None
8     puntos_compra = 0
9     puntos_venta = 0
10
11     # Evaluar recomendaciones de estrategias
12     for estrategia in recomendaciones_estrategias:
13         if estrategia['recomendacion'] == "Recomendación de compra.":
14             puntos_compra += 1
15         elif estrategia['recomendacion'] == "Recomendación de venta.":
16             puntos_venta += 1
17
18     # Evaluar indicadores técnicos
19     rsi = indicadores_economicos.get('RSI')
```



```

20     if rsi and rsi < 30:
21         puntos_compra += 1
22     elif rsi and rsi > 70:
23         puntos_venta += 1
24
25     # Evaluar soportes y resistencias para determinar precios recomendados de compra
26 y venta
27     if soportes:
28         precio_soporte_cercano = min(soportes, default=precio_actual)
29         precio_compra_recomendado = precio_soporte_cercano * 0.98 # Comprar un 2%
30 antes del soporte más cercano
31         puntos_compra += 1
32     if resistencias:
33         precio_resistencia_cercano = max(resistencias, default=precio_actual)
34         precio_venta_recomendado = precio_resistencia_cercano * 1.02 # Vender un 2%
35 después de la resistencia más cercana
36         puntos_venta += 1
37
38     # Evaluar predicciones de precio
39     precios_futuros = list(predicciones_precio.values())
40     if precios_futuros:
41         promedio_precio_futuro = sum(precios_futuros) / len(precios_futuros)
42         if promedio_precio_futuro > precio_actual:
43             puntos_compra += 1 # Tendencia a aumentar
44         else:
45             puntos_venta += 1 # Tendencia a disminuir
46
47     # Decidir recomendación basada en puntos acumulados
48     if puntos_compra > puntos_venta:
49         recomendacion_final = f"Comprar - Basado en {puntos_compra} puntos de compra
50 frente a {puntos_venta} puntos de venta."
51     elif puntos_venta > puntos_compra:
52         recomendacion_final = f"Vender - Basado en {puntos_venta} puntos de venta
53 frente a {puntos_compra} puntos de compra."
54
55     return {
56         "recomendacion": recomendacion_final,
57         "precio_compra_recomendado": precio_compra_recomendado,
58         "precio_venta_recomendado": precio_venta_recomendado
59     }
60

```

Tabla 85 Obtener recomendaciones (Back-end)

Este fragmento de código se basa en una evaluación cuidadosa de cada uno de los factores mencionados para generar una recomendación que pueda guiar al usuario en su toma de decisiones. Lo interesante aquí es cómo pondero las distintas recomendaciones y análisis para llegar a una conclusión coherente y útil para el usuario.



Comienzo por evaluar una serie de estrategias de inversión, cada una basada en principios y datos históricos distintos. Este análisis no es trivial; implica una cuidadosa revisión de cómo cada estrategia ha rendido a lo largo del tiempo, lo que me permite ponderar sus recomendaciones con precisión.

Por ejemplo, examino desde cruces de medias móviles hasta indicadores de fuerza relativa (RSI), cada uno ofreciendo una perspectiva única sobre la acción. El proceso se profundiza con el backtesting de estas estrategias, una técnica que aplico para simular cómo habrían actuado en el pasado. Esta retrospectiva es invaluable, ya que me da una visión clara de la eficacia de cada estrategia en diferentes condiciones de mercado.

No se trata solo de elegir las estrategias más exitosas, sino de entender en qué contextos específicos cada una podría ser más útil. Así, puedo ajustar las recomendaciones basadas en el comportamiento actual del mercado, aumentando la relevancia y precisión de mis consejos.

```
1 # Ruta para obtener recomendaciones basadas en estrategias de trading
2 def get_recommendation(symbol, email):
3     usuario = Usuario.query.filter_by(email=email).first()
4
5     if usuario:
6         cartera = Cartera.query.filter_by(id_usuario=usuario.id).first()
7         saldo = cartera.saldo if cartera else 10000
8     else:
9         saldo = 10000
10
11    data = yf.download(symbol, period="1y")
12    recommendation_details = []
13
14    strategies = [
15        SmaCross,
16        RsiStrategy,
17        BollingerBandsStrategy,
18        MACDStrategy,
19        CCIStrategy,
20        EMAcrossStrategy,
21        StochasticStrategy,
22        OBVStrategy,
23        HMAStrategy
24    ]
25
26    for strategy_class in strategies:
27        bt = Backtest(data, strategy_class, cash=saldo, commission=.002)
28        stats = bt.run()
29
30        recommendation = "Sin recomendación clara."
31        if not stats['_trades'].empty:
32            last_trade = stats['_trades'].iloc[-1]
33            if last_trade['Size'] > 0:
34                recommendation = "Recomendación de compra."
35            elif last_trade['Size'] < 0:
36                recommendation = "Recomendación de venta."
37
38        recommendation_details.append({
39            "estrategia": strategy_class.name,
40            "descripcion": strategy_class.description,
41            "recomendacion": recommendation,
```



```

42         "equity_final": stats['Equity Final [$)'],
43         "retorno": stats['Return [%]']
44     })
45
46     if recommendation_details:
47         recommendation_summary = "Recomendaciones: " + ", ".join(
48             [d['estrategia'] + ": " + d['recomendacion']] for d in
49             recommendation_details)
50     else:
51         recommendation_summary = "Mantener. No hay recomendaciones claras basadas en
52 las estrategias aplicadas."
53
54     return {'symbol': symbol, 'recommendations': recommendation_details, 'summary':
55 recommendation_summary}
56
57 # Ruta para realizar un backtest de una estrategia de trading para un símbolo dado
58 @recommendations_bp.route('/backtest/<string:symbol>', methods=['GET'])
59 def backtest(symbol):
60     start_date = request.args.get('start', "2020-01-01") # Obtiene la fecha de
61 inicio de la solicitud (por defecto: 1 de enero de 2020)
62     end_date = request.args.get('end', datetime.now().strftime("%Y-%m-%d")) # # Obtiene la fecha de finalización de la solicitud (por defecto: fecha actual)
63     data = yf.download(symbol, start=start_date, end=end_date) # Descarga datos de
64 mercado para el símbolo y el rango de fechas especificado
65
66     bt = Backtest(data, SmaCross, cash=10000, commission=.002) # Inicializa el
67 backtest con la estrategia de cruce de medias móviles simples y el efectivo inicial
68 de $10,000
69     stats = bt.run() # Ejecuta el backtest y obtiene las estadísticas
70     bt.plot() # Genera un gráfico del backtest
71
72     # Retorna un mensaje de éxito y las estadísticas del backtest como un objeto
73 JSON
74     return jsonify({'success': True, 'message': 'Backtesting completado', 'stats':
75 stats._trade_data})

```

Tabla 86 Obtener estrategias con backtesting (Back-end)



En cuanto a los soportes y resistencias, estos representan los pisos y techos psicológicos dentro de la dinámica de precios de las acciones. Identificar estos niveles me permite establecer puntos de compra y venta recomendados que son fundamentales para las decisiones de inversión.

Esta parte del análisis implica un minucioso estudio de los gráficos de precios, buscando esos puntos donde el precio ha mostrado una tendencia a revertirse o estancarse. Al ajustar las recomendaciones para considerar estos niveles, proporciono a los usuarios orientaciones más estratégicas, potencialmente maximizando sus ganancias o minimizando pérdidas.

```
1 def calcular_soportes_resistencias(precios_cierre, ventana=20, tolerancia=0.02):
2
3     minimos = precios_cierre.rolling(window=ventana, center=True).min()
4     maximos = precios_cierre.rolling(window=ventana, center=True).max()
5
6     soportes = []
7     resistencias = []
8
9     for i in range(len(precios_cierre)):
10         precio_actual = precios_cierre[i]
11         min_local = minimos[i]
12         max_local = maximos[i]
13
14         if abs(precio_actual - min_local) / precio_actual < tolerancia:
15             soportes.append(precio_actual)
16         elif abs(precio_actual - max_local) / precio_actual < tolerancia:
17             resistencias.append(precio_actual)
18
19     # Filtrar soportes y resistencias para reducir la proximidad
20     soportes_filtrados = filtrar_proximidad(soportes, tolerancia)
21     resistencias_filtradas = filtrar_proximidad(resistencias, tolerancia)
22
23     return soportes_filtrados, resistencias_filtradas
24
25 def filtrar_proximidad(niveles, tolerancia):
26
27     niveles_filtrados = []
28     ultimo_nivel = None
29
30     for nivel in sorted(niveles):
31         if ultimo_nivel is None or abs(nivel - ultimo_nivel) / nivel > tolerancia:
32             niveles_filtrados.append(nivel)
33             ultimo_nivel = nivel
34
35     return niveles_filtrados
```

Tabla 87 Obtener soportes y resistencias (Back-end)



Por último, la predicción de precios a través del modelo ARIMA basado en análisis estadísticos de series temporales, me permiten proyectar los movimientos futuros de los precios con un grado razonable de certeza.

Integrar estos pronósticos en mi recomendación final no solo añade una dimensión anticipatoria al consejo, sino que también alinea las expectativas de los usuarios con la realidad del mercado, aunque siempre siendo consciente de sus respectivos riesgos que puedan conllevar.

```
1 def generar_predicciones_precio(symbol):
2     # Creo un modelo ARIMA basado en datos históricos de precios utilizando mi
3     # símbolo favorito
4     data = yf.download(symbol, period="2y")
5
6     # Extraigo precios de cierre y los convierto a tipo flotante
7     precios_cierre = data['Close'].astype(float)
8
9     # Ajusto la frecuencia de los precios de cierre a días laborables, porque es más
10    # realista
11    precios_cierre = precios_cierre.asfreq('B') # 'B' representa días laborables
12
13    # Creo un modelo ARIMA y lo ajusto a mis datos para hacer predicciones de
14    # precios
15    modelo_arima = ARIMA(precios_cierre, order=(5,1,0))
16    modelo_ajustado = modelo_arima.fit()
17
18    # Decido cuántos días quiero predecir hacia el futuro
19    n_dias = 30
20
21    # Hago mis propias predicciones de precios para los próximos 'n_dias' días
22    pronostico = modelo_ajustado.forecast(steps=n_dias)
23
24    # Creo fechas futuras para mis predicciones
25    fechas_futuras = pd.date_range(start=precios_cierre.index[-1], periods=n_dias+1)
26
27    # Filtro y excluyo los valores NaN de mis predicciones, porque no me gustan los
28    # datos incorrectos
29    predicciones = pd.Series(pronostico, index=fechas_futuras)
30    predicciones_filtradas = {fecha.strftime("%Y-%m-%dT%H:%M:%S"): valor for fecha,
31    valor in predicciones.items() if not math.isnan(valor)}
32
33    return predicciones_filtradas
34
```

Tabla 88 Obtener predicciones de los precios (Back-end)



5.2.5 Sistema de Notificaciones

En la última etapa de desarrollo de nuestra aplicación, implementé un sistema de notificaciones que juega un papel crucial en la experiencia del usuario. Este sistema está diseñado para informar a los usuarios en tiempo real sobre cambios significativos en las acciones que han marcado como favoritas.

Con este sistema, cada tres horas, los usuarios reciben actualizaciones sobre la variación de dichas acciones, lo que les permite tomar decisiones informadas rápidamente.

Desde el momento en que el usuario inicia la aplicación, realizo una serie de comprobaciones para determinar si tiene activadas las notificaciones. Si es así, procedo a establecer una conexión WebSocket para iniciar el servicio de notificaciones.

Aquí está el fragmento de código que maneja esta lógica en el frontend:

```
1 if (isUserLoggedIn) {  
2     final correoElectronico = prefs.getString('userEmail') ?? '';  
3     // Iniciar conexión WebSocket aquí  
4     final socketService =  
5         SocketService(); // Asume que tienes una instancia de tu servicio de socket  
6     socketService.connectAndListen(  
7         correoElectronico); // Modifica tu método para aceptar el correo como param  
8 }
```

Tabla 89 Comprobaciones Notificaciones (Front-end)

Este código verifica si el usuario está registrado y, en caso afirmativo, inicia el servicio de notificaciones utilizando la dirección de correo electrónico del usuario como identificador único para la conexión WebSocket.

Dentro del servicio de notificaciones en el frontend, tengo una estructura compleja que maneja la conexión y escucha los eventos del socket, además de gestionar las notificaciones locales mediante el plugin FlutterLocalNotificationsPlugin. La inicialización de las notificaciones locales y la conexión al servidor por WebSocket se gestionan de la siguiente manera:

```
1 // Clase para gestionar la conexión y escuchar eventos del socket  
2 class SocketService {  
3     late IO.Socket socket; // Socket para la comunicación  
4  
5     // Plugin para gestionar las notificaciones locales  
6     final FlutterLocalNotificationsPlugin flutterLocalNotificationsPlugin =  
7         FlutterLocalNotificationsPlugin();  
8  
9     // Método para inicializar las notificaciones locales  
10    Future<void> initNotifications() async {  
11        // Configuración de inicialización para Android  
12        const AndroidInitializationSettings initializationSettingsAndroid =  
13            AndroidInitializationSettings('@mipmap/ic_launcher');  
14        final InitializationSettings initializationSettings =  
15            InitializationSettings(android: initializationSettingsAndroid);  
16        await flutterLocalNotificationsPlugin.initialize(initializationSettings);  
17    }  
18  
19    // Método para conectar y escuchar eventos del socket
```



```

20 void connectAndListen(String userEmail) async {
21     await initNotifications(); // Inicializa las notificaciones locales
22
23     // Conexión al servidor por WebSocket
24     socket = IO.io(
25         baseUrl,
26         IO.OptionBuilder()
27             .setTransports(
28                 ['websocket']) // Configura el transporte como WebSocket
29             .disableAutoConnect() // Deshabilita la conexión automática
30             .build(),
31     );
32
33     socket.connect(); // Conecta al servidor
34
35     // Evento al establecer conexión con el servidor
36     socket.onConnect((_) {
37         print('Conectado a WebSocket: me uno a la sala $userEmail');
38         socket.emit('join',
39             {'email': userEmail}); // Se une a una sala con el email del usuario
40     });
41
42     // Evento de alerta de stock
43     socket.on('stock_alert', (data) async {
44         if (data is Map<String, dynamic>) {
45             final title = 'Alerta de Acción';
46             final body = data['message']; // Mensaje de la alerta
47             await _showNotification(
48                 title, body ?? 'Detalle no disponible'); // Muestra la notificación
49             }
50     });
51
52     // Evento de respuesta de prueba
53     socket.on('test_response', (data) async {
54         final title = 'Respuesta de Test';
55         final body = data.toString(); // Cuerpo de la respuesta
56         await _showNotification(title, body); // Muestra la notificación
57     });
58 }
59
60 // Dentro de SocketService
61
62 // Método para verificar si las notificaciones están habilitadas
63 Future<bool> areNotificationsEnabled() async {
64     final SharedPreferences prefs = await SharedPreferences.getInstance();
65     // Lee el valor de la preferencia, asume true por defecto
66     return prefs.getBool('notificationsEnabled') ?? true;
67 }
68
69 // Método privado para mostrar una notificación
70 Future<void> _showNotification(String title, String body) async {
71     if (await areNotificationsEnabled()) {
72         // Configuración de la notificación para Android
73         const AndroidNotificationDetails androidPlatformChannelSpecifics =
74             AndroidNotificationDetails(
75                 'your channel id',

```



