



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática



**TFG del Grado en Ingeniería
Informática**

**título del TFG
Documentación Técnica**



Presentado por nombre alumno
en Universidad de Burgos — 9 de junio de 2024
Tutor: nombre tutor

Índice general

Índice general	i
Índice de figuras	iii
Índice de tablas	v
Apéndice A Plan de Proyecto Software	1
A.1. Introducción	1
A.2. Planificación temporal	1
A.3. Estudio de viabilidad	11
Apéndice B Especificación de Requisitos	17
B.1. Introducción	17
B.2. Objetivos generales	18
B.3. Catálogo de requisitos	18
B.4. Especificación de requisitos	21
Apéndice C Especificación de diseño	59
C.1. Introducción	59
C.2. Diseño de datos	59
C.3. Diseño procedimental	64
C.4. Diseño arquitectónico	66
Apéndice D Documentación técnica de programación	81
D.1. Introducción	81
D.2. Estructura de directorios	81
D.3. Manual del programador	83

D.4. Pruebas del sistema	91
Apéndice E Documentación de usuario	97
E.1. Introducción	97
E.2. Requisitos de usuarios	97
E.3. Instalación	97
E.4. Manual del usuario	97
Apéndice F Anexo de sostenibilización curricular	99
F.1. Introducción	99
Bibliografía	101

Índice de figuras

B.1. Diagrama de casos de uso. Fuente: elaboración propia	22
C.1. Diseño de bases de datos. Fuente: elaboración propia	60
C.2. Diagrama E/R	62
C.3. Diagrama relacional	63
C.4. Diagrama de secuencias <i>News y Analysis</i>	64
C.5. Diagrama de secuencias <i>DashBoard</i>	65
C.6. Diagrama de secuencias <i>Lab</i>	66
C.7. Diagrama de patrón MVT. Fuente: elaboración propia	67
C.8. Patrón repositorio. Fuente: elaboración propia	68
C.9. Diagrama de paquetes general.	70
C.10. Diagrama de paquete <i>FAT</i>	70
C.11. Diagrama de paquete <i>News</i>	71
C.12. Diagrama de paquete <i>Analysis</i>	72
C.13. Diagrama de paquete <i>DashBoard</i>	73
C.14. Diagrama de paquete <i>Lab</i>	74
C.15. Diagrama de paquete <i>util</i>	74
C.16. Diagrama de directorio <i>tests</i>	75
C.17. Diagrama de directorio <i>log</i>	76
C.18. Diagrama de directorio <i>htmlcov</i>	77
C.19. Diagrama de directorio <i>databases</i>	77
C.20. Diagrama de directorio <i>docs</i>	78
C.21. Diagrama de directorio <i>.github</i>	79
D.1. Descargar proyecto en formato <i>.zip</i>	84
D.2. Clonar con <i>Git</i> <i>.zip</i>	84
D.3. VS Code	85
D.4. Selección de <i>interpreter</i>	86

D.5. Activar entorno virtual	87
D.6. Lanzar el servidor	88
D.7. Número de tests realizados	92
D.8. Lanzar pruebas de interfaz	95
D.9. Resultado de pruebas de interfaz	95

Índice de tablas

A.1. Costes en RRHH	12
A.2. Costes de <i>HW</i> y <i>SW</i>	12
A.3. Costes infraestructura	13
A.4. Costes totales	13
A.5. Licencias de terceros	14
B.1. CU-1 Consultar portada.	23
B.2. CU-2 Mostrar carrusel noticias generales.	24
B.3. CU-3 Mostrar mejores y peores valores.	25
B.4. CU-4 Mostrar gráfica.	26
B.5. CU-5 Gestionar usuario.	27
B.6. CU-6 Registro.	28
B.7. CU-6 Login.	29
B.8. CU-8 Logout.	30
B.9. CU-9 <i>DashBoard</i>	31
B.10.CU-10 Crear nuevo valor en cartera.	32
B.11.CU-11 Mostrar <i>donut</i> , sectores y divisas.	33
B.12.CU-12 Ver gráfica de Markowitz, ratio de Sharpe y pesos.	34
B.13.CU-13 Ver posiciones abiertas y evolución de cartera.	35
B.14.CU-14 Eliminar valor de cartera.	36
B.15.CU-15 Crear un nuevo valor en seguimiento.	37
B.16.CU-16 Eliminar un valor de seguimiento.	38
B.17.CU-17 Consultar índice.	39
B.18.CU-18 Mostrar tabla de valores.	40
B.19.CU-19 Consultar un valor del índice.	41
B.20.CU-20 Mostrar gráfica interactiva.	42
B.21.CU-21 Mostrar distribución de retornos.	43
B.22.CU-22 Mostrar datos del último mes.	44

B.23.CU-23	Mostrar evolución del sector.	45
B.24.CU-24	Comparar con otros valores.	46
B.25.CU-25	Mostrar grafos de correlación.	47
B.26.CU-26	Mostrar noticias relacionadas.	48
B.27.CU-27	Gestionar <i>Lab</i>	49
B.28.CU-28	Trabajar con ARIMA.	50
B.29.CU-29	Buscar (p,d,q) con funciones ACF y PACF.	51
B.30.CU-30	Introducir (p,d,q) manualmente.	52
B.31.CU-31	Estimar (p,d,q) de forma automática.	53
B.32.CU-32	Hacer búsqueda por rejilla para (p,d,q).	54
B.33.CU-33	Trabajar con <i>trading</i> algorítmico.	55
B.34.CU-34	Usar algoritmo de cruce de medias.	56
B.35.CU-35	Usar estrategia basada en <i>machine learning</i>	57
D.1.	Cobertura de código	92
D.2.	Requerimientos testeados con pruebas de interfaz.	94

Apéndice A

Plan de Proyecto Software

A.1. Introducción

En un proyecto con un equipo completo de personas trabajando, podríamos definir la planificación de un proyecto software como una etapa que implica detallar los objetivos y el alcance del proyecto, identificar los requisitos y recursos necesarios, establecer un cronograma con hitos clave, asignar tareas a los miembros del equipo y prever los posibles riesgos.

Dentro del contexto de este trabajo académico, se va a detallar en qué ha consistido la planificación temporal y se tratará de hacer un estudio de viabilidad realista.

A.2. Planificación temporal

En la memoria se ha indicado que se ha utilizado una metodología ágil de gestión de proyectos, con clara fundamentación en Scrum[7]. Algunos de los aspectos más relevantes que han cubierto esta filosofía han sido:

- Desarrollo incremental a través de iteraciones llamadas *sprints*.
- Utilización de repositorio *Git* para solicitud de mejoras y para realizar un seguimiento de la evolución, con acceso para los tutores desde las fases iniciales.
- Control temporal a través de los *sprints*, valorando al inicio de cada uno cuál sería la duración adecuada basándonos en las tareas que se iban a realizar y el producto que se esperaba al final de esa iteración.
- Seguimiento de tareas en un panel *Kanban*.

Además de los *sprints* se diseñaron hitos relevantes denominados *Milestones* en *GitHub* que sirven como referencia de la producción realizada.

Sprints

A continuación se muestra un resumen de los *sprints* que han tenido lugar en las diferentes fases de este trabajo:

Sprint 0. Presentación inicial a tutores del TFG

- **Duración:** 19/10/2023 a 23/10/2023
- **Objetivo:** Presentar una idea de TFG a los posibles tutores.
- **Contexto:** Inicio del último curso del Grado de Ingeniería Informática. Se trata de contactar con los posibles tutores que controlen el desarrollo de un TFG. Este TFG estará basado en la idea de desarrollar una herramienta, escrita en Python, para realizar análisis financiero.
- **Tareas:**
 - Contactar con los posibles tutores por email.
 - Contactar por Teams con los tutores que muestren interés en el TFG e intercambiar primeras impresiones.
 - Determinar las herramientas necesarias para desarrollar un TFG.
 - Preguntar por consejos y opiniones que puedan ser relevantes.
 - Concertar cita para una futura reunión cuando se haya avanzado con los primeros pasos del TFG.

Sprint 1. Empezar TFG con selección de herramientas adecuadas

- **Duración:** 24/10/2023 a 08/11/2023
- **Objetivo:** Empezar el desarrollo del TFG y su correspondiente documentación. Hacerlo utilizando las herramientas adecuadas sugeridas por los tutores en el trascurso del sprint 0.
- **Contexto:** Tras haber mantenido una primera reunión con los tutores, se detectan ciertas necesidades en cuanto a la utilización de herramientas adecuadas para el desarrollo de un TFG.
- **Tareas:**

- Lectura completa de documentación de TFGs disponible en la plataforma UBUVirtual.
- Rellenar el formulario de *Oferta de TFG Grado de Informática Online*. Buscar referencias bibliográficas teóricas más adecuadas.
- Utilización de *Zube*[5] para gestionar un proyecto con metodología ágil. Para ello, hay que buscar la documentación adecuada que permita entender su utilización.
- Crear un repositorio en *GitHub* que permita llevar un seguimiento de las acciones realizadas.
- En el repositorio, añadir a los tutores e integrar con *Zube*.
- Determinar la mejor herramienta para la documentación del TFG. Elegir entre \LaTeX y *Word*. Justificar la elección y documentar.
- Buscar un gestor de referencias bibliográficas: *Zotero vs Mendeley*.
- Concertar cita para una futura reunión cuando se haya avanzado con los primeros pasos del TFG.

Sprint 2. Mostrar primeras tablas y gráficos en servidor web

- **Duración:** 08/11/2023 a 29/11/2023
- **Objetivo:** Comenzar a desarrollar una aplicación web, mostrando información de precios cotizados y de gráficos. Almacenar información en una base de datos SQLite3, no en DataFrames ni en archivos .csv.
- **Contexto:** Una de las primeras recomendaciones de los tutores fue no utilizar archivos .csv para almacenar la información, por lo tanto, se va a utilizar un sistema de almacenamiento en base de datos. La idea general es que dichos datos se descargan con una API, se almacenan y, posteriormente, se procesan para mostrar la información relevante al usuario en archivos HTML.
- **Tareas:**
 - Preparar entorno local para utilizar una base de datos SQLite3. Instalar una GUI para la BD (no relevante para el usuario final).
 - Preparar clases (Descargador, Vista, Ticker y/o similares) que permitan automatizar la tarea de almacenar información en una base de datos.
 - Mostrar un primer producto - no necesariamente visualmente atractivo - en un servidor web con tablas extraídas de la BD.
 - Documentar, en el capítulo 4 de la memoria, las herramientas utilizadas.

- Concertar cita para una futura reunión con los tutores.

Sprint 3. Mejorar lógica de web y proteger enlaces con registro. Mostrar gráficos y dar primeros estilos

- **Duración:** 29/11/2023 a 13/12/2023
- **Objetivo:** Mejorar la lógica de acceso a la web. Proteger los enlaces no públicos mediante registro de usuarios. Mostrar gráficos y no sólo tablas en un estilo visualmente atractivo.
- **Contexto:** Una vez llevado el proyecto a *producción* (subido a un servidor web) hay que mejorar y asegurar los enlaces para que no sean accesibles en un mal uso de los mismos. Además, es necesario hacer un registro de usuarios que permita mostrar información relevante dependiendo del rol (invitado / registrado).
- **Tareas:**
 - Crear lógica de control de usuarios registrados.
 - Asignar permiso de acceso a enlaces dependiendo del tipo de usuario.
 - Crear portada y dar primeros estilos visuales.
 - Preparar un índice o zona de *breadcrumbs* de la página para mejorar la navegación.
 - Diseñar una vista que permita mostrar gráficos según el valor cotizado escogido.

Sprint 4. Manejar imágenes estáticas. Añadir base de datos de DJ30 y buscar método para auto-actualizar las BDs

- **Duración:** 14/12/2023 a 28/12/2023
- **Objetivo:** Mejorar estética y mostrar imágenes estáticas. Crear y añadir una nueva base de datos. Hacer las bases de datos auto-actualizables.
- **Contexto:** El proyecto en local permite mostrar imágenes estáticas en la navbar y en portada, pero en *producción* (subido a un servidor web) no. Hay que mejorar este aspecto para que resulte visualmente atractivo. Además, hay que crear y añadir una nueva base de datos para el índice DJ30 y hay que buscar un método para auto-actualizar las BD en producción: con *cron*[6] o con *tasks*.

■ Tareas:

- Mostrar imágenes estáticas en producción (en servidor web).
- Crear una nueva BD y hacerla accesible.
- Investigar e implantar un método de auto-actualización de BDs en producción.

Sprint 5. Añadir un carrusel de noticias y cambiar el acceso a las BDs. Incorporar tests y mejorar los logs

■ **Duración:** 04/01/2024 a 18/01/2024

■ **Objetivo:** Añadir noticias relacionadas con los mercados financieros, para hacer un interfaz más útil y agradable para el usuario. Cambiar la lógica de acceso a las bases de datos, para evitar las sentencias en SQL y aprovechar las capacidades del patrón MVT[4] de Django. Aplicar primeros tests para mejorar el comportamiento y aumentar la seguridad.

■ **Contexto:** Ahora mismo la portada principal de la página es una imagen estática que permite hacer *login*, pero que no aporta la utilidad que se espera de este tipo de aplicaciones. Por ello, parece razonable añadir una zona de noticias relacionadas con los mercados financieros, que hagan la página más dinámica y agradable de usar. Además, aunque tengo diseñados unos pequeños tests, se tiene que mejorar toda la estructura de *testing* y ampliar el *logger*.

■ Tareas:

- Añadir una nueva aplicación al proyecto, que permita controlar lo que ocurre en la portada:
 - Mostrar noticias en portada.
 - Mostrar información sobre los mejores y peores stocks en portada.
 - Mejorar el logging para lo que ocurra en esta nueva aplicación.
- Mejorar la lógica de acceso a las BDs. Aplicar MVC y evitar consultas directas de SQL en la medida de lo posible.
- Ampliar los tests y mejorar la estructura de testing.

Sprint 6. Añadir un *dashboard* para almacenar y controlar una cartera por parte del usuario. Operaciones CRUD en BDs

■ **Duración:** 22/01/2024 a 05/02/2024

- **Objetivo:** Añadir un área personal que permita realizar el control de una cartera de acciones, con posiciones abiertas y posiciones objetivo. Poder crear informes con la composición de la cartera y mostrar porcentajes de inversión de forma agradable al usuario.
- **Contexto:** Aunque un usuario puede consultar los datos de un valor concreto y de su índice de referencia, ahora mismo no puede tener una lista de seguimiento que le permita controlar sus inversiones. Se pretende integrar una nueva aplicación que permita ofrecer un *dashboard* a los usuarios, para que puedan hacer un seguimiento de sus inversiones.
- **Tareas:**
 - Añadir una nueva aplicación al proyecto, que permita controlar un dashboard.
 - Permitir al usuario almacenar información sobre sus inversiones actuales y futuras.
 - Mostrar acciones compradas, i.e., posiciones abiertas.
 - Mostrar una lista de seguimiento.
 - Mejorar el *logging* para lo que ocurra en esta nueva aplicación.
 - Preparar los tests más relevantes para el control de esta nueva aplicación.
 - Integrar esta aplicación con las actuales.
 - Aprovechar la lógica mejorada de acceso a las BDs para facilitar todas las operaciones CRUD.
 - Ampliar los tests.

Sprint 7. Refactoring y testing. Poder eliminar stocks en seguimiento desde el DashBoard.

- **Duración:** 07/02/2024 a 14/02/2024
- **Objetivo:** Ahora mismo hay bastantes casos sin cubrir con los tests, especialmente en las *views* de las *apps* incorporadas al proyecto. Por ello, se hace necesario, antes de continuar, realizar un testeo exhaustivo para asegurar una buena base en desarrollos posteriores.

Además, voy a incorporar una nueva funcionalidad dentro del DashBoard, que permitirá al usuario eliminar valores en seguimiento (de manera similar a como se eliminan actualmente los valores en cartera).

- **Contexto:** Tras intentar seguir un enfoque TDD en la creación del DashBoard en el sprint 6, se detecta la necesidad de resolver algunos fallos y cubrir el código, al 100 %, con tests. Además, hay que mejorar el estilo y la calidad del código, así como medir dicha calidad con alguna herramienta que ofrezca información final que justifique las mejoras.
- **Tareas:**
 - Cubrir todas las normas de estilo de *PEP-8* enhancement
 - Solucionar problema formularios *DashBoard* con precios de más de 2 decimales.
 - Incluir funcionalidad de eliminación de stocks en seguimiento.
 - Completar tests unitarios de `News.views`
 - Completar tests unitarios `DashBoard.views`
 - Completar tests unitarios de `Analysis.views`
 - Medir con *flake8* y *pylint* la calidad del código.

Sprint 8. Correlaciones entre valores con NetworkX

- **Duración:** 17/02/2024 a 02/03/2024
- **Objetivo:** Permitir al usuario comparar la evolución de un valor con respecto al resto, en un tiempo determinado. La idea es mostrar al usuario un grafo creado con *NetworkX*[\[3\]](#) que permita entender las correlaciones (positiva y negativa) que haya entre diferentes valores. De esta manera se podrá hacer un análisis de diversificación de cartera, ya que tener valores muy correlacionados puede ser indicativo de no tener la cartera bien balanceada.
- **Contexto:** El usuario puede acceder a la información de un valor pero no puede compararlo con otros. Además, aunque tenga valores comprados o en seguimiento, realmente, ahora mismo, no se le está ofreciendo capacidad analítica, así que hay que proporcionársela.
- **Tareas:**
 - Añadir dos índices adicionales, con sus valores, para ampliar la información disponible y ofrecer un mayor abanico comparativo.
 - Agregar una funcionalidad que permita ver la correlación de un valor con todos los demás disponibles.
 - Como la correlación puede ser positiva o negativa, hacer diferenciación entre ambas.

- Además, puede que interese obtener información sobre las mayores correlaciones (positivas o directas; y negativas o inversas)
 - Una vez obtenidas las correlaciones, sobre los precios de cierre, ver la información en grafos de *NetworkX*.
- Adicionalmente, intentar permitir al usuario ver los gráficos de la mayor correlación de forma conjunta.
- Ampliar los tests a la nueva funcionalidad.
- Integrar los grafos en la plataforma web de forma temporal para cada usuario (sin almacenar en el servidor).

Sprint 9. Aplicación de modelos. *Forecasting* para series temporales

- **Duración:** 13/03/2024 a 27/03/2024
- **Objetivo:** Aportar una visión diferenciadora al análisis de valores cotizados, permitiendo que un usuario pueda aplicar un modelo que le permita tener una idea intuitiva de la futura evolución de la cotización. En cualquier caso, se informará al usuario de que los datos aportados por el modelo no pueden ser considerados como una fuente única para tomar decisiones de inversión.
- **Contexto:** En las webs que ofrecen información de productos cotizados no se ofrece un apartado de experimentación con modelos de *forecasting*. Puede ser un factor diferenciador para aquellos usuarios que entiendan los riesgos y las ventajas de aplicar estos modelos.
- **Tareas:**
 - Añadir el análisis a través de un modelo ARIMA.
 - Añadir el análisis a través de una red LSTM.
 - Permitir al usuario ver alguna información gráfica que facilite la comprensión de los resultados.
 - Ampliar los tests de las nuevas funcionalidades.
 - Integrar las gráficas o diagramas en la plataforma web de forma temporal para cada usuario (sin almacenar en el servidor).

Sprint 10. Documentación y limpieza de código y avanzar considerablemente en la memoria.

- **Duración:** 10/04/2024 a 24/04/2024

- **Objetivo:** Documentación de código y de la memoria.
- **Contexto:** Actualmente el proyecto está documentado con notas internas del autor y debe empezarse a documentar utilizando las herramientas y guías oficiales.
- **Tareas:**
 - Configurar *MikTex*.
 - Descargar y preparar plantilla oficial de Trabajos de Fin de Grado.
 - Añadir resumen / abstract.
 - Documentar introducción.
 - Describir objetivos del proyecto.
 - Documentar conceptos teóricos.
 - Explicar las técnicas y herramientas utilizadas.
 - Indicar aspectos relevantes del desarrollo del proyecto que se hayan dado hasta ahora.
 - Detallar trabajos relacionados.
 - Añadir referencias bibliográficas en todos los apartados de la memoria.
 - Medir la calidad del código e introducir las mejoras necesarias.

Sprint 11. Gráfica de Markowitz con frontera eficiente y ratio de Sharpe. Mejoras visuales.

- **Duración:** 09/05/2024 a 19/05/2024
- **Objetivo:** Favorecer el estudio de una cartera de valores a través de la distribución de pesos de los mismos. Mostrar una gráfica de Markowitz con la cartera ideal encontrada por simulación de Montecarlo y, también, por optimización de funciones. Realizar pequeñas mejoras estéticas y funcionales.
- **Contexto:** Es necesario mejorar el *DashBoard* de los usuarios para que reciban una información más completa y, de entre las opciones barajadas, una de las más interesantes es la de mostrar cuál sería la mejor distribución de los valores que ya tenga en cartera el usuario, basándome en los retornos y pesos en la propia cartera.
- **Tareas:**
 - Añadir nuevos métodos que permitan hacer una simulación de Montecarlo con diferentes pesos de valores en una cartera.

