UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

FAT

Financial Analysis Tool. Herramienta de análisis financiero.

Presentado por Rodrigo Merino Tovar
en Universidad de Burgos — 22 de abril de 2024
Tutores: Dra. Virginia Ahedo García y Dr. José Ignacio Santos Martín

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

Dña. Virginia Ahedo García y D. José Ignacio Santos Martín, profesores del departamento de Ingeniería de Organización, área de Organización de Empresas.

Exponen:

Que el alumno D. Rodrigo Merino Tovar, con DNI 71286910C, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado "FAT: Financial Analysis Tool. Herramienta de análisis financiero.".

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección de los que suscriben, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 22 de abril de 2024

Vº. Bº. del Tutor:	Vº. Bº. del co-tutor:
Dña. Virginia Ahedo García	D. José Ignacio Santos Martín

Resumen

Invertir nuestro capital para obtener una rentabilidad a cambio puede ser un proceso costoso si no se dispone de un acceso sencillo a la información necesaria. Además, deberemos de tener claro cuáles son el objetivo y el horizonte de nuestra inversión, así como el riesgo que estamos dispuestos a asumir y si la información de la que disponemos es suficiente.

Para abordar algunos de estos retos, este trabajo propone una herramienta digital que recopila información técnica y fundamental de empresas y sus productos cotizados, así como noticias relacionadas con algunos de los índices de referencia más relevantes. Esta herramienta presenta datos de forma agregada y permite realizar la comparación evolutiva - de precios de cierre de mercado - con el sector de referencia de una empresa cotizada o entre diferentes valores de distintos mercados.

Adicionalmente, en este trabajo se aporta una visión diferente a las webs de mercados financieros, permitiendo hacer uso de algunas utilidades poco comunes, como son la opción de hacer un análisis gráfico de correlación entre las cotizaciones de cuatro bolsas mundiales o la experimentación con predicción automática sobre series temporales, a través de *ARIMA* o redes *LSTM*.

Por otro lado, esta herramienta permite llevar el control de una cartera de inversión - realizando cambio de divisa automático a euros en los casos necesarios - y disponer de una lista de seguimiento, con precios reales de cierre diario.

La herramienta se encuentra disponible en `http://takeiteasy.pythonanywhere.com/` o, para ser utilizada de forma local, en `https://github.com/rmt0009alu/FAT`.

Descriptores

Django, SQLite, análisis financiero, noticias y RSS, seguimiento de cartera, *forecasting* de series temporales.

Abstract

Investing our capital to obtain a return can be a costly process if we don't have easy access to the necessary information. Additionally, we must be clear about the target and horizon of our investment, as well as the risk we are willing to assume and whether the information we have is sufficient.

To address some of these challenges, this work proposes a digital tool that collects technical and fundamental information about companies and their listed products, as well as news related to some of the most relevant indices. This tool presents data in an aggregated manner and allows for comparative analysis - using market closing prices - with the reference sector of a listed company or between different stocks from different markets.

Additionally, this work offers a different perspective on financial market websites, allowing the use of some uncommon utilities, such as the option to perform graphical correlation analysis between the stock quotes from four world stock exchanges, or experimentation with automatic prediction on time series, through ARIMA or LSTM networks.

On the other hand, this tool allows for the control of an investment portfolio - including automatic currency conversion to euros when necessary - and a watchlist with real closing prices on a daily basis.

The tool is available at http://takeiteasy.pythonanywhere.com/ or, for local use, at https://github.com/rmt0009alu/FAT.

Keywords

Django, SQLite, financial analysis, news and RSS, investment portfolio, time series *forecasting*.

Índice general

Índice de figuras

Índice de tablas

1. Introducción

Los mercados financieros juegan un papel fundamental en la economía global. Las empresas y Administraciones Públicas que necesitan financiarse acuden a estos mercados en busca de capital proveniente de ahorradores, que esperan obtener un rendimiento sobre el dinero aportado.

Los ahorradores prestan su dinero, depositando su confianza en una empresa, a través de la compra de acciones, bonos, pagarés y obligaciones - o productos derivados, así como materias primas -. Y para estos inversores, la toma de decisiones informada debe de ser la base principal de su estrategia de negocio.

Siguiendo las valiosas recomendaciones de Benjamin Graham [27] podremos invertir de forma sensata, realizando un análisis minucioso, basado en unos principios subyacentes que no se van a modificar sustancialmente con el paso del tiempo, pero que sí requieren de una constante actualización de la información sobre el entorno de las empresas y los mercados en los que cotizan. Por ello, en este trabajo se hace uso de la información disponible para ayudar a los inversores a formar una cartera bien diversificada, teniendo en cuenta diferentes divisas y mercados; y se aportan algunas herramientas de análisis poco frecuentes en otras plataformas web [8, 33, 38], como puede ser el análisis visual de correlaciones entre valores o la comparación gráfica con el sector de referencia, entre otros.

Por otro lado, según el Plan de Educación Financiera 2022-2025 [15] de la CNMV [16] y del Banco de España [14], existe un consenso generalizado sobre la necesidad de mejorar el nivel de cultura financiera, independientemente del país y las circunstancias de los ciudadanos, por lo que, de manera experimental, se introduce al usuario en la utilización de modelos para análisis de series temporales, con la intención de aportar herramientas

adicionales a su *backup* financiero. Concretamente, se da acceso al uso de modelos ARIMA [61] y de redes LSTM [29].