```

76     'Notificacion grownomics',
77     'Notificacion',
78     importance: Importance.max,
79     priority: Priority.high,
80     ticker: 'ticker',
81   );
82   const NotificationDetails platformChannelSpecifics =
83     NotificationDetails(android: androidPlatformChannelSpecifics);
84   await flutterLocalNotificationsPlugin.show(
85     0,
86     title,
87     body,
88     platformChannelSpecifics,
89     payload: 'item x',
90   );
91 }
92 }
```

Tabla 90 Estructura del servicio Socket (Front-end)

Este servicio se encarga de establecer la conexión con el servidor, unirse a la sala específica del usuario y escuchar los eventos de alertas de acciones, para luego mostrar las notificaciones correspondientes.

En el backend, la lógica es aún más compleja. Utilizo APScheduler para programar tareas que verifican los precios de las acciones favoritas de los usuarios cada tres horas. Cuando se detecta una variación significativa, se envía una notificación al usuario a través de email y se emite un evento de alerta de stock a través de WebSocket, que el frontend escucha y maneja adecuadamente:

```

1 # Instancia del scheduler
2 scheduler = APScheduler()
3
4 @socketio.on('connect')
5 def connect():
6     print("\n Cliente conectado al socket \n")
7
8 @socketio.on('disconnect')
9 def disconnect():
10    print('\n Cliente desconectado del socket \n')
11
12 # Definir la función para manejar el evento 'test'
13 @socketio.on('test')
14 def handle_test_event(message):
15     logger.info(f"\n Me llega este mensaje del cliente: {message} \n")
16     socketio.emit('test_response', 'Recibo tu mensaje cliente')
17
18 @socketio.on('join')
19 def on_join(data):
20     email = data['email']
21     room = email_to_room_mapping(email) # Asegúrate de que esta función devuelva
22     un identificador único de sala basado en el email
23     join_room(room)
24     print(f"{email} se ha unido a la sala {room}.")
```



```

25
26 #-----
27 -
28 #-----
29 -
30
31 # Define la configuración de tareas en APScheduler
32 def configura_tareas(app):
33     if not scheduler.running:
34         scheduler.init_app(app)
35         scheduler.start()
36
37     # Guardar la instancia de app en el scheduler para usarla luego
38     scheduler.app = app
39
40     # Agrega aquí todas las tareas y su programación
41     scheduler.add_job(id='check_favorite_stocks_prices',
42 func=check_favorite_stocks_prices, trigger='interval', minutes=180)
43     #scheduler.add_job(id='test_alert', func=test_alert, trigger='interval',
44 seconds=20)
45
46 # Tarea para verificar los precios de acciones favoritas
47 def check_favorite_stocks_prices():
48     with scheduler.app.app_context(): # Asegúrate de ejecutar dentro del contexto
49 de la aplicación
50     usuarios = Usuario.query.all()
51     for usuario in usuarios:
52         acciones_favoritas =
53 AccionesFavoritas.query.filter_by(id_usuario=usuario.id).all()
54         for favorita in acciones_favoritas:
55             # Utiliza la relación `accion` para obtener la instancia de Accion
56 asociada
57             accion = favorita.accion
58             if accion: # Comprueba que la relación haya devuelto una instancia
59 válida
60                 ticker_symbol = accion.codigoticker
61                 ticker = yf.Ticker(ticker_symbol)
62                 data = ticker.history(period="1d")
63
64                 if not data.empty:
65                     last_row = data.iloc[-1]
66                     change_percent = ((last_row['Close'] - last_row['Open']) /
67 last_row['Open']) * 100
68
69                     #if abs(change_percent) >= 5:
70                     send_notification(usuario.email, ticker_symbol,
71 change_percent)
72                     logger.info(f"Correo enviado a {usuario.email} sobre la
73 acción {ticker_symbol}.")
74
75 # Función para enviar notificaciones
76 def send_notification(email, ticker_symbol, change_percent):
77     with scheduler.app.app_context():
78
79         usuario = Usuario.query.filter_by(email=email).first()
80         if usuario:

```



```

81     mensaje = f"Tu acción favorita {ticker_symbol} ha tenido una variación
82 significativa de {change_percent:.2f}% en las últimas 24 horas."
83     nueva_notificacion = Notificacion(usuario_id=usuario.id,
84 mensaje=mensaje)
85     db.session.add(nueva_notificacion)
86     db.session.commit()
87
88     mail = scheduler.app.extensions['mail']
89
90     msg = Message("Alerta de Acción Favorita",
91                 sender=scheduler.app.config['MAIL_DEFAULT_SENDER'],
92                 recipients=[email],
93                 body=f"Tu acción favorita {ticker_symbol} ha tenido una
94 variación significativa de {change_percent:.2f}% en las últimas 24 horas.")
95
96     room = email_to_room_mapping(email) # Implementa esta función según tus
97 necesidades
98     socketio.emit('stock_alert', {
99         'message': f"Tu acción favorita {ticker_symbol} ha tenido una variación
100 significativa de {change_percent:.2f}% en las últimas 24 horas."
101     }, room=room)
102
103     mail.send(msg)
104
105 def email_to_room_mapping(email):
106     # Este es un ejemplo simple que usa el email como identificador de sala.
107     # En una implementación real, se debería considerar un identificador único más
108 seguro.
109     return email

```

Tabla 91 Estructura del servicio Socket (Back-end)



5.3. Despliegue del proyecto

Para el despliegue de nuestro proyecto, como ya he adelantado y mencionado anteriormente, he decidido utilizar los contenedores de Docker, ya que facilita la puesta en marcha y el mantenimiento a la par que es escalable.

Docker, con su descripción de contenedores ligeros y portátiles, me ofreció la flexibilidad de empaquetar mi aplicación con todas sus dependencias en un contenedor propiamente dicho, que se ejecuta consistentemente en cualquier entorno. Esto es crucial, ya que elimina el típico problema de "esto funciona en mi entorno local, pero no en producción".

Para alojar estos contenedores, realicé una investigación sobre diversas plataformas cloud y tras un análisis extenso, me encontré con Oracle Cloud Infrastructure. La plataforma de Oracle me pareció bastante buena ya que se puede observar que es bastante robusta y confiable para desplegar los servicios, además tiene una gran personalización para poder aprovechar todos sus componentes y recursos.

Con esto facilitamos la administración del backend y del entorno, sin tener que hacer uso de un sistema de gestión de clusters que podría llegar a complicar la coordinación y la disponibilidad de los recursos. También mencionar que un factor bastante importante para elegir Oracle fue la seguridad que ofrece, para poder proteger los datos y operaciones de nuestra operación.

Iremos comentando el proceso realizado para poder montar nuestro proyecto utilizando Docker, proporcionando una visión más o menos clara de cómo está configurado tanto la aplicación y de cómo se gestionan los contenedores.



El primer paso a realizar para el despliegue del servidor es acceder a la interfaz en línea de Oracle Cloud Infrastructure, como cualquier página estándar tenemos que crearnos una cuenta de Oracle o bien iniciar sesión con nuestras credenciales:



Ilustración 33 Inicio de Sesión en OCI

Una vez dentro de la propia plataforma, procedemos con la creación de una nueva instancia de servidor, para ello debíamos personalizarlo a nuestro gusto, por ejemplo, elegir el sistema operativo a implementar.

En mi caso, opté por Ubuntu 22.04 debido a su compatibilidad con las aplicaciones que planeamos desplegar y en cuanto al hardware, seleccioné un procesador Ampere con dos CPUs y 12 GB de memoria RAM, lo cual he considerado más que suficiente para poder manejar la carga de trabajo prevista.

Lo siguiente que realicé fue asegurarme que la instancia tuviera expuestos los puertos necesarios para poder realizar las conexiones necesarias a nuestra API (en este caso el puerto 5000) y cargué las claves SSH públicas para poder acceder posteriormente de manera segura:

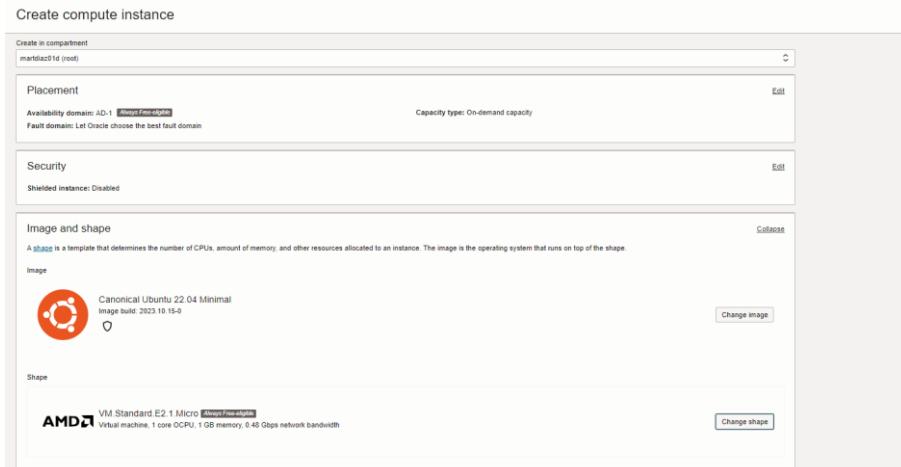


Ilustración 34 Creación de una instancia del servidor



Una vez hemos configurado todos los componentes del servidor, ya podemos iniciar el proceso de creación de nuestro servidor, donde OCI se encarga de orquestar todos los recursos necesarios. Y en cuestión de un par de minutos ya teníamos el servidor funcionando:

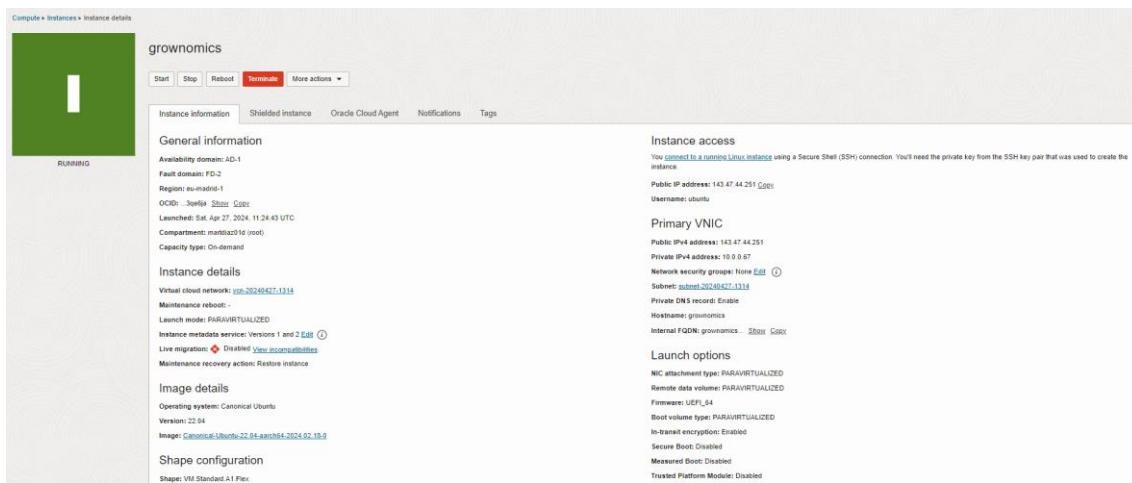


Ilustración 35 Dashboard de nuestro servidor en OCI

Finalmente, para poder acceder a nuestro servidor, utilizamos la conexión SSH que habíamos configurado anteriormente, ya que es una forma rápida y segura de poder acceder. Desde la terminal ejecutamos el comando “SSH”, especificando la clave privada y la dirección IP publicada que tiene asignado el servidor. Una vez dentro, me instale todo lo relevante a Docker para poder empezar a generar todo el entorno de contenedores que tenía ya implementada de manera local:

```

~) ssh -i ~/.ssh/clave_oracle ubuntu@143.47.44.251
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 5.15.0-1051-oracle aarch64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro

System information as of Sun Apr 28 08:42:12 UTC 2024

System load: 0.02734375   Users logged in:          0
Usage of /: 11.3% of 44.96GB   IPv4 address for br-580ddc6b560a: 172.20.0.1
Memory usage: 7%
Swap usage: 0%                IPv4 address for docker0:      172.17.0.1
Processes: 170                 IPv4 address for enp0s6:      10.0.0.67

* Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
  just raised the bar for easy, resilient and secure K8s cluster deployment.

https://ubuntu.com/engage/secure-kubernetes-at-the-edge

Expanded Security Maintenance for Applications is not enabled.

