GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

Trader Companion: Plataforma Web para Aprender y Optimizar Estrategias de Trading Algorítmico

Autor: Álvaro Guerrero Gallego

Director: David Contreras Bárcena

Madrid

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

Trader Companion: Plataforma Web para Aprender y Optimizar Estrategias de Trading Algorítmico

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el

curso académico 2022/23 es de mi autoría, original e inédito y

no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido

tomada de otros documentos está debidamente referenciada.

Fdo.: Álvaro Guerrero Gallego Fecha: 30/ 06/ 2023

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: David Contreras Bárcena Fecha: ……4…./……julio……./ ……2023……

GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

Trader Companion: Plataforma Web para Aprender y Optimizar Estrategias de Trading Algorítmico

Autor: Álvaro Guerrero Gallego

Director: David Contreras Bárcena

Madrid

TRADER COMPANION: PLATAFORMA WEB PARA APRENDER Y OPTIMIZAR ESTRATEGIAS DE TRADING ALGORÍTMICO

Autor: Guerrero Gallego, Álvaro.
Director: Contreras Bárcena, David.
Entidad Colaboradora: ICAI – Universidad Pontificia Comillas

RESUMEN DEL PROYECTO

Este proyecto presenta el desarrollo de una plataforma web para backtesting de estrategias de inversión con un enfoque educativo. La herramienta permite a los usuarios diseñar, probar y optimizar estrategias automatizadas sin necesidad de conocimientos avanzados de programación y sirve de puente para aprender a programar sus propias estrategias.

Palabras clave: Finanzas, Aplicación Web, *Trading* Algorítmico, *Backtesting*

1. Introducción

En los últimos años, la erosión del poder adquisitivo y la elevada inflación han suscitado preocupación entre la población (Eurostat, 2022). Los mercados financieros y las instituciones tradicionales han sido la opción para proteger la riqueza (Georgarakos & Pasini, 2009). Sin embargo, en periodos de alta inflación, son necesarias estrategias más activas (Cagan & Lipsey, 1978).

El *trading* algorítmico ha surgido como una solución eficaz, ya que permite automatizar las operaciones y analizar grandes cantidades de datos (Chan, 2021). Aunque esta práctica es habitual entre los grandes fondos, los inversores particulares aún se enfrentan a barreras en términos de conocimientos y tecnología.

Para superar esta limitación, este proyecto propone el desarrollo de una plataforma web educativa centrada en el backtesting de estrategias de *trading* algorítmico. El backtesting permite probar estrategias de inversión utilizando datos históricos, lo que proporciona una comprensión más completa de su eficacia potencial.

La motivación de este trabajo radica en difundir conocimientos y aumentar la accesibilidad en finanzas cuantitativas y *trading* algorítmico. El objetivo es tender un puente entre los inversores comunes y las estrategias algorítmicas proporcionando una herramienta intuitiva para probar y evaluar las estrategias de inversión.

2. Definición del Proyecto

Entre los objetivos del proyecto figuran el desarrollo de una interfaz fácil de usar, la integración de datos financieros en tiempo real, la aplicación de estrategias predefinidas y la posibilidad de cargar estrategias personalizadas. Además, la plataforma pretende ofrecer funcionalidades para la optimización de estrategias.

La metodología adoptada para el desarrollo de este trabajo de fin de proyecto se basa en el análisis de requisitos, la investigación tecnológica, el diseño de la arquitectura, el desarrollo de la interfaz de usuario, la integración de datos financieros, la investigación e implementación de estrategias, las pruebas y la validación.

La justificación del proyecto se basa en tres puntos clave. En primer lugar, se ha detectado una clara necesidad en el ámbito del *trading* algorítmico: la demanda de soluciones accesibles para diseñar, probar y optimizar estrategias de negociación automatizadas. La plataforma desarrollada responde a esta necesidad proporcionando una interfaz intuitiva y eliminando las barreras técnicas asociadas al *trading* algorítmico.

En segundo lugar, la plataforma ofrece una solución de coste cero a los usuarios, eliminando la barrera financiera que impide a muchas personas acceder a soluciones de *trading* algorítmico de calidad. Esto democratiza el campo del *trading* algorítmico y promueve la experimentación y el aprendizaje sin riesgos, fomentando la innovación y el descubrimiento de nuevas estrategias.

Por último, la plataforma tiene un enfoque educativo y puede utilizarse en el marco de un club universitario. Permite a los estudiantes aprender estrategias de negociación, experimentar en un entorno simulado y colaborar con otros miembros. Esto les brinda la oportunidad de adquirir habilidades prácticas en el *trading* algorítmico, complementando su formación académica y preparándolos para futuras carreras en el ámbito financiero.

3. Descripción de la plataforma

Trader Companion se ha desarrollado en inglés para ampliar su adopción y utilizar términos comunes en este campo. La arquitectura de la plataforma se basa en un enfoque modular, que proporciona una experiencia de usuario sin fisuras. Los sistemas principales incluyen un controlador y vistas que constituyen la interfaz de usuario, una conexión a una API externa para obtener datos de activos bursátiles, bases de datos para almacenar información y un módulo de estrategias con estrategias de inversión predeterminadas.

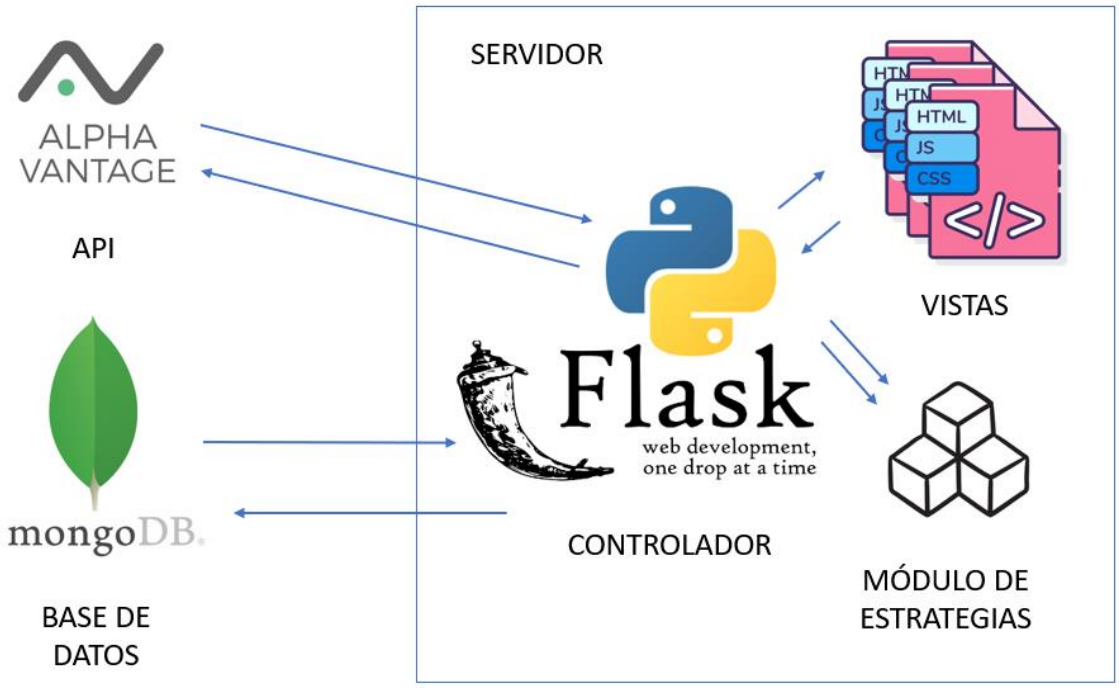


Ilustración 1 - Arquitectura global de la plataforma

La conexión a la API externa se realiza mediante la biblioteca "alpha_vantage" en Python, que recupera datos en tiempo real de los activos seleccionados por los usuarios. Las bases de datos utilizan MongoDB como sistema de gestión NoSQL, almacenando datos históricos de precios, usuarios registrados, estrategias y resultados de backtest. El módulo de estrategia permite a los usuarios implementar y ejecutar estrategias de backtesting mediante la importación dinámica de clases individuales.

La interfaz de usuario se centra en ofrecer una experiencia intuitiva y funcional. La página de inicio destaca las características clave de la plataforma, mientras que las páginas de registro e inicio de sesión ofrecen una estructura coherente y minimalista. La creación de estrategias se simplifica con un formulario interactivo, y la biblioteca de estrategias presenta fichas con acciones interactivas.

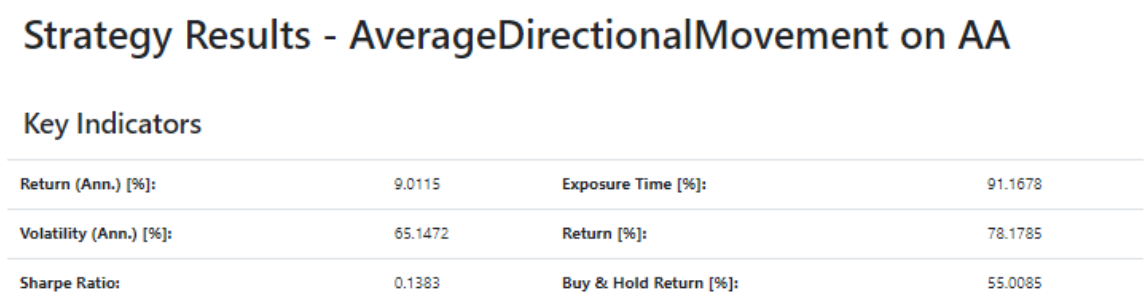
La selección de activos permite a los usuarios elegir los activos a los que pueden aplicar estrategias, y los resultados del backtest se muestran en una vista que incluye gráficos, indicadores clave y opciones para guardar y descargar.

En cuanto al almacenamiento de datos, se utilizan diferentes bases de datos para mantener información específica como datos históricos de precios, usuarios, estrategias y resultados de backtest.

4. Resultados

La plataforma está diseñada para facilitar la participación de usuarios sin conocimientos avanzados de programación en el ámbito del *trading* algorítmico. Por ello, los resultados se presentan en forma de caso de uso. En este caso de uso, el usuario puede registrarse, explorar la biblioteca de estrategias predefinidas y seleccionar una de ellas.

A continuación, el usuario personaliza su estrategia seleccionando el periodo de inversión, la frecuencia y las comisiones de las órdenes. La plataforma ofrece funciones de búsqueda para encontrar valores concretos y obtener información financiera relevante sobre ellos, de modo que el usuario pueda elegir un valor estando informado. Una vez cumplimentado el formulario, se realiza el backtest de la estrategia seleccionada y se muestran los resultados al usuario.



Strategy Results - AverageDirectionalMovement on AA

Key Indicators

Return (Ann.) [%]:	9.0115	**Exposure Time [%]:**	91.1678
Volatility (Ann.) [%]:	65.1472	**Return [%]:**	78.1785
Sharpe Ratio:	0.1383	**Buy & Hold Return [%]:**	55.0085

Ilustración 2 - Resultados del backtest. Indicadores clave

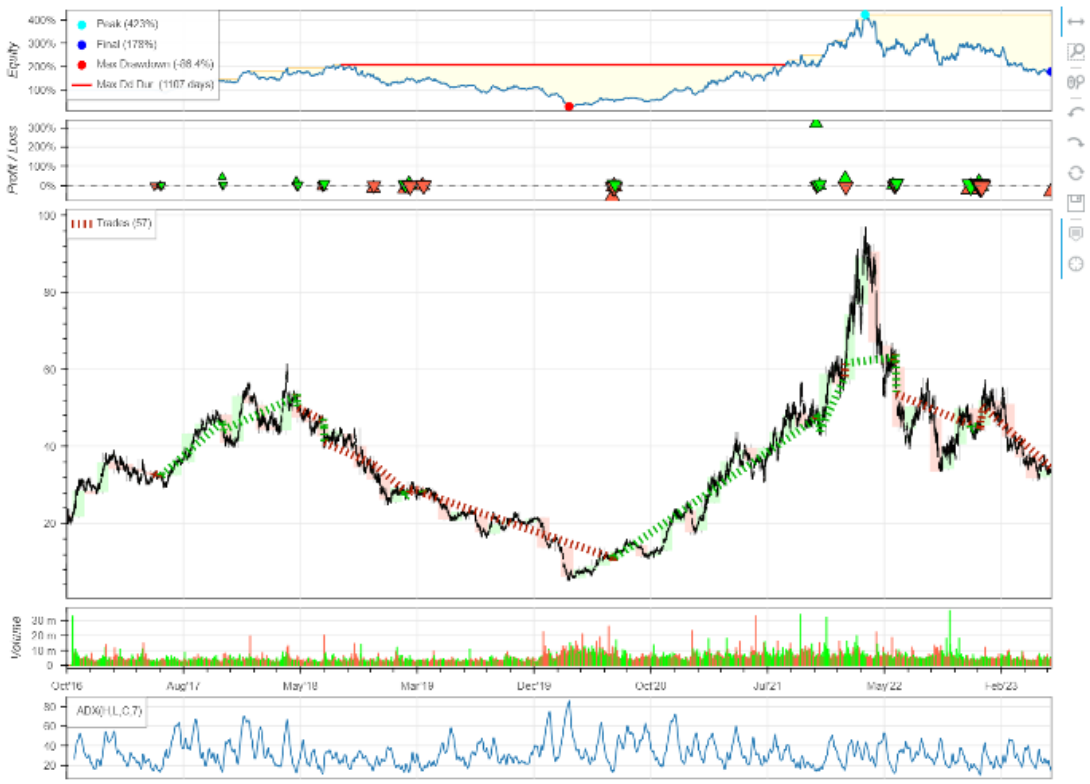


Ilustración 3 - Resultados del backtest. Gráfico interactivo

Los resultados del backtest se presentan en una vista principal que muestra los indicadores clave y un gráfico interactivo del rendimiento temporal. Los indicadores clave más importantes son la rentabilidad y la volatilidad anualizada de la estrategia, con el ratio de Sharpe como métrica principal. El rendimiento de la estrategia también se comparó con una estrategia de "comprar y mantener".

Además, se ofrece la opción de optimizar los parámetros de la estrategia. Un formulario permite optimizar los parámetros especificando los rangos de valores que se van a probar. Una vez que los usuarios obtienen los resultados deseados, pueden descargar el gráfico y las estadísticas del backtest, así como guardar los resultados en su biblioteca personalizada.

5. Conclusiones

A lo largo del desarrollo de este trabajo se han alcanzado todos los objetivos planteados en la introducción. La plataforma ofrece una biblioteca de estrategias predefinidas y herramientas de optimización que permiten a los usuarios maximizar el rendimiento de sus estrategias y adaptarlas a las condiciones cambiantes del mercado. Además, la posibilidad de descargar el código de las estrategias predefinidas y probar estrategias personalizadas dentro de la plataforma proporciona a los usuarios una experiencia de aprendizaje completa. Nuestro análisis de los resultados y las pruebas realizadas en la plataforma demuestra que los usuarios, incluso sin experiencia previa en programación, pueden beneficiarse de sus funcionalidades.

Aunque se han logrado avances significativos, es importante destacar algunas limitaciones y oportunidades de mejora identificadas durante el desarrollo de la plataforma. En primer

lugar, la plataforma está actualmente limitada al uso de valores de los principales índices estadounidenses debido a restricciones en la API utilizada. Otra limitación significativa está relacionada con el límite de velocidad impuesto por la API utilizada para las solicitudes de datos.

Además, existe una limitación en el backtesting inicial de una estrategia, ya que actualmente utiliza parámetros por defecto del código base de la estrategia. Esto restringe la capacidad del usuario para probar diferentes configuraciones de parámetros desde el principio. Además, la dependencia de la calidad de las estrategias predefinidas es también una limitación importante y la simplificación de variables y procesos en la plataforma puede limitar su capacidad para representar plenamente la complejidad del *trading* algorítmico.

En conclusión, este trabajo ha demostrado la viabilidad y el valor práctico de la plataforma Trader Companion para diseñar, probar y optimizar estrategias de negociación algorítmica, pero ha llegado con sus limitaciones. la plataforma ha abordado con éxito la necesidad de herramientas accesibles y personalizables en el ámbito del *trading* automatizado, permitiendo a los usuarios, incluso sin conocimientos avanzados de programación, aprovechar las ventajas de las estrategias automatizadas.

La plataforma también ha demostrado su potencial educativo al ofrecer a los usuarios la oportunidad de adquirir conocimientos prácticos y experiencia en *trading* algorítmico. A pesar de las limitaciones identificadas, éstas ofrecen oportunidades para futuras investigaciones y mejoras adicionales que refuercen aún más la plataforma y amplíen su alcance.

6. Referencias

[1] Cagan, P., & Lipsey, R. E. The financial effects of inflation (No. caga78-1). National Bureau of Economic Research. 1978.

[2] Chan, E. P. Quantitative trading: how to build your own algorithmic trading business. John Wiley & Sons. 2021.

[3] Eurostat. Annual inflation up to 10.6% in the euro area. Euro indicators. October 2022. https://ec.europa.eu/eurostat/documents/2995521/15265521/2-17112022-AP-EN.pdf/b6953137-786e-ed9c-5ee2-6812c0f8f07f#:~:text=The%20euro%20area%20annual%20inflation,up%20from%2010.9%25%20in%20September.

[4] Georgarakos, D., & Pasini, G. Trust, Sociability and Stock Market Participation. Behavioral & Experimental Finance (Editor's Choice) eJournal. 2009. https://doi.org/10.2139/ssrn.1397236.

TRADER COMPANION: WEB PLATFORM TO LEARN AND OPTIMIZE ALGORITHMIC TRADING STRATEGIES

Author: Guerrero Gallego, Álvaro.
Supervisor: Contreras Bárcena, David.
Collaborating Entity: ICAI – Universidad Pontificia Comillas

ABSTRACT

This project presents the development of a web platform for backtesting investment strategies with an educational approach. The tool allows users to design, test and optimize automated strategies without the need for advanced programming knowledge and serves as a bridge to learn how to program their own strategies.

Keywords: Finance, Web Application, Algorithmic Trading, Backtesting

1. Introduction

In recent years, the erosion of purchasing power and high inflation has raised concerns among the population (Eurostat, 2022). Financial markets and traditional institutions have been the option for protecting wealth (Georgarakos & Pasini, 2009). However, during periods of high inflation, more active strategies are necessary (Cagan & Lipsey, 1978).

Algorithmic trading has emerged as an efficient solution, enabling the automation of trades and the analysis of large amounts of data (Chan, 2021). Although this practice is common among large funds, individual investors still face barriers in terms of knowledge and technology.

To overcome this limitation, this project proposes the development of an educational web platform focused on backtesting algorithmic trading strategies. Backtesting allows the testing of investment strategies using historical data, providing a more comprehensive understanding of their potential effectiveness.

The motivation behind this work lies in disseminating knowledge and increasing accessibility in quantitative finance and algorithmic trading. The aim is to bridge the gap between common investors and algorithmic strategies by providing an intuitive tool for testing and evaluating investment strategies.

2. Project Definition

The objectives of the project include developing a user-friendly interface, integrating real-time financial data, implementing predefined strategies, and enabling the loading of custom strategies. Additionally, the platform aims to provide functionality for optimizing strategies.

The methodology adopted for the development of this final project work is based on requirements analysis, technology research, architecture design, user interface development, integration of financial data, strategy research and implementation, testing, and validation.

The justification for the project is based on three key points. Firstly, a clear need in the field of algorithmic trading has been identified: the demand for accessible solutions to design, test, and optimize automated trading strategies. The developed platform addresses this need by providing an intuitive interface and eliminating the technical barriers associated with algorithmic trading.