1.1. Estructura de la memoria

La memoria sigue la siguiente estructura:

- **Introducción:** breve descripción del problema a resolver y la solución propuesta. Estructura de la memoria y listado de materiales adjuntos.
- **Objetivos del proyecto:** exposición de los objetivos que persigue el proyecto.
- **Conceptos teóricos:** breve explicación de los conceptos teóricos clave para la comprensión de la solución propuesta.
- **Técnicas y herramientas:** listado de técnicas metodológicas y herramientas utilizadas para gestión y desarrollo del proyecto.
- **Aspectos relevantes del desarrollo:** exposición de aspectos destacables que tuvieron lugar durante la realización del proyecto.
- **Trabajos relacionados:** estado del arte en las aplicaciones y sitios web de bolsa y finanzas.
- **Conclusiones y líneas de trabajo futuras:** conclusiones obtenidas tras la realización del proyecto y posibilidades de mejora o expansión de la solución aportada.

Junto a la memoria se proporcionan los siguientes anexos:

- **Plan del proyecto software:** planificación temporal y estudio de viabilidad del proyecto.
- **Especificación de requisitos del software:** se describe la fase de análisis; los objetivos generales, el catálogo de requisitos del sistema y la especificación de requisitos funcionales y no funcionales.
- **Especificación de diseño:** se describe la fase de diseño; el ámbito del software, el diseño de datos, el diseño procedimental y el diseño arquitectónico.
- **Manual del programador:** recoge los aspectos más relevantes relacionados con el código fuente (estructura, compilación, instalación, ejecución, pruebas, etc.).
- **Manual de usuario:** guía de usuario para el correcto manejo de la aplicación.

1.2. Materiales adjuntos

Los materiales que se adjuntan con la memoria son:

- **Herramienta web FAT**: Financial Analysis Tool.
- *Dataset* de **vídeos de prueba**.

Además, los siguientes recursos están accesibles a través de internet:

- **Página web** del proyecto [50].
- **Repositorio** del proyecto [51].

2. Objetivos del proyecto

A continuación se detallan los objetivos que se persiguen con la realización de este proyecto:

2.1. Objetivos generales

- Desarrollar una aplicación *web* que permita a un usuario la composición de una cartera de valores cotizados bien diversificada.
- Ofrecer información agregada sobre la evolución de un valor y su sector de referencia.
- Permitir la comparación relativa entre valores cotizados.
- Aportar valor añadido a través del análisis de correlaciones entre valores.
- Facilitar la interpretación de los datos recogidos mediante representaciones gráficas.
- Dar acceso a información extra mediante el análisis de series temporales con modelos y redes neuronales.

2.2. Objetivos de carácter técnico

- Desarrollar una plataforma *web* con *Django* que permita realizar el seguimiento de valores cotizados en algunos de los principales índices de referencia mundiales.
- Crear bases de datos *SQLite* cuya actualización sea automática a través de un administrador de procesos como *cron* en un servidor *web* remoto o de forma semiautomática en un servidor local.

- Aplicar la arquitectura MVC (*Model-View-Controler*), más conocida en *Django* como MVT (*Model-View-Template*).
- Diseñar formularios que permitan la interacción con el usuario para realizar operaciones *CRUD* en la base de datos principal y operaciones de lectura en las bases de datos de los valores cotizados.
- Utilizar Git como sistema de control de versiones distribuido junto con la plataforma GitHub.
- Hacer uso de herramientas CI/CD integradas en el repositorio con *GitHub actions*. Por ejemplo, utilizar *pyllint* como herramienta de control de calidad de código, o *coverage* para testear de forma continuada el desarrollo del proyecto.
- Aplicar la metodología ágil Scrum junto con TDD (*Test Driven Development*) en los apartados que sea posible a lo largo del desarrollo del software.
- Realizar test unitarios, de integración y de interfaz.
- Utilizar Zube como herramienta de gestión de proyectos.
- Utilizar un sistema de documentación como Sphinx con el estilo de Read The Docs y con la posibilidad de subir la documentación de forma continua.

2.3. Objetivos personales

- Crear una herramienta sencilla - y no habitual - para el público general en el ecosistema de las *webs* de los mercados de valores.
- Abarcar el máximo número posible de conocimientos adquiridos durante el grado.
- Explorar metodologías, herramientas y estándares utilizados en el mercado laboral.
- Introducirme en el mundo del análisis y el *forecasting* de series temporales de datos.

3. Conceptos teóricos

En aquellos proyectos que necesiten para su comprensión y desarrollo de unos conceptos teóricos de una determinada materia o de un determinado dominio de conocimiento, debe existir un apartado que sintetice dichos conceptos.

Algunos conceptos teóricos de LaTeX [1].

3.1. Secciones

Las secciones se incluyen con el comando section.

Subsecciones

Además de secciones tenemos subsecciones.

Subsubsecciones

Y subsecciones.

3.2. Referencias

Las referencias se incluyen en el texto usando cite [**?**]. Para citar webs, artículos o libros [**?**], si se desean citar más de uno en el mismo lugar [**?** **?**].