19 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

*** System restart required ***
Last login: Sat Apr 27 14:46:33 2024 from 78.30.15.36
Ubuntu@grownomics:~$ |

```

Ilustración 36 Comando SSH para acceder a mi servidor



Y adentrándonos ya a un nivel más profundo de nuestro servidor, defini un Dockerfile que utiliza Python 3.8 como imagen base, ya que me estuve informando de que ofrece un rendimiento optimizado y estabilidad para aplicaciones web, y TA-Lib, una herramienta esencial para el análisis técnico en mi proyecto, que requiere una instalación cuidadosa que puedo controlar fácilmente en este entorno de contenedor.

El proceso de configuración en el Dockerfile va desde la instalación de TA-Lib hasta la preparación del entorno de la aplicación, descargando e instalando las dependencias necesarias del archivo “requirements.txt” y asegurando que todo esté listo para ejecutarse sin problemas una vez que el contenedor esté en funcionamiento en el puerto 5000:

```
1 # Usamos una imagen base de Python 3.8
2 FROM python:3.8
3
4 RUN curl -L https://sourceforge.net/projects/ta-lib/files/ta-lib/0.4.0/ta-lib-0.4.0-
5 src.tar.gz/download?use_mirror=deac-ams -o ta-lib.tar.gz \
6   && tar -xvzf ta-lib.tar.gz \
7   && cd ta-lib \
8   && ./configure --prefix=/usr \
9   && make \
10  && make install \
11  && cd .. \
12  && rm -rf /app/ta-lib /app/ta-lib.tar.gz
13
14 # Establecemos un directorio de trabajo
15 WORKDIR /app
16
17 # Copiamos los archivos necesarios al contenedor
18 COPY requirements.txt requirements.txt
19
20 # Instalamos las dependencias
21 RUN pip install --no-cache-dir -r requirements.txt
22
23 # Copiamos el resto del código fuente de la aplicación al contenedor
24 COPY . .
25
26 # Exponemos el puerto 5000
27 EXPOSE 5000
28
29 # Comando para ejecutar la aplicación
30 CMD ["flask", "run", "--host=0.0.0.0"]
```

Tabla 92 Código del Dockerfile



Para manejar tanto la aplicación como la base de datos PostgreSQL, recurrió a docker-compose, ya que esta herramienta me permite definir los diferentes servicios de mi proyecto, tanto sus variables de entorno como sus configuraciones específicas, facilitando un despliegue sencillo.

Por lo que, mediante este archivo, a modo general, específico cómo debe construirse la aplicación, qué puertos deben exponerse y cómo los distintos servicios dependen entre sí, garantizando una interacción correcta y fluida entre mi aplicación y la base de datos:

```
1 version: '3.8'  
2  
3 services:  
4  
5   app:  
6     build: .  
7     ports:  
8       - "5000:5000"  
9  
10    environment:  
11      - FLASK_APP=app  
12      - FLASK_ENV=development  
13    volumes:  
14      - .:/app  
15    command: flask run --host=0.0.0.0  
16  
17    depends_on:  
18      - grownomics-db  
19  
20 grownomics-db:  
21  
22   image: postgres  
23  
24   environment:  
25     POSTGRES_PASSWORD: example  
26  
27   volumes:  
28     - pgdata:/var/lib/postgresql/data  
29  
30 volumes:  
31   pgdata:
```

Tabla 93 Código del Docker-Compose.yml

Este enfoque no solo simplifica el despliegue inicial, sino que también facilita actualizaciones y mantenimiento continuo de la aplicación y sus componentes subyacentes.



Las dependencias, detalladas en el archivo **requirements.txt**, abarcan desde Flask y sus extensiones para la creación de aplicaciones web hasta bibliotecas de análisis de datos como pandas y numpy, así como el manejo de bases de datos con psycopg2.

Esta lista asegura que el entorno del contenedor esté completamente equipado para ejecutar la aplicación:

```
1 Flask
2 flask-cors
3 yfinance
4 requests
5 Flask-Admin
6 pandas
7 Flask-SQLAlchemy
8 psycopg2
9 flask_login
10 pytest
11 Flask-Testing
12 flask-restplus
13 backtesting
14 TA-Lib
15 numpy
16 statsmodels
17 Flask-Mail
18 Flask-APScheduler
19 Flask-SocketIO>=3.0.0
```

Tabla 94 Dependencias del requirements.txt

Después de definir el Dockerfile y el docker-compose.yml, el proceso de despliegue se simplifica considerablemente. Solo necesitamos construir la imagen Docker de nuestra aplicación y luego ejecutarla junto con la base de datos PostgreSQL usando Docker Compose:

```
1 docker-compose build # Construye la imagen de Docker a partir del Dockerfile.
2 docker-compose up # Levanta los servicios definidos en docker-compose.yml, incluyendo
3 la app y la base de datos.
```

Tabla 95 Comandos para montar una imagen de Docker

Este enfoque no solo facilita la puesta en producción, sino que también asegura que nuestro entorno de desarrollo se refleje fielmente en producción, reduciendo así los problemas de "funciona en mi máquina".



5.4. Pruebas

Como última etapa del desarrollo software, realizar pruebas es una fase necesaria y crítica en el que hay que tener mucho cuidado para lograr tener una alta calidad y seguridad en nuestra aplicación. Disponer de pruebas automatizadas permite a los desarrolladores detectar errores lo más rápido posible, así como cualquier incoherencia en el sistema antes de que llegue el producto a los usuarios finales.

La finalidad de esto es obtener altos estándares en el desarrollo para poder crear una base sólida y sin errores donde podamos realizar una aplicación confiable y robusta.

5.4.1 Prueba Caja Blanca

Primero comenzaremos realizando el test de caja blanca, el cual se basa en el análisis detallado del código interno y la estructura del software, por ejemplo, en el back-end incluyó pruebas unitarias que validen el correcto funcionamiento de las funciones individuales, modelos de datos y endpoints de la API.

En el front-end con Flutter, utilizo librerías como **flutter_test** para asegurarme de que los widgets muestran la información correcta basada en diferentes estados de la aplicación y reaccionan como se espera a las interacciones del usuario.

Haciendo este tipo de pruebas identifico y corrojo errores en las etapas tempranas del desarrollo, mejorar la calidad del código y asegurar que el software funcione correctamente.

5.4.2 Prueba Caja Negra

Siguiendo con las pruebas tenemos la caja negra, que se centra en evaluar el comportamiento de la aplicación desde una perspectiva externa, sin considerar su estructura interna o cómo se implementa la lógica.

Este tipo de prueba es esencial para asegurar que la aplicación cumpla con sus requisitos funcionales y que proporcione una experiencia de usuario adecuada en diferentes situaciones y entradas de datos.

En flutter realizo pruebas que simulan la interacción del usuario con la aplicación, como ingresar datos en formularios, navegar a través de las distintas pantallas y activar distintas funcionalidades con librerías como **integration_test** que permiten automatizar estas pruebas, asegurando que la aplicación se comporte como se espera desde la perspectiva del usuario final, independientemente de los detalles de implementación.

En cuanto al back-end, las pruebas de caja negra son enviar solicitudes a los endpoints de la API y evaluar las respuestas para asegurar que sean correctas y estén en el formato esperado, basándose únicamente en las especificaciones de la API.



5.4.3 Pruebas Back-end

Las pruebas de back-end se centran en la lógica del servidor, las bases de datos y cualquier otro proceso que ocurre en el lado del servidor y sirven para garantizar que la lógica de negocio se ejecute como se espera y que los datos se manejen correctamente.

A continuación, se detalla cómo se han implementado las pruebas en el contexto de este proyecto:

Se realizan pruebas sobre los modelos Usuario, Accion, Cartera, Transaccion, AccionesFavoritas, y Notificacion para verificar la creación correcta de las entidades y sus relaciones. Esto asegura que la lógica de negocio que depende de estos modelos funcione como se espera.

Por ejemplo, se prueba la creación de un nuevo usuario, la asociación de acciones favoritas, y la generación de transacciones y notificaciones, validando así la integridad y el funcionamiento adecuado de la base de datos:

```
1 from flask_testing import TestCase
2 from app import create_app, db
3 from app.models import Usuario, Accion, Cartera, Transaccion, AccionesFavoritas,
4 Notificacion
5
6 class TestModelos(TestCase):
7     # Crear una configuración de aplicación para pruebas
8     def create_app(self):
9         app = create_app()
10        app.config['TESTING'] = True
11        app.config['SQLALCHEMY_DATABASE_URI'] =
12 'postgresql://postgres@example@grownomics-db/grownomics_db'
13        return app
14
15     # Configurar la base de datos para cada prueba
16     def setUp(self):
17         db.create_all()
18
19     # Test para verificar la creación de un nuevo usuario
20     def test_nuevo_usuario(self):
21         # Crear una instancia del modelo Usuario
22         usuario = Usuario(nombre="Juan", apellido="Pérez", email="juan@example.com")
23         # Guardar la instancia en la base de datos (simulado aquí con un commit
24 falso)
25         db.session.add(usuario)
26         db.session.commit()
27
28         # Comprobar que el usuario ha sido creado con los atributos correctos
29         assert usuario.nombre == "Juan"
30         assert usuario.apellido == "Pérez"
31         assert usuario.email == "juan@example.com"
32
33     # Test para verificar la creación de una nueva acción
34     def test_nueva_accion(self):
35         # Crear una instancia del modelo Accion
36         accion = Accion(nombre="Pepito Inc.", codigoticker="PEPIT")
37         # Guardar la instancia en la base de datos
```



```

38     db.session.add(accion)
39     db.session.commit()
40
41     # Comprobar que la acción ha sido creada con los atributos correctos
42     assert accion.nombre == "Pepito Inc."
43     assert accion.codigoticker == "PEPIT"
44
45     # Test para verificar la creación de una relación de acción favorita
46     def test_nueva_accion_favorita(self):
47         usuario = Usuario(nombre="Ana", apellido="García", email="ana@example.com")
48         accion = Accion(nombre="Tesla Inc.", codigoticker="TSLA")
49         db.session.add(usuario)
50         db.session.add(accion)
51         db.session.commit()
52
53         favorita = AccionesFavoritas(id_usuario=usuario.id,
54 id_accion=accion.id_accion)
55         db.session.add(favorita)
56         db.session.commit()
57
58         # Comprobar que la acción favorita ha sido creada con los atributos
59 correctos
60         assert favorita.id_usuario == usuario.id
61         assert favorita.id_accion == accion.id_accion
62
63     ... etc

```

Tabla 96 Código de tests para modelos de la base de datos

```

Proyecto-Informatica on ② main
❯ docker exec -it backend-app-1 bash
root@8e32b36c54ca:/app# pytest
===== test session starts =====
platform linux -- Python 3.8.19, pytest-8.1.1, pluggy-1.4.0
rootdir: /app
collected 7 items

tests/test_modelos.py .....
tests/test_views.py . [ 85%]
[ 100%]

===== warnings summary =====
/usr/local/lib/python3.8/site-packages/backtesting/test/_init_.py:8
/usr/local/lib/python3.8/site-packages/backtesting/test/_init_.py:8
/usr/local/lib/python3.8/site-packages/backtesting/test/_init_.py:8: FutureWarning: The argument 'infer_datetime_format' is deprecated and w
ill be removed in a future version. A strict version of it is now the default, see https://pandas.pydata.org/pdeps/0004-consistent-to-datetime-p
arsing.html. You can safely remove this argument.
    return pd.read_csv(join(dirname(__file__), filename),
-- Docs: https://docs.pytest.org/en/stable/how-to/capture-warnings.html
===== 7 passed, 2 warnings in 4.93s =====
root@8e32b36c54ca:/app#

```

Ilustración 37 Captura de Tests pasados correctamente



5.4.4 Pruebas Front-end

En la sección de pruebas de front-end de nuestro proyecto Flutter, hemos implementado dos categorías fundamentales: pruebas con mocks y pruebas de widgets. Estas pruebas son esenciales para garantizar la confiabilidad y la robustez de la interfaz de usuario, así como la lógica de negocios que interactúa con ella.

Las pruebas con mocks son esenciales cuando queremos simular la interacción con servicios externos, como las llamadas a APIs o bases de datos, permitiéndonos validar comportamientos esperados sin depender de servicios en vivo. Estas pruebas nos ayudan a asegurar que nuestro front-end maneje correctamente los datos y los estados de la aplicación, independientemente de la infraestructura subyacente.

En nuestro caso, hemos utilizado mocks para simular el proceso de inicio de sesión. Por ejemplo, creamos un mock del UsuarioController que nos permite especificar el comportamiento esperado al recibir ciertos inputs. Este es un fragmento del código que muestra cómo hemos implementado el mock:

```
1 import 'package:grownomics/controladores/userController.dart';
2 import 'package:grownomics/modelos/Notificacion.dart';
3
4 class MockUsuarioController implements UsuarioController {
5     // Simula un comportamiento de éxito o fallo para el inicio de sesión
6     @override
7     Future<bool> iniciarUsuario(String correo, String contrasena) async {
8         if (correo == "david@gmail.com" && contrasena == "clave123") {
9             return true; // Simula un inicio de sesión exitoso
10        } else {
11            return false; // Simula un fallo en el inicio de sesión
12        }
13    }
14
15    @override
16    Future<bool> registrarUsuario(String nombre, String apellido, String correo,
17        String contrasena) async {
18        // Simula un registro exitoso
19        return true;
20    }
21
22    @override
23    Future<bool> actualizarUsuario(int id, String nombre, String apellido, String
24        correo, String nuevaContrasena) async {
25        // Simula una actualización exitosa del usuario
26        return true;
27    }
28
29    @override
30    Future<bool> solicitarResetPassword(String email) async {
31        // Simula una solicitud exitosa de restablecimiento de contraseña
32        return true;
33    }
34
35    @override
36    Future<bool> resetPassword(String email, String codigo, String newPassword)
37    async {
38        // Simula un restablecimiento de contraseña exitoso
```



```

39     return true;
40   }
41
42   @override
43   Future<List<Notificacion>> obtenerNotificacionesUsuario(String correo) async {
44     // Devuelve una lista simulada de notificaciones para el usuario
45     return [
46       Notificacion(id: 0, mensaje: "Notificación de prueba", fecha: "22/10/2024"),
47     ];
48   }
49
50   @override
51   Future<Map<String, dynamic>> obtenerDatosUsuario(String correo) async {
52     // Simula la obtención de datos del usuario
53     return {
54       "nombre": "Usuario de Prueba",
55       "correo": correo,
56     };
57   }
58 }

```

Tabla 97 Código del MockUsuario (Front-end)

Este mock se usa en las pruebas para verificar que la aplicación se comporta correctamente al iniciar sesión con credenciales válidas o inválidas, sin necesidad de interactuar con el servidor real.