Secondly, the platform offers a zero-cost solution to users, removing the financial barrier that prevents many individuals from accessing quality algorithmic trading solutions. This democratizes the field of algorithmic trading and promotes risk-free experimentation and learning, fostering innovation and the discovery of new strategies.

Lastly, the platform has an educational focus and can be used in a university club setting. It allows students to learn about trading strategies, experiment in a simulated environment, and collaborate with other members. This provides them with the opportunity to gain practical skills in algorithmic trading, complementing their academic education and preparing them for future careers in the financial field.

3. Platform Description

Trader Companion has been developed in English to broaden its adoption and use common terms in this field. The platform architecture is based on a modular approach, providing a seamless user experience. The main systems include a controller and views constituting the user interface, a connection to an external API for obtaining stock asset data, databases for storing information, and a strategy module.

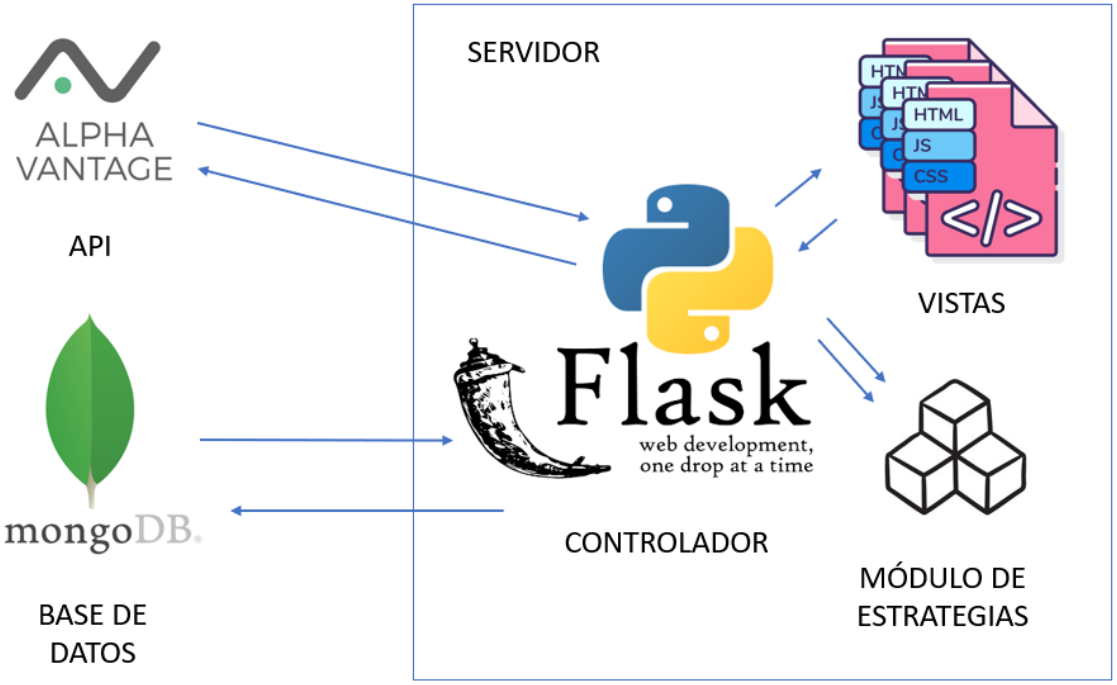


Ilustración 4 - Global platform architecture

The connection to the external API is done using the "alpha_vantage" library in Python, which retrieves real-time data for assets selected by users. The databases use MongoDB as a NoSQL management system, storing historical price data, registered users, strategies, and backtest results. The strategy module allows users to implement and execute backtesting strategies by dynamically importing individual classes.

The user interface focuses on providing an intuitive and functional experience. The homepage highlights key features of the platform, while the registration and login pages offer a coherent and minimalist structure. Strategy creation is simplified with an interactive form, and the strategy library presents cards with interactive actions.

The asset selection allows users to choose assets to which they can apply strategies, and backtest results are displayed in a view that includes charts, key indicators, and options for saving and downloading.

Regarding data storage, different databases are used to maintain specific information such as historical price data, users, strategies, and backtest results.

4. Results

The platform is designed to facilitate the involvement of users with no advanced programming knowledge in the field of algorithmic trading. Therefore, the results are presented in the form of a use case. In this use case, the user is able to register, explore the library of predefined strategies and select one of the predefined strategies.

Next, the user customizes their strategy by selecting the investment period, frequency, and order commissions. The platform provides search functionalities to find specific stocks and obtain relevant financial information about them, so the user can pick a stock being informed. Once the form is completed, the selected strategy is backtested and the results are displayed to the user.

Ilustración 5 - Backtest results. Key indicators

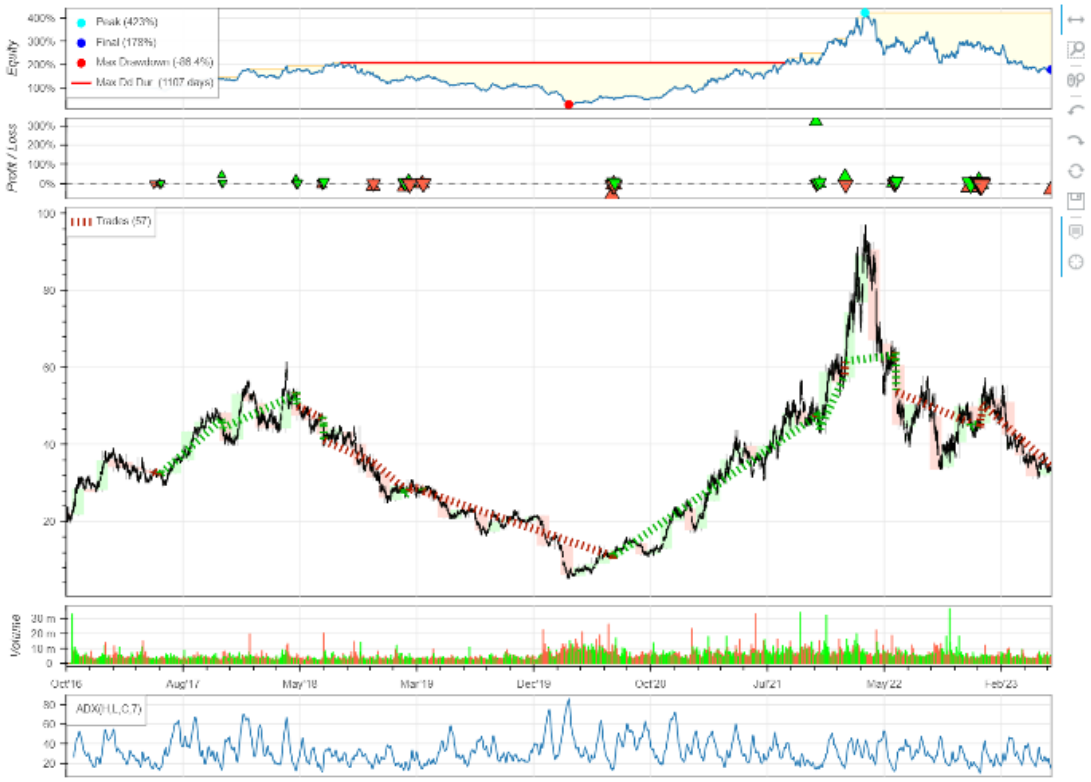


Ilustración 6 - Backtest results. Interactive graph

The backtest results are presented in a main view that displays key indicators and an interactive chart of the temporal performance. The most important key indicators are the strategy's profitability and annualized volatility, with the Sharpe ratio as the main metric. The performance of the strategy was also compared to a "buy and hold" strategy.

Additionally, the option to optimize strategy parameters is provided. A form allows for parameter optimization by specifying ranges of values to be tested. Once users obtain the desired results, they can download the backtest chart and statistics, as well as save the results to their custom library.

5. Conclusions

Throughout the development of this project, all the objectives set forth in the introduction are achieved. The platform offers a library of predefined strategies and optimization tools that allow users to maximize the performance of their strategies and adapt them to changing market conditions. Moreover, the ability to download the code of predefined strategies and test custom strategies within the platform provides users with a comprehensive learning experience. Our analysis of the results and tests conducted on the platform demonstrates that users, even without previous programming experience, can benefit from its functionalities.

Although significant progress has been made, it is important to highlight some limitations and opportunities for improvement identified during the development of the platform. Firstly, the platform is currently limited to the use of stocks from major US indices due to restrictions in the API used. Another significant limitation is related to the rate limit imposed by the API used for data requests.

Furthermore, a limitation exists in the initial backtesting of a strategy, as it currently uses default parameters from the strategy's base code. This restricts the user's ability to test different parameter configurations from the beginning. Moreover, the dependence on the quality of predefined strategies is also an important limitation and the simplification of variables and processes in the platform may limit its ability to fully represent the complexity of algorithmic trading.

In conclusion, this project has demonstrated the viability and practical value of the Trader Companion platform for designing, testing, and optimizing algorithmic trading strategies, but it has come with its limitations. The platform has successfully addressed the need for accessible and customizable tools in the field of automated trading, allowing users, even without advanced programming knowledge, to leverage the advantages of automated strategies.

The platform has also demonstrated its educational potential by providing users with the opportunity to acquire practical skills and experience in algorithmic trading. Despite the identified limitations, they offer opportunities for future research and additional improvements to further strengthen the platform and expand its reach in the field of algorithmic trading.

6. References

[1] Cagan, P., & Lipsey, R. E. The financial effects of inflation (No. caga78-1). National Bureau of Economic Research. 1978.

[2] Chan, E. P. Quantitative trading: how to build your own algorithmic trading business. John Wiley & Sons. 2021.

[3] Eurostat. Annual inflation up to 10.6% in the euro area. Euro indicators. October 2022. https://ec.europa.eu/eurostat/documents/2995521/15265521/2-17112022-AP-EN.pdf/b6953137-786e-ed9c-5ee2-6812c0f8f07f#:~:text=The%20euro%20area%20annual%20inflation,up%20from%2010.9%25%20in%20September.

[4] Georgarakos, D., & Pasini, G. Trust, Sociability and Stock Market Participation. Behavioral & Experimental Finance (Editor's Choice) eJournal. 2009. https://doi.org/10.2139/ssrn.1397236.

Índice de la memoria

Capítulo 1. Introducción 4

1.1 Motivación del proyecto 5

1.2 Planteamiento del problema y objetivos 6

1.3 Alcance y limitaciones 7

1.4 Metodología 9

Capítulo 2. Descripción de Tecnologías 11

2.1 Framework Web: Flask 11

2.2 Base de Datos: MongoDB 13

2.3 API de Datos: Alphavantage 15

Capítulo 3. Marco Teórico 18

3.1 Contexto Técnico 18

3.1.1 Origen y desarrollo de los mercados financieros 18

3.1.2 Relevancia de la Inversión Financiera 20

3.1.3 Estrategias de Inversión 21

3.1.4 Trading Algorítmico 22

3.1.5 Backtesting 23

3.2 Estado de la Cuestión 24

3.3 Justificación del Proyecto 29

3.3.1 Identificación de una necesidad en el trading algorítmico 29

3.3.2 Coste Cero 30

3.3.3 Enfoque educativo 30

Capítulo 4. Diseño de la Plataforma 32

4.1 Sistemas de alto nivel 32

4.2 Interfaz de usuario 34

4.2.1 Página Principal 35

4.2.2 Registro e Inicio de sesión 36

4.2.3 Creación de estrategias 38

4.2.4 Librería de Estrategias 40

4.2.5 Selección de acciones 41

4.2.6 Resultados del backtest ... *43*
4.2.7 Librería de Backtests ... *46*
4.3 Almacenamiento y modelo de datos ... 47
4.3.1 Precios Históricos de Acciones ... *48*
4.3.2 Usuarios ... *48*
4.3.3 Estrategias ... *48*
4.3.4 Resultados de Backtests ... *49*
Capítulo 5. Implementación de la Plataforma ... 51
5.1 Desarrollo de Funcionalidades Clave ... 51
5.1.1 Programación de Estrategias ... *52*
5.1.2 Obtención y Filtrado de Datos ... *54*
5.1.3 Backtesting y Optimización ... *55*
5.2 Implementación del Frontend y Backend ... 58
5.2.1 Frontend ... *59*
5.2.2 Backend ... *60*
Capítulo 6. Análisis de Resultados ... 64
Capítulo 7. Conclusiones y Trabajos Futuros ... 72
7.1 Limitaciones y Futuras mejoras ... 73
Capítulo 8. Bibliografía ... 76
ANEXO I: ALINEACIÓN DEL PROYECTO CON LOS ODS ... 78
ANEXO II: REPOSITORIO DE CÓDIGO ... 80

Índice de figuras

Figura 1- Esquema del patrón modelo-vista-controlador (Fuente: https://codigofacilito.com/articulos/mvc-model-view-controller-explicado)..... 11
Figura 2 - Arquitectura global de la plataforma 33
Figura 3 - Página principal de la plataforma 35
Figura 4 - Página de registro de la plataforma..... 36
Figura 5 - Página de inicio de sesión de la plataforma 36
Figura 6 - Página de creación de estrategias de la plataforma..... 38
Figura 7 - Página de librería de estrategias de la plataforma..... 40
Figura 8 - Página de selección de acciones de la plataforma 41
Figura 9 - Página de resultados del backtest de la plataforma [1/2]..... 43
Figura 10 - Página de resultados del backtest de la plataforma [2/2]..... 44
Figura 11 - Página de librería de backtests de la plataforma..... 46
Figura 12 - Modelo de datos (Fuente de los iconos: https://www.vecteezy.com/free-vector/database) 47
Figura 13 - Gráfico interactivo resultantes de un backtest 56
Figura 14 - Estadísticas principales resultantes de un backtest 57
Figura 15 - Caso de uso de selección de la acción bursátil y su periodo de inversión..... 65
Figura 16 - Caso de uso de obtención de información financiera sobre un ticker concreto 66
Figura 17 - Caso de uso de interpretación de resultados de un backtest 67
Figura 18 - Caso de uso de optimización de parámetros de un backtest 68
Figura 19 - Caso de uso de la optimización de las métricas principales tras la optimización 69
Figura 20 - Ejemplo de código de estrategia comentado disponible para su descarga por el usuario 70
Figura 21 - Caso de uso para subir a la plataforma una estrategia personalizada por el usuario 71

Capítulo 1. INTRODUCCIÓN

En los últimos años, la erosión del poder adquisitivo se ha convertido en una preocupación acuciante para una parte importante de la población, sobre todo a la luz de las tasas de inflación imperantes. En el último año en la zona euro, por ejemplo, se ha llegado a registrar una tasa de inflación interanual del 10,6% (Eurostat, 2022), un nivel que no se registraba desde 1985, previo a la creación de la Unión Europea. En consecuencia, los particulares que buscan salvaguardar su patrimonio han recurrido tradicionalmente a los mercados bursátiles y a diversas instituciones financieras (Georgarakos & Pasini, 2009). Lo que en un principio comenzó como una aventura especulativa, con la esperanza de obtener beneficios del comercio marítimo, ha evolucionado hasta convertirse en el complejo panorama inversor actual.

Dentro de este panorama, las personas con un capital limitado tienen la oportunidad de diseñar sus propias estrategias de inversión o confiar en la experiencia de "asesores expertos" para asignar sus fondos entre diversas clases de activos. A lo largo de los años, estrategias como la ampliamente reconocida de "*Buy and Hold*" han demostrado su eficacia durante los mercados alcistas, permitiendo a los inversores beneficiarse del crecimiento de los activos de sus empresas o gobiernos preferidos. Muchos pequeños inversores han utilizado con éxito esta estrategia para aumentar sus ahorros a un ritmo más rápido que la tasa de subida de los precios (Barber & Odean, 2013).

Sin embargo, durante periodos de alta inflación y elevada incertidumbre, como el actual, limitarse a mantener una posición en el mercado es insuficiente. Solo las personas con el tiempo necesario para estudiar meticulosamente las tendencias del mercado y la perspicacia para capitalizar las oportunidades específicas que surjan pueden combatir eficazmente los retos que plantea la inflación (Cagan & Lipsey, 1978).

Afortunadamente, los avances tecnológicos han revolucionado el análisis de los mercados, haciéndolo más accesible a un mayor número de personas. El desarrollo de la potencia de

cálculo y la generalización de las redes de comunicación han dado lugar a la negociación algorítmica, una nueva forma de inversión que elimina la necesidad de supervisar continuamente el mercado. La ejecución manual de órdenes bursátiles, que antes se realizaba por teléfono o a través de sitios web bancarios, puede automatizarse ahora mediante el uso de bots programados con estrategias predefinidas. Aunque los grandes fondos ya han adoptado la gestión pasiva del capital a través de la negociación algorítmica, su utilización entre los inversores particulares, que pueden beneficiarse enormemente de su aplicación, sigue siendo relativamente limitada (Chan., 2021).

Esta adopción limitada puede atribuirse al desconocimiento del inversor común respecto a la flexibilidad que ofrecen las estrategias de negociación algorítmica y a las barreras de entrada que suponen los conocimientos financieros y tecnológicos necesarios. Aunque muchas personas pueden tener la capacidad de conceptualizar estrategias de inversión con potencial en circunstancias específicas, pocas poseen las habilidades necesarias para probar y evaluar fácilmente estas estrategias. En consecuencia, este documento pretende abordar estos dos retos y superar las limitaciones asociadas a las soluciones de negociación algorítmica sin código que ofrecen las empresas.

Mediante el desarrollo de un sitio web educativo centrado en el *backtest*ing, que es la prueba de estrategias de *trading* algorítmico sobre datos históricos, esta investigación pretende salvar la brecha existente entre el inversor común y las estrategias de negociación algorítmica. El objetivo es proporcionar a las personas los medios para probar y evaluar fácilmente sus estrategias de inversión, ofreciendo así una comprensión más completa de su eficacia potencial en el mercado.

1.1 MOTIVACIÓN DEL PROYECTO

La motivación subyacente de este proyecto se fundamenta en el deseo de difundir conocimiento y aumentar la accesibilidad en el ámbito de las finanzas cuantitativas y el *trading* algorítmico. La exploración personal en estas áreas se inició a través de interacciones con colegas en ICAI y una extensa investigación autodidacta sobre el tema. Durante este

periodo, surgieron diversos foros en línea donde las personas compartían estrategias de *trading* e ideas para su optimización. No obstante, a pesar de la abundancia de información disponible, comprender las complejidades de estas discusiones resultó desafiante, lo que llevó inicialmente a participar en algunas publicaciones de los foros entre amigos y, posteriormente, a la concepción de un club universitario dedicado a fomentar un ambiente colaborativo de aprendizaje en este ámbito.

Sin embargo, surgió un obstáculo importante: al igual que nosotros al principio, muchos no entendíamos gran parte de los comentarios de los foros que leíamos, nos dimos cuenta de que los conocimientos básicos necesarios para participar en la negociación algorítmica resultaron ser considerablemente extensos, y que limitaban mucho la participación de personas sin experiencia en programación ya que estas simplemente podrían proponer estrategias pero nunca probarlas y optimizarlas, parte clave de la actividad. Por lo tanto, la motivación de esta tesis es doble: desarrollar una herramienta que proporcione una interfaz fácil de usar para probar, optimizar y ejecutar estrategias de trading, sirviendo como una guía completa para establecer el propio entorno de *trading* algorítmico, y cerrar la brecha entre la complejidad de las estrategias y las habilidades de programación necesarias para personalizarlas completamente.