[1] Créditos a los proyectos de Álvaro López Cantero: Configurador de Presupuestos y Roberto Izquierdo Amo: PLQuiz

3.3. Imágenes

Se pueden incluir imágenes con los comandos standard de LaTeX, pero esta plantilla dispone de comandos propios como por ejemplo el siguiente:

Figura 3.1: Autómata para una expresión vacía

3.4. Listas de items

Existen tres posibilidades:

- primer item.
- segundo item.

1. primer item.
2. segundo item.

Primer item más información sobre el primer item.

Segundo item más información sobre el segundo item.

-

Herramientas	App AngularJS	API REST	BD	Memoria
HTML5	X			
CSS3	X			
BOOTSTRAP	X			
JavaScript	X			
AngularJS	X			
Bower	X			
PHP		X		
Karma + Jasmine	X			
Slim framework		X		
Idiorm		X		
Composer		X		
JSON	X	X		
PhpStorm	X	X		
MySQL			X	
PhpMyAdmin			X	
Git + BitBucket	X	X	X	X
MikTEX				X
TEXMaker				X
Astah				X
Balsamiq Mockups	X			
VersionOne	X	X	X	X

Tabla 3.1: Herramientas y tecnologías utilizadas en cada parte del proyecto

3.5. Tablas

Igualmente se pueden usar los comandos específicos de LATEXo bien usar alguno de los comandos de la plantilla.

4. Técnicas y herramientas

4.1. Técnicas metodológicas

Scrum

Scrum [63] es un marco de trabajo relativamente estructurado y con roles específicos dentro de la metodología Agile (roles principales: *Product Owner*, *Scrum Master* y desarrollador) . Se puede utilizar tanto para la gestión de proyectos como para el desarrollo de productos, especialmente en el despliegue de *software*.

Con *Scrum* los proyectos se dividen en iteraciones cortas llamadas *sprints*. Al final de cada *sprint* se debe presentar un producto mínimo viable y evaluar lo que se ha hecho bien y lo que se puede mejorar.

Se ha optado por esta metodología, frente a otras como *Waterfall* [55] , porque ofrece una alta adaptabilidad y genera entrega temprana de valor, con productos viables y valorables por el usuario final desde las primeras fases.

Test-Driven Development (TDD)

TDD [56] es una metodología de desarrollo de software que se enfoca en escribir una batería de tests automatizados antes de iniciar la implementación del código fuente del propio software. Posteriormente, se hace un proceso de refactorización para mejorar o solucionar los defectos encontrados.

Mis conocimientos previos de *Django* y *SQLite* no me han permitido utilizar de forma integral esta metodología, pero sí que se ha seguido en

diferentes etapas del desarrollo, mejorando notablemente la calidad del código final.

***Behavior-Driven Development* (BDD)**

BDD [53] es una metodología que se basa en el comportamiento del software y me ha resultado útil en aquellas fases del proyecto en las que no tenía una idea preconcebida del cómo trabajar con *Django* pero sí que conocía el resultado final esperado.

La ventaja de este enfoque es que las pruebas se escriben en un lenguaje natural y es sencillo extrapolarlas a un gestor de tareas con un sistema Kanban.

Kanban

Kanban [59] es un método visual de gestión de proyectos a través de la utilización de un tablero, en el que se disponen una serie de tarjetas con las tareas pendientes, en curso o finalizadas. Esto permite crear un flujo de trabajo que prioriza aquellas tareas más urgentes o que aportan antes valor a un producto.

4.2. Patrones de diseño

***Model-View-Template* (MVT)**

Es el patrón de diseño de *Django*, Modelo-Vista-Plantilla [19, 42], que es similar al Modelo-Vista-Controlador (MVC) [62]. En *Django*, el Modelo representa la estructura de los datos, la Vista maneja la lógica de la aplicación (el controlador en MVC) y la Plantilla se encarga de la presentación de los datos (la vista en MVC).

Una de las ventajas de *Django* es que este modelo está plenamente integrado y promueve un acoplamiento débil, lo que facilita el mantenimiento y la escalabilidad de una aplicación.

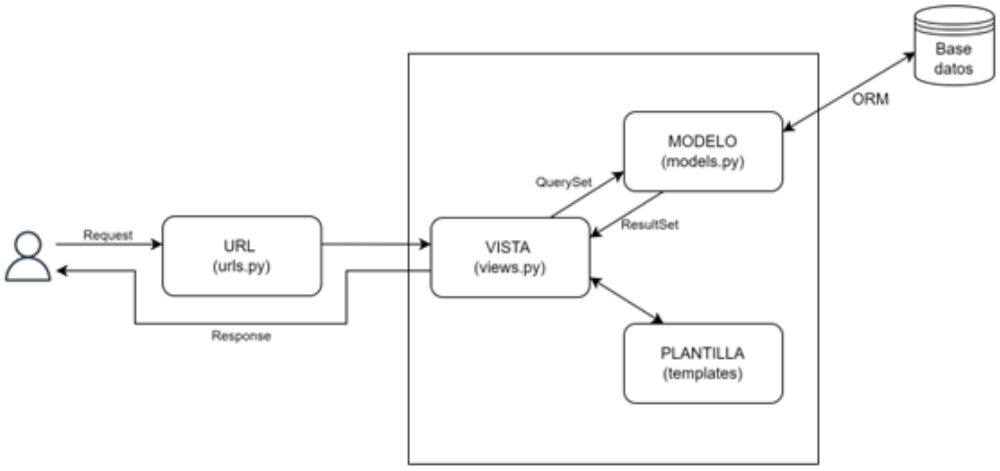

Figura 4.1: Patrón MVT. Fuente: realización propia

4.3. Control de versiones

- Herramientas consideradas: Git [22], Apache Subversion [1] y Mercurial [12].
- Herramienta elegida: Git.