- Añadir nuevos métodos al *DashBoard* para calcular la frontera eficiente.
- Desarrollar las funciones de optimización que permitan calcular la mejor distribución de pesos.
- Preparar los métodos necesarios para mostrar la gráfica de Markowitz junto con la frontera eficiente y las mejores carteras encontradas.
- Implementar los tests necesarios para las nuevas funcionalidades.
- Integrar en las plantillas existentes del *DashBoard* para mostrar.
- Documentar de forma interna en el apartado de conceptos teóricos (apartado 3 de la memoria del TFG).
- Mejoras estéticas en botones y página principal.
- Añadir referencias bibliográficas en todos los apartados de la memoria aunque sea a nivel local.
- Medir la calidad de nuevo código y testear de forma automática con *GitHub actions*.
- Desplegar nueva documentación de código de forma automática con *Sphinx*.

Sprint 12. Mejorar gráficas ARIMA. Incluir estrategias basadas en *machine learning*

- **Duración:** 22/05/2024 a 26/05/2024
- **Objetivo:** Completar documentación teórica de ARIMA en memoria, añadir apartado de *trading* algorítmico con su documentación y contemplar redes LSTM como posible mejora del proyecto.
- **Contexto:** Ahora mismo se puede acceder a un apartado de *forecasting* con redes LSTM pero no me convencen los resultados obtenidos, así que se dejará como una de las posibles mejoras del proyecto. Sin embargo, voy a añadir un apartado nuevo al *Lab* sobre *trading* algorítmico.

Además, tengo que terminar la documentación de ARIMA en la memoria y añadir la nueva documentación de *trading* algorítmico.

- **Tareas:**
 - Terminar documentación ARIMA.
 - Mejorar gráficas y plantillas HTML para ARIMA.
 - Incluir mejoras de estrategia basada en ML, en lugar de redes LSTM, y documentar.
 - Pasar redes LSTM a apartado de posibles mejoras del proyecto.

- Realizar nuevos tests para estrategias basadas en ML.
- Crear un *release* con el cambio, porque es relevante.

Sprint 13. Finalizar memoria. Empezar anexos. Correcciones en plantillas y en *cron* de servidor web

- **Duración:** 31/05/2024 a 04/06/2024
- **Objetivo:** Finalizar memoria y empezar anexos. Realizar mejoras estéticas. Corregir tarea *cron* de actualización de BDs en servidor web. Realizar la *release* que está pendiente desde el *sprint* anterior, no realizado por falta de comprobaciones.
- **Contexto:** En este momento es posible trabajar con todas las herramientas, pero quedan algunos detalles que mejorar antes de realizar un *release*. Además, es necesario avanzar con la documentación de la memoria (que ya tiene cubierta toda la parte teórica).
- **Tareas:**
 - Finalizar sección 5 de memoria.
 - Finalizar sección 6 de memoria.
 - Finalizar sección 7 de memoria.
 - Empezar anexos.
 - Mejorar gráficas y plantillas HTML para estrategias basadas en ML.
 - Ampliar tests para cubrir todos los métodos.
 - Mejorar la calidad de código. Comprobar con *pylint*.
 - Mejoras gráficas en grafos de correlaciones de *NetworkX* y adaptación de plantillas en local.
 - Corregir problemas de tarea *cron* en servidor web.
 - Crear un *release* con los cambios, siempre y cuando haya terminado todas las correcciones.

A.3. Estudio de viabilidad

Viabilidad económica

Se va a realizar una estimación de costes lo más aproximada a la realidad, como si el proyecto se hubiese realizado en un entorno empresarial.

Estudio de costes

De manera general se pueden considerar costes de personal, de *hardware* y *software* y, en caso de necesidad, costes de infraestructura. La mayor partida económica se tiene que destinar a los recursos humanos:

Concepto	Coste anual	Prorratio (6 meses)
<i>Salario bruto</i>	25.000,00€	12.500,00€
<i>Retención IRPF</i>	4.250,00€	2.125,00€
<i>Seguridad Social</i>	1.587,50€	793,75€
<i>Salario neto</i>	19.162,50€	9.581,25€

Tabla A.1: Costes en RRHH

Se aplica un 17 % de retención sobre la nómina, considerando ausencia de deducciones tributarias.

A continuación se muestran los costes de *hardware* y *software*. Para el *hardware* se estima una amortización de 5 años y una utilización de 6 meses. El *software*, por su parte, tendrá una amortización estimada de 2 años.

Concepto	Coste	Coste amortizado
<i>Ordenador portátil</i>	1.600,00€	160,00€
<i>Licencia MS Windows 10 pro</i>	279,00€	69,75€

Tabla A.2: Costes de HW y SW

Por último, se realiza un análisis de costes de infraestructura (solo asimilable en caso de necesidad) de nuevo, el coste amortizado se calcula a 6 meses:

Concepto	Coste anual	Coste amortizado
<i>Consumo eléctrico</i>	350,00€	175,00€
<i>Espacio coworking</i>	4.000,00€	2.000,0€
<i>Material de oficina</i>	10,00€	5,00€
<i>Alojamiento web</i>	72,00€	36,00€

Tabla A.3: Costes infraestructura

Los costes totales estimados son:

Concepto	Coste
<i>Costes en RRHH</i>	12.500,00€
<i>Costes de HW y SW</i>	229.75€
<i>Costes infraestructura</i>	2.216,00€
TOTAL	14.945,75€

Tabla A.4: Costes totales

Análisis de beneficios

Este trabajo no se plantea inicialmente como un mecanismo para generar beneficios económicos, sino como una herramienta gratuita para ayudar a los inversores a mejorar la distribución de sus carteras y para que los analistas financieros experimenten con diferentes técnicas.

En caso de monetizar el proyecto se podría optar por incluir publicidad en la web para generar ingresos que, al menos, cubran los gastos anuales.

Viabilidad legal

Descargo de responsabilidades

El *software* que se proporciona en este trabajo es solo para fines informativos y no debe considerarse como asesoramiento financiero. La información proporcionada no está garantizada como precisa o completa y puede cambiar sin previo aviso.

El usuario es el único responsable de sus decisiones de inversión y de las consecuencias de las mismas. No se debe confiar en este *software* para tomar

decisiones de inversión sin realizar una investigación profunda y consultar con un asesor financiero calificado.

Licencias *software*

En este trabajo se utilizan múltiples librerías de terceros. Además, aunque los datos son almacenados en bases de datos propias, hay que considerar que se hace uso de información externa a través de la API *yFinance*.

Las licencias asociadas a las librerías y APIs utilizadas, se detallan en la siguiente tabla:

Librería / API	Licencia
<i>Python</i>	OSI-Open Source
<i>Django</i>	BSD
<i>yFinance</i>	Apache 2.0
<i>Pandas</i>	BSD
<i>Numpy</i>	BSD 3-Clause
<i>Scikit-Learn</i>	BSD 3-Clause
<i>Scipy</i>	BSD
<i>NetworkX</i>	BSD 3-Clause
<i>Matplotlib</i>	Licencia libre propia
<i>Plotly</i>	MIT
<i>Statsmodels</i>	BSD 3-Clause
<i>Feedparser</i>	BSD 3-Clause
<i>NewsAPI</i>	MIT
<i>pmdarima</i>	MIT
<i>Keras</i>	Apache 2.0

Tabla A.5: Licencias de terceros

Todas las librerías y el *framework Django* utilizan licencias de código abierto permisivas - la más restrictiva sería Apache 2.0 - lo que significa que el código de *FAT: Financial Analysis Tool* se puede usar, modificar y distribuir libremente, incluyendo para fines comerciales, siempre que se cumplan los requisitos de atribución y exención de responsabilidad establecidos en cada licencia.

Por tanto, se toma la decisión de publicar este proyecto bajo licencia CC BY-NC-SA 4.0[2].

Consideraciones adicionales

Dada la condición académica de este proyecto no hay limitación en el uso de los datos financieros almacenados y no se recoge información personal; pero dependiendo del uso que se dé a este *software* es posible que se deba cumplir con regulaciones específicas, como la Ley de Protección de Datos de la Unión Europea (GDPR) o la Ley de Protección de la Privacidad de la Información Financiera (FINRA).

Apéndice B

Especificación de Requisitos

B.1. Introducción

En este anexo se recogen los requisitos y casos de uso que se han tenido en cuenta para el desarrollo de este trabajo. Se han seguido las recomendaciones del estándar IEEE 830-1998[1] como guía de buenas prácticas.

En este proyecto se ha tratado de crear una especificación de requisitos que cumpla con las siguientes condiciones:

- Completa y consistente
 - Incluir todos los requisitos y referencias necesarias.
 - Ser coherente con los propios requisitos y otros documentos de especificación.
- Claridad y accesibilidad
 - Redacción clara para evitar malas interpretaciones.
 - Uso de términos y definiciones precisos.
- Verificabilidad y trazabilidad
 - Debe de existir un método finito y sin costo para probar los requisitos.
 - Uso de términos y definiciones precisos.
- Modificabilidad y priorización
 - Fácilmente modificable.
 - Jerarquía de priorización según relevancia para el negocio (o fin).

- Correctitud y accesibilidad
 - El software debe cumplir con los requisitos de la especificación.
 - Accesibilidad y facilidad de comprensión para los usuarios y desarrolladores.

B.2. Objetivos generales

Este trabajo persigue los siguientes objetivos generales:

- Proporcionar una herramienta que otorgue capacidad crítica en inversiones personales.
- El sistema tiene que poder controlar la evolución de valores en cartera y en seguimiento.
- Tiene que mostrarse información relevante en cuanto a distribución de carteras y posibles pesos para los valores escogidos.
- Permitir trabajar, de manera experimental, con herramientas de *trading* algorítmico.
- Facilitar el acceso a la información con una página web pública y gratuita.

B.3. Catálogo de requisitos

Requisitos funcionales

- **RF-1 Mostrar portada con información general de diferentes índices bursátiles:** la web tiene que tener información agregada de todos los índices disponibles.
 - **RF-1.1 Mostrar carrusel de noticias generales:** en la portada se tiene que disponer de noticias relevantes dentro del mundo bursátil.
 - **RF-1.2 Mostrar mejores y peores valores de cada índice:** se tiene que poder consultar, de forma ágil, cuáles han sido los mejores y peores valores de cada índice.
 - **RF-1.2.1 Mostrar gráfica:** tiene que estar disponible una gráfica evolutiva junto a cada valor de los mostrados.

- **RF-2 Control de usuarios:** los usuarios tendrán acceso a información adicional si están dados de alta.
 - **RF-2.1 Permitir registro:** el usuario se podrá registrar.
 - **RF-2.2 Permitir hacer *login*:** el usuario podrá acceder con su cuenta en la web.
 - **RF-2.3 Permitir *logout*:** el usuario podrá cerrar su sesión.
- **RF-3 Gestionar *DashBoard*:** tiene que haber una zona de usuario, denominada *DashBoard*, con información relevante sobre valores seleccionados por el usuario.
 - **RF-3.1 Crear nuevo valor en cartera:** el usuario podrá añadir un nuevo valor, con fecha y precio de compra, a su cartera.
 - **RF-3.1.1 Mostrar 'donut', sectores y divisas:** si un usuario tiene valores en cartera, éste contará con diagramas de información relevante.
 - **RF-3.1.2 Ver gráfica de Markowitz, ratio de Sharpe y pesos:** si un usuario tiene valores en cartera podrá ver la mejor distribución de pesos para la misma.
 - **RF-3.1.3 Ver posiciones abiertas y evolución de cartera:** si hay valores en cartera se podrá comprobar la evolución de la misma.
 - **RF-3.2 Eliminar valor de cartera:** el usuario podrá eliminar un valor de los que tuviera almacenados.
 - **RF-3.3 Crear un nuevo valor en seguimiento:** el usuario podrá hacer seguimiento de valores de su interés.
 - **RF-3.4 Eliminar valor de seguimiento:** el usuario podrá eliminar un valor de los que tuviera en seguimiento.
- **RF-4 Consultar índice:** se debe presentar, de forma ordenada, información relevante de cada índice por separado.
 - **RF-4.1 Mostrar tabla de valores:** al consultar un índice se mostrará una tabla con todos los componentes de ese índice, con información relevante sobre la última sesión disponible.
 - **RF-4.2 Consultar un valor del índice:** se podrá consultar información de un único valor.
 - **RF-4.2.1 Dar acceso a gráfica interactiva:** se ofrecerá una gráfica interactiva, con medias móviles y volumen.

- **RF-4.2.2 Mostrar distribución de retornos:** habrá una gráfica con la distribución que hayan seguido los retornos en los últimos meses.
 - **RF-4.2.3 Mostrar datos del último mes:** disponer datos en formato tabular para ver precios de apertura y cierre de los últimos 30 días.
 - **RF-4.2.4 Ver evolución del sector:** facilitar una comparativa con el sector de referencia del valor.
 - **RF-4.2.5 Comparar con otros valores:** existirá la posibilidad de comparar gráficas de precios de cierre con otros valores.
 - **RF-4.2.6 Mostrar grafos de correlación:** mostrar grafos de alta correlación positiva y negativa con otros valores.
- **RF-4.3 Mostrar noticias relacionadas:** facilitar enlaces de fuentes RSS que estén relacionadas con el índice.
- **RF-4.4 Ver gráfica de evolución:** mostrar gráfica de evolución del índice en su conjunto.
- **RF-5 Gestionar *Lab*:** dar acceso a un laboratorio virtual.
 - **RF-5.1 Trabajar con ARIMA:** permitir realizar estimaciones y búsqueda de parámetros con ARIMA.
 - **RF-5.1.1 Buscar parámetros (p,d,q) con funciones ACF y PACF:** se mostrarán gráficas de funciones para interpretación del usuario.
 - **RF-5.1.2 Introducir (p,d,q) de forma manual:** el usuario podrá introducir los parámetros deseados para aplicar un modelo ARIMA.
 - **RF-5.1.3 Buscar (p,d,q) de forma automática:** se proporcionará una funcionalidad de búsqueda automática de los parámetros más adecuados a un valor.
 - **RF-5.1.4 Hacer búsqueda por rejilla para (p,d,q):** se podrá realizar una búsqueda de parámetros (p,d,q) de entre una serie de posibles valores preestablecidos.
 - **RF-5.2 Trabajar con *trading* algorítmico:** facilitar el acceso a herramientas de *trading* algorítmico.
 - **RF-5.2.1 Usar algoritmo de cruce de medias:** se mostrará el resultado de buscar las mejores medias móviles para un valor, en un período de tiempo concreto.

- **RF-5.2.2 Utilizar estrategia basada en *machine learning*:** permitir interactuar con modelos de regresión y clasificación para estimar la tendencia de la próxima sesión de un índice.

Requisitos no funcionales

En este apartado se tratará de dar detalle de aquellas características que no son funcionales pero que aportan un valor añadido al proyecto y a la interacción con la herramienta desarrollada:

- **RNF-1 Escalabilidad:** la web tiene que permitir y favorecer, en la medida de lo posible, la incorporación de nuevas funcionalidades y de bases de datos adicionales.
- **RNF-2 Privacidad:** los datos de usuario, como nombres o contraseñas, se debe gestionar de forma segura.
- **RNF-3 Disponibilidad:** la web debe ser compatible con los navegadores más modernos y tendrá alta disponibilidad a través de un servicio de alojamiento fiable.
- **RNF-4 Usabilidad:** la interfaz de la web será *user friendly*, resultar intuitiva y facilitará comentarios de ayuda adicional en los apartados más especializados.
- **RNF-5 Mantenibilidad:** se tiene que favorecer un mantenimiento posterior a la puesta en producción. Además, se facilitará información a los desarrolladores para que puedan realizar mejoras incrementales posteriores.

B.4. Especificación de requisitos

Casos de uso

A continuación se muestran todos los casos de uso contemplados. Muchos de estos casos de uso se plantearon en las fases iniciales del proyecto y se fueron mejorando en las sucesivas iteraciones de cada *sprint*. Se trató de realizar mejoras incrementales para satisfacer las expectativas de un inversor o potencial cliente:

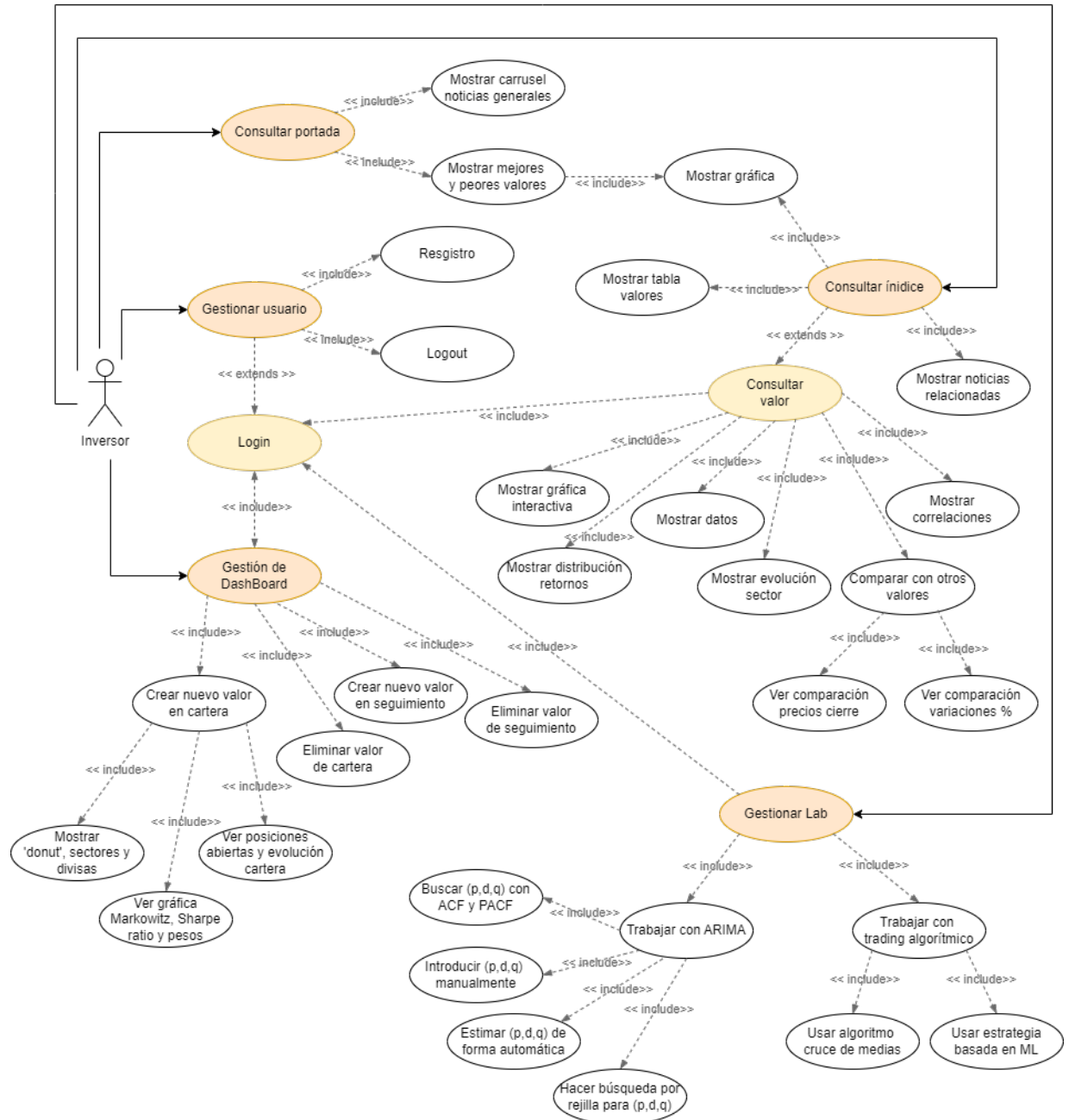


Figura B.1: Diagrama de casos de uso. Fuente: elaboración propia

CU-1	Consultar portada
Versión	1.0
Autor	Rodrigo Merino Tovar
Requisitos asociados	RF-1, RF-1.1, RF-1.2 y RF-1.2.1
Descripción	Permite al usuario ver la página principal
Precondición	La base de datos se encuentra disponible
Acciones	<ol style="list-style-type: none"> 1. El usuario accede a la página web. 2. Se muestran noticias del mundo bursátil. 3. Se listan los índices con los mejores y peores valores
Postcondición	El número de valores mostrados es múltiplo del número de índices disponibles.
Excepciones	<ul style="list-style-type: none"> ■ Error al cargar valores (mensaje de error). ■ Error al cargar una noticia (pasar a siguiente).
Importancia	Alta

Tabla B.1: CU-1 Consultar portada.

CU-2	Mostrar carrusel noticias generales
Versión	1.0
Autor	Rodrigo Merino Tovar
Requisitos asociados	RF-1, RF-1.1
Descripción	Mostrar carrusel de noticias bursátiles
Precondición	La API <i>NewsAPI</i> está activa
Acciones	<ol style="list-style-type: none"> 1. El usuario ve texto preliminar de una noticia y su imagen asociada. 2. Se <i>navega</i> entre noticias con selectores laterales o se dejan pasar automáticamente. 3. Se muestra un botón para <i>leer más</i> en la fuente original.
Postcondición	La imagen y el enlace externo se corresponden con el texto de la noticia.
Excepciones	<ul style="list-style-type: none"> ■ Error al cargar una noticia (pasar a siguiente).
Importancia	Media

Tabla B.2: CU-2 Mostrar carrusel noticias generales.