1.2 PLANTEAMIENTO DEL PROBLEMA Y OBJETIVOS

Este trabajo de fin de grado se enfoca en abordar el desafío de proporcionar una plataforma educativa para el *backtest*ing de estrategias de *trading* algorítmico. El *trading* algorítmico ha ganado popularidad en los últimos años debido a su capacidad para analizar grandes cantidades de datos y ejecutar operaciones de manera automatizada. Sin embargo, muchas personas interesadas en aprender y practicar el *trading* algorítmico se enfrentan a barreras de entrada significativas, como la falta de conocimientos técnicos y el acceso a herramientas adecuadas. Con el fin de abordar este problema se plantean los siguientes objetivos para delimitar el alcance del proyecto:

- Desarrollo de una interfaz intuitiva: El objetivo principal es crear una interfaz de usuario intuitiva y fácil de usar, que permita a los usuarios interactuar con la aplicación de manera eficiente. Se presta especial atención a la usabilidad y la experiencia del usuario para garantizar una experiencia agradable y efectiva.
- Integración de datos financieros en tiempo real: Se busca integrar datos financieros en tiempo real utilizando APIs y un almacenamiento adecuados. Esto permitirá a los usuarios probar sus estrategias utilizando datos precisos y actualizados, mejorando la calidad y la confiabilidad de los resultados obtenidos.
- Implementación de estrategias predefinidas: Se plantea implementar una variedad de estrategias de negociación predefinidas que los usuarios puedan utilizar como punto de partida para sus propias pruebas. Estas estrategias abarcarán diferentes enfoques y estilos de negociación, brindando a los usuarios una amplia gama de opciones y posibilidades de exploración.
- Personalización y carga de estrategias personalizadas: Además de las estrategias predefinidas, se desea proporcionar a los usuarios la capacidad de cargar sus propias estrategias personalizadas en la plataforma. Esto fomentará la creatividad y la experimentación, permitiendo a los usuarios adaptar las estrategias a sus necesidades y preferencias individuales.
- Optimización de estrategias: Se plantea ofrecer funcionalidades de optimización de estrategias, lo que permitirá a los usuarios encontrar los valores óptimos de los parámetros de sus estrategias para maximizar los resultados deseados. Esto facilitará el proceso de ajuste y mejora de las estrategias, lo que a su vez podría conducir a un rendimiento más sólido y rentable.

1.3 ALCANCE Y LIMITACIONES

A pesar de los objetivos mencionados anteriormente, es importante reconocer las limitaciones de este trabajo. Algunas de las limitaciones identificadas son las siguientes:

- Limitación de datos históricos: La plataforma utiliza fuentes de datos para acceder a información histórica y en tiempo real. Sin embargo, la disponibilidad y la calidad de los datos pueden estar sujetas a restricciones y limitaciones impuestas por las fuentes de datos utilizadas. Esto puede afectar la cantidad y la calidad de los datos disponibles para las pruebas y análisis de estrategias. Es importante tener en cuenta que los resultados de las pruebas y la optimización de estrategias pueden depender de la disponibilidad y calidad de los datos utilizados. Se recomienda a los usuarios considerar estas limitaciones al interpretar los resultados obtenidos en la plataforma.
- Restricciones de recursos computacionales: Dado que el *backtest*ing implica la ejecución de algoritmos y cálculos intensivos, las capacidades de procesamiento y los recursos computacionales disponibles pueden limitar el tamaño y la complejidad de las estrategias que se pueden probar de manera eficiente en la plataforma.
- Limitaciones de personalización: Aunque la plataforma permite a los usuarios cargar sus propias estrategias personalizadas, existen ciertas limitaciones en cuanto a la complejidad y las funcionalidades que se pueden implementar. Esto se debe a las restricciones de la plataforma y a la necesidad de mantener un entorno de *backtest*ing controlado y seguro.
- Limitaciones de ejecución de estrategias en tiempo real: Una limitación importante de esta plataforma es que no se implementa la ejecución de las estrategias en tiempo real utilizando brókeres financieros. Esto se debe a que trabajar con brókeres financieros implica una serie de problemas relacionados con la protección de datos y la seguridad que añadirían una capa de complejidad excesiva para el objetivo educativo del proyecto. La plataforma se centra en el *backtest*ing de estrategias y proporciona a los usuarios una herramienta para probar y optimizar sus estrategias en un entorno simulado. La ejecución en tiempo real implicaría su integración con brókeres y exigiría consideraciones adicionales en términos de seguridad, cumplimiento normativo y protección de datos financieros. Si bien la ejecución en tiempo real es una funcionalidad valiosa en entornos de inversión reales, su implementación va más allá del alcance de un proyecto educativo.

Es importante tener en cuenta todas estas limitaciones al utilizar la plataforma, ya que los resultados obtenidos a través del *backtest*ing pueden no reflejar necesariamente el rendimiento real de una estrategia en un entorno de inversión en vivo. Los usuarios deben considerar estas limitaciones al evaluar y tomar decisiones basadas en los resultados de las pruebas realizadas en la plataforma.

1.4 METODOLOGÍA

El proceso de desarrollo de este trabajo de fin de grado se basa en una serie de etapas y enfoques sistemáticos para garantizar la eficiencia y la calidad del proyecto. A continuación, se describe la metodología empleada:

1. Análisis de requisitos: En esta etapa, se realiza un análisis exhaustivo de los requisitos y objetivos de la plataforma. Se identifican las funcionalidades clave, como la creación de una interfaz intuitiva y fácil de usar, la integración de datos financieros en tiempo real y la implementación de estrategias predefinidas. También se determina la capacidad de personalización de las estrategias y se establece la opción de optimización de los parámetros. Se tiene en cuenta la importancia de abordar las limitaciones relacionadas con la disponibilidad de datos históricos, los recursos computacionales y la ejecución en tiempo real.
2. Investigación de tecnologías: Se lleva a cabo una investigación exhaustiva sobre las tecnologías más adecuadas para el desarrollo de la plataforma. Se evalúan diversos lenguajes de programación, *framework*s y bibliotecas, considerando aspectos como la eficiencia, la escalabilidad y la seguridad. Además, se investigan fuentes confiables de datos financieros en tiempo real y se seleccionan las API apropiadas para acceder a estos datos de manera precisa y actualizada.
3. Diseño de la arquitectura: Con base en el análisis de requisitos y la investigación de tecnologías, se realiza el diseño de la arquitectura de la plataforma. Se establece una estructura modular que permita la fácil integración de nuevas funcionalidades y estrategias. Se definen los protocolos de comunicación entre los distintos componentes

de la plataforma y se seleccionan las herramientas y tecnologías adecuadas para su desarrollo.

4. Desarrollo de la interfaz de usuario: Se implementa una interfaz de usuario intuitiva y atractiva utilizando tecnologías web modernas como HTML, CSS y JavaScript. Se pone especial atención en la simplicidad y la experiencia del usuario, diseñando una navegación fluida, pantallas claras y controles interactivos que faciliten la utilización de la plataforma.
5. Integración de datos financieros: Se lleva a cabo la integración de datos financieros en tiempo real utilizando las fuentes confiables de datos identificadas durante la etapa de investigación de tecnologías. Se implementan mecanismos eficientes de almacenamiento y recuperación de datos históricos, asegurando un acceso rápido y confiable. Se establecen conexiones seguras con las APIs de datos financieros para obtener información actualizada y precisa.
6. Investigación e implementación de estrategias predefinidas: Se realiza una exhaustiva investigación de estrategias básicas y ampliamente utilizadas en el ámbito del *trading* algorítmico, abarcando diferentes enfoques y estilos de negociación, como estrategias de seguimiento de tendencias, reversión a la media y rompimientos de patrones, entre otras. Estas estrategias son implementadas en la plataforma utilizando un enfoque basado en clases, con métodos y funciones específicos para cada estrategia, permitiendo a los usuarios utilizarlas como punto de partida para sus pruebas. Además, se incorporan componentes didácticos en la implementación, como comentarios y explicaciones detalladas en el código fuente, y documentación que describe la lógica y los principios subyacentes de cada estrategia.
7. Carga de estrategias personalizadas: Se diseña un sistema que permite a los usuarios cargar sus propias estrategias personalizadas en la plataforma. Se establecen protocolos de seguridad para garantizar la integridad de los archivos de estrategia y prevenir posibles riesgos o vulnerabilidades.
8. Pruebas y validación: Se llevan a cabo pruebas rigurosas en todas las etapas del desarrollo para garantizar el funcionamiento correcto y confiable de la plataforma.

Capítulo 2. DESCRIPCIÓN DE TECNOLOGÍAS

En este capítulo profundizaremos en los conceptos y herramientas tecnológicos utilizados en el desarrollo de la plataforma de negociación. Comprender estos conceptos es esencial para entender los mecanismos subyacentes y las funcionalidades de la plataforma. Exploraremos las tecnologías pertinentes que se han empleado para garantizar la recuperación, el procesamiento, el almacenamiento y el análisis eficaces de los datos. Además, presentaremos las principales herramientas y marcos utilizados para el desarrollo web y la interacción con el usuario Este capítulo tiene como objetivo facilitar la lectura y comprensión del proyecto al familiarizar al lector con los fundamentos tecnológicos necesarios

2.1 FRAMEWORK WEB: FLASK

Un *framework* web de trabajo web es una bibliotecas o marco de trabajo de software que proporciona un enfoque estructurado para el desarrollo de aplicaciones web. Ofrece un conjunto de herramientas, bibliotecas y convenciones para simplificar el proceso de creación de aplicaciones web. Los *frameworks* web suelen seguir un patrón arquitectónico específico, como el patrón modelo-vista-controlador (MVC), que ayuda a separar las tareas de gestión de peticiones HTTP, gestión de datos y representación de vistas.

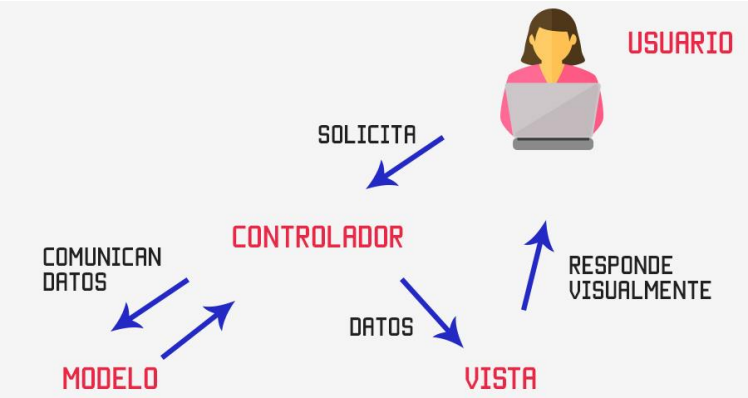


Figura 1- Esquema del patrón modelo-vista-controlador (Fuente: https://codigofacilito.com/articulos/mvc-model-view-controller-explicado)

Los *frameworks* web proporcionan a los desarrolladores una base para crear aplicaciones web dinámicas e interactivas. Se encargan de los detalles de bajo nivel de la comunicación HTTP, el enrutamiento y la gestión de solicitudes, permitiendo a los desarrolladores centrarse en la lógica de negocio de la aplicación.

En este trabajo el *framework* web utilizado es Flask, un *framework* web ligero y flexible escrito en Python. Sigue el enfoque de *framework* web del lado del servidor y junto a una base de datos como la que se define en la siguiente sección conformará nuestra arquitectura MVC. Está diseñado para ser minimalista y proporciona un conjunto potente de herramientas y funciones que permiten a los desarrolladores crear aplicaciones web de forma eficaz.

El núcleo de Flask es un servidor web que gestiona las peticiones HTTP entrantes y las envía a las funciones de vista apropiadas. Estas funciones de vista son responsables de procesar las solicitudes y devolver las respuestas correspondientes. Flask utiliza un mecanismo de enrutamiento para asignar URL a funciones de vista específicas, lo que facilita la definición de diferentes rutas y su lógica asociada.

Uno de los puntos fuertes de Flask es su sencillez. El *framework* tiene una base de código pequeña y una filosofía de diseño minimalista, lo que facilita su comprensión y el trabajo con él. Proporciona una API clara e intuitiva, lo que permite a los desarrolladores iniciarse rápidamente en la creación de aplicaciones web. Además, admite el renderizado de plantillas, que permite crear páginas web dinámicas. Las plantillas son archivos HTML que pueden incluir marcadores de posición para contenido dinámico y Flask utiliza un motor de plantillas llamado Jinja, para sustituir estos marcadores de posición por datos reales antes de enviar la respuesta al cliente.

Además, Flask ofrece una amplia gama de extensiones que se pueden utilizar para mejorar sus capacidades. Estas extensiones cubren diversas áreas, como la integración de bases de datos, la validación de formularios, la autenticación, etc. El diseño modular de Flask permite a los desarrolladores elegir e incluir sólo las extensiones que necesitan, manteniendo la aplicación ligera y eficiente.

Flask se beneficia del amplio ecosistema de Python. Como marco de trabajo de Python, aprovecha la amplia gama de bibliotecas y herramientas disponibles en Python, lo que facilita la integración con otros sistemas, bases de datos y servicios de terceros.

Su sencillez, sus extensiones y su capacidad de crear plantillas web dinámicas son las principales razones por las que Flask es la tecnología elegida para el desarrollo de nuestra plataforma. Con Flask, podemos crear una aplicación web robusta y escalable que satisfaga los requisitos de nuestro proyecto y, al mismo tiempo, disfrutar de las ventajas de un marco ligero y flexible.

2.2 BASE DE DATOS: MONGODB

En las aplicaciones informáticas modernas, las bases de datos desempeñan un papel crucial en la gestión y organización eficaz de los datos. Una base de datos es una colección estructurada de datos que sirve de repositorio centralizado para almacenar y recuperar información de forma sistemática. Para interactuar con las bases de datos, utilizamos sistemas de gestión de bases de datos (SGBD), que proporcionan herramientas y funcionalidades para la manipulación y recuperación de datos.

En esencia, una base de datos consta de tablas, filas y columnas. Las tablas organizan los datos en unidades lógicas, donde cada fila representa una instancia única y cada columna representa un atributo o propiedad específica de los datos. Este enfoque estructurado garantiza la integridad de los datos y permite un almacenamiento y una recuperación eficaces.

Las bases de datos relacionales han sido la opción tradicional para almacenar datos estructurados, donde los datos se organizan en tablas con esquemas predefinidos. Ofrecen una sólida integridad de los datos mediante relaciones y restricciones. Sin embargo, con el auge de los datos no estructurados y semiestructurados, ha surgido la necesidad de soluciones de bases de datos más flexibles.

MongoDB es un ejemplo destacado de base de datos NoSQL, concretamente una base de datos orientada a documentos. A diferencia de las bases de datos relacionales tradicionales, MongoDB almacena los datos en documentos flexibles similares a JSON, denominados documentos BSON (Binary JSON). Este enfoque basado en documentos permite el almacenamiento de estructuras de datos complejas y anidadas, lo que la hace adecuada para manejar datos no estructurados o semiestructurados.

El modelo de datos flexible de MongoDB permite a los desarrolladores iterar rápidamente y adaptar el modelo de datos a medida que evoluciona la aplicación. A diferencia de las bases de datos relacionales, que imponen un esquema fijo, los documentos de MongoDB pueden tener estructuras variables dentro de la misma colección. Esta flexibilidad es beneficiosa para el desarrollo ágil y la evolución de los requisitos de datos.

Además de su flexibilidad, MongoDB ofrece potentes capacidades de consulta. Admite un amplio conjunto de operadores de consulta y ofrece varias opciones de indexación para optimizar la recuperación de datos. Esto permite realizar consultas eficientes y rápidas, incluso con grandes conjuntos de datos.

La escalabilidad es otra característica clave de MongoDB. Puede escalarse horizontalmente distribuyendo los datos entre varios servidores o nodos, lo que permite gestionar grandes cargas de tráfico y conjuntos de datos de gran tamaño. La capacidad de fragmentación de MongoDB permite la partición y distribución automática de datos, lo que mejora el rendimiento y la tolerancia a fallos.

MongoDB también ofrece funciones avanzadas como la replicación para la redundancia de datos y la alta disponibilidad, la compatibilidad con datos geoespaciales, la búsqueda de texto y las canalizaciones de agregación para el procesamiento de datos complejos. Su completa documentación, su activa comunidad y la disponibilidad de bibliotecas y controladores para varios lenguajes de programación lo hacen accesible y fácil de integrar en diferentes pilas de aplicaciones.

Una de las razones clave por las que elegimos MongoDB para nuestra aplicación web es su compatibilidad con nuestro *framework* web elegido, Flask. La naturaleza flexible y sin esquemas de MongoDB se alinea bien con la naturaleza dinámica de las aplicaciones Flask. Como Flask permite la creación rápida de prototipos y el desarrollo ágil, el enfoque orientado a documentos de MongoDB complementa esto al permitirnos adaptar y modificar fácilmente el modelo de datos a medida que evoluciona nuestra aplicación. Además, las capacidades de escalabilidad y rendimiento de MongoDB lo convierten en una opción adecuada para gestionar el crecimiento potencial y las crecientes demandas de nuestra aplicación web. Aprovechando las características de MongoDB y su integración con Flask, podemos almacenar y recuperar datos de forma eficiente, garantizar la integridad de los datos y ofrecer una experiencia de usuario fluida.

2.3 API DE DATOS: ALPHAVANTAGE

En el mundo interconectado de hoy, las interfaces de programación de aplicaciones (API) desempeñan un papel fundamental a la hora de facilitar la comunicación y el intercambio de datos entre distintos sistemas de software. Las API permiten a los desarrolladores acceder e interactuar con las funcionalidades y los datos de aplicaciones, servicios o plataformas externas. En esta sección, exploraremos el concepto de API y profundizaremos en el funcionamiento de la API de AlphaVantage, un conocido proveedor de datos financieros.

Una API es un conjunto de reglas, protocolos y herramientas que permite a diferentes aplicaciones de software comunicarse entre sí. Define los métodos, formatos de datos y convenciones para solicitar y recibir datos o realizar acciones específicas. Las API se abstraen de los detalles de implementación subyacentes y proporcionan una interfaz estandarizada que los desarrolladores pueden utilizar para acceder a las funciones y recursos de un servicio o aplicación concretos.

Las API pueden clasificarse en distintos tipos, como API web, API de bibliotecas y API de sistemas operativos. Las API web, en particular, se utilizan habitualmente para el desarrollo

web y permiten a los desarrolladores acceder a servicios o datos remotos a través de Internet utilizando protocolos web estándar como HTTP.

La API AlphaVantage es una API web que proporciona a los desarrolladores acceso a una amplia gama de datos de los mercados financieros. Ofrece datos bursátiles históricos y en tiempo real, así como diversos indicadores técnicos, datos fundamentales y otra información financiera.

Para interactuar con la API de AlphaVantage, los desarrolladores deben enviar solicitudes HTTP a los puntos finales de la API, especificando los parámetros necesarios y las credenciales de autenticación. La API responde con los datos solicitados, normalmente en un formato estructurado como JSON o CSV. Los desarrolladores pueden entonces analizar y procesar los datos en sus aplicaciones.

La API de AlphaVantage admite varios puntos finales y funcionalidades. Por ejemplo, los desarrolladores pueden recuperar cotizaciones bursátiles en tiempo real, precios históricos de las acciones, datos de series temporales intradía y realizar análisis técnicos utilizando diferentes indicadores. La API también proporciona acceso a datos globales de renta variable y divisas, información de rendimiento sectorial y datos del mercado de criptomonedas.