Git y Mercurial son sistemas de control de versiones distribuidos (DVCS), mientras que Subversion - o SVN - es centralizado (VCS).

Una de las ventajas de Git es que permite a cada desarrollador tener una copia en local del repositorio completo y, aunque es menos eficiente para proyectos muy grandes, es más sencillo de utilizar para proyectos pequeños. Además, el sistema de ramificación de Git es más intuitivo y facilita la tarea de los desarrolladores.

4.4. Alojamiento del repositorio

- Herramientas consideradas: GitHub [24], GitLab [26] y Gitea [23].
- Herramienta elegida: GitHub.

Me he decantado por GitHub porque ya lo conocía, porque se utiliza en algunas asignaturas del Grado de Ingeniería Informática y porque es muy popular, lo que facilita la resolución de problemas gracias a su mayor comunidad.

GitHub puede ofrecer menor control sobre proyectos grandes - Gitea y GitLab permiten auto hospedaje con la configuración que más nos interese -, pero en proyectos medios o pequeños es una herramienta práctica y sencilla de utilizar, con diferentes integraciones y que facilita el uso de flujos de trabajo CI/CD.

4.5. Gestión del proyecto

- Herramientas consideradas: Zube [32], ZenHub [67], Trello [5] y Jira [4].
- Herramienta elegida: Zube.

Zube es una plataforma de gestión de proyectos que se integra muy bien con GitHub. Además, permite la sincronización en tiempo real con el repositorio de referencia que se esté utilizando y ofrece una interfaz fácil de utilizar con posibilidad de seguimiento a través de *burndonws* , *burnups* y *throughput* del equipo de desarrollo o de los desarrolladores de forma individual.

Frente a las alternativas valoradas, Zube ha sido la más intuitiva, permitiendo hacer seguimiento y planificación del proyecto en pocos pasos.

4.6. Comunicación

- Herramientas consideradas: email, GitHub y Microsoft Teams [39].
- Herramientas elegidas: todas las anteriores.

La comunicación en tiempo real, con llamadas o vídeo llamadas a través de Teams, aporta soluciones rápidas por el continuo flujo de preguntas-respuestas. Pero no siempre se pueden utilizar estos medios y es preferible hacer uso de email o de *requests* de *GitHub*. Además, recientemente, se ha dado la posibilidad de integrar MS Teams con GitHub [41] para enviar notificaciones a un grupo de trabajo.

4.7. Entorno de desarrollo integrado (IDE)

- Herramientas consideradas: Spyder IDE [30], Visual Studio Code [40].
- Herramientas elegidas: Visual Studio Code.

A pesar de que ambos entornos tienen plugins de alta calidad, VS Code ofrece mayor personalización. Además, VS Code es integrable con *GitHub* de forma sencilla y ofrece aplicaciones de terceros que facilitan tanto la implementación de código como las labores de testeo y control de calidad.

VS Code se ha utilizado en este trabajo para desarrollo de *Django*, como editor *CSS* y *HTML*, para *JavaScript* y para *Markdown*, así como medio de integracción con *GitHub*, entre otros.

4.8. Documentación de la memoria

- Herramientas consideradas: Texmaker [10] y TeXstudio [52].
- Herramientas elegidas: Texmaker.

Texmaker es un editor de texto gratuito, multiplataforma y que integra diversas herramientas necesarias para desarrollar documentos LaTeX. *Texmaker* incluye soporte *Unicode*, corrección ortográfica, auto-completado y un visor de PDF incorporado que es realmente útil.

Adicionalmente, cabe destacar que la integración de *Texmaker* con la distribución de TeX/ LaTeX*MikTex* [60] es menos problemática que con *TeXstudio*.

4.9. Documentación del código

- Herramientas consideradas: Sphinx [6] y pdoc [44].
- Herramientas elegidas: Sphinx.

Sphinx es la herramienta más extendida en la comunidad *Python* para documentar código, es compatible con varios formatos y estilos de *docstrings* - *PyDoc*, *Google* o *Numpy* entre otros - y, además, puede generar documentación de manera automática a partir de los *docstrings*.

En este trabajo, se ha escogido el formato de *Numpy* para la documentación de código y el estilo de *Read The Docs* para las plantillas de la documentación HTML.

4.10. Integración y despliegue continuos (CI/CD)

- Herramientas consideradas: GitHub actions [25], Jenkins [34] y CircleCI [11].
- Herramienta elegida: GitHub actions.

GitHub actions es una plataforma de CI/CD que permite automatizar el proceso de compilación, pruebas y despliegue de software. En este trabajo, GitHub actions se ha utilizado para tetear y comprobar la calidad del código a través de flujo de trabajo que se activan con cada evento de *push* al *branch main* del repositorio de GitHub.