CU-3	Mostrar mejores y peores valores
Versión	1.0
Autor	Rodrigo Merino Tovar
Requisitos asociados	RF-1, RF-1.2 y RF-1.2.1
Descripción	Mostrar 3 mejores y 3 peores valores de cada índice en la última sesión bursátil.
Precondición	Las bases de datos están disponibles.
Acciones	<ol style="list-style-type: none"> 1. El usuario ve valores de cierre de última sesión de mejores valores de un índice. 2. El usuario ve valores de cierre de última sesión de peores valores de un índice. 3. Se muestra gráfica asociada a cada valor.
Postcondición	Las gráficas se corresponden con los valores mejores y peores y se muestran en orden de mejor a peor
Excepciones	<ul style="list-style-type: none"> ■ Error al cargar un valor (mensaje de error).
Importancia	Media

Tabla B.3: CU-3 Mostrar mejores y peores valores.

CU-4	Mostrar gráfica
Versión	1.0
Autor	Rodrigo Merino Tovar
Requisitos asociados	RF-1, RF-1.2, RF-1.2.1 y RF-4.4
Descripción	Mostrar gráficas de valores o índices.
Precondición	Las bases de datos están disponibles.
Acciones	<ol style="list-style-type: none"> 1. El usuario ve gráficas de última sesión de mejores valores de un índice. 2. El usuario ve gráficas de última sesión de peores valores de un índice. 3. Se muestran valores de cierre asociados a cada valor. 4. El usuario consulta gráfica de índice.
Postcondición	Las gráficas se corresponden con los valores mejores y peores y se muestran en orden de mejor a peor. O la gráfica del índice consultado está disponible.
Excepciones	<ul style="list-style-type: none"> ■ Error al cargar un valor o índice (mensaje de error).
Importancia	Media

Tabla B.4: CU-4 Mostrar gráfica.

CU-5	Gestionar usuario
Versión	1.0
Autor	Rodrigo Merino Tovar
Requisitos asociados	RF-2, RF-2.1, RF-2.2 y RF-2.3
Descripción	Controlar sesión de usuario.
Precondición	La base de datos de usuarios está disponible.
Acciones	<ol style="list-style-type: none">1. El usuario se registra.2. El usuario hace <i>login</i>.3. Diferentes acciones dependiendo de la labor a realizar.4. El usuario hace <i>logout</i> para cerrar sesión.
Postcondición	Tras <i>logout</i> la cookie de sesión se inhabilita.
Excepciones	<ul style="list-style-type: none">■ Error al registrar por contraseña o nombre (mensaje informativo).■ Error en <i>login</i> por contraseña o nombre (mensaje informativo).
Importancia	Alta

Tabla B.5: CU-5 Gestionar usuario.

CU-6	Registro
Versión	1.0
Autor	Rodrigo Merino Tovar
Requisitos asociados	RF-2 y RF-2.1
Descripción	Permitir registro de usuarios.
Precondición	El usuario no está registrado.
Acciones	<ol style="list-style-type: none"> 1. El usuario se registra. 2. Mostrar un mensaje de bienvenida. 3. Dejar usuario ya activo con <i>login</i>. 4. Redirigir a página principal.
Postcondición	Nuevo usuario en base de datos.
Excepciones	<ul style="list-style-type: none"> ■ Error al registrar por contraseña o nombre (mensaje informativo). ■ Error por usuario duplicado (mensaje informativo).
Importancia	Alta

Tabla B.6: CU-6 Registro.

CU-7	Login
Versión	1.0
Autor	Rodrigo Merino Tovar
Requisitos asociados	RF-2 y RF-2.2
Descripción	Permitir <i>login</i> de usuarios.
Precondición	El usuario no está logueado.
Acciones	<ol style="list-style-type: none">1. El usuario hace <i>login</i>.2. Mostrar <i>DashBoard</i>.3. Permitir acceso a funciones adicionales.
Postcondición	Usuario logueado y con acceso a funciones adicionales.
Excepciones	<ul style="list-style-type: none">■ Error al loguear (mensaje informativo y repetición).■ Error forzado (mensaje informativo).
Importancia	Alta

Tabla B.7: CU-6 Login.

CU-8	Logout
Versión	1.0
Autor	Rodrigo Merino Tovar
Requisitos asociados	RF-2 y RF-2.3
Descripción	Permitir <i>logout</i> de usuarios.
Precondición	El usuario está logueado.
Acciones	<ol style="list-style-type: none">1. El usuario hace <i>logout</i>.2. Mostrar página principal.3. No permitir acceso a funciones adicionales.
Postcondición	Usuario no logueado y sin acceso a funciones adicionales.
Excepciones	
Importancia	Baja

Tabla B.8: CU-8 Logout.

CU-9	Gestionar <i>DashBoard</i>
Versión	1.0
Autor	Rodrigo Merino Tovar
Requisitos asociados	RF-3, RF-3.1, RF-3.1.1, RF-3.1.2, RF-3.1.3, RF-3.2, RF-3.3 y RF-3.4
Descripción	Mostrar información agregada de valores en cartera y en seguimiento de un usuario.
Precondición	El usuario está logueado y las bases de datos están disponibles.
Acciones	<ol style="list-style-type: none">1. El usuario hace <i>login</i>.2. Mostrar página de <i>DashBoard</i> con información de usuario.
Postcondición	El usuario puede consultar precios, añadir o quitar valores en cartera y añadir o quitar valores en seguimiento.
Excepciones	<ul style="list-style-type: none">■ No hay valores previos en cartera (se muestran tablas vacías).■ No hay valores previos en seguimiento (se muestran tablas vacías).
Importancia	Alta

Tabla B.9: CU-9 *DashBoard*.

CU-10	Crear nuevo valor en cartera
Versión	1.0
Autor	Rodrigo Merino Tovar
Requisitos asociados	RF-3, RF-3.1, RF-3.1.1, RF-3.1.2 y RF-3.1.3
Descripción	Añadir un valor a la cartera del usuario.
Precondición	El usuario está logueado y las bases de datos están disponibles.
Acciones	<ol style="list-style-type: none"> 1. El usuario accede al <i>DashBoard</i>. 2. Pinchar en botón <i>Nueva posición de cartera</i>.
Postcondición	El usuario ve un nuevo valor asociado a su cuenta.
Excepciones	<ul style="list-style-type: none"> ■ Valor no existe (mensaje informativo y reintento). ■ Fecha no válida (mensaje informativo y reintento). ■ Precio valor no adecuado para la fecha (mensaje informativo y reintento). ■ Valores formulario fuera de rango (mensaje informativo y reintento).
Importancia	Alta

Tabla B.10: CU-10 Crear nuevo valor en cartera.

CU-11	Mostrar <i>donut</i> , sectores y divisas
Versión	1.0
Autor	Rodrigo Merino Tovar
Requisitos asociados	RF-3, RF-3.1 y RF-3.1.1
Descripción	Mostrar información de composición de cartera en gráficas y diagramas que resulten agradables para el usuario.
Precondición	El usuario está logueado, las bases de datos están disponibles y el usuario tiene valores guardados en cartera.
Acciones	<ol style="list-style-type: none"> 1. El usuario accede al <i>DashBoard</i>. 2. Mostrar <i>donut</i> con valores en cartera. 3. Mostrar diagrama de sectores en los que se está invertido. 4. Mostrar inversión por tipo de divisa. 5. Ver cambios de divisa de última sesión disponible.
Postcondición	El usuario ve información relevante sobre el estado de su cartera.
Excepciones	<ul style="list-style-type: none"> ■ No hay valores previos en cartera (se muestran tablas vacías).
Importancia	Media

Tabla B.11: CU-11 Mostrar *donut*, sectores y divisas.

CU-12	Ver gráfica de Markowitz, ratio de Sharpe y pesos
Versión	1.0
Autor	Rodrigo Merino Tovar
Requisitos asociados	RF-3, RF-3.1 y RF-3.1.2
Descripción	Mostrar información de distribución de pesos de los valores en cartera.
Precondición	El usuario está logueado, las bases de datos están disponibles y el usuario tiene valores guardados en cartera.
Acciones	<ol style="list-style-type: none"> 1. El usuario accede al <i>DashBoard</i>. 2. Mostrar gráfica de Markowitz. 3. Ver simulación de Montecarlo de 10.000 carteras. 4. Indicar frontera eficiente. 5. Mostrar ratio de Sharpe. 6. Mostrar información de distribución de pesos en tablas.
Postcondición	El usuario ve información relevante sobre posibles distribuciones mejores para su cartera.
Excepciones	<ul style="list-style-type: none"> ■ No hay valores previos en cartera (se muestran tablas vacías).
Importancia	Alta

Tabla B.12: CU-12 Ver gráfica de Markowitz, ratio de Sharpe y pesos.

CU-13	Ver posiciones abiertas y evolución de cartera
Versión	1.0
Autor	Rodrigo Merino Tovar
Requisitos asociados	RF-3, RF-3.1 y RF-3.1.3
Descripción	Mostrar información de evolución de inversiones realizadas.
Precondición	El usuario está logueado, las bases de datos están disponibles y el usuario tiene valores guardados en cartera.
Acciones	<ol style="list-style-type: none"> 1. El usuario accede al <i>DashBoard</i>. 2. Mostrar evolución individual de valores en cartera. 3. Hacer cálculos de cambio de divisa a euros. 4. Mostrar evolución de cartera en euros.
Postcondición	El usuario ve la evolución de su cartera.
Excepciones	<ul style="list-style-type: none"> ■ No hay valores previos en cartera (se muestran tablas vacías).
Importancia	Alta

Tabla B.13: CU-13 Ver posiciones abiertas y evolución de cartera.

CU-14	Eliminar valor de cartera
Versión	1.0
Autor	Rodrigo Merino Tovar
Requisitos asociados	RF-3 y RF-3.2
Descripción	Eliminar valor guardado en cartera.
Precondición	El usuario está logueado, las bases de datos están disponibles y el usuario tiene valores guardados en cartera.
Acciones	<ol style="list-style-type: none"> 1. El usuario accede al <i>DashBoard</i>. 2. Mostrar información relativa a cartera. 3. El usuario pulsa el botón <i>Eliminar posición de cartera</i>. 4. Mostrar lista de valores en cartera. 5. Seleccionar valores a eliminar. 6. Pulsar <i>Eliminar seleccionados</i>. 7. Volver al <i>DashBoard</i>
Postcondición	El usuario ve datos actualizados de su cartera.
Excepciones	<ul style="list-style-type: none"> ■ No hay valores posteriores en cartera (se muestran tablas vacías).
Importancia	Alta

Tabla B.14: CU-14 Eliminar valor de cartera.

CU-15	Crear un nuevo valor en seguimiento
Versión	1.0
Autor	Rodrigo Merino Tovar
Requisitos asociados	RF-3 y RF-3.3
Descripción	Guardar un nuevo valor para realizar seguimiento.
Precondición	El usuario está logueado, las bases de datos están disponibles y el usuario no tiene el valor previamente en seguimiento.
Acciones	<ol style="list-style-type: none"> 1. El usuario accede al <i>DashBoard</i>. 2. Mostrar información relativa a cartera. 3. El usuario pulsa el botón <i>Nuevo valor a seguir</i>. 4. Mostrar formulario de búsqueda de valores (<i>tickers</i>). 5. Seleccionar valor a seguir. 6. Pulsar <i>Guardar</i>. 7. Buscar valores del mismo sector. 8. Volver al <i>DashBoard</i>
Postcondición	El usuario ve nuevo <i>stock</i> en seguimiento junto con aquellos valores que pertenezcan al mismo sector.
Excepciones	<ul style="list-style-type: none"> ■ No hay valores posteriores en seguimiento (se muestran tablas vacías).
Importancia	Media

Tabla B.15: CU-15 Crear un nuevo valor en seguimiento.

CU-16	Eliminar un valor de seguimiento
Versión	1.0
Autor	Rodrigo Merino Tovar
Requisitos asociados	RF-3 y RF-3.4
Descripción	Para eliminar un valor que estuviera en seguimiento.
Precondición	El usuario está logueado, las bases de datos están disponibles y el usuario tiene previamente valores en seguimiento.
Acciones	<ol style="list-style-type: none"> 1. El usuario accede al <i>DashBoard</i>. 2. Mostrar información relativa a cartera. 3. El usuario pulsa el botón <i>Eliminar valor en seguimiento</i>. 4. Mostrar valores en seguimiento). 5. Seleccionar valores a eliminar. 6. Pulsar <i>Eliminar seleccionados</i>. 7. Volver al <i>DashBoard</i>
Postcondición	El usuario ve valores en seguimiento actualizados.
Excepciones	<ul style="list-style-type: none"> ■ No hay valores previos en seguimiento (se muestran tablas vacías). ■ No hay valores posteriores en seguimiento (se muestran tablas vacías).
Importancia	Media

Tabla B.16: CU-16 Eliminar un valor de seguimiento.

CU-17	Consultar índice
Versión	1.0
Autor	Rodrigo Merino Tovar
Requisitos asociados	RF-4, RF-4.1, RF-4.2, RF-4.3, RF-4.4 y RF-1.2.1
Descripción	Para mostrar datos de valores de un índice de forma agregada.
Precondición	Las bases de datos están disponibles.
Acciones	<ol style="list-style-type: none">1. El usuario selecciona el índice que quiere consultar.2. Mostrar tabla con componentes del índice.3. Visualizar gráfica de evolución del índice en su conjunto.4. Facilitar enlaces RSS a noticias relacionadas.
Postcondición	El usuario ve información del índice.
Excepciones	<ul style="list-style-type: none">■ No hay datos de índice (mensaje de página inexistente).
Importancia	Alta

Tabla B.17: CU-17 Consultar índice.

CU-18	Mostrar tabla de valores
Versión	1.0
Autor	Rodrigo Merino Tovar
Requisitos asociados	RF-4 y RF-4.1
Descripción	Para mostrar tabla de componentes de un índice.
Precondición	Las bases de datos están disponibles.
Acciones	<ol style="list-style-type: none"> 1. El usuario selecciona el índice que quiere consultar. 2. Mostrar tabla con componentes del índice. 3. El usuario ordena según variación diaria.
Postcondición	El usuario consulta precios de todos los valores del índice.
Excepciones	<ul style="list-style-type: none"> ■ No hay datos de índice (mensaje de página inexistente).
Importancia	Alta

Tabla B.18: CU-18 Mostrar tabla de valores.

CU-19	Consultar un valor del índice
Versión	1.0
Autor	Rodrigo Merino Tovar
Requisitos asociados	RF-4, RF-4.1, RF-4.2, RF-4.2.1, RF-4.2.2 y RF-4.2.3, RF-4.2.4, RF-4.2.5 y RF-4.2.6
Descripción	Para consultar información detallada de un único valor cotizado.
Precondición	El usuario está logueado, las bases de datos están disponibles y el valor existe.
Acciones	<ol style="list-style-type: none"> 1. El usuario selecciona el índice del valor de su interés. 2. El usuario selecciona el valor que quiere consultar. 3. Dar acceso a gráfica interactiva. 4. Mostrar distribución de retornos. 5. Mostrar datos de último mes. 6. Ver evolución del sector. 7. Mostrar formulario de comparación con otros valores. 8. Mostrar grafos de correlación.
Postcondición	El usuario ve toda la información ordenada en forma de tablas o gráficas; es posible interactuar (de forma básica) con la gráfica interactiva.
Excepciones	<ul style="list-style-type: none"> ■ No hay datos del valor (mensaje de página inexistente).
Importancia	Alta

Tabla B.19: CU-19 Consultar un valor del índice.

CU-20	Mostrar gráfica interactiva
Versión	1.0
Autor	Rodrigo Merino Tovar
Requisitos asociados	RF-4, RF-4.1, RF-4.2 y RF-4.2.1
Descripción	Para interactuar con temporalidades e indicadores de un valor cotizado.
Precondición	El usuario está logueado, las bases de datos están disponibles y el valor existe.
Acciones	<ol style="list-style-type: none"> 1. El usuario selecciona el valor que quiere consultar. 2. Dar acceso a gráfica interactiva. 3. Se varían temporalidades ajustando volumen y medias móviles. 4. Se comprueban datos diarios.
Postcondición	El usuario interactúa con la gráfica del valor.
Excepciones	<ul style="list-style-type: none"> ■ No hay datos del valor (mensaje de página inexistente).
Importancia	Alta

Tabla B.20: CU-20 Mostrar gráfica interactiva.

CU-21	Mostrar distribución de retornos
Versión	1.0
Autor	Rodrigo Merino Tovar
Requisitos asociados	RF-4, RF-4.1, RF-4.2 y RF-4.2.2
Descripción	Para ver la distribución que siguen los retornos de un valor en el último año (252 sesiones).
Precondición	El usuario está logueado, las bases de datos están disponibles, el valor existe y tiene datos históricos suficientes.
Acciones	<ol style="list-style-type: none"> 1. El usuario selecciona el valor que quiere consultar. 2. Dar acceso a gráfica de distribución de retornos.
Postcondición	El usuario consulta la distribución de los retornos de un valor.
Excepciones	<ul style="list-style-type: none"> ■ No hay datos del valor (mensaje de página inexistente).
Importancia	Baja

Tabla B.21: CU-21 Mostrar distribución de retornos.

CU-22	Mostrar datos del último mes
Versión	1.0
Autor	Rodrigo Merino Tovar
Requisitos asociados	RF-4, RF-4.1, RF-4.2 y RF-4.2.3
Descripción	Para ver tabla con precios de un valor en el último mes.
Precondición	El usuario está logueado, las bases de datos están disponibles, el valor existe y tiene datos históricos suficientes.
Acciones	<ol style="list-style-type: none"> 1. El usuario selecciona el valor que quiere consultar. 2. Dar acceso a tabla con datos de último mes. 3. Mostrar tabla con colores verde (alcista) y rojo (bajista).
Postcondición	El usuario consulta la tabla de precios del último mes.
Excepciones	<ul style="list-style-type: none"> ■ No hay datos del valor (mensaje de página inexistente).
Importancia	Media

Tabla B.22: CU-22 Mostrar datos del último mes.

CU-23	Mostrar evolución del sector
Versión	1.0
Autor	Rodrigo Merino Tovar
Requisitos asociados	RF-4, RF-4.1, RF-4.2 y RF-4.2.4
Descripción	Para ver comparación - en términos relativos - entre el valor y su sector de referencia.
Precondición	El usuario está logueado, las bases de datos están disponibles, el valor existe y tiene datos históricos suficientes.
Acciones	<ol style="list-style-type: none"> 1. El usuario selecciona el valor que quiere consultar. 2. Obtener el sector de referencia del valor. 3. Obtener listado de valores del mismo sector. 4. Calcular media de evolución de todos los valores del mismo sector. 5. Ajustar datos de forma relativa (porcentual). 6. Mostrar gráfica con evolución de valor y evolución de sector.
Postcondición	El usuario consulta la gráfica comparativa de valor con sector de referencia.
Excepciones	<ul style="list-style-type: none"> ■ No hay datos del valor (mensaje de página inexistente). ■ No hay datos de valores del sector (gráfica sin comparación).
Importancia	Alta

Tabla B.23: CU-23 Mostrar evolución del sector.

CU-24	Comparar con otros valores
Versión	1.0
Autor	Rodrigo Merino Tovar
Requisitos asociados	RF-4, RF-4.1, RF-4.2 y RF-4.2.5
Descripción	Para ver comparación - en términos relativos - con otros valores seleccionados por el usuario.
Precondición	El usuario está logueado, las bases de datos están disponibles, el valor existe y tiene datos históricos suficientes.
Acciones	<ol style="list-style-type: none"> 1. El usuario selecciona el valor que quiere consultar. 2. El usuario selecciona en un formulario otro valor con el que comparar. 3. Mostrar comparación relativa de precios de cierre. 4. Mostrar comparación relativa de rentabilidades diarias.
Postcondición	Recargar página con nueva gráfica comparativa.
Excepciones	<ul style="list-style-type: none"> ■ No hay datos del valor (mensaje de página inexistente). ■ No hay datos del valor con el que comparar (mensaje de <i>ticker</i> no válido).
Importancia	Media

Tabla B.24: CU-24 Comparar con otros valores.

CU-25	Mostrar grafos de correlación
Versión	1.0
Autor	Rodrigo Merino Tovar
Requisitos asociados	RF-4, RF-4.1, RF-4.2 y RF-4.2.5
Descripción	Para ver grafos de correlación positiva y negativa con el resto de valores disponibles.
Precondición	El usuario está logueado, las bases de datos están disponibles, el valor existe y tiene datos históricos suficientes.
Acciones	<ol style="list-style-type: none"> 1. El usuario selecciona el valor que quiere consultar. 2. Crear matriz de correlación entre todos los valores disponibles. 3. Crear grafo de correlación positiva (correl. mayor a 0,75) 4. Crear grafo de correlación negativa (correl. mayor a -0,75) 5. Mostrar grafos de correlaciones.
Postcondición	El usuario consulta los grafos de correlaciones.
Excepciones	<ul style="list-style-type: none"> ■ No hay datos del valor (mensaje de página inexistente). ■ No hay correlaciones que superen los umbrales (mostrar grafo con nodo de valor sin más datos).
Importancia	Alta

Tabla B.25: CU-25 Mostrar grafos de correlación.