```

1 import 'package:flutter/material.dart';
2 import 'package:flutter_test/flutter_test.dart';
3 import 'package:grownomics/logins/loginPage.dart';
4 import 'package:grownomics/paginas/inicio.dart';
5 import 'mocks/mockUser.dart';
6
7 void main() {
8   group('Login Page Tests', () {
9     late MockUsuarioController mockUsuarioController; // Utiliza el mock manual
10
11     setUp(() {
12       mockUsuarioController = MockUsuarioController(); // Inicializa el mock
13     });
14
15     // Configura el mock para simular un inicio de sesión exitoso
16
17     testWidgets('Test de login exitoso', (WidgetTester tester) async {
18
19       // Renderiza la PaginaInicioSesion en el árbol de widgets
20       await tester.pumpWidget(MaterialApp(home: PaginaInicioSesion())); // Asegúrate
21       de pasar el mock al constructor
22
23       // Ingresa credenciales válidas
24       await tester.enterText(find.byKey(Key('correoField')), 'david@gmail.com');
25       await tester.enterText(find.byKey(Key('contrasenaField')), 'clave123');
26
27       expect(find.text('david@gmail.com'), findsOneWidget); // Verifica que el
28       correo electrónico se ingresó correctamente
29

```



```

30     expect(find.text('clave123'), findsOneWidget); // Verifica que la contraseña
31 se ingresó correctamente
32
33     // Asegúrate de esperar el Future y luego comparar el valor
34     expectLater(mockUsuarioController.iniciarUsuario("david@gmail.com",
35 "clave123"), completion(equals(true)));
36
37     // Toca el botón de iniciar sesión
38     await tester.pumpWidget(MaterialApp(home: PantallaInicio()));
39
40     // Verifica la navegación a la página de inicio
41     expect(find.byType(PantallaInicio), findsOneWidget);
42   );
43 };
44 }

```

Tabla 98 Código del test de inicio de sesión

La prueba comienza con la configuración del entorno de prueba y la inicialización de los mocks, seguido por la ejecución de un escenario de inicio de sesión dentro de un entorno de aplicación simulado. Se ingresan datos en los campos de texto correspondientes y se verifica que la operación asíncrona de inicio de sesión se complete con éxito.

Y por otro lado también tenemos las pruebas de widgets, que nos permiten validar que los widgets de Flutter se construyen como se espera en respuesta a diversas entradas y que interactúan correctamente con el resto del sistema. Es aquí donde probamos que la interfaz de usuario sea coherente y funcione adecuadamente bajo diferentes escenarios.

Por ejemplo, hemos realizado pruebas en el widget BalanceCard para asegurarnos de que los saldos y transacciones se muestran correctamente. Estas pruebas cargan el widget dentro de la infraestructura de pruebas de Flutter y simulan interacciones del usuario como toques en la pantalla:

```

1 import 'package:flutter/material.dart';
2 import 'package:flutter_test/flutter_test.dart';
3 import 'package:grownomics/paginas/Home/Widgets/balanceWidget.dart';
4
5 void main() {
6   group('BalanceCard Widget Tests', () {
7     testWidgets('BalanceCard muestra los valores por defecto correctamente',
8       (WidgetTester tester) async {
9         // Carga el widget en el árbol de widgets de pruebas con un correo no válido
10        await tester.pumpWidget(MaterialApp(
11          home: Scaffold(
12            body: BalanceCard(
13              userEmail: 'correo_no_valido@example.com',
14            ),
15          ),
16        ));
17
18        // Simula la carga de datos
19        await tester.pumpAndSettle();
20
21        // Define los valores por defecto esperados
22        final saldoEsperado = '0.00€';

```



```

23     final totalDepositadoEsperado = '0.00€';
24     final totalRetiradoEsperado = '0.00€';
25     final totalTransaccionesEsperado = '0';
26
27     // Verifica que los textos esperados están en el árbol de widgets
28     expect(
29         find.byWidgetPredicate(
30             (Widget widget) =>
31                 widget is Text &&
32                 widget.data!.contains(saldoEsperado) &&
33                 widget.style!.fontSize ==
34                     30, // Asumiendo que el saldo se muestra con tamaño de fuente 30
35             ),
36             findsOneWidget,
37         );
38         expect(find.text('Balance Actual'), findsOneWidget);
39     );
40 );
41 }

```

Tabla 99 Código del test Widget BalanceCard

Las pruebas se realizan en un entorno de aplicación de prueba, donde el widget BalanceCard se carga y muestra. Se simula la espera por la carga de datos y se realizan comprobaciones para asegurar que los textos, como "Balance Actual" y los valores monetarios, aparezcan como se espera y con el formato adecuado.



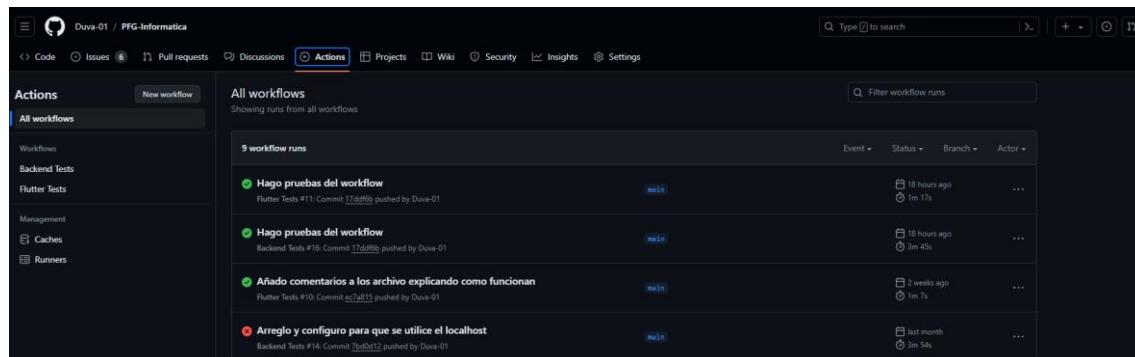
5.4.5 Github Actions (Workflows)

La automatización de las pruebas es una buena práctica para realizar el desarrollo de un proyecto, ya que así nos aseguramos la calidad y la estabilidad del software antes de su despliegue.

Para lograr esto utilizamos GitHub Actions, esta se trata de una herramienta de integración y despliegue continuo (CI/CD), en nuestro caso, lo utilizaremos para realizar pruebas automatizadas cada vez que se realizan cambios en el código con los “commits” para comprobar que todo está funcionando correctamente.

Con esto logramos detectar los errores de una manera prematura y mejorar sobre todo la calidad del código. En nuestro caso, configuramos los flujos de trabajo que se activan en respuesta a “pushes” en el repositorio, con el fin de asegurarnos que las nuevas implementaciones cumplen con los estándares de calidad antes de que sean integradas en la rama principal.

Y también nos ahorra la necesidad de hacer pruebas manuales repetitivas y mejora la eficiencia en el proceso de desarrollo. Este sistema integrado de pruebas en Github Actions es una parte fundamental para nuestro enfoque de desarrollo ágil con Scrum, ya que es vital tener un feedback continuo y la mejora de la calidad del producto a entregar de manera iterativa con la automatización y la estandarización de los tests.



The screenshot shows the GitHub Actions interface for the repository "Dava-01 / PFG-Informatica". The "Actions" tab is selected. On the left, there's a sidebar with "All workflows" expanded, showing categories like Workflows, Backend Tests, Flutter Tests, Management, Caches, and Runners. The main area displays "All workflows" with 9 workflow runs listed:

Workflow	Event	Status	Branch	Actor	
Hago pruebas del workflow	Flutter Tests #11: Commit 17dd9fb pushed by Dava-01	main	18 hours ago	1m 17s	...
Hago pruebas del workflow	Backend Tests #16: Commit f7fd6ff pushed by Dava-01	main	18 hours ago	3m 46s	...
Añado comentarios a los archivo explicando como funcionan	Flutter Tests #10: Commit ec7a811 pushed by Dava-01	main	2 weeks ago	1m 7s	...
Arreglo y configuro para que se utilice el localhost	Backend Tests #14: Commit 2ed0d11 pushed by Dava-01	main	last month	3m 56s	...

Ilustración 38 Workflows de Github Actions



5.4.5.1 Automatización pruebas en el Front-end

En el caso del front-end, desarrollado en Flutter, hemos establecido una serie de pruebas automatizadas que se ejecutan a través de un flujo de trabajo definido en un archivo, con el que se configuran las “acciones” necesarias para instalar las dependencias, el entorno de Flutter y ejecutar todas las pruebas que hayamos asignado automáticamente.

```
1 name: Flutter Tests
2
3 on:
4   push:
5     branches:
6       - main
7     paths:
8       - 'grownomics/**'
9   pull_request:
10    branches:
11      - main
12    paths:
13      - 'grownomics/**'
14
15 jobs:
16   build-and-test:
17     runs-on: ubuntu-latest
18
19     steps:
20       - uses: actions/checkout@v4.1.2
21
22       - uses: subosito/flutter-action@v2
23         with:
24           flutter-version: '3.x'
25
26       - name: Install Dependencies
27         run: |
28           cd grownomics
29           flutter pub get
30
31       - name: Run Flutter Tests
32         run: |
33           cd grownomics
34           flutter test
35
```

Tabla 100 Código para automatización pruebas en el Front-end

Como podemos ver en este caso se activa tanto por eventos ‘push’ como ‘pull request’ en la rama main y en la carpeta grownomics donde está alojado el código del front-end. En cuanto a la configuración específica, podemos ver que la maquina virtual a utilizar es Ubuntu en su versión más reciente.

Se utilizan acciones predefinidas como ‘checkout’ para clonar el repositorio de código en la máquina virtual, también configuramos el entorno de flutter con la versión 3 o mayor, y por último se ejecutan el conjunto de pruebas definidas.



5.4.5.1 Automatización pruebas en el Back-end

De manera similar ocurre en el back-end, hemos configurado el archivo para que este flujo de trabajo se active con cada ‘push’ o ‘pull request’ que afecte al código del back-end, pero esta especialmente diseñado para crear un entorno en Docker y con una instancia de PostGreSQL replicando así el entorno de producción de manera fiel.

```
1 name: Backend Tests
2
3 on:
4   push:
5     branches:
6       - main
7     paths:
8       - 'backend/**'
9   pull_request:
10    branches:
11      - main
12    paths:
13      - 'backend/**'
14
15 jobs:
16   build-and-test:
17     runs-on: ubuntu-latest
18
19   services:
20     postgres:
21       image: postgres:latest
22       env:
23         POSTGRES_USER: postgres
24         POSTGRES_PASSWORD: example
25         POSTGRES_DB: grownomics_db
26       ports:
27         - 5432:5432
28       options: >-
29         -v /home/runner/work/Proyecto-Informatica/Proyecto-
30 Informatica/backend/backup.sql:/docker-entrypoint-initdb.d/backup.sql
31
32   steps:
33     - uses: actions/checkout@v2
34
35     - name: Install Docker Compose
36       run: |
37         sudo curl -L
38         "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
39         sudo chmod +x /usr/local/bin/docker-compose
40
41     - name: Set up Docker Compose
42       run: docker-compose -f backend/docker-compose.yml up -d
43
44     - name: Mostrar logs de la base de datos
45       run: docker-compose -f backend/docker-compose.yml logs grownomics-db
46
47     - name: Wait for DB to be ready
48       run: |
```