4.11. Calidad y consistencia de código

- Herramientas consideradas: Pylint [36] y Flake8 [13].
- Herramienta elegida: Pylint.

Pylint es una herramienta de análisis de código estático para *Python*, diseñada para detectar errores y mejorar la calidad del código. Este analizaor de código se utilizar para verificar sintaxis, semántica y obliga a seguir las convenciones de estilo de *Python*.

Se ha escogido *Pylint* frente a *Flake8* porque, adicionalmente, permite medir la calidad del código en términos de complejidad y legibilidad, lo que favorece un mantenimiento posterior. Además, con *Pylint* podemos hacer informes que señalan todos los fallos, - aumentando la productividad - y es posible integrarlo con *GitHub actions*, como se ha hecho en este proyecto.

4.12. Cobertura de código

- Herramientas consideradas: Coverage [7] y Pytest-cov [46].
- Herramienta elegida: Coverage.

Coverage es una herramienta que permite medir la cobertura de código en *Python*. Tiene la ventaja de que está bien integrada con proyectos de *Django* y permite reconocer los *django.test.TestCase*. Además, se puede incorporar a *GitHub actions*, tal y como se ha realizado en este proyecto, a través de un documento *YAML*.

Uno de los aspectos relevantes de *coverage* es que, con pocos comandos, permite generar un informe HTML muy intuitivo que guía al desarrollador hacia los fallos detectados.

4.13. Framework web

- Herramientas consideradas: Django [21] y Flask [43].
- Herramienta elegida: Django.

Django es un *framework web* muy completo que incluye diversas características por defecto. También adopta un elevado nivel de seguridad y es altamente escalable. Es menos ligero que *Flask* pero, a cambio, ofrece mayor nivel de personalización y control sobre el sitio web, así como una mejor estructuración general de un proyecto.

4.14. Sistema gestor de bases de datos

- Herramientas consideradas: SQLite [48] y PotgreSQL [28].
- Herramienta elegida: SQLite.

SQLite es una librería de código escrita en lenguaje C, que implementa un motor de bases de datos pequeño y rápido. Se califican a sí mismos como un sistema gestor de bases de datos ligero y multiplataforma. Tiene la ventaja de estar muy extendido y utilizan un único archivo en el sistema de almacenamiento, lo que favorece su distribución y uso.

Por su parte, *PostgreSQL* tiene opciones más avanzadas que *SQLite* y está pensado para soportar alta concurrencia y bases de datos grandes. Pero *SQLite* está perfectamente integrado con *Django* y no requiere de configuración adicional como sí requeriría *PostgreSQL*. Además, *SQLite* es suficiente para las expectativas de este trabajo - la migración a *PostgreSQL* sería recomendable en caso de escalar el proyecto -.

4.15. Bibliotecas y librerías relevantes

En este apartado se indican las bibliotecas y librerías más relevantes dentro del proyecto. Hay otras muchas que forman parte del conjunto de dependencias y que se pueden consultar en el archivo de *requirements*.

Pandas

Pandas [31] es una biblioteca de código abierto para *Python* especializada en análisis de datos. Es útil para cargar datos desde diversas fuentes - como una base de datos o una API -, permite tratar los datos y transformarlos o incluso crear visualizaciones.

Plotly

Plotly [45] es una biblioteca de código abierto que se utiliza para crear visualizaciones de datos interactivas en *Python*. Permite crear gráficos de barras o líneas, entre otros, y permite dar una alta personalización a la información que recibe el usuario final.

Matplotlib

Matplotlib [18] es una librería para crear visualizaciones estáticas, animadas e interactivas en *Python*. Es una herramienta de código abierto, multiplataforma y está orientada a objetos.

La curva de aprendizaje de *Matplotlib* puede ser más compleja que en otras herramientas similares, pero ofrece mayor control sobre los datos y la forma de representarlos, especialmente cuando se utiliza para gráficos estadísticos.

yFinance

yFinance [3] es una biblioteca de código abierto para *Python* que permite el acceso y procesado de datos financieros. Permite la recuperación de datos históricos y en tiempo real de cotizaciones de acciones, índices bursátiles, divisas, criptomonedas y otros instrumentos financieros. En este trabajo se utiliza para recuperar datos de valores cotizados y sus divisas de referencia - siempre con precios de cierre de mercado -.

Entre las ventajas que ofrece es que su uso está bastante extendido y tiene una comunidad que facilita la resolución de problemas. Además, es fácil de utilizar y dispone de múltiples ejemplos en su documentación que permiten que la curva de aprendizaje sea progresiva.

News API

News API [2] es una biblioteca de código abierto, de *Python*, que facilita el acceso y la integración de noticias en una aplicación. Es un servicio web

que proporciona una amplia variedad de artículos y noticias actualizadas, ordenadas por diferentes categorías, países y fuentes de información.

Feedparser

Feedparser [66] es una librería que permite analizar y procesar *feeds RSS* y *Atom*. En este trabajo se utiliza para obtener información relativa a índices bursátiles desde enlaces *RSS*. Los *feeds* de los datos se proveen con archivos en formato *XML* y proceden de fuentes relacionadas con mercados financieros nacionales e internacionales (en todos los casos se utilizan fuentes conocidas y contrastadas).