CU-26	Mostrar noticias relacionadas
Versión	1.0
Autor	Rodrigo Merino Tovar
Requisitos asociados	RF-4 y RF-4.3
Descripción	Para facilitar enlaces de fuentes RSS que estén relacionadas con el índice.
Precondición	Las bases de datos están disponibles y las fuentes RSS están activas.
Acciones	<ol style="list-style-type: none"> 1. El usuario selecciona el índice que quiere consultar. 2. Mostrar listado de noticias relacionadas de fuentes RSS. 3. Pulsar el botón <i>leer más</i> de una fuente. 4. Redirigir a web externa.
Postcondición	El usuario ve listado de noticias relacionadas relevantes y puede seleccionar <i>leer más</i> en fuente externa.
Excepciones	<ul style="list-style-type: none"> ■ No hay datos de fuentes RSS (mensaje de error).
Importancia	Baja

Tabla B.26: CU-26 Mostrar noticias relacionadas.

CU-27	Gestionar <i>Lab</i>
Versión	1.0
Autor	Rodrigo Merino Tovar
Requisitos asociados	RF-5, RF-5.1, RF-5.1.1, RF-5.1.2, RF-5.1.3, RF-5.1.4, RF-5.2, RF-5.2.1 y RF-5.2.2
Descripción	Para controlar una sección de <i>laboratorio</i> virtual con el que experimentar sobre los datos disponibles.
Precondición	El usuario está logueado y las bases de datos están disponibles.
Acciones	<ol style="list-style-type: none"> 1. El usuario pulsa en en el apartado de <i>Lab</i>. 2. Se muestran opciones de modelado ARIMA y de <i>trading</i> algorítmico. 3. Para los usuarios del repositorio en local se muestra una opción de <i>forecasting</i> con redes LSTM. 4. Al pulsar cualquier opción disponible se facilitan los accesos a las funcionalidades concretas. 5. Al lado de las funcionalidades disponibles se muestra información de ayuda al usuario.
Postcondición	El usuario puede consultar las herramientas disponibles dentro del <i>Lab</i> .
Excepciones	
Importancia	Alta

Tabla B.27: CU-27 Gestionar *Lab*.

CU-28	Trabajar con ARIMA
Versión	1.0
Autor	Rodrigo Merino Tovar
Requisitos asociados	RF-5, RF-5.1, RF-5.1.1, RF-5.1.2, RF-5.1.3 y RF-5.1.4
Descripción	Para tratar de hacer <i>forecasting</i> de series temporales con modelos ARIMA .
Precondición	El usuario está logueado y las bases de datos están disponibles.
Acciones	<ol style="list-style-type: none"> 1. El usuario entra en el <i>Lab</i>. 2. Se pulsa el botón <i>ARIMA</i>. 3. Se muestran las opciones para trabajar con modelos ARIMA. 4. Se despliegan los botones de las herramientas disponibles: funciones ACF y PACF, modelado manual, modelado automático y modelado con búsqueda de parámetros por rejilla. 5. Al lado de cada opción se muestra una breve ayuda. 6. Pulsar de la opción deseada. 7. Redirigir a la web con el formulario correspondiente.
Postcondición	El usuario es dirigido a la plantilla con el formulario adecuado.
Excepciones	<ul style="list-style-type: none"> ■ La redirección no es adecuada (mensaje de error inesperado).
Importancia	Alta

Tabla B.28: CU-28 Trabajar con ARIMA.

CU-29	Buscar (p,d,q) con funciones ACF y PACF:
Versión	1.0
Autor	Rodrigo Merino Tovar
Requisitos asociados	RF-5, RF-5.1 y RF-5.1.1
Descripción	Para buscar los parámetros (p,d,q) de un modelo ARIMA de forma visual, a través del uso de funciones ACF y PACF.
Precondición	El usuario está logueado, las bases de datos están disponibles, el valor buscado existe.
Acciones	<ol style="list-style-type: none"> 1. El usuario accede a la sección de ARIMA dentro del <i>Lab</i>. 2. Se pulsa el botón <i>Buscar parámetros (p,d,q) con funciones ACF y PACF</i>. 3. Se accede al formulario de interacción adecuado. 4. El usuario introduce el valor buscado y la cantidad de sesiones previas que desea utilizar para el modelo. 5. El usuario pulsa el botón <i>Mostrar resultados</i>. 6. Se muestran las diferenciaciones más habituales: d=1, d=2 y d=3. 7. Se muestran las gráficas de ACF y PACF junto con precios de cierre diferenciados.
Postcondición	El usuario puede consultar la información y sacar conclusiones de los mejores parámetros para la serie temporal escogida.
Excepciones	<ul style="list-style-type: none"> ■ Se indica un valor que no existe (mensaje de error y reintento). ■ Se indica una cantidad inadecuada de sesiones (mensaje de error y reintento).
Importancia	Alta

Tabla B.29: CU-29 Buscar (p,d,q) con funciones ACF y PACF.

CU-30	Introducir (p,d,q) manualmente:
Versión	1.0
Autor	Rodrigo Merino Tovar
Requisitos asociados	RF-5, RF-5.1 y RF-5.1.2
Descripción	Para realizar <i>forecasting</i> con un modelo ARIMA que tenga los parámetros (p,d,q) deseados por el usuario.
Precondición	El usuario está logueado, las bases de datos están disponibles, el valor buscado existe y el usuario conoce los parámetros que va a utilizar.
Acciones	<ol style="list-style-type: none"> 1. El usuario accede a la sección del <i>Lab</i>. 2. Se pulsa el botón <i>Parámetros (p,d,q) introducidos por usuario</i>. 3. Se accede al formulario de interacción adecuado. 4. El usuario introduce valor buscado y cantidad de sesiones que desea utilizar con el modelo. 5. El usuario introduce la cantidad de datos que quiere utilizar para entrenar el modelo. 6. El usuario introduce los parámetros (p,d,q). 7. El usuario pulsa el botón <i>Estimar</i>. 8. Calcular intervalos de confianza y validación <i>WF</i>. 9. Calcular aciertos de tendencia al usar validación <i>WF</i>. 10. Se hace predicción de ARIMA sobre el porcentaje de datos de entrenamiento. 11. Se muestran métricas de error: MSE y RMSE. 12. Se muestra comparación con una predicción naïf. 13. Se muestra gráfica con validación <i>walk forward</i>. 14. Mostrar resultados sobre datos de test con validación <i>WF</i>.
Postcondición	El usuario puede consultar la información.
Excepciones	<ul style="list-style-type: none"> ■ Se indica un valor que no existe (mensaje de error y reintento). ■ Se indica una cantidad inadecuada de sesiones (mensaje de error y reintento). ■ Se indica un porcentaje de datos de entrenamiento no válido (mensaje de error y reintento). ■ Se indican parámetros (p,d,q) no válidos (mensaje de error y reintento).
Importancia	Alta

Tabla B.30: CU-30 Introducir (p,d,q) manualmente.

CU-31	Estimar (p,d,q) de forma automática:
Versión	1.0
Autor	Rodrigo Merino Tovar
Requisitos asociados	RF-5, RF-5.1 y RF-5.1.3
Descripción	Para realizar <i>forecasting</i> con un modelo ARIMA que tenga los parámetros (p,d,q) calculados de forma automática.
Precondición	El usuario está logueado, las bases de datos están disponibles y el valor buscado existe.
Acciones	<ol style="list-style-type: none"> 1. El usuario accede a la sección de ARIMA dentro del <i>Lab</i>. 2. Se pulsa el botón <i>Parámetros (p,d,q) estimados automáticamente</i>. 3. Se accede al formulario de interacción adecuado. 4. El usuario introduce el valor buscado y la cantidad de sesiones previas que desea utilizar para el modelo. 5. El usuario introduce la cantidad de datos que quiere utilizar para entrenar el modelo. 6. El usuario pulsa el botón <i>Estimar</i>. 7. Se utiliza <i>auto_arima</i> para estimar parámetros. 8. Calcular intervalos de confianza y validación <i>WF</i>. 9. Calcular aciertos de tendencia al usar <i>walk forward</i>. 10. Se hace predicción de ARIMA sobre el porcentaje de datos de entrenamiento. 11. Se muestran métricas de error: MSE y RMSE. 12. Se muestra comparación con una predicción naïf. 13. Se muestra gráfica con validación <i>walk forward</i>. 14. Mostrar resultados sobre datos de test con <i>WF</i>. 15. Se muestra informe ARIMA/SARIMAX.
Postcondición	El usuario puede consultar la información.
Excepciones	<ul style="list-style-type: none"> ■ Se indica un valor que no existe (mensaje de error y reintento). ■ Se indica una cantidad inadecuada de sesiones (mensaje de error y reintento). ■ Se indica un porcentaje de datos de entrenamiento no válido (mensaje de error y reintento).
Importancia	Alta

Tabla B.31: CU-31 Estimar (p,d,q) de forma automática.

CU-32	Hacer búsqueda por rejilla para (p,d,q):
Versión	1.0
Autor	Rodrigo Merino Tovar
Requisitos asociados	RF-5, RF-5.1 y RF-5.1.4
Descripción	Para realizar <i>forecasting</i> con un modelo ARIMA que tenga los parámetros (p,d,q) obtenidos tras una búsqueda por rejilla de entre unos valores preestablecidos.
Precondición	El usuario está logueado, las bases de datos están disponibles, el valor buscado existe, los posibles valores de los parámetros están preestablecidos.
Acciones	<ol style="list-style-type: none"> 1. El usuario accede a la sección de ARIMA. 2. Se pulsa el botón <i>Parámetros (p,d,q) creados con búsqueda por rejilla</i>. 3. Se accede al formulario de interacción adecuado. 4. El usuario introduce el valor buscado y la cantidad de sesiones previas que desea utilizar. 5. El usuario selecciona posibles valores (p,d,q). 6. El usuario pulsa el botón <i>Estimar</i>. 7. Se evalúan todas las posibles combinaciones de parámetros y se selecciona el modelo con menor MSE (es decir, el mejor modelo posible). 8. Calcular intervalos de confianza y validación <i>walk forward</i>. 9. Se calculan aciertos de tendencia al usar <i>WF</i>. 10. Se hace predicción de ARIMA sobre el porcentaje de datos de entrenamiento. 11. Se muestran métricas de error: MSE y RMSE. 12. Se muestra comparación con una predicción naïf. 13. Se muestra gráfica con validación <i>walk forward</i>. 14. Se muestran resultados sobre datos de test con <i>walk forward</i> e informe ARIMA/SARIMAX.
Postcondición	El usuario puede consultar la información.
Excepciones	<ul style="list-style-type: none"> ■ Se indica un valor que no existe (mensaje de error y reintento). ■ Se indica una cantidad inadecuada de sesiones (mensaje de error y reintento). ■ Se indica un porcentaje de datos de entrenamiento no válido (mensaje de error y reintento). ■ Se indican posibles valores de (p,d,q) no válidos (mensaje de error y reintento).
Importancia	Media

Tabla B.32: CU-32 Hacer búsqueda por rejilla para (p,d,q).

CU-33	Trabajar con <i>trading</i> algorítmico:
Versión	1.0
Autor	Rodrigo Merino Tovar
Requisitos asociados	RF-5, RF-5.2, RF-5.2.1 y RF-5.2.2
Descripción	Para poder utilizar técnicas de <i>trading</i> algorítmico aplicadas a los valores disponibles.
Precondición	El usuario está logueado, las bases de datos están disponibles y el valor buscado existe.
Acciones	<ol style="list-style-type: none"> 1. El usuario entra en el <i>Lab</i>. 2. Se pulsa el botón <i>Trading algorítmico</i>. 3. Se muestran las opciones para trabajar con técnicas de <i>trading</i> algorítmico. 4. Se despliegan los botones de las herramientas disponibles: algoritmo de cruce de medias y estrategia basada en <i>machine learning</i>. 5. Al lado de cada opción se muestra una breve ayuda. 6. Pulsar de la opción deseada. 7. Redirigir a la web con el formulario correspondiente.
Postcondición	El usuario puede consultar la información y sacar conclusiones sobre los resultados tanto en gráficas como en resumen descriptivo del modelo.
Excepciones	<ul style="list-style-type: none"> ■ La redirección no es adecuada (mensaje de error inesperado).
Importancia	Alta

Tabla B.33: CU-33 Trabajar con *trading* algorítmico.

CU-34	Usar algoritmo de cruce de medias:
Versión	1.0
Autor	Rodrigo Merino Tovar
Requisitos asociados	RF-5, RF-5.2 y RF-5.2.1
Descripción	Para utilizar el algoritmo de cruce de medias sobre un valor seleccionado.
Precondición	El usuario está logueado, las bases de datos están disponibles y el valor buscado existe.
Acciones	<ol style="list-style-type: none"> 1. El usuario accede a la sección de <i>trading</i> algorítmico dentro del <i>Lab</i>. 2. Se pulsa el botón <i>Algoritmo de cruce de medias</i>. 3. Se accede al formulario de interacción adecuado. 4. El usuario introduce el valor buscado y la cantidad de sesiones previas que desea utilizar para el algoritmo. 5. Se hace una búsqueda por rejilla calculando las mejores rentabilidades con distintos cruces de medias preestablecidos ([30, 50, 200] y [10, 15, 20, 50]). 6. Se muestra el mejor resultado posible. 7. Se hace una comparación con una estrategia <i>buy and hold</i> en el mismo número de sesiones seleccionado.
Postcondición	El usuario puede ver un resultado porcentual de haber seguido la estrategia de cruce de medias en el período seleccionado y puede comparar con una estrategia <i>buy and hold</i> para sacar conclusiones.
Excepciones	<ul style="list-style-type: none"> ■ Se indica un valor que no existe (mensaje de error y reintento). ■ Se indica una cantidad inadecuada de sesiones (mensaje de error y reintento).
Importancia	Alta

Tabla B.34: CU-34 Usar algoritmo de cruce de medias.

CU-35	Usar estrategia basada en <i>machine learning</i> :
Versión	1.0
Autor	Rodrigo Merino Tovar
Requisitos asociados	RF-5, RF-5.2 y RF-5.2.2
Descripción	Para hacer una predicción para la siguiente sesión de un índice con modelos de regresión y de clasificación.
Precondición	El usuario está logueado, las bases de datos están disponibles y el índice sobre el que predecir existe.
Acciones	<ol style="list-style-type: none"> 1. El usuario accede a la sección de <i>trading</i> algorítmico dentro del <i>Lab</i>. 2. Se pulsa el botón <i>Estrategia basada en ML</i>. 3. Se accede al formulario de interacción adecuado. 4. El usuario introduce el valor buscado y la cantidad de sesiones previas que desea utilizar para el algoritmo. 5. El usuario indica el tipo de modelo. 6. El usuario indica el porcentaje de datos para entrenar el modelo. 7. Se seleccionan los datos de los valores que representan al índice para el mismo período (mismo número de sesiones). 8. Se entrena el modelo y se calculan los <i>scores</i> en entrenamiento y en test. 9. Se hace una predicción adicional, fuera de los datos de test. 10. Se muestra al usuario información tabular con resultados obtenidos y con predicción para la siguiente sesión.
Postcondición	El usuario puede consultar una tabla de resultados tras entrenar el modelo y la predicción.
Excepciones	<ul style="list-style-type: none"> ■ Se indica una cantidad inadecuada de sesiones (mensaje de error y reintento). ■ Se indica un índice que no existe (mensaje de error y reintento). ■ Se indica un tipo de modelo no disponible (mensaje de error y reintento). ■ Se indica un porcentaje de datos de entrenamiento no válido (mensaje de error y reintento).
Importancia	Alta

Tabla B.35: CU-35 Usar estrategia basada en *machine learning*.

Apéndice C

Especificación de diseño

C.1. Introducción

En este anexo se describe la resolución de los requisitos y casos de uso previos. Además, se muestra cómo se almacenan y manejan los datos, detalles procedimentales y la estructura interna del proyecto, entre otros.

C.2. Diseño de datos

En este proyecto de *Django* se han creado cinco aplicaciones, intentado separar la lógica y funcionalidades de cada una de ellas de manera coherente con los resultados esperados. Hay una aplicación principal, denominada *FAT* que se encarga de la gestión y configuración general del proyecto, y otras cuatro aplicaciones que se llaman *News*, *Analysis*, *DashBoard* y *Lab*.

Por otro lado, es importante destacar que se han utilizado cinco bases de datos. Una de ellas dedicada a la gestión de usuarios y sus datos asociados, así como a tablas comunes de datos entre valores: divisas, sectores, etc. Las otras cuatro bases de datos almacenan información histórica de los componentes de cada índice:

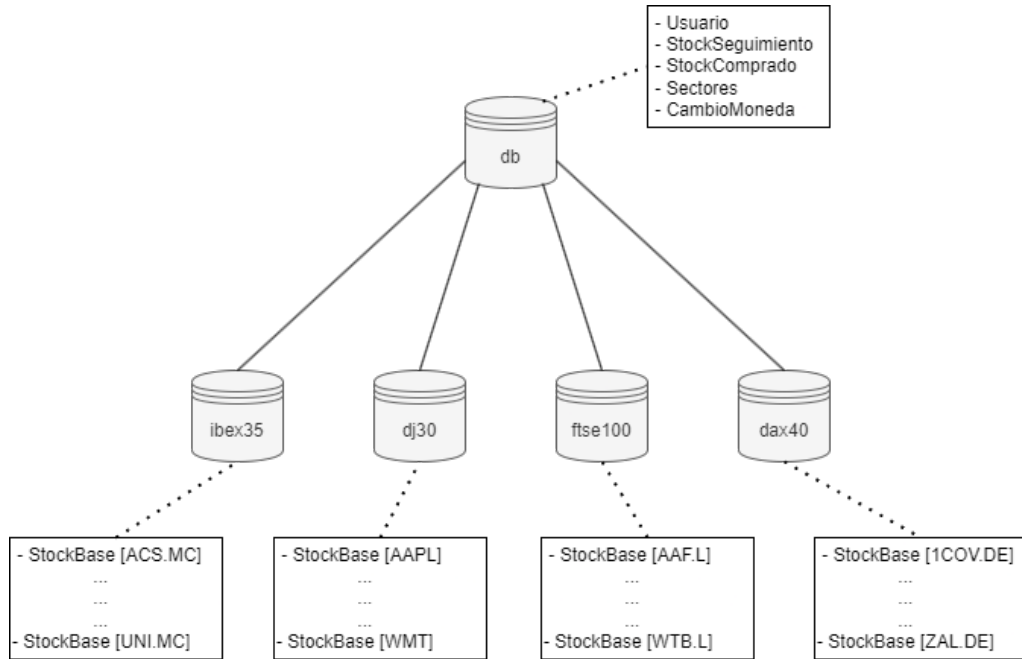


Figura C.1: Diseño de bases de datos. Fuente: elaboración propia

La decisión de diseño inicial ha implicado una adaptación consecuente de los datos, en donde se crean las siguientes entidades:

- **Usuario** tiene un nombre y un identificador únicos, una marca que indica si es *superusuario* y una contraseña que debe cumplir unas condiciones de complejidad. A su vez puede estar asociado a grupos de usuarios y tener permisos de acceso según su rol. Esta entidad viene predefinida por *Django* y de ella no se utilizan los grupos.
- **StockComprado**: representa a un valor que tenga el usuario en cartera. Se define a través de: *usuario*, *ticker_bd*, *bd*, *ticker*, *nombre_stock*, *fecha_compra*, *num_acciones*, *precio_compra*, *moneda*, *sector*, *ult_cierre* y *objects* (para *Django*). Los nombres de los *ticker* se diferencian entre el nombre con formato de punto y el nombre con formato de '_', en todos los casos, para referenciar a las tablas en las bases de datos y evitar conflictos de notación.
- **StockSeguimiento**: representa a un valor que tenga el usuario en seguimiento. sus atributos son: *usuario*, *ticker_bd*, *bd*, *ticker*, *nom-*

bre_stock, *fecha_inicio_seguimiento*, *precio_entrada_deseado*, *moneda*, *sector* y *objects*.

- **Sectores:** para almacenar de forma conjunta todos los valores disponibles en todas las bases de datos junto con su sector de referencia. Este modelo se crea única y exclusivamente para mejorar el rendimiento y evitar la búsqueda recurrente de valor-sector. Sus atributos son: *id*, *ticker_bd*, *bd*, *ticker*, *nombre*, *sector* y *objects*.
- **CambioMoneda:** se utiliza para guardar la información de la última sesión de las divisas afectadas por las bases de datos y sus cambios de divisa. Es una entidad muy pequeña, pero necesaria para el correcto funcionamiento de las aplicaciones. Su estructura interna viene definida por: *id*, *ticker_forex*, *date*, *ultimo_cierre*, *objects*.
- **StockBase:** representa a cada uno de los valores cotizados. Estos valores tienen los siguientes atributos: *id*, *date*, *open*, *high*, *low*, *close*, *volume*, *dividends*, *stock_splits*, *ticker*, *previous_close*, *percent_variance*, *mm20*, *mm50*, *mm200*, *name*, *sector*, *currency* y *objects*. La mayoría de estos atributos vienen impuestos por la API *yFinance* utilizada para la obtención de los datos. Además, es recomendable no modificar los nombres para facilitar la compatibilidad con otras librerías, como *plotly*.