```
50      until docker exec $(docker-compose -f backend/docker-compose.yml ps -q
51 grownomics-db) pg_isready ; do sleep 5 ; done
52
53 - name: Run Tests
54   run: docker-compose -f backend/docker-compose.yml exec -T app python -m pytest
55 tests/
```

Tabla 101 Código para automatización pruebas en el Back-end

En este archivo, además de clonarnos el repositorio y preparar el entorno de Docker, tenemos que configurar servicios adicionales, como es el contenedor de PostGreSQL. Esto es necesario para poder realizar las pruebas con la base de datos, ya que es lo principal de lo que se compone el back-end.

El flujo de trabajo empleo “Docker compose” para configurar los servicios de manera que las pruebas se puedan ejecutar de manera similar a una operación real del servidor.

Las demás configuraciones están diseñadas para garantizar que el back-end funcione de manera óptima y que las APIs responda correctamente bajo diversas condiciones, verificando tanto la robustez como la seguridad de nuestro software.



6. Conclusiones

6.1. Temporización real

Llegando a la etapa de reflexión y de retrospectiva en el tiempo sobre el proyecto, lo principal que podemos analizar es realmente el tiempo invertido en el desarrollo de la aplicación en comparación con lo que había planeado inicialmente. Así se consigue comprender como fluyen las tareas y como de precisas han sido las estimaciones de tiempo en la etapa de planificación.

Por tanto, si comparamos los dos diagramas de Gantt, tanto el inicial como el real, he podido identificar varias diferencias significativas entre la planificación prevista y la realidad:

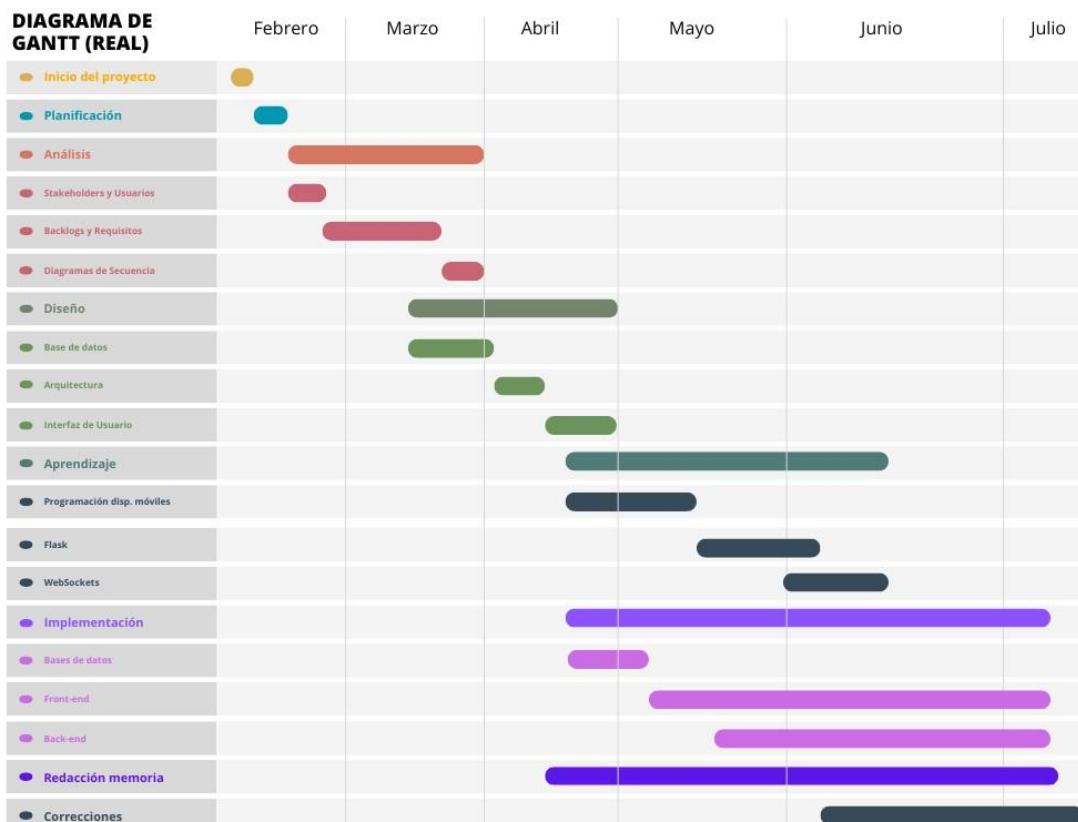


Ilustración 39 Diagrama de Gantt Real

En la fase inicial, distribuí las tareas de manera equilibrada y diseñé el proyecto para que siguiera un flujo secuencial y ordenado. Tenía planeado, por ejemplo, completar el "Análisis" y el "Diseño" antes de siquiera comenzar con la implementación de la "Base de datos" y la "Arquitectura".

No obstante, al mirar el diagrama real, es evidente que varias tareas necesitaron más tiempo del previsto para su finalización. El alargamiento de fases como "Análisis", "Stakeholders y Usuarios" y "Backlogs y Requisitos" demuestran que me encontré con desafíos al definir el alcance del proyecto y al recolectar los requisitos necesarios.

Estos imprevistos provocaron retrasos en tareas posteriores, como el "Diseño" y la "Arquitectura", que también se vieron afectadas.



Me llamó la atención que algunas tareas, que en mi planificación inicial tenían un inicio y un final claros, como la "Programación disp. móviles" y "Flask", terminaron superponiéndose y extendiéndose más de lo que había estimado.

Esto refleja que tuve que hacer mejoras en ciertos aspectos del proyecto mientras avanzaba, debido al feedback de los usuarios y a los obstáculos técnicos que no había previsto.

Además, en el diagrama real, es evidente que inicié con la implementación incluso antes de terminar completamente con el "Diseño" y la "Base de datos".

Ya que al seguir la metodología scrum, no suponía ningún problema tener una adaptación hacia un enfoque más ágil, donde comencé a implementar sin tener todas las especificaciones finalizadas, algo que, retrospectivamente, veo como algo ventajoso por ejemplo a la hora de detectar problemas prácticos desde una etapa temprana.



6.2. Objetivos alcanzados

Haciendo un pensamiento critico sobre el trabajo emprendido en este Proyecto de Fin de Grado, he llegado a la conclusión de que he conseguido alcanzar satisfactoriamente los objetivos propuestos al inicio de este proyecto.

Tanto en lo técnico, con el desarrollo y la implementación de las funcionalidades, como en el personal, aun suponiendo un trabajo elaborado. A continuación, desglosó cómo cada uno de estos objetivos se ha materializado en el proyecto final:

En primer lugar, la creación de una interfaz de usuario intuitiva ha sido completada de manera correcta, he construido una interfaz que es fácil de manejar para usuarios de diversos niveles de habilidad informática, logrando así un acceso sin barreras a las funcionalidades de la aplicación.

En cuanto a la implementación de datos financieros en tiempo real en la aplicación ha sido realizada correctamente, ya que los usuarios ahora pueden acceder a información del mercado actualizada al instante, para que puedan tomar decisiones sobre una base fundamentada.

Sin embargo, he tenido complicaciones con el objetivo 3, ya que, aunque el plan inicial era desarrollar algoritmos que ofrecieran recomendaciones de inversión personalizadas adaptadas a cada perfil, durante el desarrollo del proyecto opté por una solución alternativa. En lugar de personalizar el análisis bursátil para cada usuario, he implementado un sistema de recomendaciones generales que provee una perspectiva amplia de diferentes estrategias de inversión.

Por otro lado, he incluido con éxito una herramienta de gestión de cartera de valores, la cual permite a los usuarios supervisar y administrar sus inversiones de manera eficiente.

En cuanto al objetivo de incorporar educación y asesoramiento financiero en la aplicación, debo admitir que este aspecto no se ha desarrollado con la profundidad inicialmente prevista. Aunque se han integrado secciones educativas básicas y se ofrece algún asesoramiento en inversiones, estas características son más elementales de lo que se había concebido en la etapa de planificación.

Con vistas al futuro, he estructurado la aplicación para asegurar su escalabilidad y su fácil adaptación a cambios y expansiones futuras. Este enfoque previsor garantiza que el software seguirá siendo relevante y útil a medida que evolucionen los mercados financieros.

En relación con el objetivo de asegurar la privacidad y seguridad de los datos de los usuarios, he establecido las medidas básicas de seguridad, cubriendo los requisitos esenciales en este ámbito, aunque soy consciente de que aún hay espacio para fortalecer y expandir las estrategias de seguridad aplicadas.



6.3. Lecciones aprendidas

Y tras echar un vistazo sobre el camino recorrido en este proyecto, me siento profundamente agradecido por todas las lecciones que he aprendido, tanto en destrezas técnicas como en metodologías de gestión de proyectos.

Comencé el desarrollo de una aplicación móvil con una base de conocimientos básica en ese campo, teniendo una curva de aprendizaje bastante inclinada, pero a cada paso, fui capaz de coger más soltura y tener una comprensión sólida que finalizó con el desarrollo de una aplicación móvil completamente funcional. Esto se consiguió principalmente por la paciencia, dedicación y el aprendizaje continuo que he cultivado durante todo el proceso.

El uso de la metodología Scrum ha sido todo un acierto, ya que, en comparación con la metodología tradicional, que opta por un trabajo más fijo y preestablecido, he podido gestionar el proyecto con una gran flexibilidad adaptándolo a mis circunstancias y de forma iterativa, realizando cambios y ajustándome a las prioridades principales en tiempo real.

Además, me he dado cuenta de la importancia de la ingeniería software y de la etapa de diseño, ya que generalmente cuando hago unas prácticas para un entorno específico no hay que pensar mucho en estos conceptos, pero cuando quieras estructurar un proyecto tan grande se vuelven unas herramientas esenciales, aunque puedan parecer tediosas de realizar.

Por tanto, el diseño cuidadoso y la planificación son una parte imprescindible de cualquier proyecto que se quiera realizar, ya que son las fases que transforman nuestras complejas ideas en una estructura clara, visible y manejable.

Sin estas fases previas al desarrollo, probablemente me tendría que haber enfrentado a obstáculos imprevistos y que hubieran supuesto grandes complicaciones tanto en tiempo como en soluciones viables.

Estas experiencias, me han hecho darme cuenta de que hay que tener una perspectiva más amplia sobre el desarrollo de proyectos, y la satisfacción de poder completar y alcanzar ciertos objetivos propuestos se ve magnificado por la gran cantidad de conocimientos adquiridos.



6.4. Trabajos futuros

Soy consciente de que el trabajo realizado todavía tiene mucho que pulirse en muchos aspectos, por lo que esto es solo un inicio hacia la expansión y la mejora continua de la aplicación.

Además, ya tengo pendientes una serie de funcionalidades que me gustaría implementar en algún futuro no muy lejano, para poder mejorar tanto la experiencia del usuario como la utilidad de este:

- La implementación del protocolo HTTPS con certificados para mejorar la seguridad y robustez de la plataforma, con el fin de proteger los datos de los usuarios y que haya una mayor confianza en nuestra aplicación frente a los posibles ataques ciberneticos y a las vulnerabilidades.
- Un foro para inversores, lo que permitirá que los usuarios compartan sus perspectivas sobre las últimas noticias y fluctuaciones del mercado, fomentando así una comunidad de aprendizaje y soporte mutuo.
- Contacto con el staff técnico, que ayudará a resolver dudas o problemas que surjan durante el uso de la aplicación.
- Crear perfiles más detallados que reflejen las preferencias y comportamientos de inversión de cada usuario permitirá que nuestro sistema de recomendación sea aún más preciso y útil.
- La implementación de un modo oscuro no solo es una característica ampliamente solicitada por los usuarios por su estética, sino también por su comodidad visual.
- Mejorar el sistema de notificaciones actual, para ofrecer alertas instantáneas incluso cuando la aplicación no está activa o se encuentra en segundo plano.



7. Bibliografía

7.1 Referencias

- [1] G. Soros, «El problema no es lo que uno no sabe, sino lo que uno cree que sabe estando equivocado».
- [2] G. T. Doran, «There's a S.M.A.R.T. Way to Write Management's Goals and,» 1981. [En línea]. Available: <https://community.mis.temple.edu/mis0855002fall2015/files/2015/10/S.M.A.R.T-Way-Management-Review.pdf>.
- [3] U. d. Alicante, «CÓMO REDACTAR Y ESTRUCTURAR EL TFG DE INGENIERÍA INFORMÁTICA.,» 2019. [En línea]. Available: https://rua.ua.es/dspace/bitstream/10045/90049/36/Redaccion-y-estructura-TFG-INGENIERIA-INFORMATICA_2018_2019.pdf.
- [4] A. Navarro Cadavid y J. D. Fernández Martínez, «Revisión de metodologías ágiles para el desarrollo de software,» 2013. [En línea]. Available: <https://www.redalyc.org/articulo.oa?id=496250736004>.
- [5] H. V. Cevallos, «Metodologías ágiles frente a las tradicionales en el proceso de desarrollo de software,» 2018. [En línea]. Available: https://www.researchgate.net/publication/327537074_Metodologias_agiles_frente_a_las_tradicionales_en_el_proceso_de_desarrollo_de_software.
- [6] P. t. d. proyecto, «Planificación temporal del proyecto,» 2020. [En línea]. Available: <https://ingsoftware2020.webcindario.com/segunda-unidad/planificacion-de-proyectos-de-software/planificacion-temporal-del-proyecto.html>.