NetworkX

NetworkX [17] es una biblioteca que se utiliza para el estudio y análisis de redes complejas. Entre sus funciones principales se pueden encontrar las de crear, manipular y analizar grafos. Estos grafos son estructuras matemáticas que modelan relaciones entre objetos y, en el caso de este trabajo, se utiliza para visualizar las mayores correlaciones - positiva y negativa - entre las cotizaciones de todos los valores de los índices estudiados.

Statsmodels

Statsmodels [35] es una biblioteca de *Python* que se utiliza para el análisis y modelado estadístico de datos. Permite explorar, estimar y evaluar modelos estadísticos complejos; y ofrece herramientas para el análisis de series temporales. En este proyecto se utiliza para configurar y aplicar modelos ARIMA.

Scikit-learn

Scikit-learn [47] es una bilioteca de código abierto para el aprendizaje automático en *Python*. Se utiliza frecuentemente para clasificación de datos, regresión, agrupación y reducción de dimensionalidad. Sin embargo, en este trabajo se utiliza sólo para calcular el *MSE* (Mean Squared Error) entre datos de test y predicciones de modelos y para normalizar datos en una escala concreta.

Keras

Keras [65] es una biblioteca de código abierto para aprendizaje profundo escrita en *Python* - en este proyecto se ejecuta sobre *TensorFlow* [49] -. *Keras* sirve para crear y entrenar redes neuronales y para experimentar con diferentes arquitecturas de estas redes. Su modularidad permite crear redes muy complejas y, además, es muy eficiente tanto en entrenamiento como en inferencia. En este trabajo se utiliza para crear redes *LSTM* y realizar análisis de series temporales.

4.16. Desarrollo web

HTML

HTML [57] es el lenguaje de marcado de hipertexto estándar para crear páginas web. Es un lenguaje que utiliza etiquetas para definir la esrtructura y el contenido de una página web.

CSS

CSS [54] es un lenguaje de hojas de estilo que se utiliza para dar un aspecto y diseño agradables a una página web. Se utiliza junto con *HTML*.

Bootstrap

Bootstrap [9] es un *framewrok* de código abierto para el desarrollo web *front-end*. Proporciona un conjunto de herramientas y componentes predefinidos que permiten a los desarrolladores crear plantillas e interfaces atractivas y responsivas.

JavaScript

JavaScript [58] es un lenguaje de programación interpretado, orientado a objetos y débilmente tipado. En este trabajo se utiliza para desarrollo web *front-end* para agregar interactividad y dinamismo a la página web.

4.17. Otras herramientas

python-dotenv

python-dotenv [64] es una biblioteca que permite gestionar variables de entorno para aplicaciones *Python*. Se puede utilizar, entre otros, para securizar las *API keys* de *Django* y *News API*.

Draw.io

Draw.io [37] es una herramienta de diseño de diagramas y esquemas que incluye una amplia variedad de formas predefinidas. Permite exportar el resultado en varios formatos, lo que facilita la integración con diferentes editores de texto.

Mendeley

Mendeley [20] es una herramienta de gestión de referecnias bibliográficas y colaboración académica. Entre sus características cabe destacar la capacidad de organizar las referencias y exportarlas a un archivo que se puede utilizar desde LaTeX.

5. Aspectos relevantes del desarrollo del proyecto

Este apartado pretende recoger los aspectos más interesantes del desarrollo del proyecto, comentados por los autores del mismo. Debe incluir desde la exposición del ciclo de vida utilizado, hasta los detalles de mayor relevancia de las fases de análisis, diseño e implementación. Se busca que no sea una mera operación de copiar y pegar diagramas y extractos del código fuente, sino que realmente se justifiquen los caminos de solución que se han tomado, especialmente aquellos que no sean triviales. Puede ser el lugar más adecuado para documentar los aspectos más interesantes del diseño y de la implementación, con un mayor hincapié en aspectos tales como el tipo de arquitectura elegido, los índices de las tablas de la base de datos, normalización y desnormalización, distribución en ficheros3, reglas de negocio dentro de las bases de datos (EDVHV GH GDWRV DFWLYDV), aspectos de desarrollo relacionados con el WWW... Este apartado, debe convertirse en el resumen de la experiencia práctica del proyecto, y por sí mismo justifica que la memoria se convierta en un documento útil, fuente de referencia para los autores, los tutores y futuros alumnos.

6. Trabajos relacionados

Este apartado sería parecido a un estado del arte de una tesis o tesina. En un trabajo final grado no parece obligada su presencia, aunque se puede dejar a juicio del tutor el incluir un pequeño resumen comentado de los trabajos y proyectos ya realizados en el campo del proyecto en curso.

7. Conclusiones y Líneas de trabajo futuras

Todo proyecto debe incluir las conclusiones que se derivan de su desarrollo. Éstas pueden ser de diferente índole, dependiendo de la tipología del proyecto, pero normalmente van a estar presentes un conjunto de conclusiones relacionadas con los resultados del proyecto y un conjunto de conclusiones técnicas. Además, resulta muy útil realizar un informe crítico indicando cómo se puede mejorar el proyecto, o cómo se puede continuar trabajando en la línea del proyecto realizado.

Bibliografía

[1] Apache. *Apache Subversion*. https://subversion.apache.org/, 2024. Online; Accedido el 20-Abr-2024.