Diagrama E/R

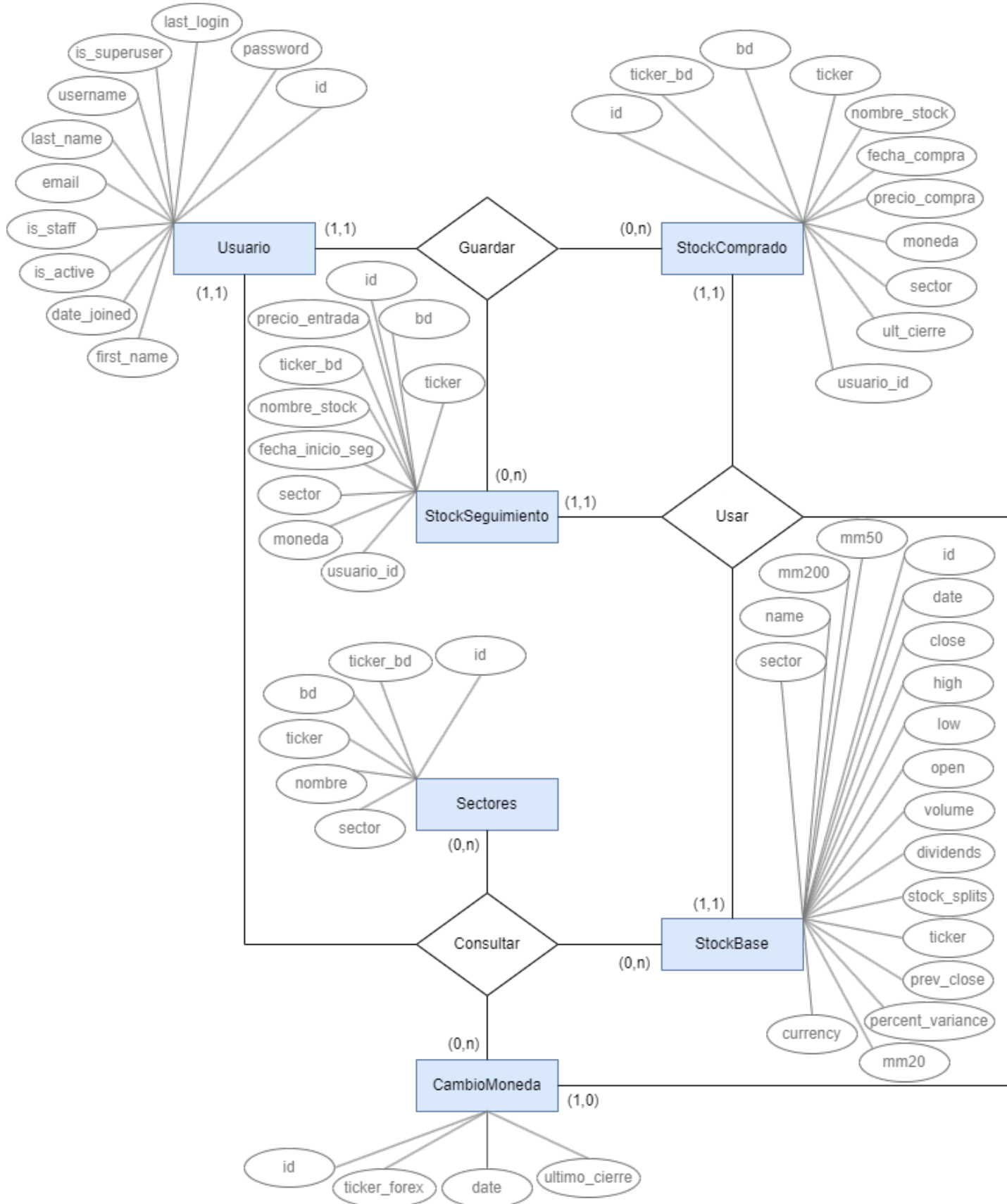


Figura C.2: Diagrama E/R

Diagrama relacional

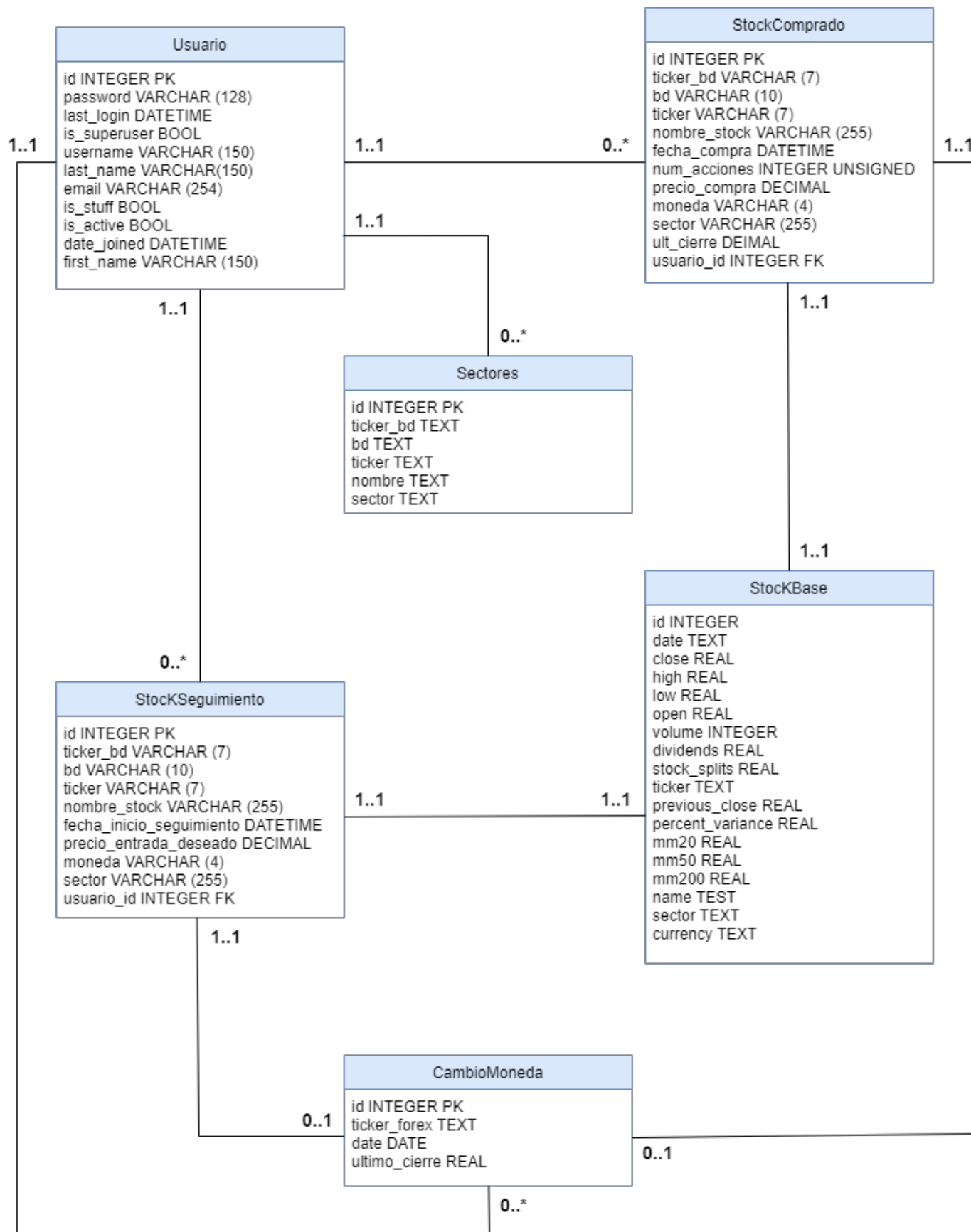


Figura C.3: Diagrama relacional

C.3. Diseño procedimental

Diagrama de secuencias *News* y *Analysis*

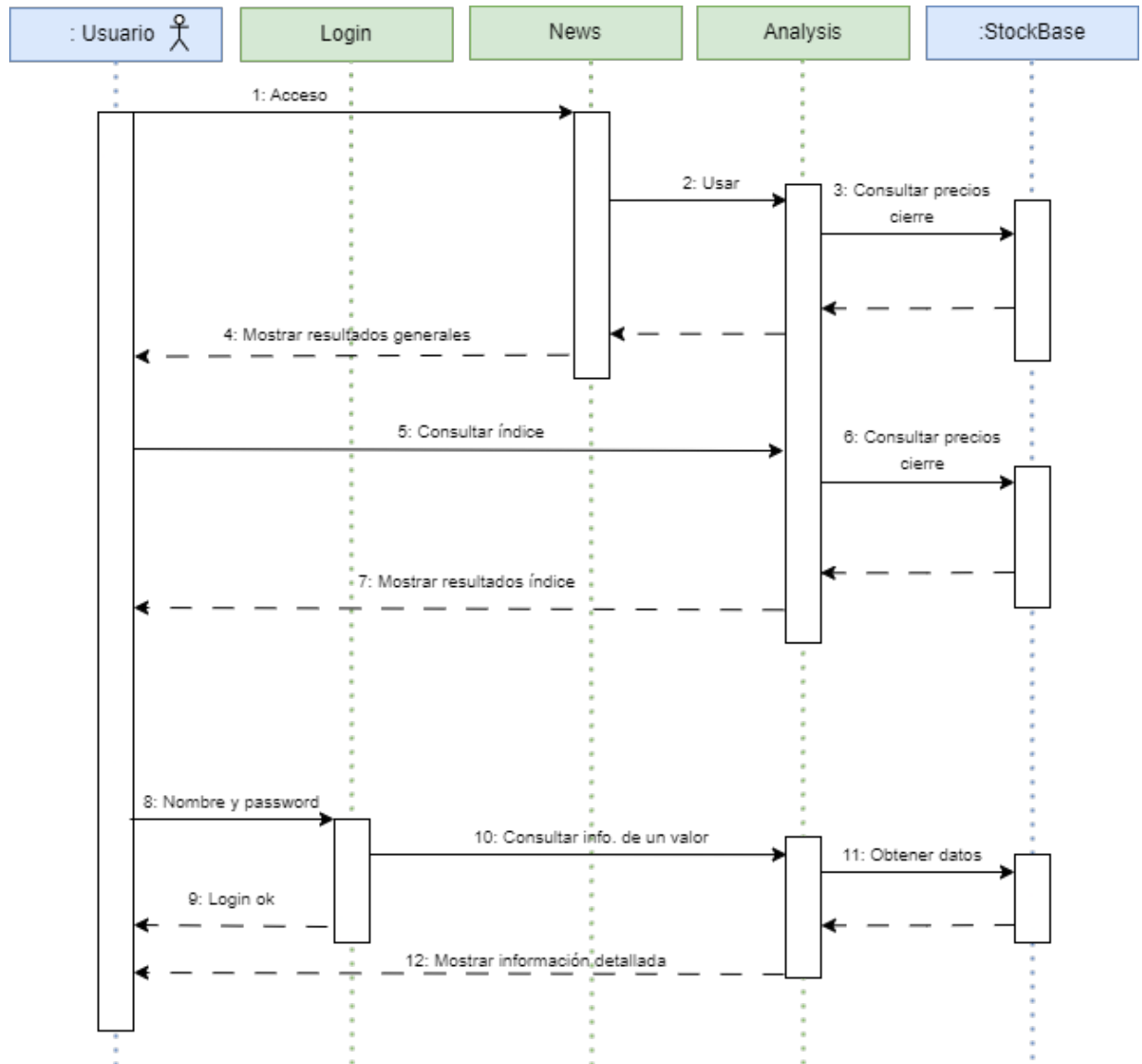


Figura C.4: Diagrama de secuencias *News* y *Analysis*

Diagrama de secuencias *DashBoard*

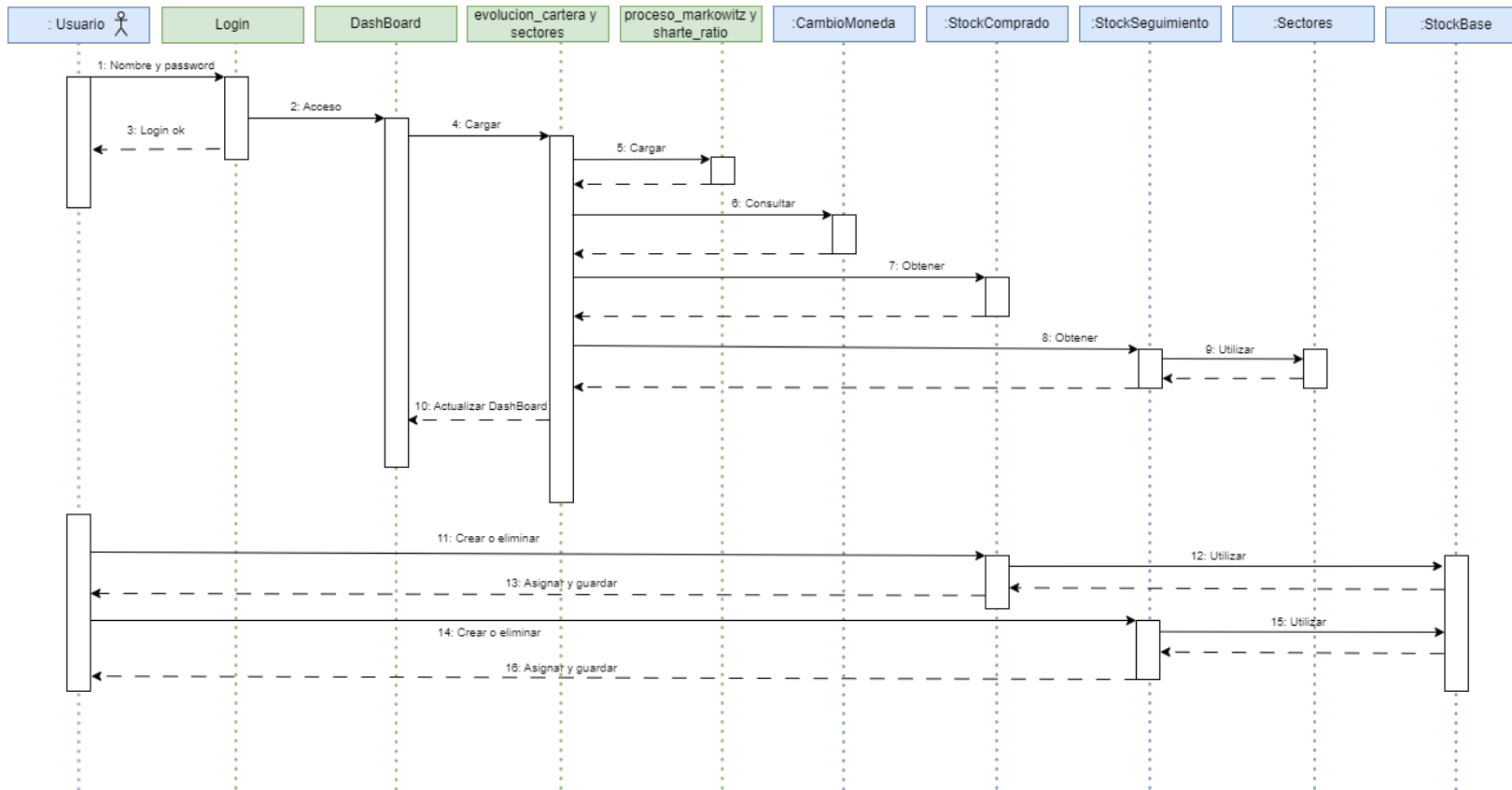
Figura C.5: Diagrama de secuencias *DashBoard*

Diagrama de secuencias *Lab*

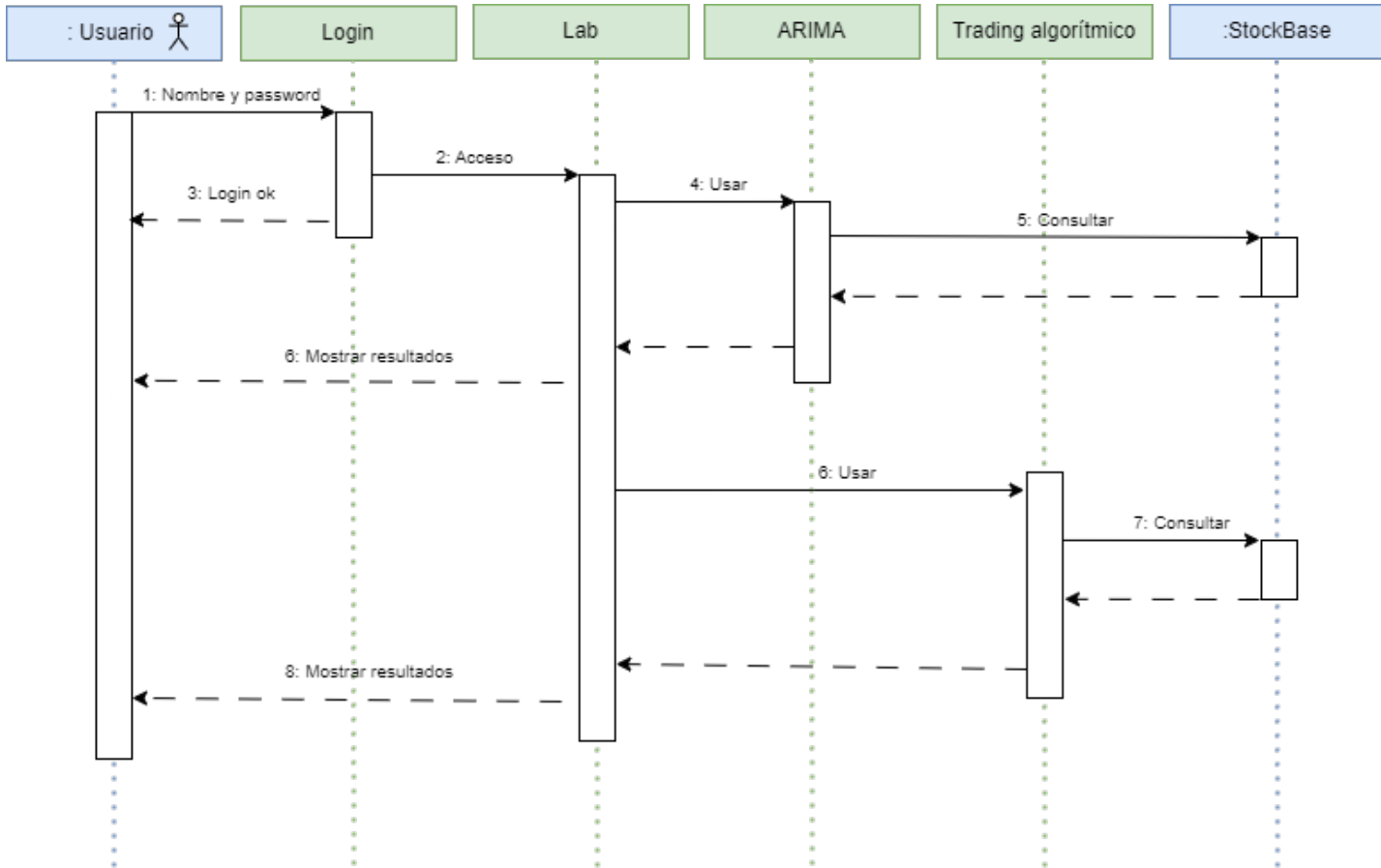


Figura C.6: Diagrama de secuencias *Lab*

C.4. Diseño arquitectónico

Patrón MVT

Al realizar un proyecto en *Django* hay varias decisiones de diseño que vienen impuestas por el propio *framework*. La característica más importante es que se sigue el patrón MVT (*Model View Template*).

El patrón MVT es una arquitectura de desarrollo web donde *Model* maneja la lógica de la base de datos, *View* procesa las solicitudes y devuelve respuestas y *Template* define la presentación visual. MVT facilita la separación de responsabilidades.

ción de responsabilidades, permitiendo un desarrollo mejor organizado y un mantenimiento más eficiente. Las comunicaciones entre estos componentes son gestionadas por el *framework*, lo que simplifica la creación de aplicaciones web.

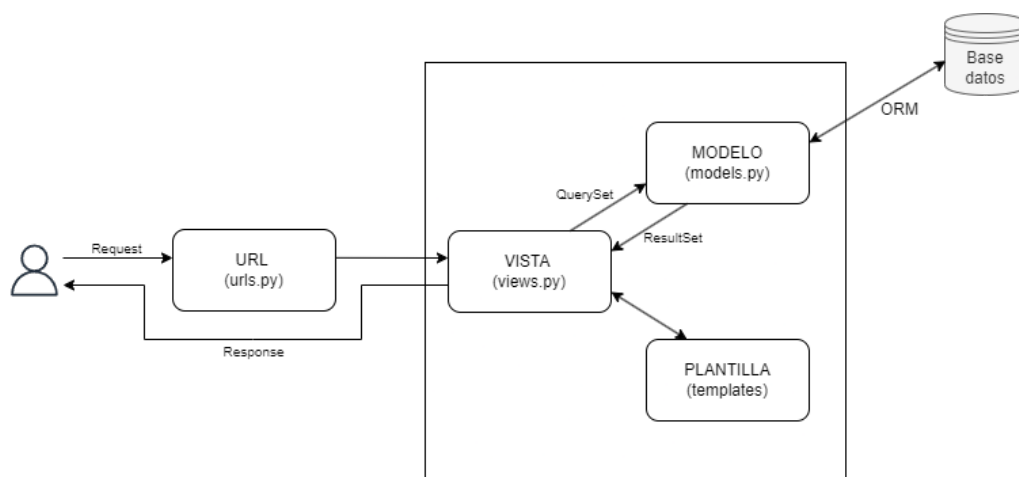


Figura C.7: Diagrama de patrón MVT. Fuente: elaboración propia

Patrón *Repository*

Django trabaja con el patrón *Repository* de manera implícita a través de su sistema de modelos y el ORM (Object-Relational Mapping). Aunque no se implementa este patrón de manera estricta, su ORM y sus *managers* proporcionan una funcionalidad muy similar.