Para garantizar un acceso seguro y fiable, la API de AlphaVantage requiere que los desarrolladores obtengan una clave API, que sirve como token de autenticación. Esta clave se incluye en las solicitudes de la API para identificar y rastrear el uso de los desarrolladores individuales. Dependiendo del plan específico o suscripción, la clave API puede tener ciertos límites de uso, tales como el número de solicitudes por minuto o por día.

Para nuestro proyecto, elegimos específicamente utilizar la API de AlphaVantage para obtener precios históricos de las acciones e información sobre las empresas. La API ofrece una amplia gama de datos históricos, incluidas series de precios diarias, semanales y mensuales, así como información fundamental como resúmenes de empresas, estados financieros y datos sobre beneficios. Además, la API de AlphaVantage ofrece puntos finales fáciles de usar y es compatible con varios lenguajes de programación, lo que la convierte en

una opción adecuada para nuestra aplicación. La disponibilidad de datos en tiempo real, indicadores técnicos y datos de rendimiento del sector consolidaron aún más nuestra decisión de utilizar la API de AlphaVantage. Al aprovechar las capacidades de esta API, podemos incorporar datos financieros precisos y actualizados a nuestra plataforma de negociación, lo que permite a los usuarios tomar decisiones de inversión informadas y adaptadas a los últimos datos existentes.

Capítulo 3. MARCO TEÓRICO

El marco teórico constituye un componente fundamental de los proyectos de investigación académica, ya que proporciona una base sólida para comprender los trabajos y soluciones existentes en el campo. El propósito de este capítulo consiste en presentar un marco teórico integral para el proyecto, el cual abarque el desarrollo del conocimiento hasta la fecha actual y proporcione respuestas a interrogantes fundamentales, como la existencia de soluciones similares en el mercado y el grado de éxito alcanzado en investigaciones previas para lograr los resultados deseados. En este capítulo también se abordará la justificación del proyecto desde un punto de vista más pragmático y de mercado.

3.1 CONTEXTO TÉCNICO

3.1.1 ORIGEN Y DESARROLLO DE LOS MERCADOS FINANCIEROS

Los mercados financieros tienen sus raíces en la necesidad de intercambiar capital y recursos entre diferentes actores económicos. A lo largo de la historia, han surgido y evolucionado para facilitar transacciones y proporcionar instrumentos financieros que permitan a las empresas y gobiernos obtener financiamiento. A continuación, se presenta un breve resumen del origen de los mercados financieros.

El origen de los mercados financieros se remonta a la antigüedad, cuando las civilizaciones antiguas, como Mesopotamia y Babilonia, desarrollaron sistemas rudimentarios para el intercambio de bienes y servicios. En estos sistemas, surgieron prácticas de préstamos, créditos y contratos que sentaron las bases para las transacciones financieras. Sin embargo, fue durante el Renacimiento y la Revolución Industrial cuando se produjeron avances significativos en los mercados financieros. Durante este período, las ciudades comerciales de Europa, como Ámsterdam y Londres, se convirtieron en centros financieros importantes. Se establecieron bolsas de valores y se emitieron acciones y bonos para financiar proyectos

comerciales y coloniales. Estos desarrollos sentaron las bases para los mercados financieros modernos.

En el siglo XX, los mercados financieros experimentaron importantes cambios y desafíos. Eventos como el Crash de 1929, la Segunda Guerra Mundial y la posterior creación de instituciones financieras internacionales, como el Fondo Monetario Internacional y el Banco Mundial, tuvieron un impacto significativo en la regulación y supervisión de los mercados financieros a nivel global (Romer, 1990). A medida que la economía mundial se volvía cada vez más interconectada, los mercados financieros se expandieron y se diversificaron. Surgieron nuevos instrumentos financieros, como los derivados, los fondos de inversión y los mercados de divisas, que brindaron a los inversores oportunidades de inversión más amplias y complejas.

En referencia a las características de los mercados financieros, es importante destacar la distinción entre los mercados primarios y secundarios. Los mercados primarios son aquellos en los que se emiten por primera vez los valores financieros, permitiendo a las empresas y gobiernos obtener financiamiento directamente de los inversores. Por otro lado, los mercados secundarios son aquellos en los que los inversores compran y venden valores financieros entre sí, sin la participación directa de la entidad emisora. Estos mercados proporcionan liquidez y facilitan la transferencia de activos financieros (Garbade & Silber, 1979).

En cuanto a los principales mercados financieros del mundo, encontramos centros financieros clave como la Bolsa de Nueva York (NYSE) en Estados Unidos, la Bolsa de Londres (LSE) en el Reino Unido y la Bolsa de Tokio en Japón. Estos mercados se caracterizan por su alto volumen de transacciones y su influencia en la economía global. Además, los índices financieros, como el índice Dow Jones Industrial Average (DJIA) en Estados Unidos o el FTSE 100 en el Reino Unido, sirven como referencia para evaluar el desempeño general de los mercados financieros y reflejar la salud económica de un país o región (Nowbutsing & Odit, 2009).

Los mercados financieros ejercen una influencia de amplio alcance en la vida de las personas, al reflejar de manera imperceptible pero significativa la capacidad económica de

los ciudadanos. Un ejemplo destacado de su impacto reciente es la crisis financiera de 2008, conocida como la "crisis de las hipotecas subprime", que tuvo su origen en el sector hipotecario de Estados Unidos y se propagó a nivel global. Esta crisis puso de manifiesto la intrincada interconexión y complejidad de los mercados financieros internacionales, así como los riesgos asociados a productos financieros complejos y una gestión inadecuada del riesgo. Sus repercusiones trascendieron el ámbito financiero, afectando la economía real y generando una profunda recesión económica (Grusky et al., 2011).

3.1.2 RELEVANCIA DE LA INVERSIÓN FINANCIERA

Siguiendo con el desarrollo de los mercados financieros, es importante entender el porqué de la inversión financiera y reside el impacto que puede generar en los individuos que la ejercen, para entender al completo la funcionalidad de este proyecto. Los inversores buscan aprovechar las oportunidades y generar retornos económicos a través de la asignación estratégica de sus recursos financieros. Existen diversas motivaciones que llevan a los individuos a invertir y se consideran características necesarias para ser un inversor exitoso. Sin embargo, también es importante abordar las razones por las cuales algunas personas son reacias a invertir o tienen tabúes respecto de la inversión.

Una de las principales motivaciones que llevan a un individuo a invertir es la búsqueda de crecimiento y rentabilidad financiera a largo plazo. La inversión puede permitir a las personas incrementar su patrimonio y alcanzar metas financieras, como la jubilación, la adquisición de bienes duraderos o la financiación de proyectos personales (Chatterjee, Finke, & Harness, 2011). Además, la inversión ofrece la oportunidad de diversificar los riesgos y protegerse contra la inflación, ya que determinados instrumentos de inversión pueden ofrecer rendimientos superiores a los de los ahorros tradicionales (Bajtelsmit & VanDerhei, 1995).

Para ser un inversor exitoso, es necesario contar con ciertas características y habilidades. En primer lugar, la capacidad de tomar decisiones informadas y basadas en el análisis de información financiera es fundamental. Los inversores deben tener una comprensión sólida de los principios y conceptos financieros, así como una capacidad para evaluar y gestionar los riesgos asociados con las inversiones (Graham & Dodd, 2009). Además, la paciencia, la

disciplina y la capacidad para mantener una perspectiva a largo plazo son características importantes para resistir la volatilidad y las fluctuaciones inherentes a los mercados financieros (Ping, 1995).

Sin embargo, es importante reconocer que muchas personas aún son reacias a invertir o tienen tabúes respecto de la inversión. Una de las razones clave es la falta de educación financiera. La mayoría de las personas no han recibido una formación adecuada en términos de conocimientos financieros básicos y no están familiarizadas con los conceptos de inversión y gestión de riesgos. Esta falta de educación puede generar temor, desconfianza y una sensación de que la inversión es complicada o reservada solo para expertos financieros (Lusardi, 2008). Además, existen tabúes culturales y sociales arraigados que rodean la inversión, lo que puede influir en la reticencia de las personas a explorar esta opción financiera (Pahlevi & Oktaviani, 2018).

3.1.3 ESTRATEGIAS DE INVERSIÓN

Tradicionalmente, los inversores tomaban decisiones de asignación basadas en la información disponible sobre empresas individuales o confiando en su intuición. En esos casos, la construcción de un marco sistemático para estrategias de *trading* no era necesaria, ya que las decisiones se basaban en informes financieros periódicos emitidos por las empresas cotizadas. Sin embargo, a medida que el campo de la inversión financiera evolucionó, surgió la noción de implementar un marco estructurado para obtener una ventaja sobre el mercado. Esto se basa en el reconocimiento de que los participantes del mercado no pueden predecir perfectamente el comportamiento del mercado únicamente en función de la información fundamental de las empresas, lo que da lugar a ineficiencias explotables.

La hipótesis de mercado eficiente (EMH, por sus siglas en inglés), propuesta por Fama (1970), sostiene que los mercados financieros son eficientes y que los precios reflejan completamente toda la información disponible. Según esta hipótesis, sería imposible superar consistentemente al mercado mediante estrategias de trading, ya que cualquier información relevante ya está incorporada en los precios. No obstante, diversos estudios empíricos han

revelado desviaciones de la EMH, indicando la presencia de anomalías de mercado y oportunidades de beneficio.

Una perspectiva alternativa, la finanza conductual, incorpora factores psicológicos y cognitivos en los procesos de toma de decisiones financieras (Barberis & Thaler, 2003). Reconoce que los participantes del mercado no siempre toman decisiones racionales y pueden verse influenciados por sesgos y emociones, lo que a su vez puede llevar a patrones predecibles en los precios de los activos. Al comprender y aprovechar estos sesgos conductuales, se pueden diseñar estrategias de *trading* para generar rendimientos excesivos (Shefrin, 2002). En línea con Chan (2019), el objetivo principal de construir una estrategia de *trading* es generar ganancias capitalizando las ineficiencias del mercado. Las ineficiencias del mercado surgen cuando el precio de mercado de un activo se desvía de su valor intrínseco, que representa el verdadero valor del activo basado en sus características fundamentales, como las ganancias, los dividendos y las perspectivas de crecimiento. Las estrategias de *trading* buscan identificar y explotar estas ineficiencias para generar ganancias, y este trabajo busca lograr precisamente eso.

Por ende, resulta imperativo comprender plenamente el propósito de este proyecto y la atracción que conlleva el desarrollo de estrategias de inversión que trasciendan la mera adhesión a los resultados periódicos de una empresa y la inversión basada en el seguimiento del mercado. En cambio, se busca aprovechar la noción de que los mercados presentan ineficiencias que pueden ser explotadas mediante un enfoque adecuado. Al adoptar este enfoque, se reconoce la existencia de oportunidades de inversión que se desprenden de las desviaciones del mercado, lo que motiva la búsqueda y el desarrollo de estrategias que permitan capitalizar estas ineficiencias.

3.1.4 TRADING ALGORÍTMICO

El *trading* algorítmico es un enfoque de inversión que utiliza algoritmos y sistemas automatizados para tomar decisiones de compra y venta de activos financieros en los mercados. Este enfoque se basa en la programación de reglas y condiciones específicas que permiten ejecutar operaciones de manera rápida y eficiente.

El *trading* algorítmico ha ganado popularidad en los últimos años debido a su capacidad para analizar grandes cantidades de datos en tiempo real y tomar decisiones basadas en patrones identificados por los algoritmos. Estos algoritmos pueden tener en cuenta una variedad de variables, como precios, volúmenes de negociación, indicadores técnicos y noticias, entre otros.

Este enfoque tiene como objetivo principal mejorar la velocidad y la precisión de las transacciones, así como reducir los sesgos emocionales y los errores humanos en la toma de decisiones. Al utilizar algoritmos y sistemas automatizados, se busca aprovechar las oportunidades del mercado de manera más eficiente y optimizar el rendimiento de las estrategias de inversión.

El *trading* algorítmico ha sido ampliamente adoptado por instituciones financieras, como bancos de inversión y fondos de cobertura, pero también ha ganado popularidad entre los inversores individuales y las plataformas de inversión en línea. A medida que la tecnología continúa avanzando, se espera que el *trading* algorítmico siga evolucionando y desempeñe un papel cada vez más importante en los mercados financieros.

3.1.5 *BACKTESTING*

El *backtest*ing es un proceso utilizado en el desarrollo y evaluación de estrategias de *trading* algorítmico. Consiste en probar una estrategia de inversión utilizando datos históricos para evaluar su desempeño y determinar su viabilidad antes de implementarla en tiempo real.

Durante el *backtest*ing, se utilizan datos pasados para simular la ejecución de la estrategia y calcular los resultados que se habrían obtenido en el pasado. Esto permite evaluar cómo habría funcionado la estrategia en condiciones históricas y proporciona una visión retrospectiva de su rendimiento.

La importancia del *backtest*ing radica en varios aspectos clave. En primer lugar, permite a los inversores y desarrolladores de estrategias evaluar la rentabilidad potencial de una estrategia antes de invertir tiempo y recursos en su implementación en el mercado real. Al

simular la estrategia utilizando datos históricos, se pueden identificar posibles fortalezas y debilidades, así como realizar ajustes y optimizaciones para mejorar su rendimiento.

Además, el *backtest*ing proporciona una base objetiva para tomar decisiones informadas. Al analizar los resultados del *backtest*ing, los inversores pueden evaluar la consistencia de la estrategia a lo largo del tiempo, identificar patrones y tendencias, y comprender mejor el riesgo asociado con la estrategia.

El *backtest*ing también permite la comparación y selección de estrategias. Al probar múltiples estrategias utilizando el mismo conjunto de datos históricos, los inversores pueden comparar su desempeño y seleccionar la estrategia más adecuada para sus objetivos y tolerancia al riesgo.

Es importante destacar que el *backtest*ing tiene limitaciones y suposiciones inherentes. Los resultados obtenidos durante el *backtest*ing se basan en datos históricos y asumen que las condiciones pasadas se repetirán en el futuro. Sin embargo, los mercados financieros son dinámicos y pueden experimentar cambios significativos que pueden afectar el rendimiento de una estrategia en tiempo real. Por lo tanto, el *backtest*ing debe utilizarse como una herramienta complementaria en la toma de decisiones y no como una garantía de resultados futuros.

3.2 ESTADO DE LA CUESTIÓN

En la actualidad, el interés por invertir en el mercado de forma automatizada ha crecido significativamente, impulsado por el auge de la era de la información. Como resultado, han surgido diversas herramientas y plataformas que ofrecen alternativas para realizar inversiones automatizadas. Para entender las soluciones actuales en el mercado, es importante entender que es deseable a la hora de elegir una plataforma de *trading*.

Al buscar una plataforma de *trading* algorítmico, existen varias características clave a considerar. En primer lugar, la plataforma debe proporcionar acceso a datos en tiempo real del mercado y de las empresas, ya que los algoritmos de *trading* se basan en esta información

para tomar decisiones. Además, la plataforma debe ser capaz de conectarse a diferentes mercados y procesar los datos en diversos formatos para adaptarse a las necesidades del inversor. La latencia es otro factor crucial, ya que cualquier retraso en la transmisión de datos puede afectar la efectividad de las estrategias. Por lo tanto, la plataforma debe ofrecer una conectividad rápida y mantener la latencia lo más baja posible. La configurabilidad y personalización son también importantes, ya que permiten a los inversores adaptar las estrategias a sus necesidades y preferencias específicas. La capacidad de escribir programas personalizados, realizar backtesting en datos históricos y la integración con la interfaz de *trading*, son otras características esenciales a tener en cuenta. Por último, la plataforma debe ser compatible con diferentes herramientas de *trading* y ofrecer documentación detallada sobre la lógica subyacente del software (Seth, 2022).

En el campo del *trading* algorítmico, siguiendo estas características, han surgido diversos tipos de plataformas que ofrecen herramientas y servicios para el diseño, desarrollo y ejecución de estrategias automatizadas en los mercados financieros. Estas plataformas han facilitado el acceso de los inversores a la automatización y han permitido el desarrollo de estrategias más sofisticadas. A continuación, se presentan algunas de las principales tipos de programa utilizadas en la actualidad:

- Robots de negociación: Estos programas autónomos, también conocidos como "asesores expertos", están gestionados por algoritmos secretos desarrollados por expertos en *trading*. Los clientes pueden invertir en estos robots a cambio de una tarifa periódica. Aunque ofrecen la ventaja de no requerir conocimientos de *trading* o programación, los inversores depositan su capital en una "*black box*", lo que significa que no tienen pleno conocimiento de cómo se invierte su dinero y en qué instrumentos financieros. Además, en ocasiones, estos algoritmos secretos pueden basarse en estrategias de inversión simples, lo que puede no ser ideal para los inversores que buscan sofisticación en sus inversiones.
- Constructores de estrategias visuales: Estas herramientas permiten a los inversores construir sistemas de *trading* algorítmico utilizando bloques de construcción de indicadores y operadores. A menudo, estos sistemas visuales se convierten en scripts

estandarizados que se ejecutan en plataformas gratuitas como MT4, que ofrecen optimización y backtesting limitados. La ventaja de estas herramientas es que no requieren programación y son fáciles de aprender. Sin embargo, suelen estar limitadas a una biblioteca fija de indicadores estándar y estrategias simplificadas. Además, la construcción visual puede ser más lenta que la codificación manual, y las capacidades de optimización y backtesting pueden ser limitadas.

- Plataformas de negociación al por menor: Estas plataformas están diseñadas principalmente para la negociación manual, pero también admiten lenguajes de programación para el *trading* algorítmico. Estos lenguajes suelen ser propietarios, como EasyLanguage™, MetaStock™ o PineScript™. Ofrecen mayor flexibilidad que los constructores de estrategias visuales, ya que permiten a los usuarios agregar indicadores y funciones personalizadas. Muchas de estas plataformas también admiten métodos más avanzados de backtesting y no se limitan a los pares de divisas y contratos habituales. Sin embargo, su principal desventaja es que requieren conocimientos de programación, a veces en lenguajes propietarios poco prácticos. Además, la integración de bibliotecas externas para análisis de datos o aprendizaje automático puede ser complicada.
- *Trading* algorítmico con lenguajes de programación: Esta opción implica el uso de lenguajes de programación informáticos que ofrecen una amplia gama de librerías y paquetes para análisis de datos, conexión con brókeres, aprendizaje automático, optimización y backtesting. Estos lenguajes no están limitados al comercio y pueden ser utilizados para la investigación y el análisis en diversos campos. Ofrecen una flexibilidad superior a las plataformas de *trading* automatizadas y suelen ser gratuitos. Además, brindan una amplia variedad de bibliotecas y un sólido soporte para el aprendizaje automático. Permiten la implementación de casi cualquier algoritmo de *trading* o proyecto de investigación de mercado. No obstante, su principal limitación es que el usuario debe contar con su propia capacidad computacional y mantenerla operativa las 24 horas del día para operar en los mercados en tiempo real. Los fallos de software o hardware pueden ocasionar pérdidas significativas, lo que limita su uso en entornos de *trading* en vivo. Además,

para algoritmos altamente complejos, puede ser necesario contar con equipos potentes y costosos.