[2] News API. *News API*. https://newsapi.org/docs/client-libraries/python, 2024. Online; Accedido el 22-Abr-2024.

[3] Ran Aroussi. *yFinance*. https://pypi.org/project/yfinance/, 2024. Online; Accedido el 22-Abr-2024.

[4] Atlassian. *Jira*. https://www.atlassian.com/es/software/jira, 2024. Online; Accedido el 22-Abr-2024.

[5] Atlassian. *Trello*. https://trello.com/es, 2024. Online; Accedido el 22-Abr-2024.

[6] Sphinx Authors. *Sphinx*. https://www.sphinx-doc.org/en/master/, 2024. Online; Accedido el 22-Abr-2024.

[7] Ned Batchelder. *Coverage*. https://coverage.readthedocs.io/en/7.4.4/, 2024. Online; Accedido el 22-Abr-2024.

[8] Bloomberg. *Global media company that covers business, finance, markets, politics and more*. https://www.bloomberg.com/, 2024. Online; Accedido el 10-Abr-2024.

[9] Bootstrap. *Bootstrap*. https://getbootstrap.com/docs/5.3/getting-started/introduction/, 2024. Online; Accedido el 22-Abr-2024.

[10] Pascal Brachet. *Texmaker*. https://www.xm1math.net/texmaker/, 2024. Online; Accedido el 22-Abr-2024.

[11] CircleCI. *CircleCI*. https://circleci.com/docs/language-python/, 2024. Online; Accedido el 22-Abr-2024.

[12] Mercurial community. *Mercurial*. https://www.mercurial-scm.org/, 2024. Online; Accedido el 20-Abr-2024.

[13] Ian Stapleton Cordasco. *Flake8*. https://flake8.pycqa.org/en/latest/, 2024. Online; Accedido el 22-Abr-2024.

[14] Banco de España. *Página oficial del Banco de España*. https://www.bde.es/wbe/es/, 2023. Online; Accedido el 06-Nov-2023.

[15] Comisión Nacional del Mercado de Valores. *Plan de Educación Financiera 2022-2025*. https://www.cnmv.es/docportal/publicaciones/planeducacion/planeducacionfinanciera_22_25es.pdf, 2022. Online; Accedido el 06-Nov-2023.

[16] Comisión Nacional del Mercado de Valores. *Página oficial de la Comisión Nacional del Mercado de Valores*. https://www.cnmv.es/portal/home.aspx, 2023. Online; Accedido el 06-Nov-2023.

[17] NetworkX developers. *NetworkX*. https://networkx.org/, 2024. Online; Accedido el 22-Abr-2024.

[18] The Matplotlib development team. *Matplotlib*. https://matplotlib.org/, 2024. Online; Accedido el 22-Abr-2024.

[19] Django. *Django documentation*. https://docs.djangoproject.com/en/5.0/faq/general/, 2023. Online; Accedido el 20-Abr-2024.

[20] Elsevier. *Draw.io*. https://www.mendeley.com/, 2024. Online; Accedido el 22-Abr-2024.

[21] Django Software Foundation. *Django*. https://www.djangoproject.com/, 2024. Online; Accedido el 22-Abr-2024.

[22] Git. *Git*. https://git-scm.com/, 2024. Online; Accedido el 20-Abr-2024.

[23] Gitea. *Gitea*. https://about.gitea.com/, 2024. Online; Accedido el 20-Abr-2024.

[24] GitHub. *GitHub*. https://github.com/, 2024. Online; Accedido el 20-Abr-2024.

[25] GitHub. *GitHub actions*. https://docs.github.com/en/actions, 2024. Online; Accedido el 22-Abr-2024.

[26] GitLab. *GitLab*. https://about.gitlab.com/, 2024. Online; Accedido el 20-Abr-2024.

[27] Benjamin Graham. *El inversor inteligente*. Deusto, Grupo Planeta, 10th edition, 2007.

[28] The PostgreSQL Global Development Group. *PostgreSQL*. https://www.postgresql.org/, 2024. Online; Accedido el 22-Abr-2024.

[29] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, nov 1997.

[30] Spyder IDE. *Spyder IDE*. https://www.spyder-ide.org/, 2024. Online; Accedido el 22-Abr-2024.

[31] Num FOCUS Inc. *Pandas*. https://pandas.pydata.org/docs/, 2024. Online; Accedido el 22-Abr-2024.

[32] Pivit Inc. *Zube*. https://zube.io/, 2024. Online; Accedido el 22-Abr-2024.

[33] Investing. *Website that provides real-time data, news, analysis and tools for global financial markets*. https://www.investing.com/, 2024. Online; Accedido el 10-Abr-2024.

[34] Jenkins. *Jenkins*. https://www.jenkins.io/solutions/python/, 2024. Online; Accedido el 22-Abr-2024.

[35] Jonathan Taylor y statsmodels-developers Josef Perktold, Skipper Seabold. *statsmodels*. https://www.statsmodels.org/stable/index.html, 2024. Online; Accedido el 22-Abr-2024.

[36] Pylint Logilab. *Pylint*. https://www.pylint.org/, 2024. Online; Accedido el 22-Abr-2024.