El patrón *Repository* actúa como una capa intermedia entre la lógica de negocio y la capa de acceso a datos. Sus principales objetivos son:

- **Encapsulamiento** encapsula el acceso a la base de datos, proporcionando un entorno accesible para la lógica de negocio.
- **Desacoplamiento** desacopla la lógica de negocio de los detalles de persistencia.
- **Consistencia** proporciona una forma de trabajar con colecciones de objetos de dominio sin exponer detalles de la base de datos.

En este proyecto, de manera adicional, se siguen ideas basadas en este patrón y se crea un esquema muy similar al de un patrón *Repository*:

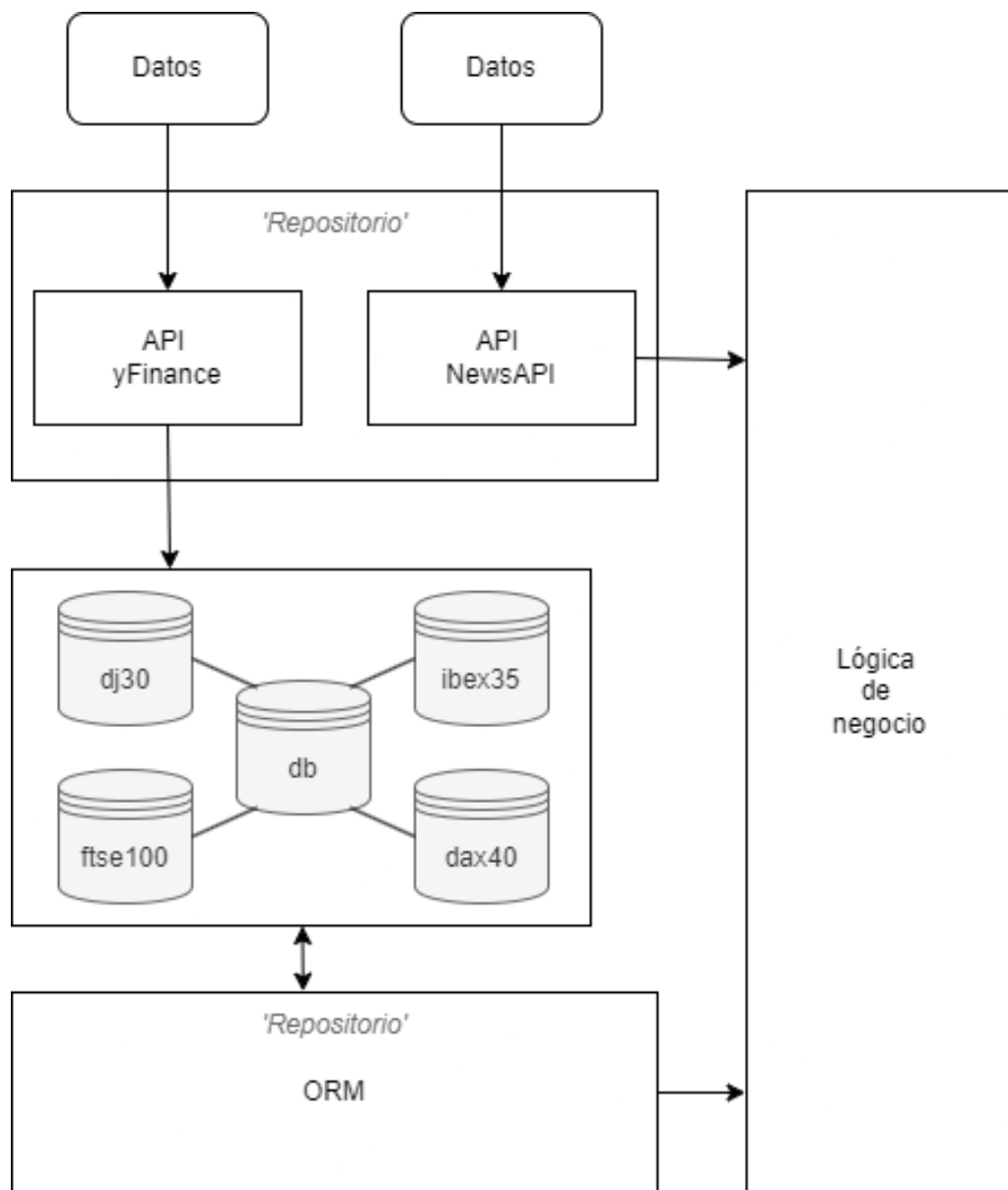


Figura C.8: Patrón repositorio. Fuente: elaboración propia

Patrón *Factory*

Para la creación de los objetos *StockBase* se utiliza el patrón *Factory*. La idea subyacente de la aplicación de este patrón al proyecto es que se

crea una clase común sin definir completamente y, según la necesidad, se va utilizando de forma dinámica para crear las tablas de los valores cotizados.

La lógica para crear las clases de modelos se encapsula en un bucle que itera sobre los *tickers* disponibles, abstrayendo el proceso de creación de instancias concretas de modelos.

Este enfoque permite mejorar la mantenibilidad, ya que no será necesario crear nuevas clases por cada valor que queramos añadir, sino que se pueden actualizar las listas de valores¹ (llamados *tickers* por sencillez de uso a lo largo del proyecto) y, a través de este patrón, se generarán las clases y tablas oportunas.

Patrón *Strategy*

Aunque no sigue exactamente este patrón en el diseño del proyecto, he de mencionar que sí se utiliza un enfoque similar a la hora de enrutar la información hacia las bases de datos.

El *router*² clasifica las bases de datos en función del modelo y define estrategias para:

- Lectura de Datos (*db_for_read*).
- Escritura de Datos (*db_for_write*).
- Permitir Relaciones (*allow_relation*).
- Migraciones (*allow_migrate*).

La mayor ventaja de seguir ideas basadas en este patrón es que las estrategias se pueden cambiar fácilmente, modificando la implementación de los métodos sin afectar a otras partes del proyecto.

Diagramas de paquetes y directorios

En el contexto de *Django*, cada aplicación (*app*) puede considerarse un paquete. *Django* está diseñado de manera modular, permitiendo que cada *app* sea una unidad independiente y reutilizable dentro del proyecto y, por tanto, se va a mostrar la estructura de paquetes basada en esta concepción:

¹Ver `/util/tickers/Tickers_BDs.py` para más información.

²Ver `/FAT/routers/router_bases_datos.py` para más información

Directorio general de todo el proyecto. Distribución de paquetes

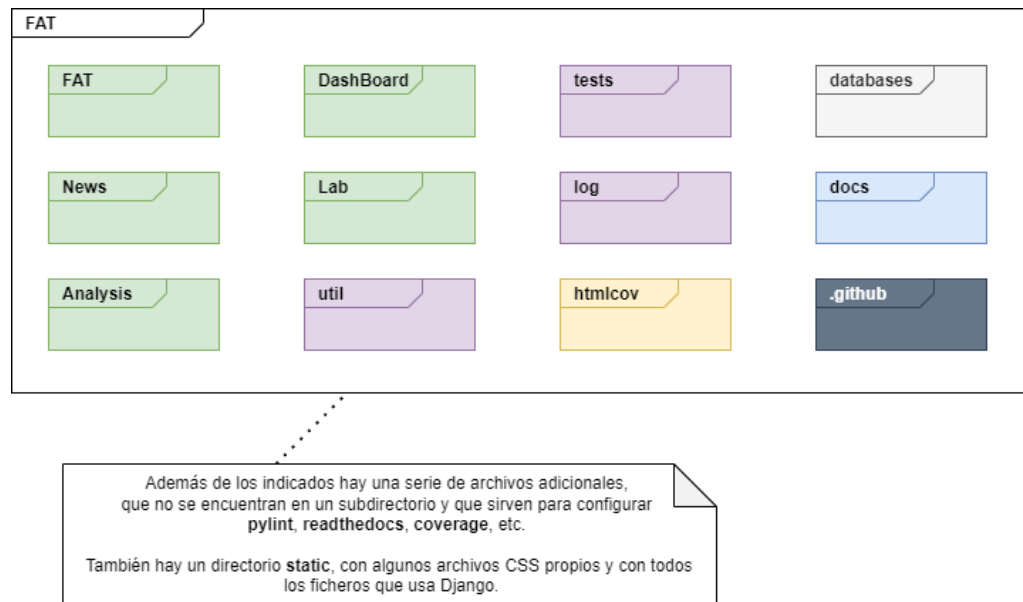


Figura C.9: Diagrama de paquetes general.

Paquete *FAT*

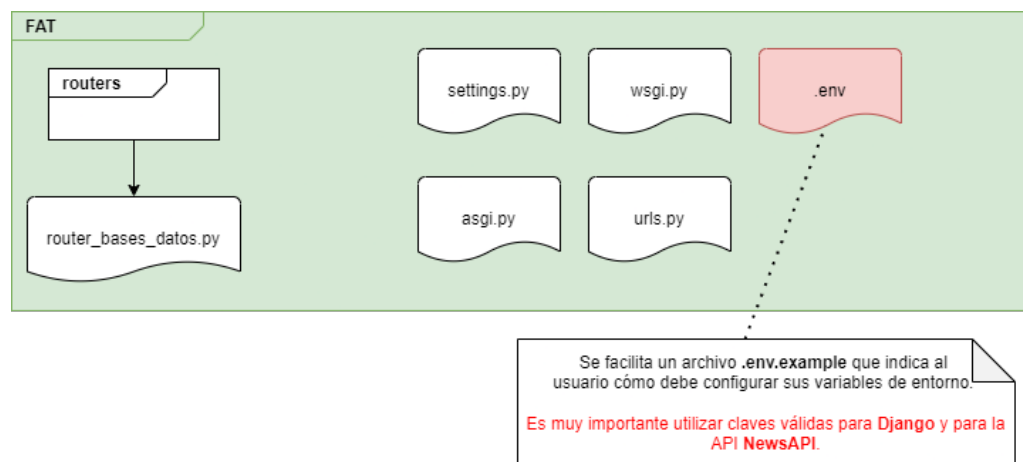


Figura C.10: Diagrama de paquete *FAT*.

Como se ve en la figura anterior, este paquete cumple las siguientes funciones:

- Configuración general del proyecto (*settings*) y direccionamiento de *URLs*.
- Gestión de *router* para las bases de datos.
- Almacenamiento de variables de entorno secretas (*.env*)

Paquete *News*

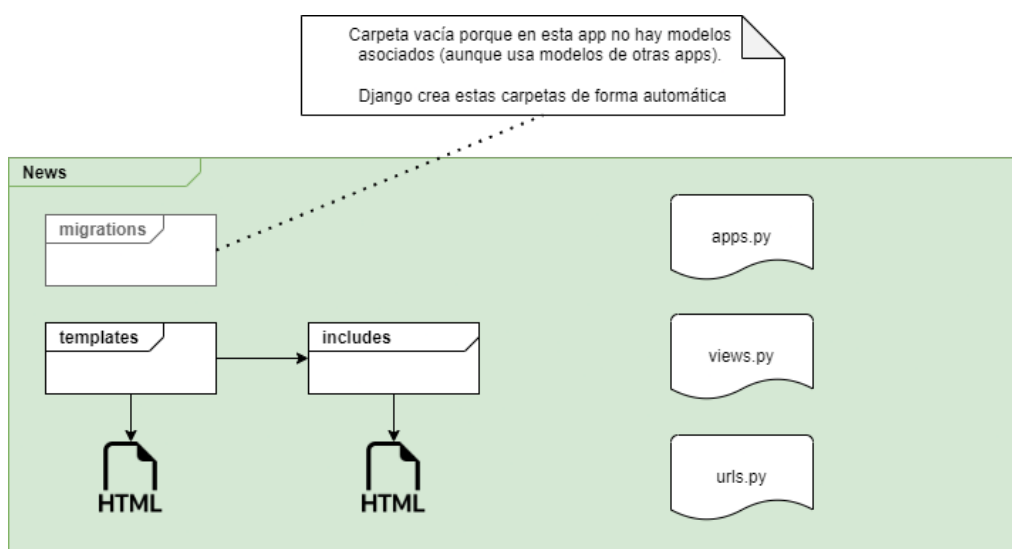


Figura C.11: Diagrama de paquete *News*.

En *News* se gestiona, fundamentalmente, la conexión con la API *NewsAPI*, que permite ofrecer noticias en tiempo real al usuario en portada. Además, permite realizar una búsqueda de los mejores y peores valores de cada índice y dispone de una función de representación gráfica que se integra con otras *apps* del proyecto³.

En este proyecto la cuenta utilizada para la obtención de noticias es gratuita (cualquiera puede adquirir una *API key* gratuita⁴) y los contenidos son, en muchos casos, relacionados con India y Estados Unidos, pero como muestra de integración en una web, parece interesante incluirlo al trabajo por los fines didácticos.

³Ver `/News/views/_generar_figura()` para más información.

⁴Visitar <https://newsapi.org/>.

Paquete *Analysis*

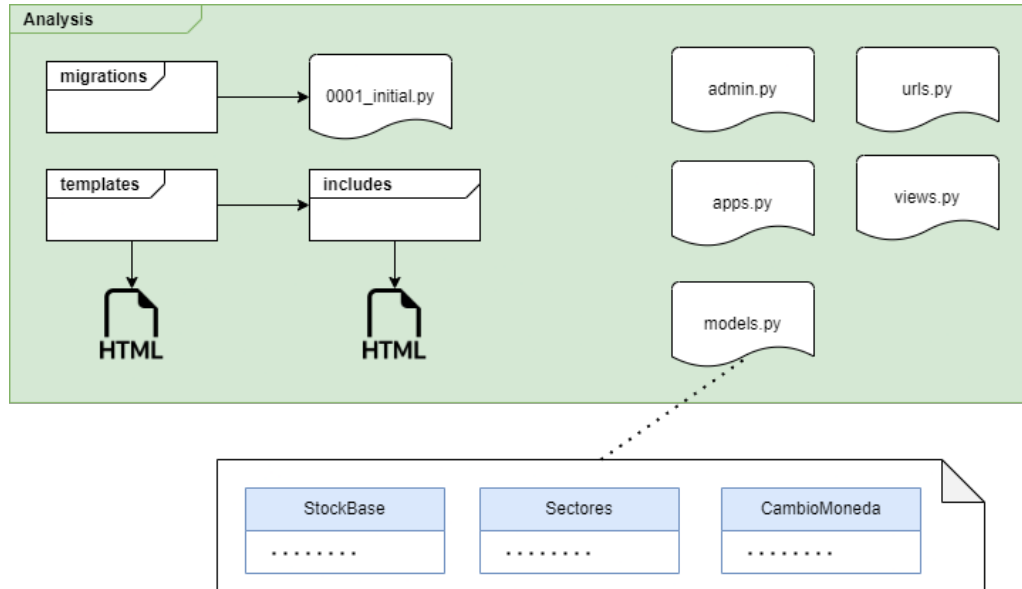


Figura C.12: Diagrama de paquete *Analysis*.

Este paquete cumple las siguientes funciones:

- Creación y gestión de modelos *StockBase*, *Sectores* y *CambioMoneda*.
- Controlar las vistas de registro, *login* y *logout*.
- Control de gráficas dinámicas de valores cotizados.
- Gestión de datos asociados a valores y su sector de referencia.
- Manejo de datos de índices y fuentes RSS asociadas.

Paquete *DashBoard*

El paquete *DashBoard* se encarga de:

- Controlar la evolución de la cartera de un usuario en la divisa seleccionada (euros).
- Creación y gestión de modelos *StockComprado* y *StockSeguimiento*.
- Control de gráficas de Markowitz y distribución de pesos con ratio de Sharpe.
- Manejo de plantillas de zona privada de usuario para ofrecer información agregada.

Además, en esta aplicación e incluyen varios formularios de interacción con el usuario, aprovechando el potencial de *Django* para este tipo de tareas.

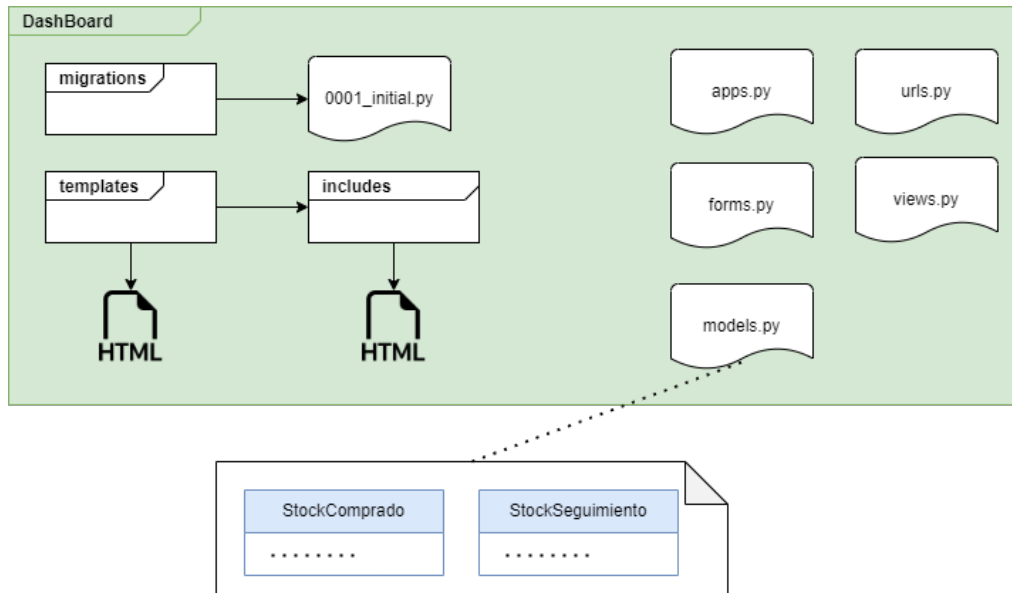


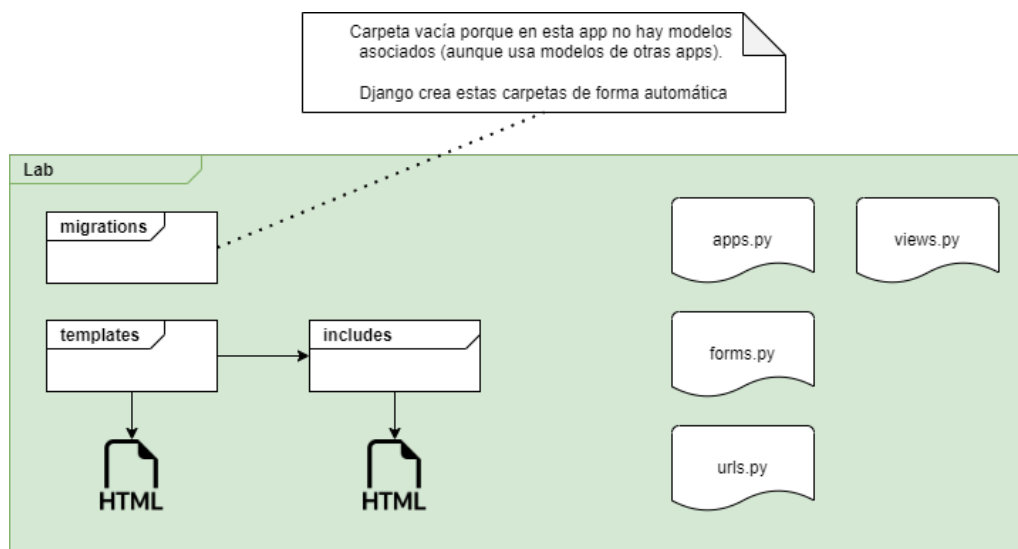
Figura C.13: Diagrama de paquete *DashBoard*.

Paquete *Lab*

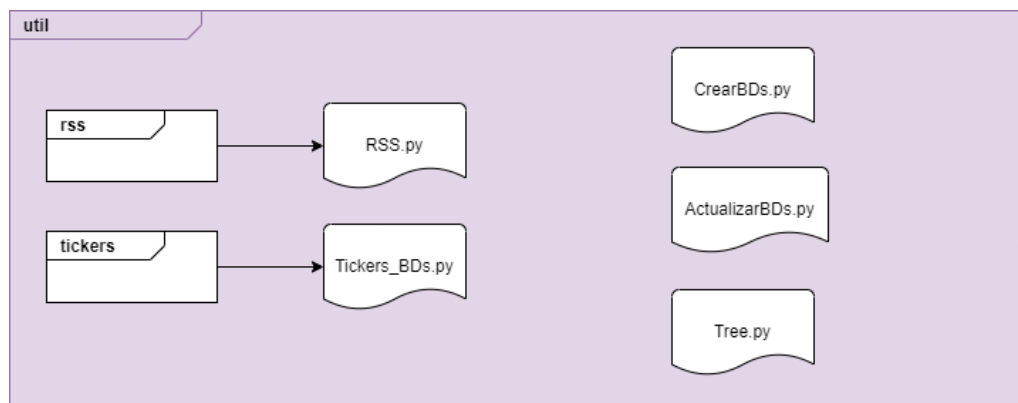
Este paquete cumple con las siguientes funcionalidades⁵:

- Control de *forecasting* con modelos ARIMA.
- Control de funciones de *trading* algorítmico.