De estas categorías, lo más practicado es realizar el *trading* algorítmico con lenguajes de programación comunes como C++, Matlab o Python y por lo tanto las plataformas más utilizadas y conocidas son las que se centran en brindar servicios de ejecución y gestión de estrategias automatizadas escritas en estos lenguajes (Oberoi, 2023). Estas plataformas, como TradeStation e Interactive Brokers, ofrecen acceso directo a los mercados financieros y permiten a los usuarios ejecutar operaciones automatizadas a través de APIs. Además, proporcionan herramientas de análisis y monitoreo en tiempo real para evaluar el rendimiento de las estrategias y realizar ajustes según sea necesario (TradeStation Group, Inc., 2021; Interactive Brokers LLC, 2021).

Aunque en el contexto de este proyecto, las plataformas dedicadas a la ejecución de estrategias de *trading* no son tan relevantes como las de backtesting de estrategias con datos históricos. Por lo tanto, la atención se centrará en plataformas específicamente diseñadas para backtesting, definiendo las 5 plataformas más populares en la actualidad.

La primera plataforma destacada de backtesting es TrendSpider, una plataforma de análisis técnico relativamente nueva que ofrece herramientas únicas que no se encuentran en otros lugares. Además de su escáner de acciones, la plataforma proporciona capacidades de backtesting potentes. Lo que distingue a TrendSpider es su capacidad para incorporar la mayoría de sus herramientas exclusivas, incluido el reconocimiento automatizado de patrones, en las estrategias evaluadas. Además, la creación de estrategias en TrendSpider es intuitiva y no requiere programación, aunque también se ofrece la opción de escribir código, aunque en su propio lenguaje de programación simplificado (TrendSpider, 2023).

Otra plataforma popular de análisis técnico es TradingView, que cuenta con una amplia comunidad de inversores que comparten sus ideas y estrategias en la plataforma. Esta plataforma basada en la nube incluye todas las características de análisis técnico básicas, y cubre la mayoría de las acciones y otros activos globales. Aunque el backtesting en TradingView no es el más avanzado, ofrece una funcionalidad fácil de usar y presenta los

resultados de forma visualmente comprensible. La creación de estrategias en TradingView requiere programación en Pine Script, el lenguaje de programación nativo de la plataforma. Sin embargo, adaptar e implementar estrategias compartidas por otros inversores es relativamente sencillo (TradingView, 2023).

Trade Ideas es una plataforma avanzada de inteligencia de mercado que se basa en gran medida en la inteligencia artificial. Incorpora numerosos algoritmos impulsados por IA que generan ideas de *trading* y que pueden integrarse en una estrategia de *trading*. Trade Ideas también incluye datos sociales que se integran en los algoritmos. El módulo de herramientas de backtesting en Trade Ideas es intuitivo y no requiere conocimientos de programación. Además, proporciona un análisis muy informativo de los resultados y sugerencias para optimizar una estrategia (Trade Ideas, 2023).

FinViz se centra principalmente en un escáner de acciones que permite filtrar acciones utilizando una combinación de criterios descriptivos, técnicos y fundamentales. En total, existen 70 criterios que se pueden utilizar para reducir el mercado a una lista de vigilancia más manejable. FinViz también cuenta con otras herramientas útiles para ayudarte a estar al tanto del mercado de valores. El módulo de herramientas de backtesting en FinViz es bastante básico y solo incluye indicadores de acción del precio, en lugar de filtros fundamentales y descriptivos. Sin embargo, puede ser una herramienta útil para desarrollar estrategias de *trading* para las acciones que encuentres utilizando el escáner. Esta plataforma también presenta una API integrada con distintos lenguajes de programación como Python o Node.js (FinViz, 2023).

Por último, QuantConnect representa una nueva generación de plataformas para inversores cuantitativos y algorítmicos. Es una plataforma basada en la nube y de código abierto que fomenta la colaboración. QuantConnect ofrece datos fundamentales, de precios y volumen, y permite el *trading* automático. El desarrollo de estrategias, el backtesting y el *trading* con QuantConnect requieren escribir código en Python o C#. Si bien la curva de aprendizaje puede ser pronunciada y se requiere compromiso, QuantConnect ofrece la ventaja de evaluar estrategias basadas en datos tanto fundamentales como de precios (QuantConnect, 2023).

Estas plataformas de backtesting representan el estado del arte en el *backtesting* de estrategias de *trading* algorítmico. Al utilizar estas plataformas, los inversores pueden evaluar sus estrategias y obtener información sobre su posible rendimiento antes de implementarlas en escenarios de *trading* en tiempo real. Es importante destacar que todas estas plataformas tienen costos asociados, y a cambio ofrecen una amplia gama de capacidades y funcionalidades para respaldar la toma de decisiones de *trading* automatizado.

3.3 JUSTIFICACIÓN DEL PROYECTO

En este apartado, se presentarán los fundamentos que respaldan el desarrollo de la plataforma, basados en consideraciones técnicas y de mercado. La justificación busca demostrar la relevancia y el valor que la plataforma aporta a potenciales clientes, emprendedores e inversores.

3.3.1 IDENTIFICACIÓN DE UNA NECESIDAD EN EL TRADING ALGORÍTMICO

El primer punto de justificación se basa en la identificación de una necesidad clara en el ámbito del *trading* algorítmico. A través de un análisis exhaustivo de investigaciones previas, se ha constatado que existe una demanda creciente de soluciones que permitan a los usuarios, incluso sin conocimientos avanzados de programación, diseñar, probar y optimizar estrategias de *trading* automatizadas y aprender poco a poco a programarlas en lenguajes de programación clásicos (Python, C++, Matlab, etc.).

La plataforma desarrollada aborda esta necesidad al proporcionar una interfaz intuitiva y accesible, eliminando las barreras técnicas tradicionalmente asociadas al *trading* algorítmico. Los usuarios, independientemente de su nivel de experiencia en programación, pueden beneficiarse de la capacidad de crear estrategias personalizadas y probar su efectividad en un entorno simulado.

Además, el punto diferencial de este proyecto es que, a pesar de no requerir conocimientos de programación, guía al usuario hacia un aprendizaje de estos conocimientos en un lenguaje tan utilizado como Python. Aprender a programar este tipo de estrategias es fundamental

para cualquier usuario que quiera pasar a invertir, ya que como se ha demostrado en el estado de la cuestión, la mayoría de las plataformas de ejecución de órdenes de *trading* algorítmico hacen uso de estos lenguajes.

3.3.2 COSTE CERO

El segundo punto de justificación se centra en el aspecto económico de la plataforma. Al ofrecer un coste cero para los usuarios, la plataforma elimina la barrera financiera que a menudo impide a muchos individuos y pequeñas empresas acceder a soluciones de *trading* algorítmico de calidad. Esto amplía significativamente el alcance y la democratización del *trading* algorítmico, permitiendo que un mayor número de personas participe en este campo y aproveche sus beneficios potenciales.

Además, el coste nulo de la plataforma también fomenta la experimentación y el aprendizaje sin riesgos para los usuarios. Los individuos interesados en el *trading* algorítmico pueden probar diferentes estrategias y enfoques sin preocuparse por incurrir en gastos financieros significativos. Esto promueve la innovación y el descubrimiento de nuevas estrategias, lo que a su vez puede generar ideas valiosas y contribuir al avance del campo del *trading* algorítmico.

Al eliminar la barrera económica, la plataforma fomenta la participación y la inclusión de diversos perfiles de usuarios, desde aquellos con amplia experiencia en *trading* hasta aquellos que se están iniciando en el campo. Esto contribuye a la creación de una comunidad diversa y enriquecedora, donde los usuarios pueden compartir conocimientos, colaborar y aprender unos de otros.

3.3.3 ENFOQUE EDUCATIVO

El tercer punto de justificación se centra en el enfoque educativo de la plataforma y su posible uso en un club universitario en el futuro. Se ha reconocido la importancia de fomentar el aprendizaje y la participación de los estudiantes en el *trading* algorítmico como una forma de ampliar sus conocimientos financieros y desarrollar habilidades prácticas relevantes para el campo.

La plataforma desarrollada se posiciona como una herramienta valiosa para un club universitario dedicado al *trading* algorítmico. Los estudiantes pueden utilizar la plataforma para aprender sobre estrategias de *trading*, experimentar con diferentes enfoques y evaluar su efectividad en un entorno simulado.

Además, la plataforma ofrece la posibilidad de compartir estrategias y resultados entre los miembros del club, lo que fomenta la colaboración y el intercambio de conocimientos. Los estudiantes pueden trabajar juntos en proyectos de investigación, competiciones de *trading* simulado o incluso en la creación de estrategias personalizadas.

El enfoque educativo de la plataforma brinda a los estudiantes una oportunidad única de adquirir habilidades prácticas en *trading* algorítmico, complementando su formación académica y preparándolos para futuras carreras en el campo financiero.

En resumen, los tres puntos de justificación abordan aspectos distintos y complementarios. Se destaca la identificación de una necesidad en el *trading* algorítmico, el coste cero de la plataforma, así como su enfoque educativo y su posible uso en un club universitario. Estos elementos buscan convencer a potenciales clientes, emprendedores e inversores sobre la relevancia y el potencial éxito de la plataforma en el mercado del *trading* algorítmico y su utilidad en un contexto educativo.

Capítulo 4. DISEÑO DE LA PLATAFORMA

En este capítulo se presenta el diseño de la plataforma Trader Companion, donde se describen los diversos sistemas que la componen y cómo interactúan entre sí para brindar la funcionalidad requerida. Es fundamental destacar que la plataforma ha sido desarrollada en inglés con el objetivo de fomentar su adopción por parte de un público más amplio y facilitar el uso de términos comúnmente utilizados en el ámbito del *trading* algorítmico, los cuales suelen estar en inglés.

4.1 SISTEMAS DE ALTO NIVEL

El diseño de la plataforma se ha concebido cuidadosamente para proporcionar una experiencia fluida y eficiente a los usuarios, permitiéndoles aprovechar al máximo las herramientas y funcionalidades disponibles. Para lograr este objetivo, se han considerado aspectos como la modularidad, la flexibilidad y la escalabilidad en la arquitectura de la plataforma.

La plataforma se basa en un enfoque modular, lo que significa que se compone de diferentes sistemas interrelacionados. Estos sistemas incluyen un controlador, un conjunto de vistas, una conexión a una API externa, un conjunto de bases de datos y un módulo de estrategias. Cada uno de estos sistemas cumple un papel específico y colabora en conjunto para brindar una experiencia integral a los usuarios.

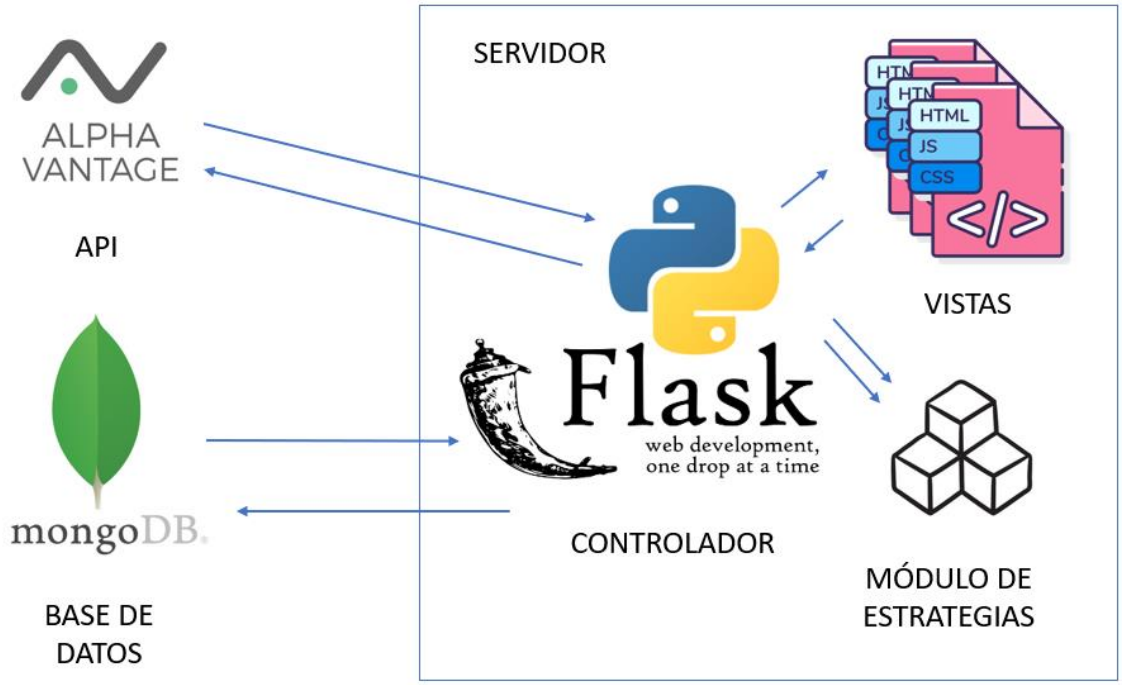


Figura 2 - Arquitectura global de la plataforma

El controlador y el conjunto de vistas constituyen la interfaz de usuario de la plataforma. El controlador se encarga de procesar las solicitudes del usuario y coordinar la ejecución de las operaciones correspondientes. Las vistas, por otro lado, son responsables de presentar la información al usuario de forma visualmente atractiva y fácil de entender.

La conexión a la API externa se establece mediante la librería "alpha_vantage" en Python. Esta API proporciona acceso a los datos de los activos bursátiles, que son necesarios para ejecutar las estrategias de *backtest*ing. La plataforma utiliza esta conexión para obtener los datos actualizados de los activos seleccionados por el usuario y realizar los cálculos correspondientes.

El conjunto de bases de datos juega un papel fundamental en el almacenamiento y recuperación de la información relevante para la plataforma. En este caso, se utiliza MongoDB como sistema de gestión de bases de datos NoSQL. Se utilizan cuatro bases de datos diferentes para almacenar datos relacionados con los precios históricos de los activos bursátiles, los usuarios registrados, las estrategias y los resultados de los *backtests*. Cada

base de datos tiene una estructura específica para almacenar los datos de manera eficiente y permitir su posterior procesamiento.

El módulo de estrategias es una parte clave de la plataforma, ya que permite a los usuarios implementar y ejecutar estrategias de *backtest*ing. Este módulo se basa en un enfoque modular, donde las estrategias se definen como clases individuales ubicadas en el directorio "modules". Estas estrategias son importadas dinámicamente por la plataforma y utilizadas para ejecutar los *backtests* solicitados por los usuarios.

En resumen, la plataforma sigue una arquitectura modular y flexible que facilita la adición de nuevas estrategias, la gestión de usuarios y la generación de resultados de *backtest*ing de manera eficiente y escalable. Los diferentes sistemas trabajan en conjunto para proporcionar una experiencia fluida y funcional a los usuarios, permitiéndoles ejecutar estrategias de *backtest*ing y analizar los resultados de manera efectiva.

4.2 INTERFAZ DE USUARIO

En esta sección, se describirán a alto nivel la interfaz gráfica elegida para cumplir con los objetivos de nuestra plataforma en cada una de las vistas de la aplicación web. La interfaz de usuario de Trader Companion consta de 8 vistas programadas en HTML, cada una diseñada para proporcionar una experiencia intuitiva y funcional a los usuarios. A continuación, se detallará cada una de las vistas acompañada por una imagen de ejemplo.

4.2.1 PÁGINA PRINCIPAL

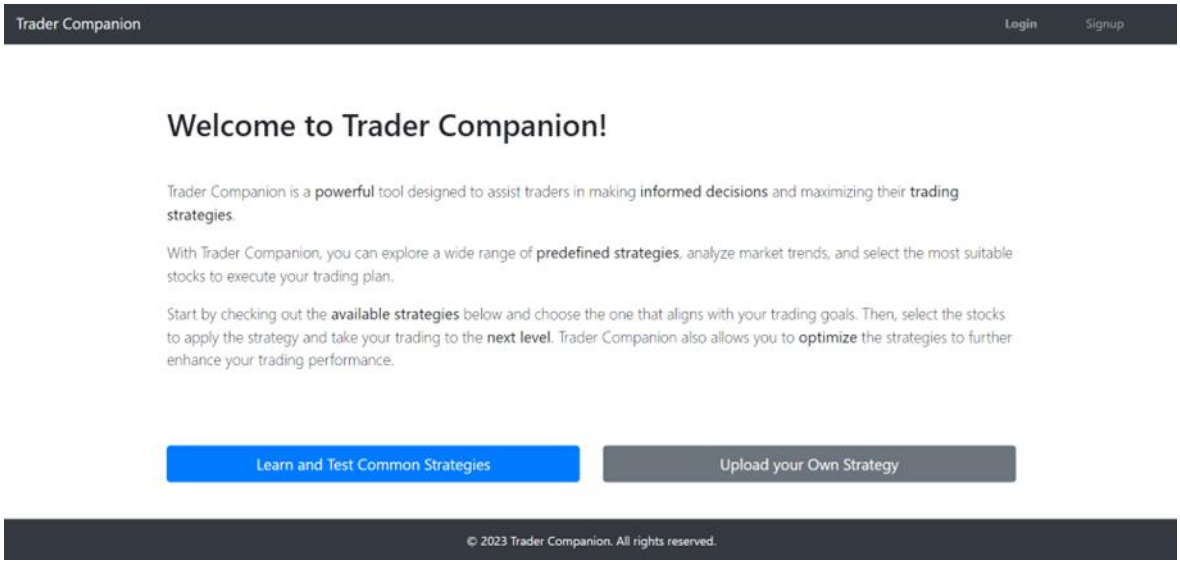


Figura 3 - Página principal de la plataforma

El diseño web de la página principal de Trader Companion ha sido desarrollado estratégicamente para brindar a los usuarios una experiencia intuitiva y atractiva, al mismo tiempo que les permite acceder y utilizar las funcionalidades ofrecidas por la plataforma de manera efectiva. Este diseño se enfoca en resaltar las características clave y facilitar la navegación del usuario. A continuación, se describe cómo el diseño web se relaciona con las funcionalidades proporcionadas:

El encabezado de la página presenta una barra de navegación bien estructurada que permite a los usuarios moverse sin dificultad por las diferentes secciones de la plataforma, como los *backtests* y las páginas de inicio de sesión y registro. Esto garantiza que los usuarios puedan acceder rápidamente a las funcionalidades relevantes.

El diseño web utiliza un enfoque flexible para garantizar que la página se adapte a diferentes dispositivos y tamaños de pantalla. Esto es fundamental para permitir a los usuarios acceder y utilizar la plataforma desde una variedad de dispositivos, como computadoras de escritorio, tabletas y dispositivos móviles.

El contenido principal de la página se centra en presentar de manera destacada las características clave de Trader Companion. Los párrafos de descripción están diseñados para captar la atención del usuario, resaltando la capacidad de explorar estrategias predefinidas, analizar tendencias del mercado y cargar estrategias propias. Esto ayuda a los usuarios a comprender rápidamente las funcionalidades disponibles y los beneficios que pueden obtener.

El diseño web utiliza botones llamativos para invitar a los usuarios a interactuar con la plataforma. Estos botones están estratégicamente ubicados y diseñados visualmente para resaltar su importancia. Al hacer clic en estos botones, los usuarios pueden acceder a secciones específicas de la plataforma, como aprender y probar estrategias comunes o cargar sus propias estrategias.