[37] JGraph Ltd. *Draw.io*. https://www.drawio.com/, 2024. Online; Accedido el 22-Abr-2024.

[38] MarketScreener. *Financial information for investors and traders*. https://www.marketscreener.com/, 2024. Online; Accedido el 10-Abr-2024.

[39] Microsoft. *MS Teams*. https://www.microsoft.com/es-es/microsoft-teams/group-chat-software, 2024. Online; Accedido el 20-Abr-2024.

[40] Microsoft. *Visual Studio Code*. https://code.visualstudio.com/, 2024. Online; Accedido el 22-Abr-2024.

[41] microsoft teams. *Integrar GitHub con MS Teams*. https://github.com/integrations/microsoft-teams, 2023. Online; Accedido el 20-Abr-2024.

[42] Vinayak Nishant. *Django MVT Architecture: A Fresh Take on Classic MVC*. https://www.askpython.com/django/django-mvt-architecture, 2020. Online; Accedido el 20-Abr-2024.

[43] Pallets. *Flask*. https://flask.palletsprojects.com/en/3.0.x/, 2024. Online; Accedido el 22-Abr-2024.

[44] pdoc Developers. *pdoc*. https://pdoc.dev/, 2024. Online; Accedido el 22-Abr-2024.

[45] plotly. *plotly*. https://plotly.com/python/, 2024. Online; Accedido el 22-Abr-2024.

[46] pytest-cov contributors. *Pytest-cov*. https://pytest-cov.readthedocs.io/en/latest/readme.html, 2024. Online; Accedido el 22-Abr-2024.

[47] scikit-learn developers. *scikit-learn*. https://scikit-learn.org/stable/index.html, 2024. Online; Accedido el 22-Abr-2024.

[48] SQLite. *SQLite*. https://www.sqlite.org/index.html, 2024. Online; Accedido el 22-Abr-2024.

[49] Google Brain Team. *TensorFlow*. https://www.tensorflow.org/?hl=es, 2024. Online; Accedido el 22-Abr-2024.

[50] Rodrigo Merino Tovar. *Herramienta web Financial Analysis Tool*. http://takeiteasy.pythonanywhere.com/, 2024. Online; Accedido el 10-Abr-2024.

[51] Rodrigo Merino Tovar. *Repositorio de GitHub del Trabajo de Fin de Grado Financial Analysis Tool*. https://github.com/rmt0009alu/FAT, 2024. Online; Accedido el 10-Abr-2024.

[52] Benito van der Zander. *Texstudio*. https://www.texstudio.org/, 2024. Online; Accedido el 22-Abr-2024.

[53] Wikipedia. *Behavior-driven development*. https://en.wikipedia.org/wiki/Behavior-driven_development, 2024. Online; Accedido el 20-Abr-2024.

[54] Wikipedia. *CSS*. https://es.wikipedia.org/wiki/CSS, 2024. Online; Accedido el 22-Abr-2024.

[55] Wikipedia. *Desarrollo en cascada*. https://es.wikipedia.org/wiki/Desarrollo_en_cascada, 2024. Online; Accedido el 22-Abr-2024.

[56] Wikipedia. *Desarrollo guiado por pruebas*. https://es.wikipedia.org/wiki/Desarrollo_guiado_por_pruebas, 2024. Online; Accedido el 20-Abr-2024.

[57] Wikipedia. *HTML*. https://es.wikipedia.org/wiki/HTML, 2024. Online; Accedido el 22-Abr-2024.

[58] Wikipedia. *JavaScript*. https://es.wikipedia.org/wiki/JavaScript, 2024. Online; Accedido el 22-Abr-2024.

[59] Wikipedia. *Kanban (desarrollo)*. https://es.wikipedia.org/wiki/Kanban_(desarrollo), 2024. Online; Accedido el 20-Abr-2024.

[60] Wikipedia. *MiKTeX*. https://es.wikipedia.org/wiki/MiKTeX, 2024. Online; Accedido el 22-Abr-2024.

[61] Wikipedia. *Modelo autorregresivo integrado de media móvil*. https://es.wikipedia.org/wiki/Modelo_autorregresivo_integrado_de_media_m%C3%B3vil, 2024. Online; Accedido el 10-Abr-2024.

[62] Wikipedia. *Modelo–vista–controlador*. https://es.wikipedia.org/wiki/Modelo%E2%80%93vista%E2%80%93controlador, 2024. Online; Accedido el 20-Abr-2024.

[63] Wikipedia. *Scrum (desarrollo de software)*. https://es.wikipedia.org/wiki/Scrum_(desarrollo_de_software), 2024. Online; Accedido el 20-Abr-2024.

[64] Saurabh Kumar y Bertrand Bonnefoy-Claudet. *python-dotenv*. https://pypi.org/project/python-dotenv/, 2024. Online; Accedido el 22-Abr-2024.

[65] François Chollet y desarrolladores de Google. *Keras*. https://keras.io/, 2024. Online; Accedido el 22-Abr-2024.

[66] Kurt McKee y Mark Pilgrim. *Feedparser*. https://feedparser.readthedocs.io/en/latest/introduction.html, 2024. Online; Accedido el 22-Abr-2024.

[67] ZenHub. *ZenHub*. https://www.zenhub.com/, 2024. Online; Accedido el 22-Abr-2024.