⁵Para los usuarios del repositorio y no de la web también está disponible la utilización de redes LSTM con funciones muy básicas.

Figura C.14: Diagrama de paquete *Lab*.

Paquete *util*

Figura C.15: Diagrama de paquete *util*.

Aquí se almacenan los siguientes módulos de utilidad:

- Directorio *rss* con las direcciones de los *feeds* asociados a los índices.
- Directorio *tickers* con métodos que permiten acceder, de forma sencilla, a todos los *tickers* disponibles en la aplicación. Si se quiere añadir un

nuevo índice, es recomendable empezar por añadir aquí los *tickers* de dicho índice, ya que esto facilitaría mucho los pasos posteriores.

- Archivos para crear y actualizar las bases de datos de los valores.
- Archivo para crear un árbol de directorios del proyecto⁶.

Directorio *tests*



Figura C.16: Diagrama de directorio *tests*.

⁶Requiere de la instalación de `seedir` (y `emoji` si se desea).

Como se puede observar en la figura anterior, en *tests* se almacenan todos los tests del proyecto, tratando de seguir la misma estructura de carpetas para mayor mantenibilidad en futuras mejoras.

Se recomienda el uso de **coverage** para lanzar los tests.

Directorio *log*

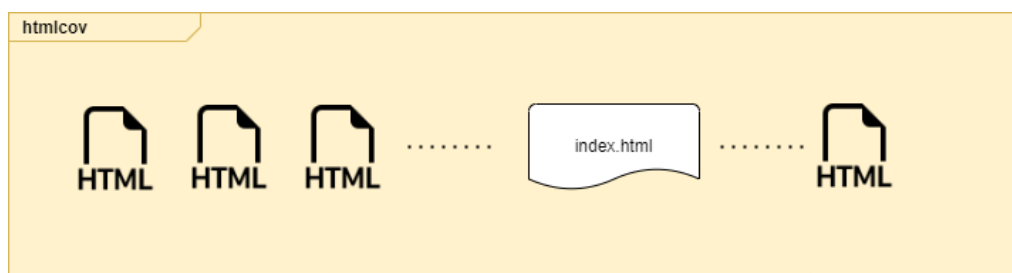
En este paquete se almacenan una serie de archivos de *log* que permiten comprobar lo que ha ocurrido con los tests. Son meramente informativos para el desarrollador. Además, aquí se incluye la configuración del *logger* para todos los tests.



Figura C.17: Diagrama de directorio *log*.

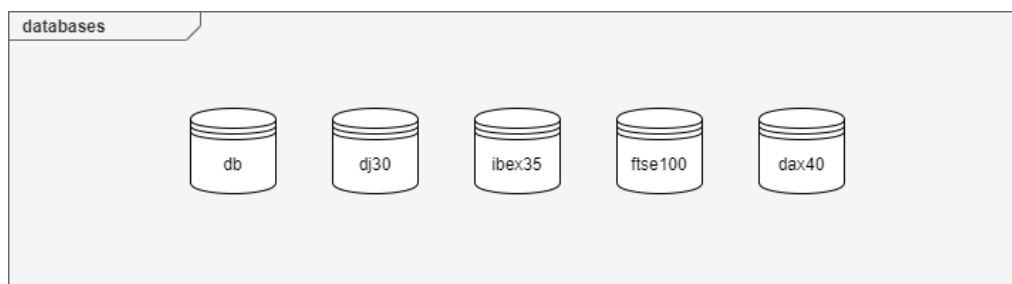
Directorio *htmlcov*

Al utilizar la librería **coverage** podemos generar un informe HTML que permite visualizar las partes de código que quedan sin cubrir y las que ya están cubiertas. Este informe está accesible en este paquete y es consultable haciendo doble *click* sobre el archivo *index.html*:

Figura C.18: Diagrama de directorio *htmlcov*.

Directorio *databases*

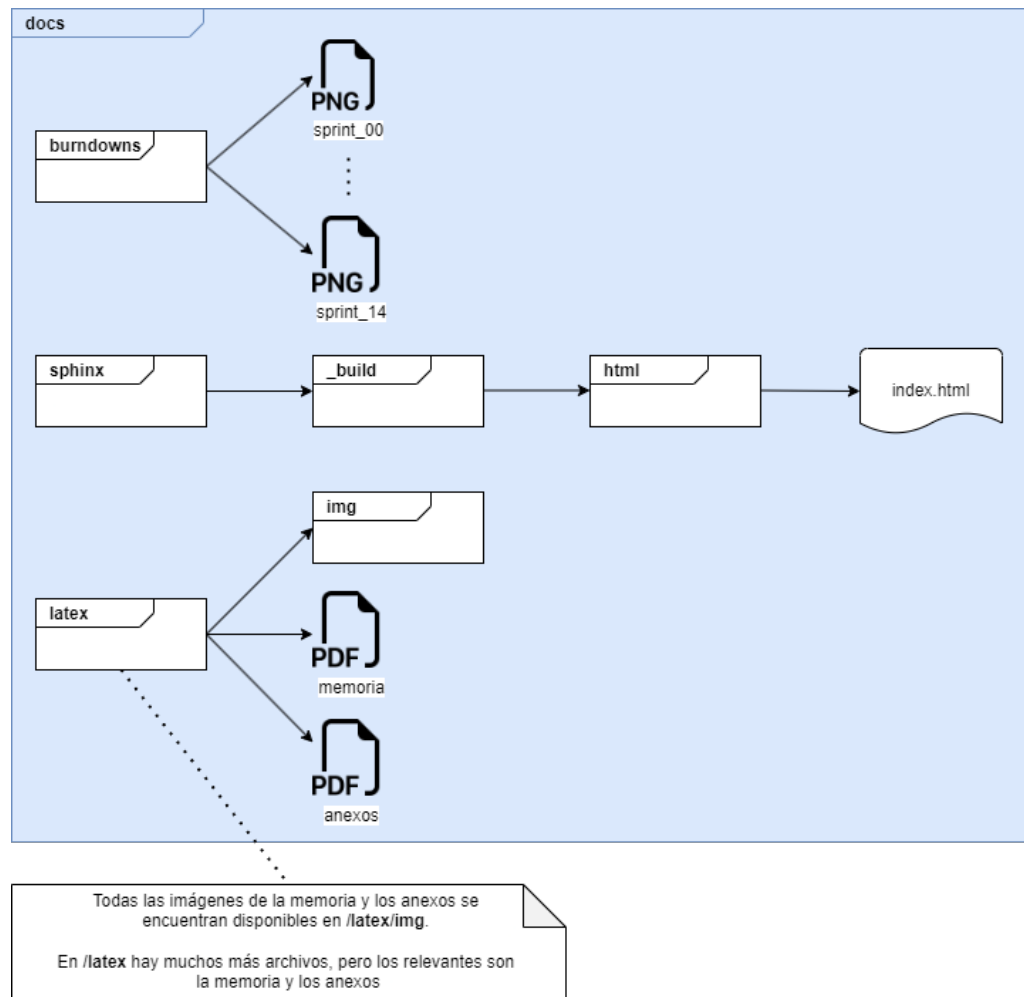
Las bases de datos utilizadas por las aplicaciones se encuentran en este directorio:

Figura C.19: Diagrama de directorio *databases*.

Directorio *docs*

La documentación del proyecto puede ser consultada en este directorio. Los archivos están distribuidos de la siguiente manera:

- */burndowns*: recoge imágenes de la evolución de los diferentes *sprints*.
- */sphinx*: al ejecutar el archivo */sphinx/_build/html/index.html* se puede consultar la documentación del código con el mismo formato que el disponible en *ReadTheDocs*.
- */latex*: almacena la memoria y los anexos, así como los archivos \LaTeX utilizados y las imágenes asociadas.

Figura C.20: Diagrama de directorio *docs*.

Directorio *.github*

Al utilizar *GitHub actions* es necesario configurarlas previamente. Los pasos que se deben realizar en las acciones - cada vez que se hace un *push to origin/main* - se describen en los archivos que podemos encontrar en este directorio.

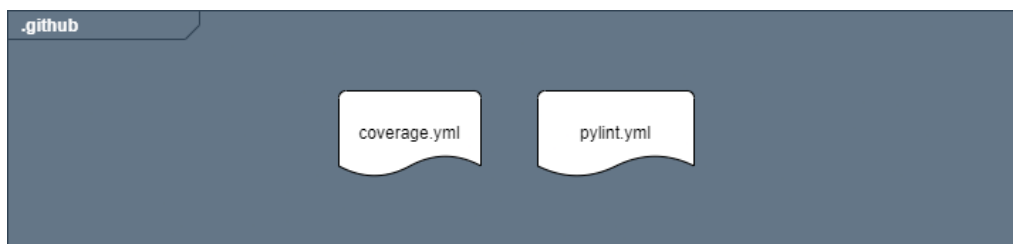


Figura C.21: Diagrama de directorio `.github`.

Las *GitHub actions* están configuradas para generar archivos `.zip`, llamados *Artifacts*, que permitan descargar los informes de resultados tanto de `pylint` como de `coverage`.

Apéndice *D*

Documentación técnica de programación

D.1. Introducción

En este anexo se recoge la documentación técnica de programación, que incluye recomendaciones para el entorno de desarrollo, la estructura de directorios, los procesos de compilación, la configuración de integración e instalación de dependencias y las baterías de tests realizadas.

D.2. Estructura de directorios

El repositorio del proyecto tiene la siguiente distribución de directorios:

- `/`: directorio raíz. Además de todas las rutas de aplicaciones y utilidades, que se detallan a continuación, hay una serie de archivos que cumplen las siguientes funciones:
 - `manage.py`: archivo de utilidad de línea de comandos para tareas administrativas de *Django*.
 - `.gitignore`: configuración de los archivos que no se encuentran en el repositorio de *GitHub*. Queda a criterio del desarrollador modificar este archivo según las necesidades.
 - `.pylintrc`: archivo con configuración general de la herramienta de medición de calidad de código `pylint`. `README.md`: archivo con recomendaciones de instalación y licencia para mostrar en la por-

tada del repositorio de *GitHub*. `requirements.txt`: dependencias del proyecto.

- `/.github`: archivos de configuración de *GitHub actions*.
- `/Analysis`: archivos de la *app Analysis*. Entre otros, aquí se pueden ver los modelos *StockBase*, *Sectores* y *CambioMoneda*.
- `/Analysis/migrations`: archivos, con migraciones de modelos a bases de datos, generados automáticamente por *Django*.
- `/Analysis/templates`: plantillas HTML que permiten ver múltiple información de un valor y su sector asociado.
- `/Analysis/templates/includes`: plantillas HTML auxiliares que aportan mejor organización al proyecto. Todas las plantillas tienen nombres con referencias a las funciones que cumplen para mejorar la mantenibilidad.
- `/DashBoard`: archivos de la *app* que gestiona el área de usuario, donde se puede realizar, por ejemplo, el control de una cartera y ver la distribución de pesos más adecuada.
- `/DashBoard/migrations`: archivos, con migraciones de modelos a bases de datos, generados automáticamente por *Django*.
- `/DashBoard/templates`: plantillas HTML que permiten ver información agregada sobre la cartera del usuario. También están disponibles los formularios de interacción con el usuario para las funcionalidades de esta *app*.
- `/DashBoard/templates/includes`: plantillas HTML auxiliares.
- `/databases`: directorio en el que se encuentran las bases de datos de la aplicación.
- `/docs`: carpeta de documentación.
- `/docs/burndowns`: donde se almacenan las imágenes que corresponden a la evolución de los diferentes *sprints*.
- `/docs/latex`: documentación de la memoria y los anexos del Trabajo de Fin de Grado que ha llevado al desarrollo de este proyecto.
- `/docs/sphinx/_build/html`: documentación en formato HTML de todo el código del proyecto siguiendo el estilo de *ReadTheDocs*.
- `/FAT`: aplicación principal para gestión del proyecto. Aquí se encuentra el archivo de `settings.py` junto con otros archivos relevantes para el correcto funcionamiento del resto de aplicaciones. También se puede ver la configuración del enrutamiento a las bases de datos. Además, aquí se encuentra el archivo `.env` donde se guardan las claves secretas de *Django* y *NewsAPI*.
- `/htmlcov`: informe de tests realizado con `coverage`.

- `/Lab`: archivos de la *app* que gestiona el laboratorio virtual. Aquí se encuentran los archivos de control de modelos ARIMA, estrategias basadas en *machine learning*, etc.
- `/Lab/migrations`: directorio vacío porque esta *app* utiliza modelos ya creados para otras aplicaciones del mismo proyecto; se crea automáticamente por *Django*. Aquí podemos ver los formularios *Django* que se utilizan para interactuar con el usuario.
- `/Lab/templates`: plantillas HTML que permiten ver información y mostrar con los formularios con los campos predefinidos en `forms.py`.
- `/Lab/templates/includes`: plantillas HTML auxiliares.
- `/log`: configuración del *logger* y archivos de *log* de los tests realizados. Estos archivos pueden ser útiles para futuros desarrolladores.
- `/News`: archivos de la *app* que gestiona las noticias y la información de valores de la portada web.
- `/News/migrations`: directorio vacío, igual que en `/Lab/migrations`.
- `/News/templates`: plantillas HTML que permiten ver información de portada.
- `/News/templates/includes`: diseño de botones HTML de utilidad.
- `/static`: archivos estáticos.
- `/static/admin`: archivos de *Django* que permiten el acceso, a través de `http://127.0.0.1:8000/admin/`, al área de gestión visual de administración.
- `/static/css`: archivos CSS utilizados en el proyecto para mantener los mismos estilos en las diferentes aplicaciones.
- `/static/icons`: diferentes iconos para mostrar en la herramienta.
- `/tests`: todos los tests de las distintas aplicaciones y pruebas de algunas de las herramientas de utilidad creadas para favorecer el desarrollo.
- `/util`: herramientas de utilidad para controlar los índices bursátiles y valores disponibles. Además, aquí se almacenan las referencias de los *feed* RSS y los archivos que sirven para crear y/o actualizar las bases de datos.

D.3. Manual del programador

En este manual se explican los pasos más relevantes que se han seguido para la creación de este proyecto y que pueden servir como referencia para futuros desarrolladores. En él se explica cómo preparar el entorno de desarrollo y qué dependencias necesitaremos. Además, se detallan los pasos más relevantes en cuanto a compilación, instalación y ejecución.

Se recomienda consultar la [documentación](#) y, especialmente, el apartado de *Instalación en local*, donde se encuentra disponible un resumen de este manual. El mismo resumen se puede encontrar en la portada del [repositorio de este proyecto](#).

A continuación se detallan los pasos lógicos que se pueden seguir para trabajar con este proyecto:

Paso 1. Descargar, clonar o hacer un *fork* del repositorio

Podemos descargar los archivos desde el [repositorio](#).

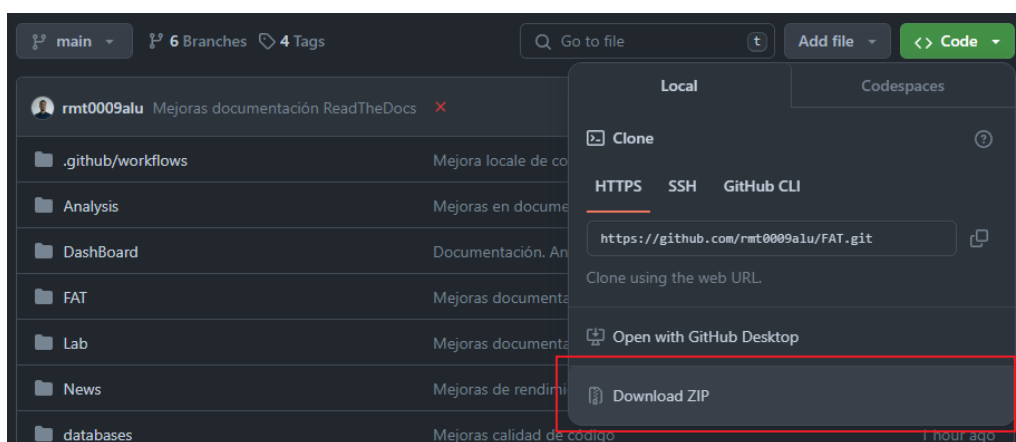


Figura D.1: Descargar proyecto en formato .zip

Si tenemos instalado un sistema de control de versiones distribuido como [Git](#), podremos hacer un *clon* del repositorio para trabajar con él en local. Este método es ideal si no se pretenden hacer contribuciones con el proyecto original. Sólo debemos crear un directorio donde queramos instalar los archivos del repositorio y lanzar el siguiente comando:

```
MINGW64/d/Proyecto
R@Equipo-R MINGW64 /d/Proyecto
$ git clone https://github.com/rmt0009alu/FAT.git
Cloning into 'FAT'...
remote: Enumerating objects: 18018, done.
remote: Counting objects: 100% (4808/4808), done.
remote: Compressing objects: 100% (2578/2578), done.
remote: Total 18018 (delta 2185), reused 4762 (delta 2139), pack-reused 13210
Receiving objects: 100% (18018/18018), 424.57 MiB | 8.56 MiB/s, done.
Resolving deltas: 100% (4909/4909), done.
Updating files: 100% (463/463), done.
```

Figura D.2: Clonar con *Git* .zip

Otra opción es que se haga un *fork* para guardar los datos en un repositorio del desarrollador. De esta manera se podrá contribuir fácilmente con el desarrollo realizado hasta ahora.

Paso 2. Intalar Python

Una vez tenemos los archivos en nuestro equipo tendremos que instalar **Python**. En el desarrollo de este proyecto se ha utilizado Python 3.11.5.

Paso 3. Instalar entorno

Instalar un IDE del gusto del desarrollador. Se recomienda el uso de **VS Code**.

VS Code ya se puede considerar un entorno de desarrollo integrado. Aunque inicialmente fue concebido como un editor de código fuente ligero, ha evolucionado para ofrecer muchas de las funcionalidades que se esperan de un IDE; y en el desarrollo de este proyecto no han sido necesarias más herramientas.

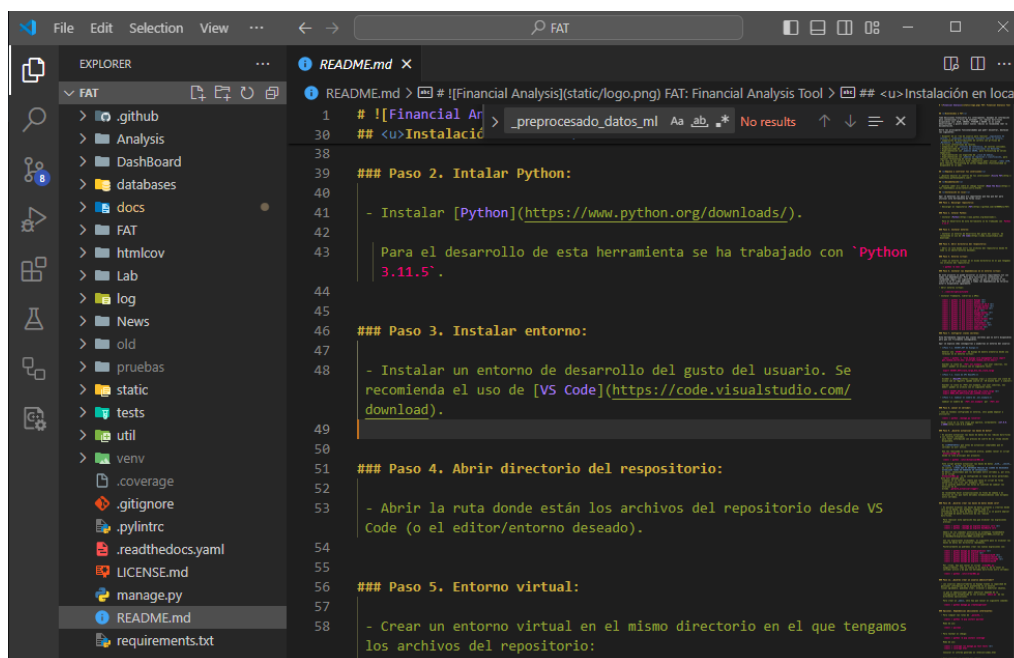


Figura D.3: VS Code

Paso 4. Abrir directorio del repositorio

Abrir la ruta donde están los archivos del repositorio desde VS Code (o el editor/entorno deseado).

Paso 5. Entorno virtual

Es altamente recomendable trabajar con un entorno virtual. Su instalación es sencilla y puede ayudarnos a controlar mejor las dependencias necesarias.