4.2.2 REGISTRO E INICIO DE SESIÓN

Figura 4 - Página de registro de la plataforma

Figura 5 - Página de inicio de sesión de la plataforma

Las dos vistas, "Login" y "Signup", comparten un diseño web coherente y siguen una estructura similar para mantener la consistencia visual en la plataforma Trader Companion. El encabezado en ambas vistas es consistente y presenta la marca "Trader Companion", lo que ayuda a los usuarios a identificar y reconocer fácilmente la plataforma en ambas páginas. Esto proporciona una experiencia de usuario cohesiva y refuerza la asociación entre las funcionalidades y la marca.
El diseño web de ambas vistas utiliza una barra de navegación simple y minimalista en el encabezado, que se mantiene constante en todas las páginas. Esto permite a los usuarios moverse sin problemas entre diferentes secciones de la plataforma, incluyendo las páginas de inicio de sesión y registro. La barra de navegación mantiene la coherencia y la accesibilidad, asegurando que los usuarios puedan acceder rápidamente a estas funcionalidades clave.

El contenido principal en ambas vistas se presenta en un formato de formulario que solicita información del usuario, ya sea para iniciar sesión o registrarse. Los formularios están diseñados de manera clara y concisa, con etiquetas y campos de entrada bien definidos para garantizar que los usuarios comprendan qué información deben proporcionar. Además, se incluyen mensajes de error en caso de que haya problemas con los datos ingresados, lo que ayuda a los usuarios a corregir y resolver los errores.

El diseño web utiliza estilos consistentes y una estructura de diseño similar en ambas vistas. Esto ayuda a los usuarios a familiarizarse rápidamente con la plataforma y a navegar sin problemas entre las páginas de inicio de sesión y registro. La consistencia visual mejora la usabilidad y proporciona una experiencia de usuario fluida.

4.2.3 CREACIÓN DE ESTRATEGIAS

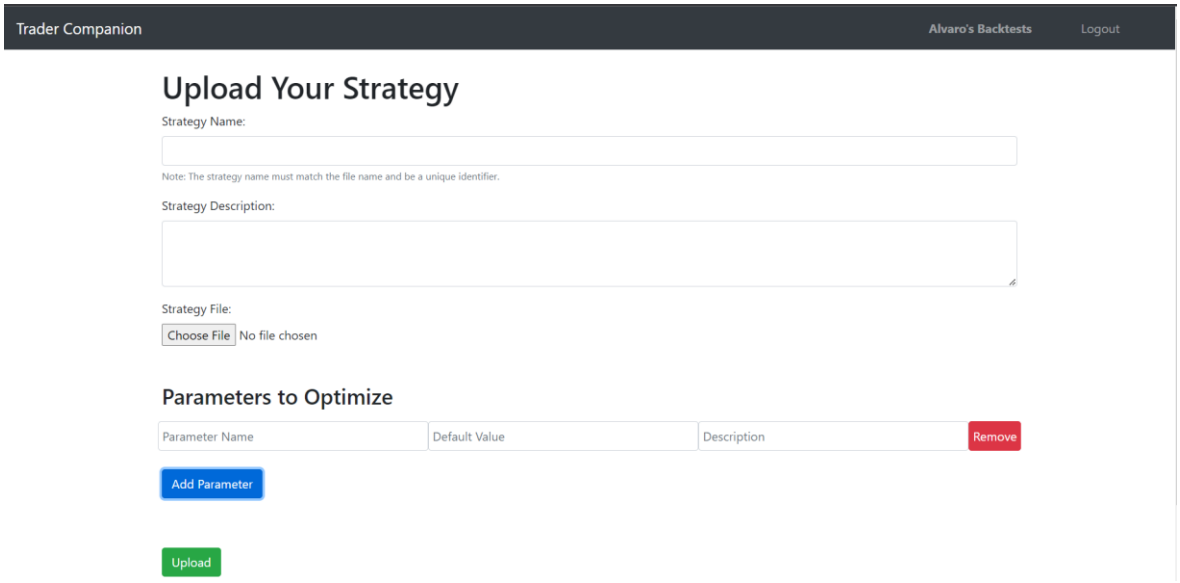


Figura 6 - Página de creación de estrategias de la plataforma

La vista de creación de estrategias en Trader Companion ha sido cuidadosamente diseñada para ofrecer a los usuarios la capacidad de cargar estrategias personalizadas en la plataforma. El diseño web se ha centrado en proporcionar una interfaz intuitiva y fácil de usar, con el objetivo de simplificar el proceso de carga de estrategias.

La página web presenta un diseño limpio y bien estructurado, que permite a los usuarios comprender rápidamente los pasos necesarios para cargar una estrategia. El contenido principal de la vista se encuentra en el cuerpo de la página, donde se presenta un formulario interactivo que guía a los usuarios a través del proceso de carga de la estrategia. El formulario consta de varios campos que solicitan información relevante para la estrategia. Estos campos incluyen:

- "Strategy Name": Este campo requiere que los usuarios ingresen un nombre descriptivo para la estrategia que están cargando. Proporcionar un nombre claro y significativo ayuda a los usuarios a identificar y recordar sus estrategias de manera efectiva.

- "Strategy Description": En este campo, los usuarios pueden proporcionar una descripción detallada de la estrategia que están cargando. Esto les permite explicar el propósito, la lógica y las características clave de su estrategia, lo que facilita la comprensión para ellos mismos y para otros usuarios.
- "Strategy File": Este campo permite a los usuarios seleccionar y cargar el archivo de la estrategia en formato ".py". Al proporcionar una opción para cargar el archivo de la estrategia, los usuarios pueden transferir sus estrategias existentes de manera conveniente y eficiente.

Además de los campos básicos, el formulario ofrece la capacidad de agregar parámetros para la optimización de la estrategia. Esto permite a los usuarios personalizar su estrategia al especificar los valores de los parámetros y proporcionar descripciones relevantes. Los campos de parámetros se pueden agregar dinámicamente haciendo clic en el botón "Add Parameter". Cada campo de parámetro incluye espacios para ingresar el nombre del parámetro, el valor predeterminado y una descripción.

El diseño visual de la página utiliza la biblioteca de estilos Bootstrap para garantizar una apariencia moderna y coherente en todos los dispositivos. Se ha elegido una combinación de colores cuidadosamente seleccionados para mejorar el contraste y la legibilidad del contenido. Además, se ha utilizado una tipografía clara y legible para facilitar la lectura y comprensión de la información presentada.

4.2.4 LIBRERÍA DE ESTRATEGIAS

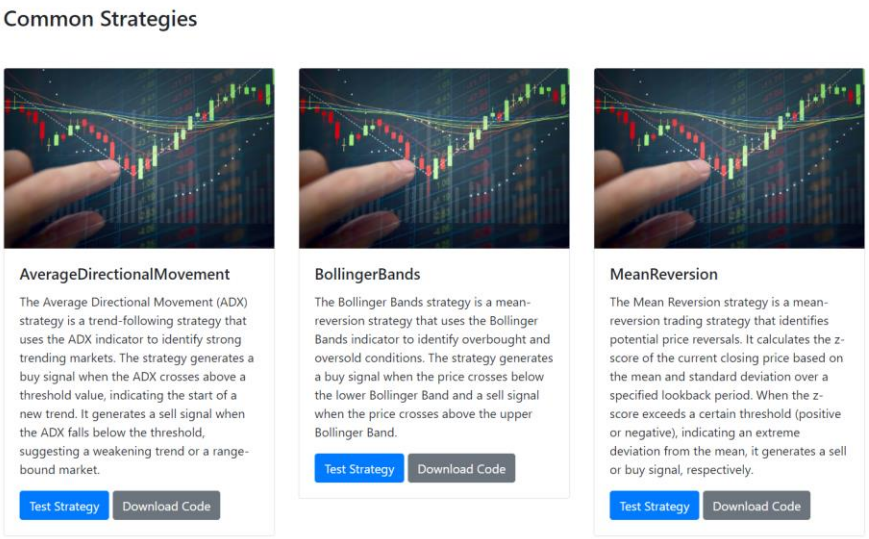


Figura 7 - Página de librería de estrategias de la plataforma

La vista de librería de estrategias en Trader Companion se ha diseñado cuidadosamente para ofrecer a los usuarios una interfaz web atractiva y funcional. El diseño web se ha centrado en proporcionar una experiencia fluida y accesible, brindando a los usuarios las herramientas necesarias para explorar y evaluar las estrategias disponibles en la plataforma. A continuación, se describen las características principales del diseño de esta vista.

La presentación de las estrategias se realiza mediante tarjetas individuales que contienen información esencial sobre cada estrategia. Cada tarjeta muestra el identificador de la estrategia y una breve descripción que resume sus características principales. Estas tarjetas se organizan en columnas, lo que permite una visualización clara y ordenada de las estrategias disponibles. Además, se incluyen imágenes ilustrativas en cada tarjeta para ofrecer una representación visual de las estrategias y mejorar la experiencia del usuario.

Las tarjetas de estrategia presentan acciones interactivas que permiten a los usuarios llevar a cabo acciones específicas. Estas acciones incluyen la posibilidad de "Try Strategy" y "Test Code", lo que permite a los usuarios evaluar y poner en práctica las estrategias seleccionadas. Estos elementos interactivos están diseñados para ser fácilmente identificables y accesibles, brindando a los usuarios un control directo sobre las funcionalidades de la plataforma.

En el caso de los usuarios registrados, se incluye una sección personalizada que muestra las estrategias específicas creadas por cada usuario. Esta sección personalizada facilita a los usuarios el acceso y la gestión de sus propias estrategias, ofreciendo una vista rápida y conveniente de su cartera de estrategias.

El diseño visual de la vista se basa en una paleta de colores oscuros que se alinea con la identidad de marca de Trader Companion. Se utiliza una combinación de elementos visuales coherentes en toda la plataforma para asegurar una apariencia atractiva y crear una experiencia de usuario consistente.

4.2.5 SELECCIÓN DE ACCIONES

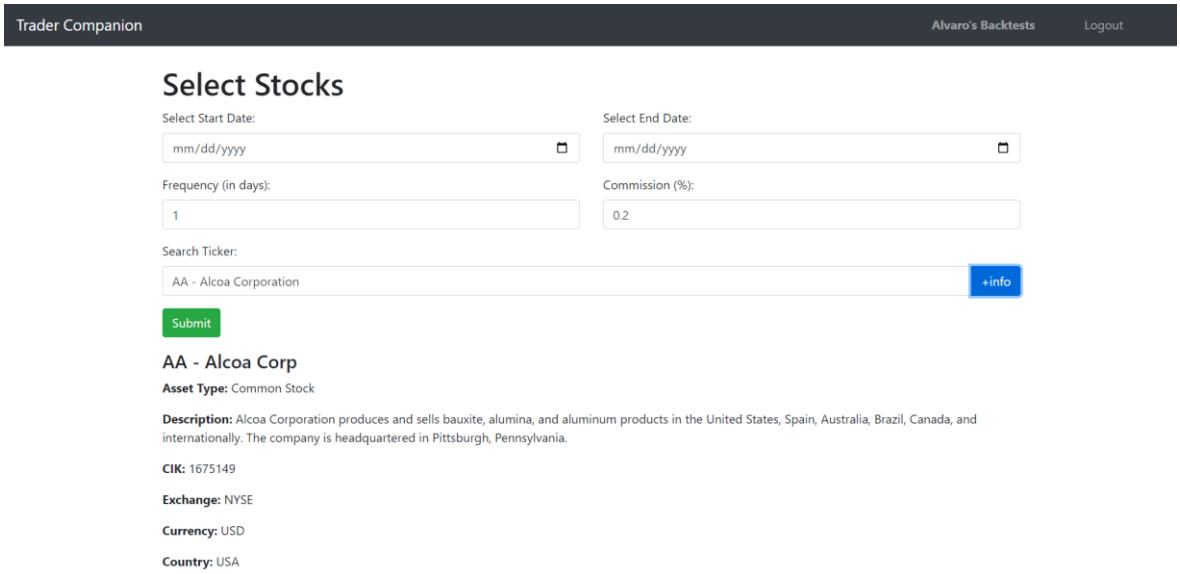


Figura 8 - Página de selección de acciones de la plataforma

La vista de selección de acciones representa una parte integral del diseño web de la plataforma. Su objetivo principal es permitir a los usuarios elegir las acciones en las que desean aplicar una estrategia, brindándoles una experiencia interactiva y amigable.

Esta vista se presenta como un formulario interactivo, el cual se encuentra cuidadosamente diseñado para facilitar la configuración de los parámetros necesarios para el *backtest*ing de estrategias. En primer lugar, se destacan dos campos de fecha, "Select Start Date" y "Select End Date", que permiten a los usuarios establecer el período de tiempo en el cual desean

evaluar la estrategia. Estos campos están diseñados de manera intuitiva y proporcionan a los usuarios un medio eficiente para especificar fechas válidas dentro de un rango predefinido.

Además de los campos de fecha, el formulario incluye otras opciones importantes que permiten a los usuarios ajustar el *backtest*ing de acuerdo con sus necesidades. Estos elementos adicionales incluyen la posibilidad de especificar la frecuencia de los datos utilizados en el *backtest*ing y la comisión aplicada durante las operaciones. La frecuencia se presenta como un campo numérico en días, con un valor predeterminado de 1, lo que permite a los usuarios seleccionar el intervalo de tiempo deseado entre cada punto de datos utilizado en el *backtest*ing. Por otro lado, el campo de comisión permite a los usuarios establecer el porcentaje que se aplicará durante las transacciones, brindando flexibilidad para adaptar la estrategia a las condiciones y preferencias del usuario.

Para facilitar la selección de acciones, se ha incluido un cuadro de búsqueda que permite a los usuarios ingresar el *ticker* de las acciones deseadas. Este cuadro de búsqueda se ha mejorado mediante la implementación de la funcionalidad de autocompletado, lo que proporciona sugerencias en tiempo real a medida que los usuarios ingresan el *ticker*. Esta característica agiliza el proceso de selección y garantiza una experiencia de usuario más fluida.

Una vez que los usuarios han ingresado el *ticker* de una acción, se muestra una sección adicional que presenta información detallada sobre la acción seleccionada. Esta información incluye datos relevantes como el símbolo del *ticker*, el nombre de la acción, el tipo de activo, una descripción, el intercambio en el que se negocia la acción, la moneda utilizada, el país de origen, el sector al que pertenece, la industria relacionada, la dirección de la empresa, la capitalización de mercado, las ganancias por acción (EPS) y los ingresos generados en los últimos 12 meses (Revenue TTM). Estos detalles proporcionan a los usuarios una visión completa y precisa de cada acción, lo que les permite tomar decisiones informadas al seleccionar las acciones adecuadas para su estrategia.

Una vez que los usuarios han completado el formulario, pueden enviar su selección de acciones y otros parámetros relacionados haciendo clic en el botón de envío "Submit". Este

botón inicia el proceso de *backtest*ing y redirige a los usuarios a la página de resultados correspondiente, donde podrán visualizar los resultados obtenidos.

4.2.6 RESULTADOS DEL *BACKTEST*

Figura 9 - Página de resultados del backtest de la plataforma [1/2]

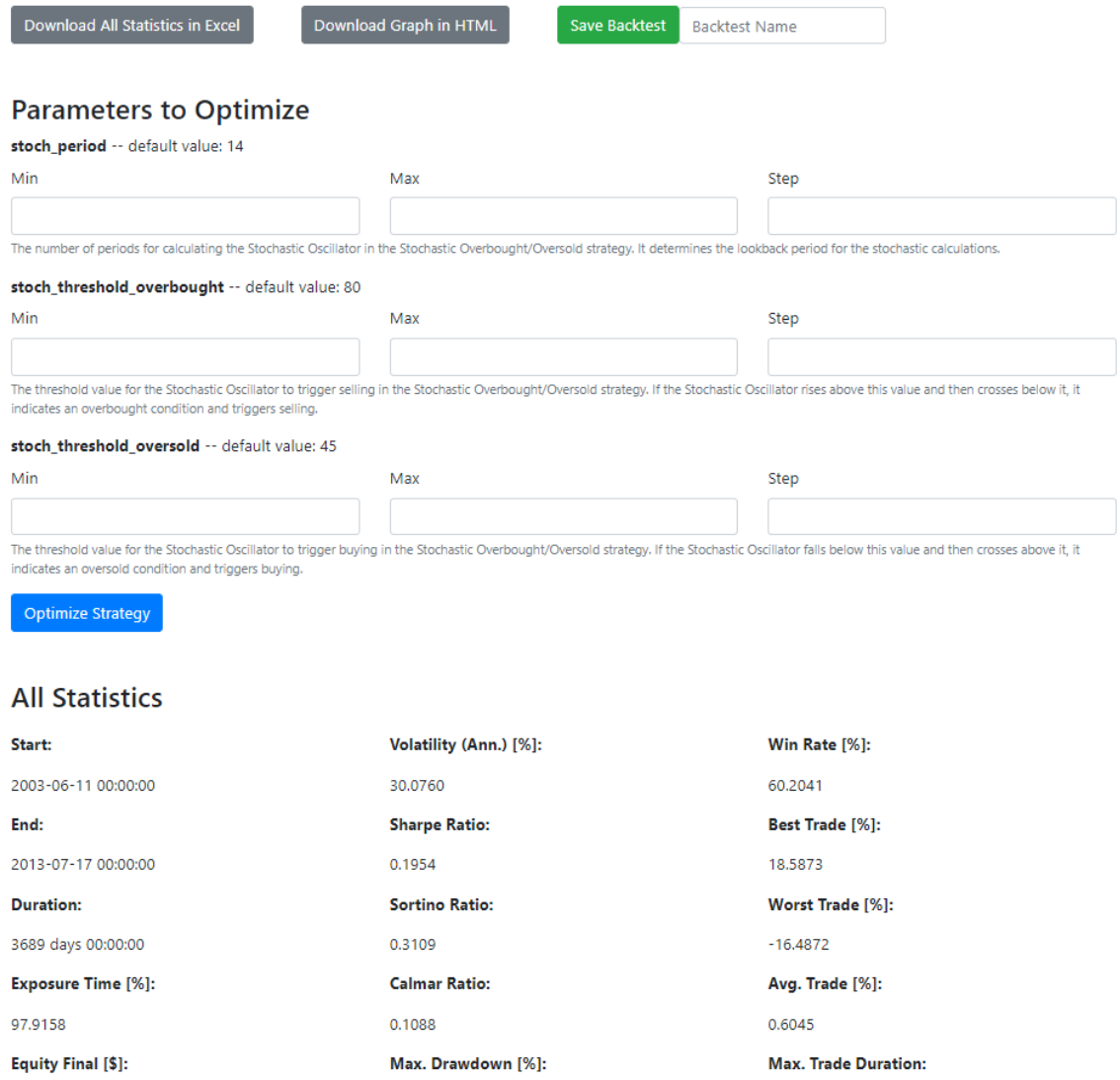


Figura 10 - Página de resultados del backtest de la plataforma [2/2]

La vista de resultados del *backtest* en Trader Companion muestra los resultados de una estrategia específica ejecutada en la plataforma. Esta vista ha sido diseñada cuidadosamente para proporcionar a los usuarios una visualización clara y concisa de los resultados de su estrategia. A continuación, se describen las características clave del diseño de esta vista.

La vista muestra una pantalla de carga al principio, lo que brinda a los usuarios una indicación visual de que la estrategia se está ejecutando. Esta pantalla de carga se muestra utilizando estilos personalizados definidos en la sección de estilos CSS.