- [7] «GlassDoor,» 22 01 2024. [En línea]. Available: https://www.glassdoor.es/Sueldos/junior-software-developer-sueldo-SRCH_K00,25.htm.
- [8] A. E. B. O. d. Estado, «Ley 35/2006, de 28 de noviembre, del Impuesto sobre la Renta de las Personas Físicas y de modificación parcial de las leyes de los Impuestos sobre Sociedades, sobre la Renta de no Residentes y sobre el Patrimonio.,» 2006. [En línea]. Available: <https://www.boe.es/buscar/act.php?id=BOE-A-2006-20764>.
- [9] C. Laborales, «Porcentaje de cotización a la Seguridad Social del trabajador y empresa,» [En línea]. Available: <https://www.cuestioneslaborales.es/porcentaje-de-cotizacion-a-la-seguridad-social-del-trabajador-y-empresa/>.
- [10] A. B. Carroll, «Understanding Stakeholder Thinking: Themes from a Finnish Conference,» 1997. [En línea]. Available: https://www.researchgate.net/publication/229484860_Understanding_Stakeholder_Thinking_Themes_from_a_Finnish_Conference.
- [11] A. Modeling, «Usage Scenarios: An Agile Introduction,» [En línea]. Available: <https://agilemodeling.com/artifacts/usagescenario.htm>.
- [12] S. Laoyan, «Planning poker: la estrategia integral para la estimación ágil,» 2022. [En línea]. Available: <https://asana.com/es/resources/planning-poker>.
- [13] G. L. Alexander Menzinsky, «Historias de Usuario - Ingeniería de Requisitos Ágil,» 2022. [En línea]. Available: https://www.scrummanager.com/files/scrum_manager_historias_usuario.pdf.
- [14] J. Canales, «Sprints: ¿Cómo Definir las Entregas de un Proyecto con Scrum?,» 2019. [En línea]. Available: <https://medium.com/@jimecs/sprints-c%C3%B3mo-definir-las-entregas-de-un-proyecto-con-scrum-77b018d54319>.



- [15] Scrum.org, «Learn About the Scrum Artifact: Product Backlog,» [En línea]. Available: <https://www.scrum.org/resources/what-is-a-product-backlog>.
- [16] MountainGoatSoftware, «Sprint Review Meeting,» [En línea]. Available: <https://www.mountaingoatsoftware.com/agile/scrum/meetings/sprint-review-meeting>.
- [17] mountaingoatsoftware, «Sprint Planning Meeting,» [En línea]. Available: <https://www.mountaingoatsoftware.com/agile/scrum/meetings/sprint-planning-meeting>.
- [18] mountaingoatsoftware, «Sprint Retrospective,» [En línea]. Available: <https://www.mountaingoatsoftware.com/agile/scrum/meetings/sprint-retrospective>.
- [19] mountaingoatsoftware, «Sprint Backlog,» [En línea]. Available: <https://www.mountaingoatsoftware.com/agile/scrum/scrum-tools/sprint-backlog>.
- [20] E. Malinowski, «EL USO DEL MODELO ENTIDAD-RELACIÓN EN EL DISEÑO DE LA REPRESENTACIÓN DE DATOS EN LAS BASES DE DATOS ORIENTADAS A OBJETOS,» 1998. [En línea]. Available: https://www.researchgate.net/publication/280821509_EL_USO_DEL_MODELO_ENTIDAD-RELACION_EN_EL_DISENO_DE_LA_REPRESENTACION_DE_DATOS_EN_LAS_BASES_DE_DATOS_ORIENTADAS_A_OBJETOS.
- [21] Drawio, «Security-first diagramming for teams.,» [En línea]. Available: <https://app.diagrams.net/>.
- [22] <https://flanagan.ugr.es/>, «Seminario 4 - Diseño y Gestión de Bases de Datos,» [En línea]. Available: <https://flanagan.ugr.es/docencia/2010-2011/fiic/apuntes/FI-IC-Seminario4.BasesDeDatos.pdf>.
- [23] H. C. Torrejón, «Cómo realizar la normalización de bases de datos y por qué,» 2022. [En línea]. Available: <https://openwebinars.net/blog/como-realizar-la-normalizacion-de-bases-de-datos-y-por-que/>.
- [24] S. T. P. Glenn E. Krasner, «A Cookbook for Using the Model-View-Controller User Paradigm in Smalltalk-80,» 1988. [En línea]. Available: <https://ics.uci.edu/~redmiles/ics227-SQ04/papers/KrasnerPope88.pdf>.
- [25] C. Team, «MVC: Model, View, Controller,» [En línea]. Available: <https://www.codecademy.com/article/mvc>.
- [26] C. A. Melanie Perkins, «Una Suite Visual para todo el mundo - Canva,» [En línea]. Available: <https://www.canva.com/>.
- [27] L. Baker, «What is the importance of UI and UX for a website?,» 2023. [En línea]. Available: <https://makeitclear.com/what-is-the-importance-of-ui-and-ux-for-a-website/>.
- [28] Atlassian, «Jira | Software de seguimiento de proyectos e incidencias,» [En línea]. Available: <https://www.atlassian.com/es/software/jira>.
- [29] GanttPro, «Diagrama de Gantt online,» [En línea]. Available: <https://ganttpro.com/es/>.
- [30] Flutter, «Flutter - Build apps for any screen,» [En línea]. Available: <https://flutter.dev/>.
- [31] Flask. [En línea]. Available: <https://flask.palletsprojects.com/en/3.0.x/>.
- [32] PostgreSQL. [En línea]. Available: <https://www.postgresql.org/>.
- [33] Docker, «Docker: Accelerated Container Application Development,» [En línea]. Available: <https://www.docker.com/>.
- [34] Dart, «Dart programming language | Dart,» [En línea]. Available: <https://dart.dev/>.
- [35] dart.dev, «A composable, Future-based library for making HTTP requests.,» [En línea]. Available: <https://pub.dev/packages/http>.
- [36] flutter4fun.com, «A highly customizable Flutter chart library that supports Line Chart, Bar Chart, Pie Chart, Scatter Chart, and Radar Chart.,» [En línea]. Available: https://pub.dev/packages/fl_chart.
- [37] Flutter.dev, «A Flutter K Chart.,» [En línea]. Available:



- https://pub.dev/packages/k_chart.
- [38] simc.dev, «Flutter widget that automatically resizes text to fit perfectly within its bounds.,» [En línea]. Available: https://pub.dev/packages/auto_size_text/versions.
- [39] flutter.dev, «Wraps platform-specific persistent storage for simple data (NSUserDefaults on iOS and macOS, SharedPreferences on Android, etc.),» [En línea]. Available: https://pub.dev/packages/shared_preferences.
- [40] flutter.dev, «On iOS the WebView widget is backed by a WKWebView. On Android the WebView widget is backed by a WebView.,» [En línea]. Available: https://pub.dev/packages/webview_flutter.
- [41] Flutter, «socket.io-client-dart - Port of awesome JavaScript Node.js library,» [En línea]. Available: https://pub.dev/packages/socket_io_client.
- [42] dexterx.dev, «flutter_local_notifications - A cross platform plugin for displaying local notifications.,» [En línea]. Available: https://pub.dev/packages/flutter_local_notifications.
- [43] dart.dev, «Provides internationalization and localization facilities,» [En línea]. Available: <https://pub.dev/packages/intl>.
- [44] tagnote.app, «A Markdown viewer widget for Flutter. It renders Markdown string to rich text output.,» [En línea]. Available: https://pub.dev/packages/markdown_viewer.
- [45] Canva, «Una Suite Visual para todo el mundo - Canva,» [En línea]. Available: <https://www.canva.com/>.
- [46] Figma, «Figma: The Collaborative Interface Design Tool,» [En línea]. Available: <https://www.figma.com/>.
- [47] ranaroussi, «Download market data from Yahoo! Finance's API,» [En línea]. Available: <https://pypi.org/project/yfinance/>.
- [48] TA-Lib, «Use TA-Lib to add technical analysis to your own financial market trading applications,» [En línea]. Available: <https://ta-lib.org/>.
- [49] NumPy, «The fundamental package for scientific computing with Python,» [En línea]. Available: <https://numpy.org/>.
- [50] kernc, «Backtest trading strategies with Python.,» [En línea]. Available: <https://pypi.org/project/Backtesting/>.
- [51] pandas, «pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool,» [En línea]. Available: <https://pandas.pydata.org/>.
- [52] Flask-SocketIO, «Flask-SocketIO gives Flask applications access to low latency bi-directional communications between the clients and the server.,» [En línea]. Available: <https://flask-socketio.readthedocs.io/en/latest/>.



8. Anexos

8.1. Manual de Usuario

8.1.1 Objetivo

Este manual tiene por objetivo dar soporte a los usuarios de la aplicación “grownomics”, con el fin de ofrecer un control efectivo e información oportuna sobre los requerimientos y funcionalidades que la aplicación necesita.

8.1.2 Requerimientos

- **Sistema Operativo:** Compatible con dispositivos iOS y Android.
- **Espacio de Almacenamiento:** Suficiente espacio para instalar la aplicación y almacenar datos de usuario y cache.
- **Conexión a Internet:** Necesaria para actualizar datos en tiempo real y acceso a contenido online.



8.1.3 Inicio de Sesión y Registro

Lo primero que podemos encontrar al iniciar la aplicación, es un inicio de sesión, donde se solicita tanto el correo electrónico como la contraseña al usuario para garantizar un acceso seguro al inicio.

También podemos hallar diferentes opciones como un checkbox para “Recordar” el inicio de sesión y facilitar futuros accesos, y por otro lado tenemos enlaces para recuperar la contraseña en caso de olvidarla o si no tenemos cuenta a la página de registro para poder introducir nuestras credenciales y darnos de alta en el sistema.

Si aun así solo queremos probar cómo funciona la aplicación antes de registrarse en cualquier sitio, podemos saltarnos esta parte y acceder al inicio, pero con funcionalidades limitadas.

Siendo esta primera toma de contacto muy importante y donde demostramos la simplicidad y eficiencia para que puedas explorar todas las funcionalidades que ofrecemos sin demora.



Ilustración 40 Capturas de pantalla del Inicio de Sesión y de Registro



8.1.4 Recuperación de Contraseña

En la sección de cambio de contraseña, podemos ver que es un entorno muy simple y que está completamente guiado de los pasos a seguir, donde en un primer lugar, encontraremos un recuadro donde introducir nuestro correo.

Tras introducir el correo, encontramos un botón verde que invita a solicitar un código de verificación, si lo pulsamos y resulta que nuestro correo es un usuario valido del sistema, se enviará un correo al usuario con un código de 6 dígitos para poder verificar que el usuario es el poseedor de dicho correo.

A continuación, en la pantalla se mostrarán tres campos de textos donde se deberán introducir tanto el código de verificación solicitado como la nueva contraseña que se querrá introducir. Si todo el proceso ha sido realizado con éxito, debería de haberse cambiado la contraseña del usuario y se podrá acceder con normalidad dentro de la aplicación sin problemas.

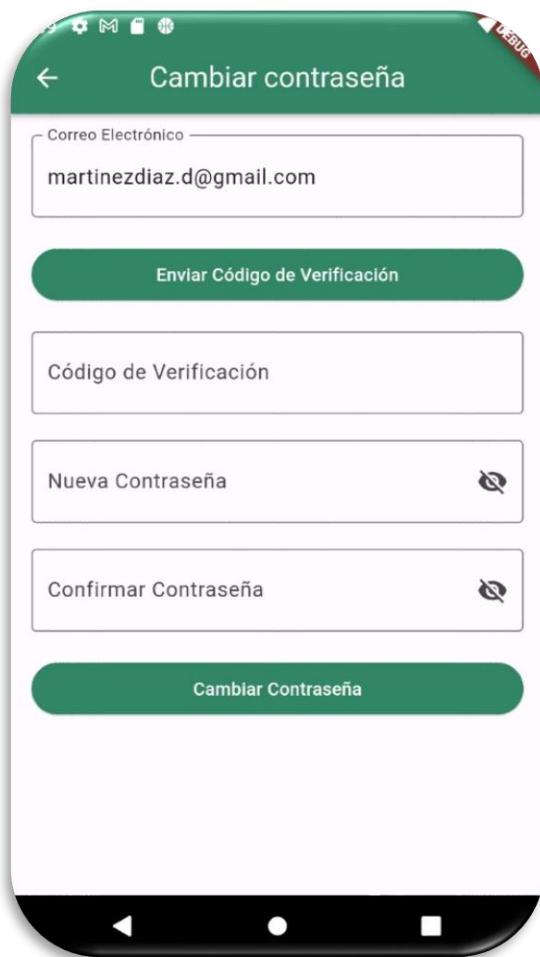


Ilustración 41 Captura de pantalla de Recuperar Contraseña



8.1.5 Página Principal (Home)

Tras haber completado el registro o el inicio de sesión (o también el saltarse esta parte), serás recibido en la página de inicio de la aplicación, si vamos por partes en primer lugar, en la parte superior descubrirás una barra de navegación que facilita el acceso al menú principal, que será la forma de navegar entre las diferentes funcionalidades de la app.

También encontraremos en la parte superior izquierda el ícono de una “campana” para poder acceder a las notificaciones de nuestras acciones favoritas.