Pulsamos **F1** y podemos escribir *Interpreter* para seleccionar el *interpreter* de *Python* que se adapte a las características del proyecto:

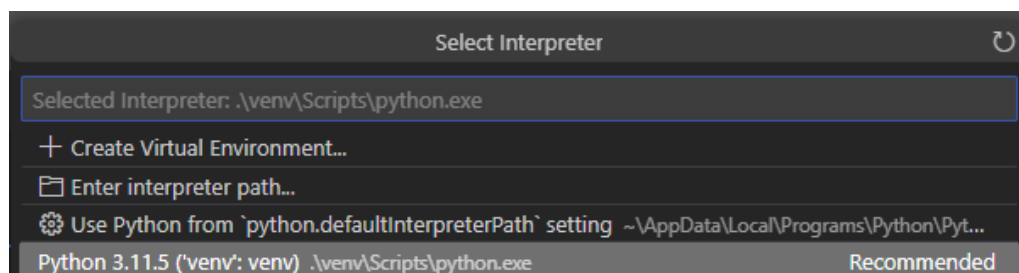


Figura D.4: Selección de *interpreter*

Luego volvemos a pulsar **F1** y escribimos **Terminal** para abrir una nueva terminal. Desde esa terminal creamos nuestro entorno virtual, en el mismo directorio en el que tengamos los archivos del repositorio:

```
> python -m venv venv
```

Paso 6. Instalar las dependencias en el entorno virtual

En este proyecto se puede encontrar un archivo *requirements.txt* con todas las dependencias. Pero para facilitar la instalación se recomienda seguir los siguientes pasos, ya que se instalarán las librerías en el orden adecuado y todas las dependencias de terceros estarán disponibles igualmente:

Primero abrimos el entorno virtual que hemos creado en el paso previo:

```
> .\venv\Scripts\activate
```

Veremos algo similar a lo siguiente:

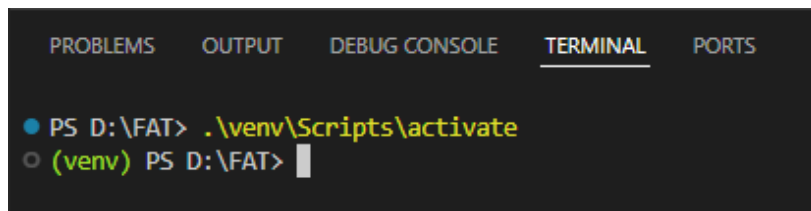


Figura D.5: Activar entorno virtual

Luego se instalan, en ese entorno virtual, el *framework*, las librerías y las APIs necesarias:

```
(venv) > python -m pip install Django
(venv) > python -m pip install pandas
(venv) > python -m pip install plotly==5.18.0
(venv) > python -m pip install newsapi-python
(venv) > python -m pip install -U matplotlib
(venv) > python -m pip install mpld3
(venv) > python -m pip install django-pandas
(venv) > python -m pip install feedparser
(venv) > python -m pip install yfinance
(venv) > python -m pip install python-dotenv
(venv) > python -m pip install networkx
(venv) > python -m pip install statsmodels
(venv) > python -m pip install scikit-learn
(venv) > python -m pip install pmdarima
(venv) > python -m pip install tensorflow
```

Paso 7. Configurar claves secretas

Este proyecto requiere dos claves secretas que no están disponibles, pero que se pueden conseguir de manera sencilla.

Aquí se explica cómo conseguirlas y añadirlas al entorno del usuario:

- **Paso 7.1. SECRET_KEY de Django** Generar una `SECRET_KEY` de *Django* de manera aleatoria desde una terminal en el entorno virtual:

```
(venv) > python -c 'from django.core.management.utils
↪ import get_random_secret_key;
↪ print(get_random_secret_key())'`
```

Guardar la clave en `/FAT/.env.example`, sin usar comillas, nos deberá quedar un archivo con el siguiente texto:

```
export SECRET_KEY=clave_larga_123456_clave_larga
```

■ Paso 7.2. Clave de API NewsAPI

Acceder a [NewsAPI](#) y solicitar una clave de acceso con un registro.

Guardar la clave en `/FAT/.env.example` - sin usar comillas - nos deberá quedar un archivo con el siguiente texto:

```
export SECRET_KEY=clave_larga_123456_clave_larga
export NEWS_API_KEY=clave_api_123456_clave_api
```

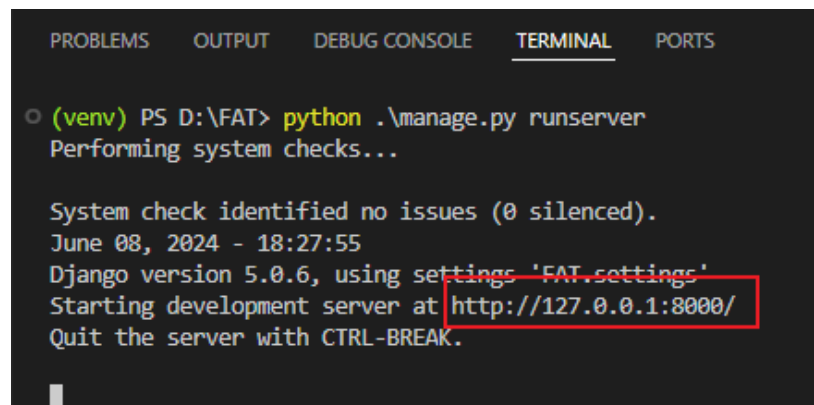
- **Paso 7.3. Cambiar el nombre de `.env.example`** Cambiar el nombre de `/FAT/.env.example` por `/FAT/.env`.

Paso 8. Lanzar el servidor

Como ya tenemos configurado el entorno, sólo queda empezar a utilizarlo:

```
(venv) > python .\manage.py runserver
```

Hacer click en la ruta local que aparece, normalmente [localhost](#).



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

○ (venv) PS D:\FAT> python .\manage.py runserver
Performing system checks...

System check identified no issues (0 silenced).
June 08, 2024 - 18:27:55
Django version 5.0.6, using settings 'FAT.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Figura D.6: Lanzar el servidor

Paso 9. ¿Quieres actualizar las bases de datos?

Es posible actualizar las bases de datos de los índices bursátiles y sus valores cotizados, para tener información con precios de cierre de la última sesión disponible.

Es **importante** que antes de actualizar compruebes que el servidor no está activo.

Puedes lanzar el *script* `ActualizarBDs.py`, una vez realizada la comprobación previa, desde la ruta principal del proyecto:

```
(venv) > python .\util\ActualizarBDs.py
```

Este *script* permite actualizar las bases de datos `dj30`, `ibex35`, `ftse100` y `dax40` con precios de cierre. **Para que se obtengan precios de cierre es necesario actualizar fuera de horarios de cotización.** Es decir, necesitamos que los mercados estén cerrados y, por ello, en el *script* `ActualizarBDs.py` se ha configurado un rango de horas permitidas. Esto está pensado para trabajar en un servidor remoto que lanza el *script* de forma automática (con *cron*, por ejemplo), pero si se quieren modificar las horas, sólo es cuestión de cambiar los horarios en el método `_permite_actualizar(logger)`.

Se recomienda hacer actualizaciones en fines de semana o en horarios en los que tanto mercados estadounidenses como europeos estén cerrados.

Paso 10. ¿Quieres crear las bases de datos desde cero?

Es posible eliminar las bases de datos actuales y crearlas desde cero si se desea, por ejemplo, limpiar toda la información de la base de datos por defecto o si se quiere ampliar la cantidad de datos históricos de los índices bursátiles.

Para realizar esta operación hay que eliminar las migraciones previas:

```
(venv) > python .\manage.py migrate Analysis zero
(venv) > python .\manage.py migrate DashBoard zero
```

Además de los comandos anteriores es altamente recomendable eliminar los archivos `/Analysis/migrations/0001_initial.py` y `/DashBoard/migrations/0001_initial.py`.

Con las migraciones eliminadas, el siguiente paso es eliminar las bases de datos del directorio `/databases`.

Posteriormente ya podremos crear las nuevas migraciones con:

```
(venv) > python manage.py makemigrations
(venv) > python manage.py migrate
(venv) > python manage.py migrate --database=dj30
(venv) > python manage.py migrate --database=ibex35
(venv) > python manage.py migrate --database=ftse100
(venv) > python manage.py migrate --database=dax40
```

Por último, hay que lanzar el *script* `CrearBDs.py`, asegurándonos, como se indica en el paso 9, de no tener el servidor activo y de que los mercados bursátiles están cerrados:

```
(venv) > python .\util\CrearBDs.py
```

Paso 11. ¿Quieres crear un usuario administrador?

Los usuarios administradores en *Django* tienen la capacidad de gestionar información de la plataforma en un entorno visual agradable, pudiendo crear, eliminar y modificar objetos.

Lo que el administrador podrá ver depende de la configuración establecida en los archivos `admin.py` de las diferentes aplicaciones.

Para crear un *admin* sólo hay que lanzar el siguiente comando:

```
(venv) > python manage.py createsuperuser
```

Opcional. Dependencias adicionales interesantes

- Para limpiar las rutas de `_pycache_`:

```
(venv) > python -m pip install pyclean
```

Modo de uso:

```
(venv) > pyclean .
```

- Para testear el código:

```
(venv) > python -m pip install coverage
```

Modo de uso:

```
(venv) > coverage run manage.py test tests
(venv) > coverage html
```

Consular el informe generado en `/htmlcov/index.html`.

- Para comprobar la calidad del código:

```
(venv) > python -m pip install pylint
```

Modos de uso:

```
(venv) > pylint .\DashBoard\views.py
(venv) > pylint .\DashBoard
(venv) > pylint .
```

- Para crear documentación del estilo de ReadTheDocs:

```
(venv) > python -m pip install sphinx
(venv) > python -m pip install sphinxcontrib-django
(venv) > python -m pip install sphinx_rtd_theme
(venv) > python -m pip install recommonmark
```

Modo de uso:

```
cd .\docs\sphinx\
(venv) \docs\sphinx\ > .\make.bat html
```

En `/docs/sphinx/_build/html/index.html` se puede consular la documentación del código generada.

D.4. Pruebas del sistema

En este trabajo se han realizado múltiples pruebas, creando instancias *adhoc* para los tests en muchos casos. Inicialmente se tomó la decisión de no utilizar *mocks*, de esta manera se simulaba que se realizan pruebas de integración con las bases de datos de forma continua. Esta decisión implica que el rendimiento de los propios tests disminuye, ya que se están creando instancias reales, pero favorece el control de los datos y obliga al desarrollador a entender lo que está ocurriendo en cada momento.

Tests unitarios

En este trabajo hay 250 tests unitarios, que cubren prácticamente todo el código, como se puede ver en el informe HTML realizado con **coverage** y que está disponible en `htmlcov/index.html`:

```
(venv) PS > coverage run .\manage.py test tests
Found 250 test(s).
Creating test database for alias 'default'...
Creating test database for alias 'dj30'...
Creating test database for alias 'ibex35'...
Creating test database for alias 'ftse100'...
Creating test database for alias 'dax40'...
System check identified no issues (0 silenced).
.....
-----
Ran 250 tests in 544.265s
OK
Destroying test database for alias 'default'...
Destroying test database for alias 'dj30'...
Destroying test database for alias 'ibex35'...
Destroying test database for alias 'ftse100'...
Destroying test database for alias 'dax40'...
```

Figura D.7: Número de tests realizados

Al final del informe de **coverage** podremos ver el siguiente resultado:

Module	statements	missing	excluded	coverage
Total	4122	11	0	100 %

Tabla D.1: Cobertura de código

Con estos tests se han realizado:

- Pruebas de caja blanca, que implican el conocimiento del código:
 - Pruebas de interfaces entre modelos y métodos.
 - Pruebas de estructuras de datos locales.
 - Pruebas de camino básico (en algunos casos).

- Pruebas de condiciones límite, especialmente para el control de entrada de datos de usuarios, tanto en *login* y registro como para trabajar con las aplicaciones *DashBoard* y *Lab*.

- Pruebas de caja negra, como si no se conocieran los datos:

De forma intensiva se han probado todos los contextos con los datos a renderizar en cada plantilla, sin tener en cuenta el código interno.

En estas pruebas se incluyen los siguientes conceptos:

- Partición de equivalencia: se ha tratado de dividir la entrada en aplicaciones y cada aplicación en vistas, formularios, etc.
- Análisis de valores límite.

Además, se puede comprobar un *log*, que fui realizando para un mejor control y que guarda un comentario de casi todas aquellas pruebas que se van pasando. La estructura del *log* se controla con un *logger* y se pueden comprobar estos datos en el directorio **/log**.

Pruebas de integración

Como se ha mencionado en el apartado anterior, se ha tratado de comprobar continuamente la correcta integración con las bases de datos. Pero, de forma adicional, en las fases iniciales del proyecto se realizó una batería básica de tests para comprobar la correcta creación y actualización de las bases de datos, así como el enrutamiento y la configuración general del proyecto.

Estos tests están disponibles en **/tests/databases** y en **/tests/FAT**.

Su ejecución se realiza a través de **coverage** tal y como se indica en el apartado anterior.

Pruebas de interfaz

Se han realizado algunas pruebas básicas de interfaz utilizando la librería *selenium*[?] y el *driver* de *Chrome*. Estos tests se pueden encontrar en el archivo **tests/tests_ui/tests_ui.py** junto con el *chromedriver.exe*[?] para el navegador web Chrome versión 125.0.xxxx.

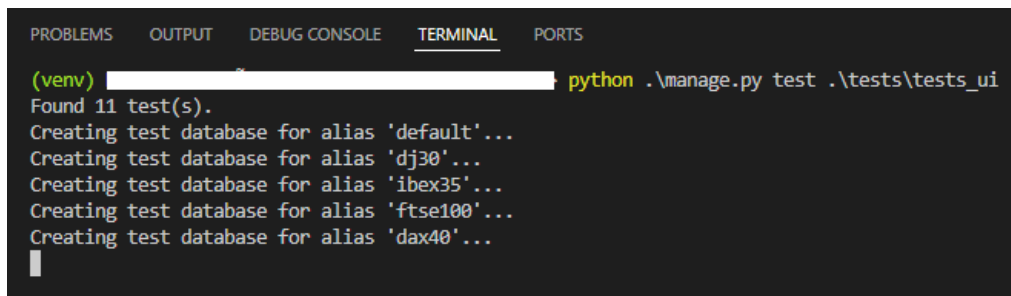
Con estas pruebas se trató de simular interacciones con usuarios, especialmente para funcionalidades de registro, *login* y *logout*. También se probó la interacción con algunos formularios de la web - no todos -.

La batería de pruebas de interfaz es reducida, ya que las diferentes casuísticas estaban recogidas en los tests unitarios, pero han cubierto los siguientes requerimientos:

Test	Requerimiento
<code>test_pag_principal</code>	RF-1 Mostrar portada con información general.
<code>test_pag_principal_artículos</code>	RF-1.1 Mostrar carrusel de noticias generales.
<code>test_pag_principal_mejores_peores</code>	RF-1.2 Mostrar mejores y peores valores de cada índice.
<code>test_login</code>	RF-2 Control de usuarios. RF-2.2 Permitir hacer login.
<code>test_logout</code>	RF-2 Control de usuarios. RF-2.2 Permitir hacer logout.
<code>test_registro</code>	RF-2 Control de usuarios. RF-2.2 Permitir hacer registro.
<code>test_mostrar_tabla_valores</code>	RF-4 Consultar índice. RF-4.1 Mostrar tabla de valores.
<code>test_consultar_un_valor</code>	RF-4 Consultar índice. RF-4.2 Consultar un valor del índice.
<code>test_dashboard_nueva_compra</code>	RF-3 Gestionar DashBoard. RF-3.1 Crear nuevo valor en cartera.
<code>test_dashboard_nuevo_seguimiento</code>	RF-3 Gestionar DashBoard. RF-3.3 Crear un nuevo valor en seguimiento.
<code>test_arima_manual</code>	RF-5.1 Trabajar con ARIMA. RF-5.1.2 Introducir (p,d,q) de forma manual.

Tabla D.2: Requerimientos testeados con pruebas de interfaz.

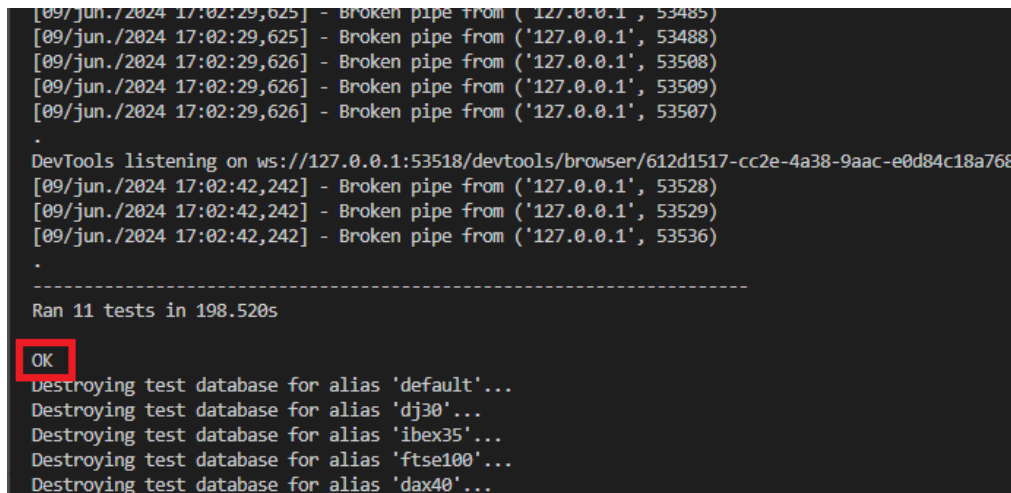
Para lanzar estas pruebas se realiza lo siguiente:



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
(venv) python .\manage.py test .\tests\tests_ui
Found 11 test(s).
Creating test database for alias 'default'...
Creating test database for alias 'dj30'...
Creating test database for alias 'ibex35'...
Creating test database for alias 'ftse100'...
Creating test database for alias 'dax40'...
```

Figura D.8: Lanzar pruebas de interfaz

Y el resultado esperado es:



```
[09/jun./2024 17:02:29,625] - Broken pipe from ('127.0.0.1', 53485)
[09/jun./2024 17:02:29,625] - Broken pipe from ('127.0.0.1', 53488)
[09/jun./2024 17:02:29,626] - Broken pipe from ('127.0.0.1', 53508)
[09/jun./2024 17:02:29,626] - Broken pipe from ('127.0.0.1', 53509)
[09/jun./2024 17:02:29,626] - Broken pipe from ('127.0.0.1', 53507)
.
DevTools listening on ws://127.0.0.1:53518/devtools/browser/612d1517-cc2e-4a38-9aac-e0d84c18a768
[09/jun./2024 17:02:42,242] - Broken pipe from ('127.0.0.1', 53528)
[09/jun./2024 17:02:42,242] - Broken pipe from ('127.0.0.1', 53529)
[09/jun./2024 17:02:42,242] - Broken pipe from ('127.0.0.1', 53536)
.
-----
Ran 11 tests in 198.520s
OK
Destroying test database for alias 'default'...
Destroying test database for alias 'dj30'...
Destroying test database for alias 'ibex35'...
Destroying test database for alias 'ftse100'...
Destroying test database for alias 'dax40'...
```

Figura D.9: Resultado de pruebas de interfaz

Apéndice E

Documentación de usuario

- E.1. Introducción
- E.2. Requisitos de usuarios
- E.3. Instalación
- E.4. Manual del usuario

Apéndice F

Anexo de sostenibilización curricular

F.1. Introducción

Este anexo incluirá una reflexión personal del alumnado sobre los aspectos de la sostenibilidad que se abordan en el trabajo. Se pueden incluir tantas subsecciones como sean necesarias con la intención de explicar las competencias de sostenibilidad adquiridas durante el alumnado y aplicadas al Trabajo de Fin de Grado.

Más información en el documento de la CRUE https://www.crue.org/wp-content/uploads/2020/02/Directrices_Sostenibilidad_Crue2012.pdf.

Este anexo tendrá una extensión comprendida entre 600 y 800 palabras.

Bibliografía

- [1] IEEE Recommended Practice for Software Requirements Specifications. *IEEE Std 830-1998*, pages 1–40, 1998.
- [2] Creatice commons. *Licencia CC BY-NC-SA 4.0*. <https://creativecommons.org/licenses/by-nc-sa/4.0/?ref=chooser-v1>, 2024. Online; Accedido el 04-Jun-2024.
- [3] NetworkX developers. *NetworkX*. <https://networkx.org/>, 2024. Online; Accedido el 22-Abr-2024.
- [4] Django. *Django documentation*. <https://docs.djangoproject.com/en/5.0/faq/general/>, 2023. Online; Accedido el 20-Abr-2024.
- [5] Pivit Inc. *Zube*. <https://zube.io/>, 2024. Online; Accedido el 22-Abr-2024.
- [6] Wikipedia. *cron (Unix)*. [https://es.wikipedia.org/wiki/Cron_\(Unix\)](https://es.wikipedia.org/wiki/Cron_(Unix)), 2024. Online; Accedido el 04-Jun-2024.
- [7] Wikipedia. *Scrum (desarrollo de software)*. [https://es.wikipedia.org/wiki/Scrum_\(desarrollo_de_software\)](https://es.wikipedia.org/wiki/Scrum_(desarrollo_de_software)), 2024. Online; Accedido el 20-Abr-2024.



Esta obra está bajo una licencia Creative Commons
Reconocimiento-NoComercial-CompartirIgual 4.0 Internacional
([CC BY-NC-SA 4.0 DEED](https://creativecommons.org/licenses/by-nc-sa/4.0/deed.es)).