Una vez que la estrategia ha terminado de ejecutarse, se muestra el contenido principal de la vista. El contenido principal está oculto al principio y se muestra una vez que se obtienen los resultados de la estrategia. El contenido principal se divide en varias secciones:

- "Key Indicators" - Esta sección muestra los indicadores clave generados por la estrategia. Los indicadores clave se presentan en forma de una tabla con nombres de indicadores y sus respectivos valores. Los indicadores clave proporcionan una visión general de alto nivel de los resultados de la estrategia.
- “Graph” – Esta sección muestra el gráfico generado con la librería “Backtesting.py” que contiene el comportamiento en detalle de la estrategia a lo largo del periodo de tiempo elegido.
- "Saving and Download Buttons" - Esta sección contiene botones que permiten a los usuarios guardar el *backtest* en su librería de resultados, y descargar los resultados en formato Excel o la gráfica en formato HTML. Los usuarios pueden proporcionar un nombre personalizado para el *backtest* que deseen guardar.
- "Optimization" - Esta sección muestra un formulario con los parámetros que se pueden optimizar de la estrategia y sus valores por defecto. El usuario puede introducir rangos de valores para estos parámetros, que son comprobados por el servidor y la página se recarga con la mejor combinación.
- "All Statistics" - Esta sección muestra todas las estadísticas generadas por la estrategia. Las estadísticas se presentan en forma de columnas, con cada columna que contiene múltiples estadísticas relacionadas. Las estadísticas se muestran en forma de pares clave-valor, donde el nombre de la estadística se muestra en negrita y su valor correspondiente se muestra al lado.

El diseño visual de la página utiliza la biblioteca de estilos Bootstrap para garantizar una apariencia moderna y coherente en todos los dispositivos. Se han aplicado estilos personalizados utilizando CSS adicional para controlar el tamaño y el diseño del contenido.

4.2.7 LIBRERÍA DE *BACKTESTS*

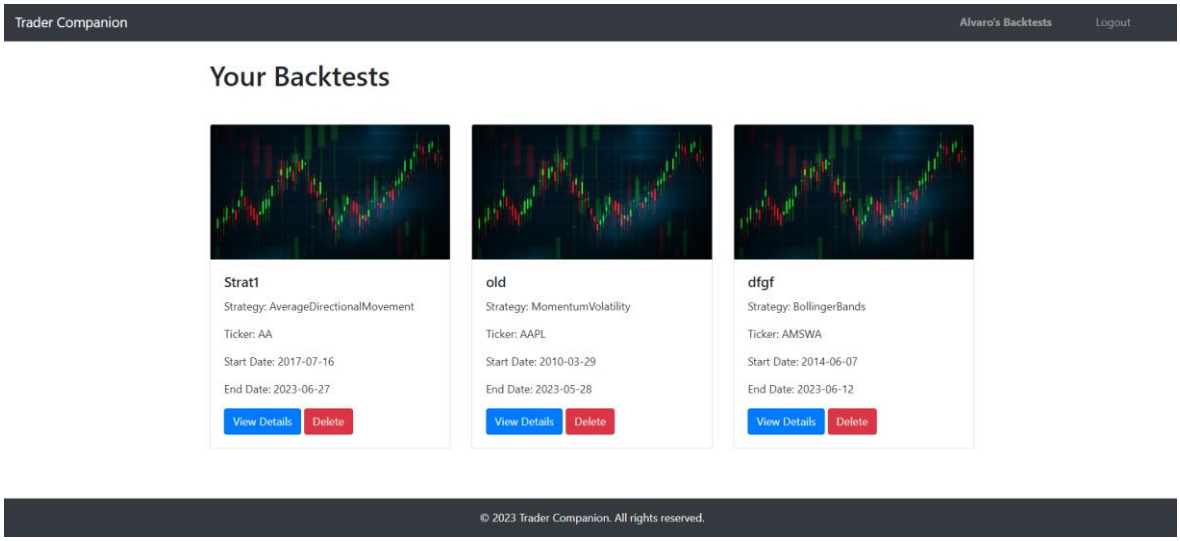


Figura 11 - Página de librería de backtests de la plataforma

La vista de librería de *backtests* en Trader Companion ha sido diseñada con el objetivo de proporcionar a los usuarios una interfaz web clara y funcional para gestionar sus pruebas de estrategias. El diseño web se centra en facilitar la visualización y gestión de los *backtests* realizados por los usuarios. A continuación, se describen las principales características del diseño de esta vista.

La interfaz de usuario se presenta de manera ordenada y estructurada, lo que permite una fácil comprensión de la información. La barra de navegación superior proporciona acceso rápido a otras secciones relevantes de la plataforma, como el perfil de usuario, el registro de *backtests* y las opciones de inicio de sesión y cierre de sesión. Esto permite a los usuarios navegar de manera fluida y acceder a las funcionalidades de la plataforma de forma intuitiva.

El contenido principal de la vista se encuentra en la sección central, donde se presenta una lista de los *backtests* realizados por el usuario. Los *backtests* se muestran en forma de tarjetas individuales que contienen información clave sobre cada prueba. Cada tarjeta muestra el nombre del *backtest*, la estrategia asociada, el *ticker* utilizado, la fecha de inicio y la fecha de finalización del *backtest*. Esta presentación estructurada permite a los usuarios tener una visión clara de sus pruebas y acceder rápidamente a los detalles relevantes.

Además de la información básica del *backtest*, las tarjetas incluyen botones interactivos que permiten a los usuarios llevar a cabo acciones específicas. Por ejemplo, se incluye un botón "View Details" que redirige a los usuarios a una página donde pueden ver los detalles completos del *backtest*, como los resultados, gráficos y estadísticas. También se incluye un botón "Delete" que permite a los usuarios eliminar un *backtest* específico de su lista. Estos elementos interactivos están diseñados para ser fácilmente identificables y accesibles, brindando a los usuarios un control directo sobre la gestión de sus pruebas.

4.3 ALMACENAMIENTO Y MODELO DE DATOS

En esta sección se hará una descripción detallada de la estructura de almacenamiento empleada en este proyecto. Aunque como se ha comentado previamente existen un módulo de estrategias dentro del servidor donde se almacenan clases de Python que no son serializables, la mayoría de los datos utilizados en la web son serializables y se almacenan en MongoDB. En concreto, en este proyecto se hace uso de 4 bases de datos con objetivos distintos y cuyos datos están interconectados.

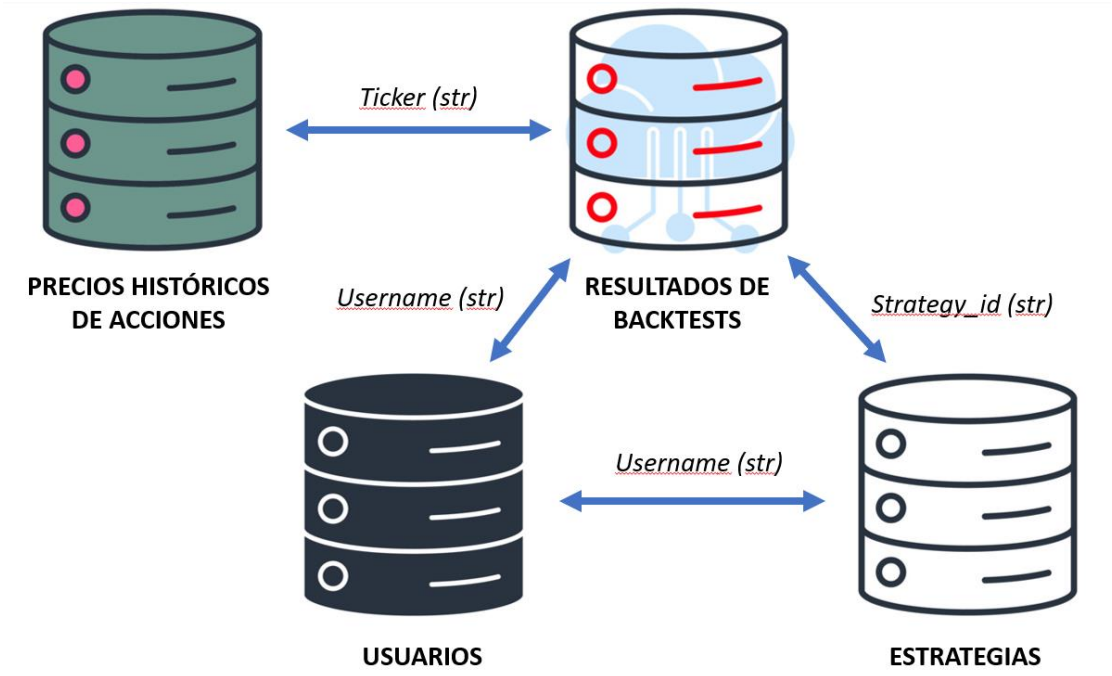


Figura 12 - Modelo de datos (Fuente de los iconos: https://www.vecteezy.com/free-vector/database)

4.3.1 PRECIOS HISTÓRICOS DE ACCIONES

Los datos de precios históricos se almacenan en MongoDB en su propia base de datos llamada "stock_cache_db". Según el esquema seguido en esta base de datos cada colección representa la caché de datos bursátiles para un *ticker* específico, y cada documento dentro de la colección representa una única entrada de datos bursátiles para una fecha específica. El nombre de cada colección se identifica por el *ticker*, que es único para cada activo en el que se puede invertir. La información almacenada cada vez que un usuario utiliza una estrategia sobre un nuevo activo es la siguiente:

- **`_id:`** El identificador único de cada documento de datos bursátiles.
- **`Date:`** La fecha de entrada de los datos bursátiles.
- **`Open:`** El precio de apertura de la acción en la fecha dada.
- **`High:`** Precio máximo de la acción en la fecha indicada.
- **`Low:`** Precio mínimo de la acción en la fecha indicada.
- **`Close:`** El precio de cierre de la acción en una fecha determinada.
- **`Volume:`** El volumen de negociación de la acción en la fecha indicada.

4.3.2 USUARIOS

Los datos de usuarios registrados se almacenan en MongoDB en su propia base de datos llamada "user_database". Esta base de datos presenta una única colección con el nombre de "users", donde se almacena la siguiente información cada vez que un usuario se registra:

- **`_id:`** El identificador único para cada documento de usuario.
- **`Username:`** El nombre de usuario del usuario.
- **`Password:`** La contraseña hash del usuario.

4.3.3 ESTRATEGIAS

Los datos de las estrategias tanto por defecto como registradas por los usuarios se almacenan en MongoDB en su propia base de datos llamada "strategies_db". Esta base de datos presenta

una única colección con el nombre de "strategies", donde se almacena la siguiente información:

- **`_id:`** El identificador único para cada documento de estrategia.
- **`Username:`** Campo que indica el usuario que tiene acceso a la estrategia. Existe un valor predeterminado "all" en las estrategias por defecto, indicando que la estrategia está disponible para todos los usuarios. El término "all se trata de un usuario reservado para el administrador de la plataforma y por ello no es posible introducir nuevas estrategias por defecto siendo un usuario común.
- **`Strategy_id:`** El identificador de la estrategia.
- **`Description:`** Una descripción de la estrategia, proporcionando una visión general de su propósito y metodología. Atiende a la faceta pedagógica del proyecto
- **`Parameters:`** Un diccionario que almacena los detalles de los parámetros de la estrategia. Cada parámetro está representado por un par clave-valor, donde la clave es el nombre del parámetro y el valor es un diccionario que contiene el valor por defecto y una descripción del parámetro.

En esta base de datos se puede observar la gran flexibilidad de MongoDB respecto a otras bases de datos, ya que el campo parameters está compuesto por un diccionario de tamaño variable en función de la estrategia, esquema que no sería posible en bases de datos convencionales y relacionales.

4.3.4 RESULTADOS DE BACKTESTS

Los resultados de los *backtest* realizados por los usuarios se almacenan en MongoDB en la base de datos llamada "*backtests*_db". Esta base de datos presenta una única colección con el nombre de "*backtests*", donde se almacena la siguiente información:

- **`_id:`** El identificador único para cada entrada del *backtest*.
- **`Name:`** El nombre que el usuario da a ese *backtest*.
- **`Username:`** El nombre de usuario del usuario que realizó el *backtest*.

- **`Bt_object:`** Los datos binarios serializados que representan el objeto del *backtest*. Es necesario almacenar el objeto al completo para poder realizar optimizaciones de parámetros sobre él posteriormente
- **`Commission:`** La comisión aplicada durante el *backtest*.
- **`End_date:`** La fecha de finalización del periodo de *backtest*.
- **`Frequency:`** La frecuencia de los datos utilizados en el *backtest*.
- **`Permanent:`** Una variable binaria que indica si el *backtest* es permanente o temporal. Es utilizada para guardar temporalmente el *backtest* caso de que el usuario quiera optimizarlo sin haberlo guardado en su librería.
- **`Start_date:`** Fecha de inicio del periodo de *backtest*.
- **`Strategy_id:`** El identificador de la estrategia de negociación utilizada en el *backtest*.
- **`Ticker:`** Símbolo del activo bursátil utilizado en el *backtest*.

Una vez más se muestra el potencial inmenso de utilizar MongoDB como base de datos, ya que nos permite guardar un objeto de nuestro programa serializado. De esta forma se mantiene toda la información del objeto y no es necesario utilizar recursos computacionales para recrearlo, es posible trabajar sobre el objeto previamente creado.

Capítulo 5. IMPLEMENTACIÓN DE LA PLATAFORMA

En este capítulo, se aborda en detalle la implementación de Trader Companion, la plataforma diseñada para ofrecer a los usuarios una experiencia didáctica el mundo del *trading* algorítmico. A lo largo de este capítulo, se explorarán las funcionalidades clave desarrolladas, el enfoque utilizado en el proceso de implementación y la integración del *frontend* y *backend* para brindar una experiencia fluida y atractiva. Es importante mencionar que todo el proyecto se ha realizado en un entorno de desarrollo en el disco local de un ordenador, por lo que no hay indicaciones de implementación en un entorno de producción, que queda fuera del alcance del proyecto.

5.1 DESARROLLO DE FUNCIONALIDADES CLAVE

En esta sección, se presentará el núcleo de la plataforma Trader Companion, que permite a los usuarios crear, probar y optimizar estrategias de *trading* utilizando la librería de Python denominada "*Backtest*ing.py". Esta librería despliega un *framework* especializado en la evaluación de la viabilidad de estrategias de negociación basadas en datos históricos, simplificando el proceso de desarrollo, prueba y obtención de métricas relevantes.

La librería "*Backtest*ing.py" se posiciona como una herramienta fundamental en este proyecto, en el cual se ha integrado con otras tecnologías y recursos complementarios. En particular, se emplea la librería "TA-Lib" para obtener funciones predefinidas sobre series temporales que actúan como indicadores en las estrategias, y se hace uso de la API de AlphaVantage como fuente de datos históricos de precios. Esta sinergia entre las diferentes tecnologías permite establecer una lógica sólida y versátil en la plataforma, brindando a los usuarios una experiencia completa y confiable.

El enfoque central en el desarrollo de funcionalidades clave en Trader Companion garantiza que los usuarios tengan a su disposición una plataforma poderosa y eficiente para el diseño, evaluación y mejora de estrategias de *trading*. A lo largo de esta sección, se examinará

detalladamente la aplicación de la librería "*Backtest*ing.py", así como su integración con las librerías "TA-Lib" y la API de AlphaVantage, permitiendo a los usuarios aprovechar al máximo estas herramientas y alcanzar resultados óptimos en sus actividades de *trading*.

5.1.1 PROGRAMACIÓN DE ESTRATEGIAS

En esta sección, se abordará la programación de estrategias utilizando la librería "*Backtest*ing.py" en el marco de la plataforma Trader Companion. Para ilustrar el proceso, se presentará un ejemplo práctico de una estrategia denominada "Volatility Breakout", la cual busca capturar los quiebres de precios basados en la volatilidad reciente.

```
from backtesting import Strategy
import talib as ta

class VolatilityBreakout(Strategy):
    lookback_period = 20
    volatility_factor = 1.5
    stop_loss_percentage = 0.02
    take_profit_percentage = 0.02

    def init(self):
        # Calculate the highest high and lowest low over the lookback period
        self.high_high = self.I(ta.MAX, self.data.High, self.lookback_period)
        self.low_low = self.I(ta.MIN, self.data.Low, self.lookback_period)

    def next(self):
        close = self.data.Close[-1]
        # Calculate stop loss adn take profit signals
        long_sl = close * (1 - self.stop_loss_percentage)
        long_tp = close * (1 + self.take_profit_percentage)
        short_sl = close * (1 + self.stop_loss_percentage)
        short_tp = close * (1 - self.take_profit_percentage)
        # Calculate breakout levels
        breakout_high = self.high_high[-2] * self.volatility_factor
        breakout_low = self.low_low[-2] * self.volatility_factor

        # Buy signal: Close price exceeds the breakout high level, and include
stop loss adn take profit signals on the order
        if close > breakout_high:
            self.buy(sl=long_sl, tp=long_tp)
        # Sell signal: Close price falls below the breakout low level, and
include stop loss adn take profit signals on the order
        elif close < breakout_low:
            self.sell(sl=short_sl, tp=short_tp)
```

En el código anterior, se define una clase llamada "VolatilityBreakout" que hereda de la clase "Strategy" del *framework*. Dentro de esta clase, se definen los parámetros y variables necesarios para la estrategia.

El método “init” se utiliza para realizar la inicialización de variables y cálculos requeridos antes de ejecutar la estrategia. En este método se han de hacer todos los cálculos posibles de indicadores de forma vectorizada, para disminuir los recursos empleados. En este ejemplo, se calculan los precios máximos y mínimos en el periodo seleccionado, 20 días, utilizando las funciones ta.MAX y ta.MIN de la librería "TA-Lib". Estos valores serán los indicadores de la futura estrategia.

El método “next” es el núcleo de la estrategia, donde se implementa la lógica de toma de decisiones. En este método se incluye el código que se ejecuta en cada punto del tiempo en el que se comprueba la estrategia. La frecuencia de tiempo entre llamadas a esta función dependerá de la frecuencia de los datos de entrada, que es elegida por el usuario como se describió en las interfaces. En este caso, se calculan los niveles de *stop loss* y *take profit*, así como los niveles de quiebre (breakout levels) basados en la volatilidad reciente. Luego, se generan señales de compra (self.buy) y venta (self.sell) cuando se cumplen las condiciones establecidas. Es relevante denotar que dentro de los métodos self.buy y self.sell se pueden incluir directamente los niveles de *stop loss* y *take profit*, esto es porque hereda otra clase de la librería llamada “Order” que permite este tipo de abstracciones, simplificando la creación de estrategias tanto para nosotros como para los usuarios de la plataforma.

Por lo tanto, se puede observar como la programación de estrategias en el *framework* "Backtesting.py" se basa en la creación de clases que heredan de la clase Strategy. Dentro de estas clases, se definen los parámetros, se realizan cálculos y se implementa la lógica de la estrategia en el método “next”. Esto permite a los usuarios de Trader Companion crear estrategias personalizadas y evaluar su desempeño utilizando datos históricos.

5.1.2 OBTENCIÓN Y FILTRADO DE DATOS

Una vez la estrategia está creada, el siguiente paso es la obtención y el ajuste de los datos obtenidos acorde a la estructura que la librería de *backtest*ing espera obtener. El proceso de obtención y filtrado de datos se realiza utilizando la API de AlphaVantage para obtener datos históricos de los activos bursátiles necesarios para el *backtest*ing. Se utiliza la librería "alpha_vantage" para establecer una conexión con la API y realizar las solicitudes correspondientes.