Pero centrándonos en el inicio, lo primero que nos encontramos a simple vista, es un resumen de los principales índices y los sectores en movimiento para saber cómo se encuentra el mercado actualmente y también una sección de las últimas noticias financieras para saber qué está pasando en la actualidad.

Si bajamos por la pantalla, podremos tener visión de otras funcionalidades siempre y cuando hayamos iniciado sesión, para poder ver un resumen personalizado de tu perfil, incluyendo tu nombre y el saldo de tus simulaciones financieras, brindándote una vista rápida de tu actividad reciente y tus últimas transacciones. Esto se complementa con secciones como tus acciones favoritas para un acceso rápido a tus inversiones máspreciadas.

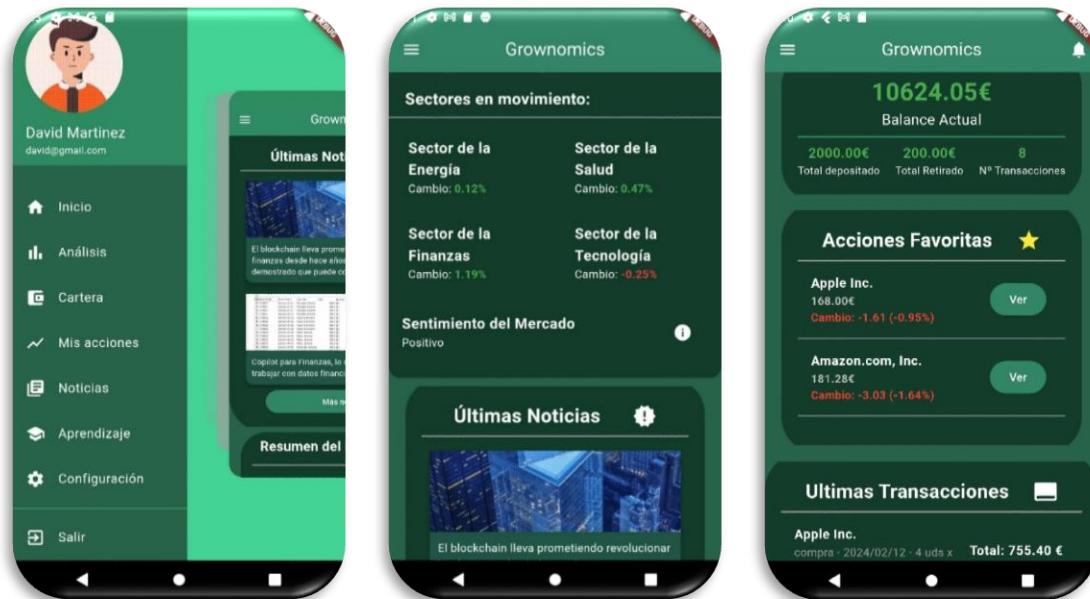


Ilustración 42 Captura de pantalla de la Página de Inicio



8.1.6 Página de Acciones

Si navegamos por el menú a la primera sección disponible, entraremos en la funcionalidad de analizar las acciones del mercado. Donde lo primero que encontraremos es una barra de búsqueda con el podremos filtrar por el nombre o el ticker de la acción y acceder más rápido a las acciones que más nos interesen.

Bajo esta herramienta de búsqueda, encontrarás los botones "Todos" y "Favoritas", diseñados para facilitar la navegación entre la extensa lista de acciones disponibles y aquellas que has marcado como de especial interés, respectivamente.

Y, por último, tendríamos una lista de las acciones del mercado, que se muestran en tarjetas donde se puede ver información general sobre la acción. También podrás ver la figura de una estrella, la cual sirve para identificar las acciones que tienes seleccionadas como favoritas, para agregar o desagregar alguna acción, lo único que hay que hacer es pulsarlo y ver cómo cambia el ícono.

Al final de cada acción habrá un botón de "Ver", que te invita a explorar detalles más profundos sobre la misma y poder acceder a su respectivo análisis.

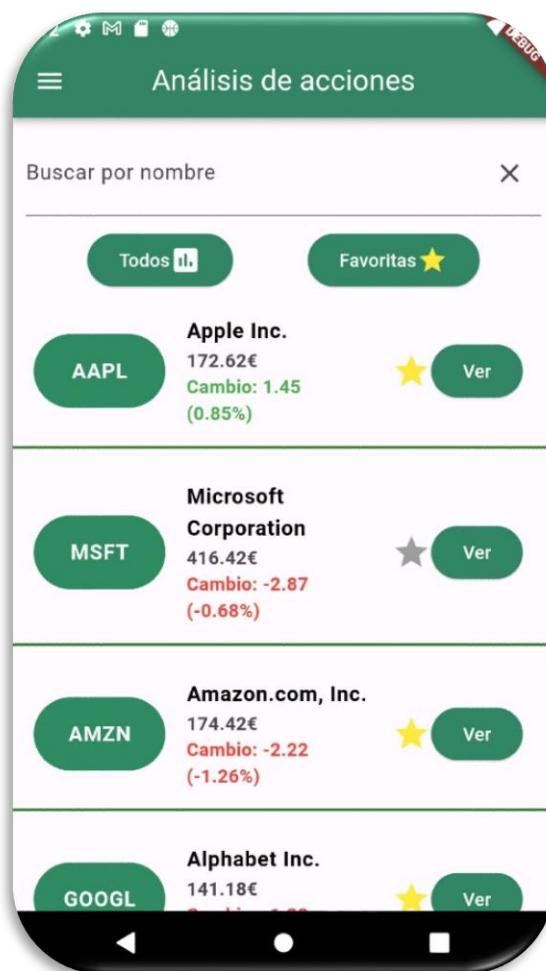


Ilustración 43 Captura de pantalla de la Página de Acciones



8.1.7 Página de Acción Específica

Una vez accedemos dentro de una acción, encontraremos muchas opciones, en la parte superior, tendremos un menú de secciones para poder ver las distintas herramientas de análisis.

En primer lugar, encontramos el resumen donde se puede ver la información general de la acción, como sus datos históricos en un gráfico de velas para elegir los intervalos de tiempo y una recomendación final basado en un sistema de recomendación diseñado a través de una heurística que tiene en cuenta varios aspectos, como los precios, soportes y resistencias, predicciones de precios e indicadores económicos.

También encontraremos otras secciones como análisis técnicos donde se podrán observar en gráficos soportes y resistencias, cruces de SMAs, indicadores económicos como el RSI, SMA, EMA... etc. Análisis fundamental para ver por ejemplo el ROI o el rendimiento del dividendo entre otras cosas.

También estará disponible la sección de estrategias y su respectiva comprobación con el back tracking para saber el rendimiento que ofrece cada estrategia utilizando los datos pasados y un listado de predicciones de precios utilizando un modelo predictivo.

Por último, encontraremos un desplegable para poder "Comprar" o "Vender", los cuales abren la puerta a simulaciones virtuales de trading, con la que se pueden experimentar con compras y ventas ficticias, ofreciendo una visión de cómo podrían resultar tales movimientos o estrategias en el mercado real.

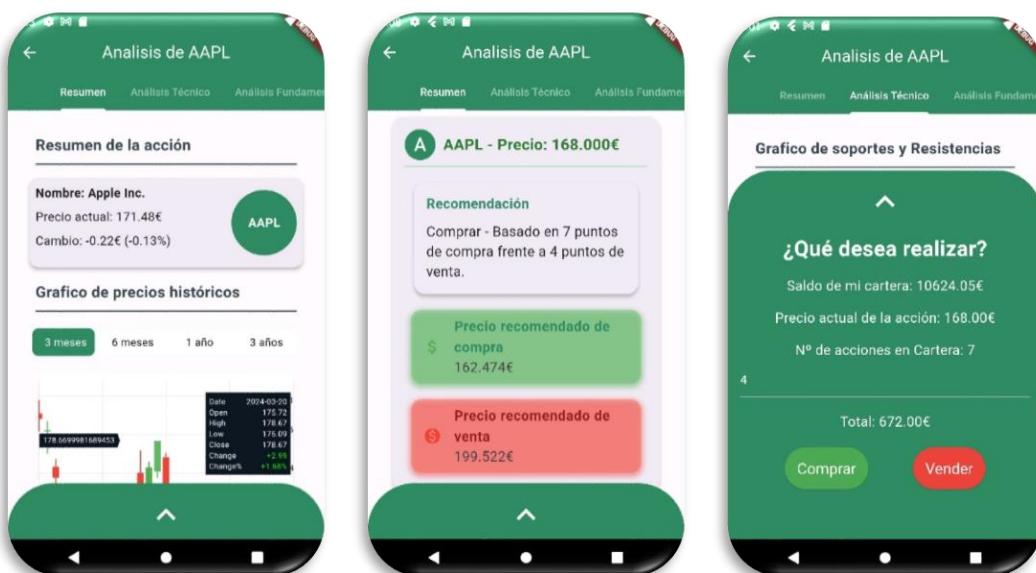


Ilustración 44 Capturas de pantalla de la Página de Acción Específica



8.1.8 Página de Cartera

En cuanto a la página de la Cartera Virtual, lo que se puede observar es una vista amplia y detallada del rendimiento financiero en el entorno virtual y poder ver un resumen completo del saldo actual.

Por ello, podemos encontrar una réplica de una tarjeta donde ver la información de la cartera, como el beneficio, balance, el total depositado... etc. Y también encontramos dos botones para gestionar fondos virtuales, uno para depositar y otro para retirar dinero virtual para poder participar en la simulación pase lo que pase.

Y justo debajo, disponemos de un historial detallado de tu rendimiento financiero, presentando una línea de tiempo de cómo tu balance ha evolucionado a través de diversas estrategias de inversión y decisiones tomadas.

Y profundizando aún más, encontramos un listado de las últimas transacciones realizadas, junto con un botón para poder acceder a un historial completo de todas las transacciones, estando organizadas por días para poder diferenciar bien las diferentes operaciones.



Ilustración 45 Captura de pantalla de la Página de Cartera



8.1.9 Página de Mis Acciones

La página "Mis Acciones" es un espacio dedicado a ofrecer una vista panorámica de como está compuesta la cartera virtual, diseñado con el objetivo de clarificar y de diversificar los activos que la componen. Para ello, implementamos un gráfico circular que desglosa tu cartera por porcentaje de inversión por activo.

También encontraremos un desplegable con información básica y general sobre porque es importante diversificar la cartera para darle una cierta base teórica al usuario de la finalidad y su utilidad en el aspecto financiero.

La segunda parte importante de la página es mostrar el valor actual de tu cartera según las acciones que tengas en posesión mediante una tarjeta con los colores de la aplicación para saber cuál es el valor de tus acciones en el momento.

Además, se muestra en conjunto con una lista de las acciones que componen tu cartera cada una de ellas con un despegable con la funcionalidad de mostrar todas las transacciones respecto a dicha acción seleccionada.

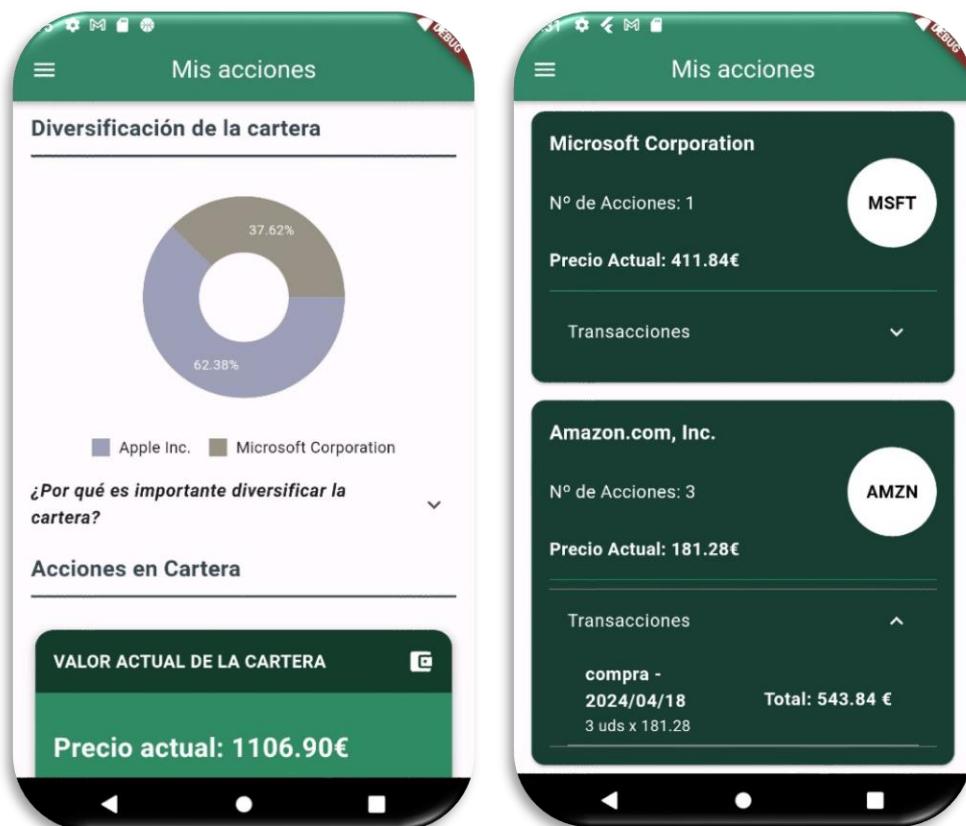


Ilustración 46 Captura de pantalla de la Página Mis Acciones



8.1.10 Página de Noticias

En la sección de noticias es donde los usuarios pueden explorar las últimas noticias delo que está ocurriendo en el mundo de las finanzas. Cuando se accede a este se muestra en la parte superior las opciones para filtrar las noticias, teniendo en primer lugar el buscador para cuando se quiere buscar por nombre o por algún elemento clave y justo debajo, hay una serie de categorías para buscar por temáticas de manera eficiente.

Algún ejemplo de estas categorías son las siguientes:

- Finanzas
- Economía
- Inversiones
- Bolsa

Y a continuación se presenta unas series de artículos y noticias financieras organizadas principalmente por relevancia y popularidad, con un diseño que facilita la comprensión y la lectura de los contenidos más importantes.



Ilustración 47 Captura de pantalla de la Página Noticias



8.1.11 Página de Aprendizaje

En cuanto a la sección de aprendizaje, esta página está diseñada para proporcionar una base de conceptos financieros, desde los más básicos hasta los más avanzados. Tenemos varias opciones y secciones educativas tanto tutoriales de cómo funciona la aplicación como conceptos introductorios y lecciones sobre inversión.

Para explorar los temas disponibles:

1. Utilice las pestañas superiores para alternar entre diferentes temas como 'Tutoriales', 'Introducción a las Finanzas' e 'Inversiones'.
2. Desplácese a través de la página para ver los diferentes módulos educativos disponibles.
3. Seleccione el módulo de interés con solo tocarlo; esto abrirá el contenido completo para su lectura.



Ilustración 48 Captura de pantalla de la Página Aprendizaje



8.1.12 Página de Configuración

Por último, encontramos la página de configuración, que será el centro de control personal para ajustar las preferencias de su cuenta y aplicación. Para acceder a esta, seleccione el ícono del menú en la esquina superior izquierda de la pantalla principal de la aplicación y luego elija la opción ‘Configuración’.

Encontramos varias opciones, por ejemplo, en la sección de ‘Editar Perfil’, puede actualizar su información personal como el nombre, apellidos, contraseña... etc. Simplemente toque ‘Editar Perfil’ para modificar sus detalles. También podrá decidir si recibir notificaciones automáticas activando el interruptor de ‘Activar notificaciones’, para poder estar al tanto de sus acciones favoritas y de cómo fluctúan en el mercado.

En la parte inferior de la Configuración, podemos encontrar enlaces a ‘Sobre nosotros’, ‘Política de Privacidad’ y ‘Términos y Condiciones’ para poder aprender más sobre nosotros los que estamos detrás de la aplicación, cómo protegemos su privacidad y los términos del servicio que acuerda al usar nuestra aplicación.

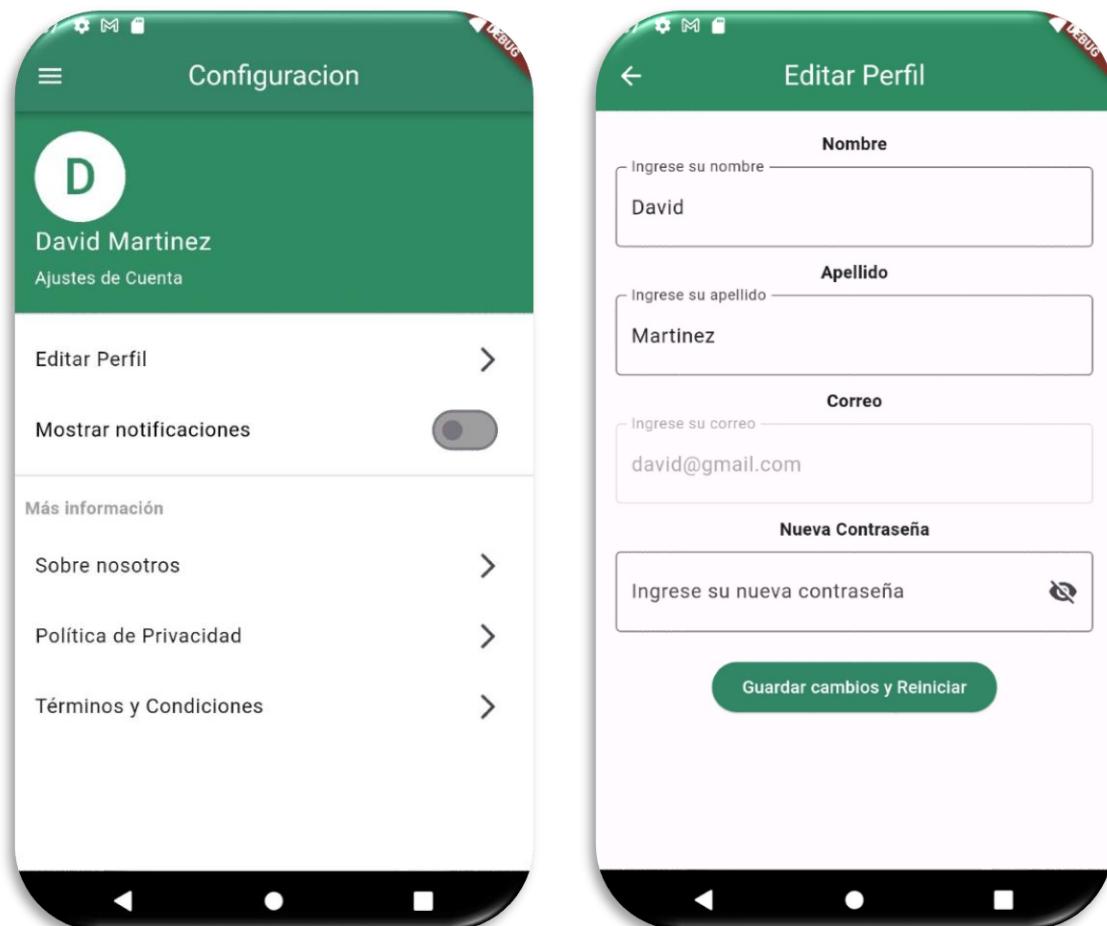


Ilustración 49 Captura de pantalla de la Página Configuración



8.2. Vista del sitio web (Back-end)

8.2.1 Página de inicio

En la página de inicio, te doy la bienvenida con una descripción clara y concisa de lo que Grownomics puede ofrecerte. La misión de Grownomics es empoderar a los usuarios para tomar decisiones de inversión informadas y eficaces, proporcionándoles acceso a datos financieros en tiempo real y análisis exhaustivos.

Aquí se destacan los objetivos principales de la plataforma y se invita a los usuarios a explorar nuevas oportunidades de inversión y gestionar sus carteras con herramientas avanzadas. Mi intención es que los usuarios se sientan motivados a navegar por el sitio y descubrir todas las funcionalidades que Grownomics tiene para ofrecer.

En la sección "Acerca de Grownomics", explico los orígenes y la misión de la plataforma. Fundada en 2024, Grownomics es el resultado de la colaboración entre expertos en tecnología y finanzas, con el objetivo de democratizar el acceso al asesoramiento financiero.

La sección "Nuestros Servicios" presenta las diversas funcionalidades que Grownomics ofrece. Estas incluyen:

- Gestión de Cartera: Herramientas para crear y gestionar una cartera de inversiones diversificada.
- Simulación de Inversiones: Plataforma avanzada para probar diferentes estrategias de inversión en un entorno seguro.
- Análisis en Tiempo Real: Datos y análisis en tiempo real proporcionados por expertos.
- Educación Financiera: Recursos educativos como seminarios web, tutoriales y blogs.
- Alertas Personalizadas: Notificaciones sobre cambios importantes en el mercado o movimientos en las inversiones.

Los usuarios pueden ver un video explicativo de la plataforma para visualizar cómo funciona Grownomics y sus diversas funcionalidades. Además, proporciono un enlace directo a la aplicación Grownomics en Google Play. Los usuarios pueden descargar la app fácilmente utilizando el enlace con la imagen oficial de Google Play.

Finalmente, proporciono información sobre mí, incluyendo una foto y detalles de contacto. Los usuarios pueden conocer más sobre mi background en Ingeniería Informática y Administración y Dirección de Empresas, así como cómo ponerse en contacto conmigo para cualquier consulta o soporte.





Bienvenido a Grownomics

La plataforma innovadora que transforma tu experiencia financiera. En Grownomics, nos dedicamos a empoderarte para tomar decisiones de inversión informadas y eficaces, proporcionándote acceso a datos financieros en tiempo real y análisis exhaustivos. Explora nuevas oportunidades de inversión, gestiona tu cartera con herramientas avanzadas, y aprende a navegar el mercado financiero con confianza. Nuestra misión es simplificar el complejo mundo de las inversiones, haciéndolo accesible, comprensible y operativo para inversores de todos los niveles.

Acerca de Grownomics

Grownomics nació en 2024, fruto de la colaboración entre expertos en tecnología y finanzas con un objetivo común: democratizar el acceso al asesoramiento financiero. Fundada por un grupo de profesionales apasionados por la tecnología y la educación financiera, nuestra plataforma integra funcionalidades inteligentes con una interfaz amigable para facilitar la gestión de inversiones. Con un fuerte enfoque en la seguridad, la precisión y la educación, Grownomics se dedica a ofrecer una experiencia personalizada que ayuda a nuestros usuarios a alcanzar sus metas financieras con mayor claridad y menos riesgos.

Con una sólida base en la investigación y desarrollo, continuamos innovando y adaptándonos a las necesidades cambiantes del mercado para ofrecer las mejores soluciones posibles. Nuestro equipo de expertos trabaja incansablemente para asegurar que la información proporcionada sea actualizada, relevante y fácil de entender.

Nuestros Servicios

Gestión de Cartera Accede a herramientas para crear y gestionar una cartera de inversiones diversificada.	Simulación de Inversiones Prueba diferentes estrategias de inversión en un entorno seguro con nuestra plataforma de simulación avanzada.	Análisis en Tiempo Real Mantente informado con datos en tiempo real y análisis profundos de la mano de nuestros expertos.	Educación Financiera Mejora tus conocimientos con recursos educativos. Desde seminarios web hasta tutoriales y blogs, te proporcionamos la información y las herramientas para entender mejor el mundo financiero.	Alertas Personalizadas Configura alertas personalizadas para recibir notificaciones sobre cambios importantes en el mercado o movimientos en tus inversiones. Reactiva rápidamente a las condiciones del mercado con información actualizada.
---	--	---	--	---

Demostración en Video

Explora más sobre nuestra plataforma viendo el siguiente video:



Descarga la App

Obtén la aplicación desde Google Play.



Contacto

David Martínez Díaz
Estudiante del doble grado en Ingeniería Informática y Administración y Dirección de Empresas. Especializado en la rama de la Ingeniería de Software
Correo: martinezdiaz01@gmail.com

© 2024 Grownomics. Todos los derechos reservados.



Ilustración 50 Página de Inicio de la Web (Back-end)



8.2.2 Página de inicio de sesión Web (Back-end)

En la página de Login de Grownomics, proporciona una interfaz sencilla y fácil de usar para que los administradores puedan acceder a sus cuentas de manera segura. La sección principal de la página de Login está dedicada al formulario de inicio de sesión, que consta de dos campos esenciales:

- Email: Un campo de entrada para que los administradores ingresen su dirección de correo electrónico.
- Contraseña: Un campo de entrada para que los usuarios ingresen su contraseña.

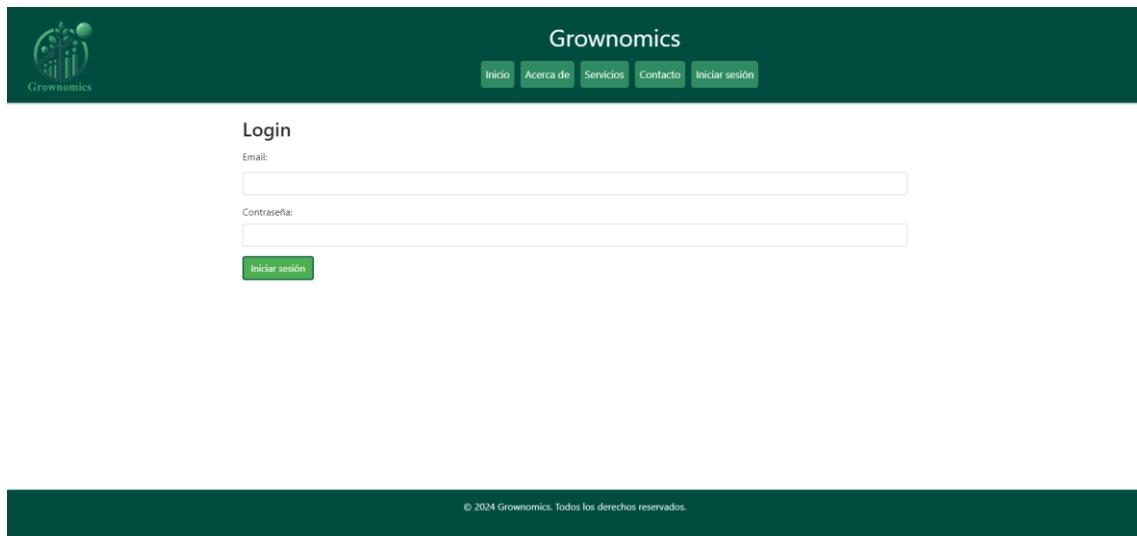


Ilustración 51 Página de Inicio de sesión Web (Back-end)



8.2.3 Página de gestión de la base de datos

En esta página, los administradores tienen acceso a una vista completa y detallada de todas las transacciones, tanto de compra como de venta, que se realizan en la plataforma. La información clave de cada transacción, como el tipo, la cantidad, el precio y la fecha, se presenta en una tabla organizada y fácil de navegar.

Esto permite a los administradores monitorear la actividad de los usuarios y asegurar la integridad y precisión de los datos financieros.

Grownomics					
	Home	Usuario	Cartera	Transaccion	Notificaciones
				Accion	Acciones Favoritas
				Articulos de Aprendizaje	
List (15)	Create	With selected▼			
	Tipo	Cantidad	Precio	Fecha	
<input type="checkbox"/>	compra	4	188.85000610351562	2024-02-12 14:21:20.133656	
<input type="checkbox"/>	compra	3	188.85000610351562	2024-02-12 14:21:28.472531	
<input type="checkbox"/>	venta	3	188.85000610351562	2024-02-12 14:21:38.049814	
<input type="checkbox"/>	compra	2	420.54998779296875	2024-02-12 14:21:48.637008	
<input type="checkbox"/>	venta	1	420.54998779296875	2024-02-12 14:21:55.898622	
<input type="checkbox"/>	compra	0	182.52000427246094	2024-02-24 18:51:41.653991	
<input type="checkbox"/>	compra	3	182.52000427246094	2024-02-24 19:01:59.936197	
<input type="checkbox"/>	venta	3	182.52000427246094	2024-02-24 19:05:07.898669	
<input type="checkbox"/>	compra	5	3000.0	2024-04-01 09:49:09.569538	
<input type="checkbox"/>	compra	5	3000.0	2024-04-01 09:57:29.827963	
<input type="checkbox"/>	compra	5	3000.0	2024-04-01 10:00:22.434894	
<input type="checkbox"/>	compra	5	3000.0	2024-04-03 16:12:55.024652	
<input type="checkbox"/>	compra	5	3000.0	2024-04-03 16:32:28.721545	
<input type="checkbox"/>	compra	3	0.0	2024-04-18 09:49:29.512305	
<input type="checkbox"/>	compra	3	181.27999877929688	2024-04-18 10:30:25.479891	

Ilustración 52 Página de gestión de la base de datos



8.3. Tableros Scrum con Jira Software

8.3.1 Tablero Sprint 1

Proyectos / Grownomics
Tablero Sprint 1
Establecer los requisitos del proyecto, conseguir conocimientos sobre la investigación de mercado y definición de las funcionalidades claves de la aplicación y crear prototipos básicos.

PLANIFICACIÓN

- Cronograma
- Backlog
- Tablero**
- Calendario NOVEDAD
- Lista
- Objetivos
- Incidencias AI
- + Añadir vista

DESARROLLO

- Código
- Páginas de proyecto...
- Añadir acceso rápido
- Configuración del pro...

Ilustración 53 Tablero Sprint 1

8.3.2 Tablero Sprint 2

Proyectos / Grownomics
Tablero Sprint 2
Desarrollo de la arquitectura del software y diseño de la interfaz de usuario, centrándonos principalmente en la experiencia del usuario.

PLANIFICACIÓN

- Cronograma
- Backlog
- Tablero**
- Calendario NOVEDAD
- Lista
- Objetivos
- Incidencias AI
- + Añadir vista

DESARROLLO

- Código
- Páginas de proyecto...
- Añadir acceso rápido
- Configuración del pro...

Ilustración 54 Tablero Sprint 2



8.3.3 Tablero Sprint 3

The screenshot shows the Jira Software interface for the 'Grownomics' project. The left sidebar is titled 'Proyectos / Grownomics' and includes sections for 'PLANIFICACIÓN' (Cronograma, Backlog), 'DESARROLLO' (Calendario NOVEDAD, Lista, Objetivos, Incidencias AI), and 'DESPARROLLO' (Código). The main area is titled 'Tablero Sprint 3' and displays three columns: 'POR HACER 1 DE 3' (HU.17 Como usuario, quiero establecer metas financieras y recibir recomendaciones para alcanzarlas. SCRUM-17), 'EN CURSO 1 DE 3' (HU.8 HU SPIKE - Diseñar una interfaz intuitiva y fácil de usar. SCRUM-8), and 'LISTO 4 DE 5' (HU.6 Como usuario, quiero obtener sugerencias de estrategias de inversión. SCRUM-6, HU.7 Como usuario, quiero realizar simulaciones de inversiones. SCRUM-7, HU.24 Como usuario, quiero acceder a un resumen del mercado diario para tener una visión general del comportamiento del mercado. SCRUM-24, HU.15 Como usuario, quiero recibir noticias financieras relevantes para tomar decisiones informadas. SCRUM-15). A search bar and filter options are at the top.

Ilustración 55 Tablero Sprint 3

8.3.4 Tablero Sprint 4

The screenshot shows the Jira Software interface for the 'Grownomics' project. The left sidebar is identical to the previous one. The main area is titled 'Tablero Sprint 4' and displays three columns: 'POR HACER 1 DE 4' (HU.11 Como usuario, quiero recibir recomendaciones basadas en mi perfil financiero y objetivos. SCRUM-11), 'EN CURSO 1 DE 4' (HU.19 Como usuario, quiero tener acceso a un historial completo de mis transacciones e inversiones. SCRUM-19), and 'LISTO 4 DE 9' (HU.9 Como usuario, quiero analizar datos detallados y gráficos. SCRUM-9, HU.10 Como usuario, quiero acceder a herramientas avanzadas de análisis. SCRUM-10, HU.21 Como usuario, quiero ver la evolución de mi balance a través de gráficas. SCRUM-21, HU.25 Como usuario, quiero acceder a una sección de "Mis Acciones" para ver el desempeño de mis inversiones individuales. SCRUM-25). A search bar and filter options are at the top.

Ilustración 56 Tabla Sprint 4



8.3.4 Tablero Sprint 5

The screenshot shows the Jira Software interface for the 'Grownomics' project under the 'Proyectos' section. The 'Tablero Sprint 5' board is displayed, featuring three columns: 'POR HACER 1 DE 5', 'EN CURSO 1 DE 5', and 'LISTO 3 DE 12'. Each column contains several user stories with their status, priority, and assignee.

Categoría	Tarea	Estado	Prioridad	Asignado a
POR HACER	HU.20 Como usuario, quiero poder compartir información sobre mis inversiones con asesores financieros.	Pendiente	Bajo	SCRUM-20
	HU.12 Como usuario, quiero poder realizar un seguimiento detallado de mis inversiones.	Pendiente	Bajo	SCRUM-12
EN CURSO	HU.4 Como usuario, quiero recibir alertas en tiempo real sobre mis inversiones.	En curso	Bajo	SCRUM-4
	HU.18 HU SPIKE - Implementar un sistema de notificaciones push para alertas importantes.	En curso	Bajo	SCRUM-18
LISTO	HU.27 Como usuario, quiero ver un historial detallado de las alertas recibidas para hacer un seguimiento de la información importante.	Finalizado	Bajo	SCRUM-27
	HU.19 Como usuario, quiero...	Finalizado	Bajo	SCRUM-19

Ilustración 57 Tablero Sprint 5



8.4. Código fuente

El desarrollo completo del proyecto, incluyendo cada una de las contribuciones realizadas a lo largo del tiempo, está disponible y puede ser consultado a través del siguiente enlace a GitHub: <https://github.com/Duva-01/PFG-Informatica>

Aquí encontrarás todos los commits realizados durante el proceso de creación y desarrollo del proyecto.



8.5. Enlace a la Web, Google Play Store y Video Explicativo

En esta sección, presentamos un video explicativo y los enlaces necesarios para descargar nuestra aplicación y acceder a nuestro servidor.

El siguiente video proporciona una visión general de nuestra aplicación para asesoramiento financiero bursátil. En el video, se explican las funcionalidades principales y los beneficios de usar nuestra aplicación:



Enlace alternativo: <https://www.youtube.com/watch?v=Xo1dega6FgY>

Para descargar la aplicación en tu dispositivo Android, haz clic en la imagen a continuación. Serás redirigido a Google Play Store, donde podrás instalar la aplicación de forma gratuita:



Enlace alternativo:

https://play.google.com/store/apps/details?id=com.david.grownomicspfg&pcampaignid=web_share

Para acceder a nuestro servidor y explorar más sobre los servicios que ofrecemos, visita el siguiente enlace: [Servidor de la Aplicación](#).