En primer lugar, se establece la conexión con la API de AlphaVantage utilizando la clave de acceso proporcionada al registrase en su web. Esta clave permite autenticar la solicitud y acceder a los datos requeridos. Una vez establecida la conexión, se procede a solicitar los datos utilizando la función correspondiente de la librería.

Los datos obtenidos de la API se devuelven en un formato específico, como un diccionario o un objeto JSON. Para facilitar su manipulación y análisis, se convierten estos datos en un DataFrame de Pandas. El DataFrame es una estructura de datos tabular que permite organizar y manipular los datos de manera eficiente.

Una vez convertidos los datos en un DataFrame, se procede al ajuste de los mismos. Para ello, se ha desarrollado una función llamada "adjust_stock_data" que realiza el ajuste de los precios. En esta función, se calcula un factor de ajuste dividiendo el precio de cierre original por el precio de cierre ajustado. Después, se crean nuevas columnas en el DataFrame ajustado, donde se aplican las transformaciones necesarias a las columnas de precios ("Open", "High", "Low", "Close") y el volumen.

```
def adjust_stock_data(df):
    df['factor'] = df['4. close'] / df['5. adjusted close']
    adjusted_df = pd.DataFrame({
        'Open': df['1. open'] / df['factor'],
        'High': df['2. high'] / df['factor'],
        'Low': df['3. low'] / df['factor'],
        'Close': df['4. close'] / df['factor'],
        'Volume': df['6. volume'] * df['factor']
    })
    return adjusted_df
```

Este ajuste es clave para la correcta interpretación de las estrategias, ya que la API originaria de los datos otorga todos los datos sin ajustar a dividendos y *splits* de las acciones, pero la librería de *backtest*ing espera recibir todos los datos ajustados, por lo que la falta de este paso podría llevar a una incorrecta evaluación de la viabilidad de las estrategias.

Después del ajuste de los datos, se realiza el filtrado del DataFrame para seleccionar un período de tiempo específico. En el ejemplo proporcionado, se ha utilizado la función "loc" de Pandas para filtrar los registros comprendidos entre dos fechas específicas. Esta función permite especificar un rango de fechas y seleccionar únicamente los registros que se encuentren dentro de ese rango. Finalmente, para ajustar la frecuencia de los datos al valor deseado, se realiza un proceso de remuestreo (resample) del DataFrame filtrado. Este proceso permite seleccionar los registros cada cierto intervalo de tiempo y reducir la cantidad de datos para su posterior análisis.

En resumen, el proceso de obtención y filtrado de datos se basa en el uso de la API de AlphaVantage para obtener los datos originales, seguido de un proceso de ajuste y filtrado utilizando las funciones y transformaciones necesarias. Esto garantiza que los datos se ajusten a los requisitos de la plataforma Trader Companion y estén listos para su uso en el *backtest*ing de estrategias. Es importante mencionar que, debido a las limitaciones de la API gratuita utilizada, la plataforma real almacena los datos ajustados requeridos por los usuarios en MongoDB para su posterior utilización. Sin embargo, con fines de simplificación en la explicación de las funcionalidades clave, esta consideración no ha sido mencionada en esta sección.

5.1.3 *BACKTEST*ING Y OPTIMIZACIÓN

Una vez se ha programado la estrategia y ajustado los datos, el siguiente paso es llevar a cabo el *backtest*ing y la optimización de la misma. El *backtest*ing es un proceso fundamental que permite evaluar el desempeño de una estrategia utilizando datos históricos y simular su aplicación en condiciones pasadas. Por otro lado, la optimización busca encontrar los valores óptimos de los parámetros de la estrategia que maximicen el rendimiento o cumplan ciertos criterios predefinidos.

Para llevar a cabo el *backtest*ing, se utiliza la función "*Backtest*" de la librería. Esta función recibe como argumentos el DataFrame filtrado con los datos ajustados, la estrategia a probar, el capital inicial disponible, la comisión aplicada en cada operación y otras configuraciones relevantes. Al ejecutar la función, se realiza el *backtest*ing de la estrategia sobre los datos históricos, generando un objeto que almacena los resultados del *backtest*.

```
from backtesting import Backtest

bt = Backtest(filtered_df, VolatilityBreakout, cash=10000, commission=0.002,
exclusive_orders=True)

output = bt.run()
bt.plot()
```

Una vez finalizado el *backtest*ing, se pueden obtener y analizar los resultados utilizando los métodos y atributos del objeto de *backtest*. Entre ellos, se pueden mencionar la función "run()", que ejecuta el *backtest* y devuelve una serie de métricas y estadísticas, como el capital final, el rendimiento, las operaciones realizadas, entre otros. Además, se puede utilizar la función "plot()" para visualizar gráficamente el rendimiento de la estrategia a lo largo del tiempo.

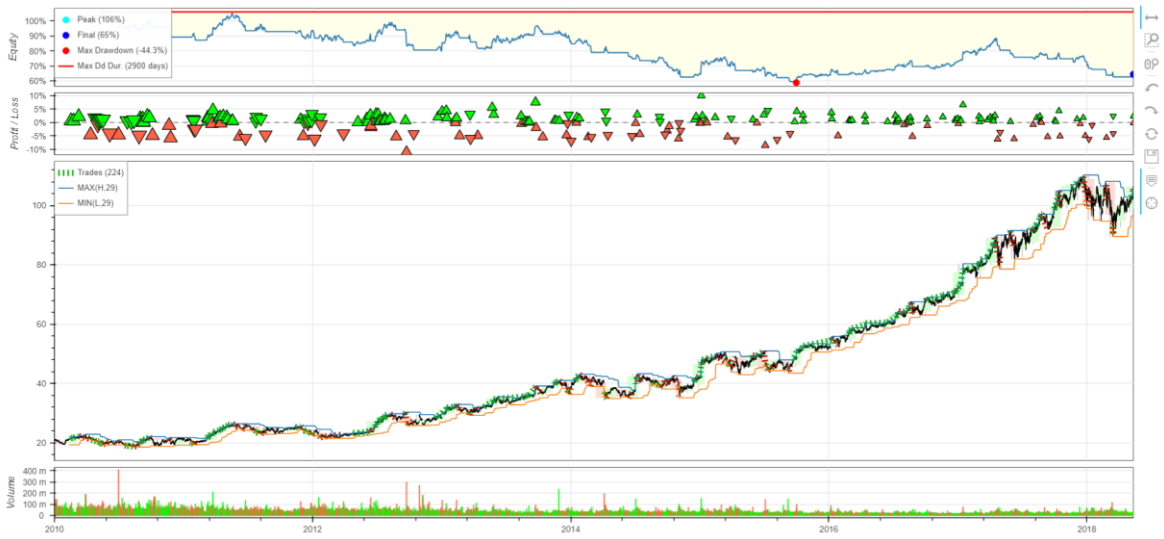


Figura 13 - Gráfico interactivo resultantes de un backtest

```
Start                     2010-11-01 00:00:00
End                       2019-02-22 00:00:00
Duration                   3035 days 00:00:00
Exposure Time [%]                   62.075562
Equity Final [$]                  6459.152497
Equity Peak [$]                  10598.63964
Return [%]                         -35.408475
Buy & Hold Return [%]               410.27222
Return (Ann.) [%]                  -5.131284
Volatility (Ann.) [%]               15.923006
Sharpe Ratio                              0.0
Sortino Ratio                             0.0
Calmar Ratio                              0.0
Max. Drawdown [%]                  -44.320579
Avg. Drawdown [%]                  -13.124372
Max. Drawdown Duration     2900 days 00:00:00
Avg. Drawdown Duration      747 days 00:00:00
# Trades                                  224
Win Rate [%]                        65.178571
Best Trade [%]                       10.02447
Worst Trade [%]                    -11.020007
Avg. Trade [%]                     -0.197864
Max. Trade Duration          77 days 00:00:00
Avg. Trade Duration           8 days 00:00:00
Profit Factor                       0.872871
...
```

Figura 14 - Estadísticas principales resultantes de un backtest

La optimización de la estrategia se lleva a cabo con el objetivo de encontrar los valores óptimos de los parámetros de la estrategia que maximicen el rendimiento o cumplan ciertos criterios predefinidos. Para ello, se utiliza la función "optimize()" del objeto de *backtest*. Esta función recibe como argumentos un diccionario que contiene los rangos de valores posibles para cada parámetro de la estrategia.

```
import numpy as np
param_ranges = {
        'lookback_period':  list(np.arange(5, 30, 2)),
        'volatility_factor': list(np.arange(1, 2, 0.1)),
        'stop_loss_percentage': 0.05,
        'take_profit_percentage': 0.05
    }
a = bt.optimize(**param_ranges,
            maximize='Equity Final [$]')
bt.plot()
```

En este ejemplo, se utiliza un diccionario llamado "param_ranges" que contiene los rangos de valores posibles para los parámetros "lookback_period" y "volatility_factor". La función "optimize()" realiza un proceso de búsqueda exhaustiva en el espacio de parámetros definido, evaluando el rendimiento de la estrategia para cada combinación de valores. Se puede especificar si se desea maximizar o minimizar una determinada métrica de desempeño, en este caso se utiliza "Equity Final [$]" como métrica a maximizar.

Una vez finalizada la optimización, se puede obtener el mejor conjunto de parámetros encontrado utilizando el objeto "a" que almacena los resultados de la optimización. Además, se puede utilizar la función "plot()" nuevamente para visualizar el rendimiento de la estrategia utilizando los parámetros óptimos encontrados.

En conclusión, la librería "*Backtest*ing.py" se convierte en el pilar fundamental para la funcionalidad de la plataforma Trader Companion al brindar la posibilidad de realizar un *backtest*ing y una optimización exhaustiva y precisa de las estrategias. Su integración en la aplicación web permite a los usuarios evaluar y mejorar sus estrategias de forma rigurosa antes de implementarlas en un entorno de tiempo real. Con esta herramienta, se fomenta la toma de decisiones informadas y se potencia la eficacia y la confianza en la ejecución de las estrategias de *trading*.

5.2 IMPLEMENTACIÓN DEL FRONTEND Y BACKEND

La implementación del *frontend* y el *backend* se lleva a cabo para crear una interfaz de usuario amigable y funcional en la plataforma Trader Companion. El *frontend* se encarga de presentar la información y permitir la interacción del usuario, mientras que el *backend* se encarga de procesar las solicitudes del usuario, manejar la lógica de negocio y proporcionar los datos necesarios. Nuestra plataforma presenta una estructura modular que facilita a interacción entre ambas partes de la aplicación, que se describirán a continuación.

5.2.1 *FRONTEND*

El *frontend* de la plataforma Trader Companion desempeña un papel crucial en la creación de una interfaz intuitiva y fácil de usar para la aplicación. Utiliza una combinación de HTML, CSS y JavaScript, junto con el *framework* Bootstrap para mejorar el atractivo visual y los elementos interactivos. Profundicemos en los detalles específicos de la implementación del *frontend*.

El código del *frontend* comienza importando las bibliotecas necesarias, incluida Bootstrap, para garantizar que la funcionalidad y el estilo requeridos estén disponibles. Se definen estilos CSS para personalizar el aspecto de los distintos elementos de la plataforma, proporcionando una interfaz de usuario cohesionada y visualmente atractiva.

La estructura HTML de cada página se crea para organizar el contenido de forma eficaz. Suele constar de una cabecera, una pantalla de carga, la sección de contenido principal y un pie de página. La cabecera suele contener una barra de navegación con el logotipo de "Trader Companion" y opciones para el inicio de sesión y el registro del usuario. Dependiendo del estado de autenticación del usuario, pueden aparecer elementos personalizados como el nombre del usuario y opciones adicionales.

JavaScript se utiliza ampliamente para manejar la lógica y la interactividad del *frontend*. Facilita la comunicación con el *backend* realizando peticiones de obtención a rutas específicas y gestionando las respuestas. Estas peticiones permiten al *frontend* recuperar información dinámica basada en la entrada del usuario o desencadenar acciones específicas en el *backend*.

Los formularios desempeñan un papel importante en la interacción del usuario con el *frontend*. Permiten a los usuarios introducir datos, seleccionar opciones y enviar solicitudes al *backend* para su procesamiento. El código JavaScript se encarga de validar las entradas de los formularios, gestionar los envíos y procesar las respuestas recibidas del *backend*. Esto permite a la plataforma proporcionar información en tiempo real y actualizar dinámicamente la interfaz de usuario en función de las acciones realizadas.

El *frontend* también incorpora varias técnicas de animación para mejorar la experiencia del usuario. Por ejemplo, se muestran pantallas de carga durante las operaciones que consumen muchos recursos, como el *backtest*ing o la optimización, para indicar que la plataforma está procesando la tarea solicitada. Esto proporciona información visual al usuario, reduciendo el tiempo de espera percibido.

La gestión de errores es otro aspecto crucial de la implementación del *frontend*. El código JavaScript es responsable de capturar y mostrar cualquier mensaje de error devuelto por el *backend*. Estos mensajes pueden informar al usuario sobre solicitudes fallidas, entradas incorrectas u otros problemas encontrados durante la ejecución de una tarea específica. Un tratamiento adecuado de los errores garantiza una experiencia de usuario fluida al proporcionar información clara y guiar a los usuarios para resolver los errores encontrados.

En general, la implementación *frontend* en Trader Companion se centra en la creación de una interfaz intuitiva y visualmente atractiva utilizando HTML, CSS, JavaScript y Bootstrap. Aprovecha la potencia de JavaScript para permitir interacciones dinámicas con el usuario, el envío de formularios y la gestión de errores. Estos elementos trabajan en armonía para proporcionar una experiencia de usuario sin fisuras al tiempo que garantiza una comunicación eficiente con el *backend* para procesar las solicitudes de los usuarios y recuperar los datos pertinentes.

5.2.2 *BACKEND*

El *backend* de la plataforma Trader Companion, construido con el *framework* Flask, desempeña un papel fundamental en la gestión de la lógica empresarial y la interacción con la base de datos. Sirve como columna vertebral de la aplicación, garantizando un procesamiento y recuperación de datos sin fisuras. Exploremos el último párrafo para ofrecer una visión general de la implementación del *backend*.

El código del *backend* comienza importando las librerías necesarias, incluyendo Flask, que permite la creación de aplicaciones web. Se importan bibliotecas adicionales para permitir el acceso a la base de datos, realizar operaciones de *backtest*ing y manipular datos. Se

definen valores constantes, como las claves API para acceder a la plataforma AlphaVantage y una lista de indicadores clave para el análisis de estrategias. Se habla de claves API en plural puesto que se utilizan dos claves distintas para la obtención de precios históricos y para la obtención de información de un *ticker*. Esto es así para evitar llegar al límite de peticiones API existentes por minuto en AlphaVantage.

El código establece una conexión con la base de datos MongoDB y accede a diferentes colecciones para almacenar y recuperar datos según sea necesario. Se crea la aplicación Flask y se configuran las opciones de sesión y cookies para autenticar a los usuarios. Estas configuraciones se importan desde un archivo de configuración con formato JSON, lo que garantiza la flexibilidad y un fácil mantenimiento.

A continuación, el *backend* define las rutas que corresponden a las diferentes vistas de Trader Companion. Estas rutas incluyen la página de inicio, la visualización de estrategias, el inicio de sesión y el registro de usuarios, la selección de valores para probar estrategias, la ejecución y optimización de estrategias, la descarga de datos y la gestión de *backtest*. Cada ruta está cuidadosamente definida para manejar peticiones HTTP GET, asegurando que los usuarios reciben la vista apropiada cuando acceden a estas rutas.

Se implementa la lógica de autenticación de usuarios, incorporando el cifrado de contraseñas mediante la biblioteca “bcrypt”. Se definen funciones auxiliares para manejar el ajuste de datos para acciones, la verificación y almacenamiento de datos en MongoDB, y la obtención de valores optimizados a partir de una cadena de estrategia.

Por último, el *backend* se centra en la implementación de la lógica central de la aplicación. Se emplean diversos métodos, cada uno responsable de una tarea específica y diseñado para interactuar con el *frontend* mediante la recepción de parámetros en peticiones realizadas a través de *fetch*. Estos métodos procesan los parámetros o los utilizan para interactuar con las colecciones de MongoDB. Algunos métodos clave incluyen:

- **`/search_ticker`**: gestiona las solicitudes para obtener el nombre completo de una acción de bolsa y su *ticker* a partir unas pocas letras. Utiliza una API creada por

Yashwanth (2018) y presente en su repositorio de GitHub para obtener la información pertinente.

- **`/get_stock_info`:** obtiene información adicional sobre una acción específica, como información general de la empresa y datos financieros, utilizando la API de AlphaVantage y la devuelve en la respuesta de la solicitud.
- **`/check_data_availability`:** comprueba si los datos de valores requeridos están disponibles en la base de datos MongoDB para un *ticker* y una fecha de finalización determinados. Si no es así, obtiene los datos de la API de AlphaVantage, realiza los ajustes necesarios y los guarda en la base de datos.
- **`/get_strategies`:** obtiene las estrategias disponibles de la colección de estrategias en la base de datos. Devuelve las estrategias como JSON, incluyendo tanto las estrategias disponibles para todos los usuarios como las estrategias específicas para el usuario conectado.
- **`/get_backtests`:** obtiene los *backtests* guardados por el usuario que se encuentran almacenados en la colección de *backtests* de la base de datos. Devuelve los resultados del *backtest* como JSON.
- **`/delete_non_permanent_backtests`:** borra los *backtests* no permanentes (*backtests* no guardados) del usuario autenticado de la colección de *backtests* de la base de datos.
- **`/delete_backtest/<backtest_id>`:** elimina un *backtest* específico con el ID de *backtest* dado para el usuario autenticado de la colección de *backtests* en la base de datos.
- **`/download_strategy/<string:strategyFilename>`:** permite a los usuarios descargar el archivo Python del módulo de estrategia aportando en el identificador único de la estrategia.
- **`/handle_upload`:** gestiona el envío del formulario de creación de estrategias. Guarda el archivo Python de estrategia subido por el usuario en el módulo de estrategias, extrae la información de la estrategia y la almacena en la colección de estrategias de la base de datos, asociada al usuario.

- **`/execute_strategy`:** ejecuta una estrategia de negociación basada en los parámetros de entrada proporcionados por el usuario. Realiza un *backtest*ing utilizando la librería de *backtest*ing y devuelve los resultados del *backtest* como JSON. La explicación concreta de como se hace el *backtest*ing ya ha sido desarrollada en la sección previa.
- **`/save_backtest`:** se encarga de guardar el resultado de un *backtest*. Actualiza la información del *backtest* en la colección de *backtests* de la base de datos, permitiendo a los usuarios guardar y reutilizar los resultados obtenidos.
- **`/download_data`:** permite a los usuarios descargar las estadísticas del *backtest* en formato Excel. Obtiene los datos del *backtest* de la colección de *backtests* de la base de datos, genera un archivo Excel utilizando "xlsxwriter" y lo envía como respuesta.
- **`/get_optim_parameters`:** obtiene las descripciones y valores por defecto de los parámetros de la estrategia para la optimización basada en el identificador único de la estrategia. Devuelve las descripciones de los parámetros como JSON para que el usuario pueda entender los parámetros utilizados y optimizarlos.
- **`/optimize_strategy`:** gestiona la optimización de una estrategia de negociación basada en los rangos de parámetros especificados. Realiza una optimización sistemática utilizando la biblioteca de *backtest*ing, almacena el *backtest* optimizado en la colección de *backtests* de la base de datos y devuelve el resultado como un JSON para que se muestren por pantalla al usuario.

Con estos módulos *backend*, que gestionan las solicitudes *frontend*, Trader Companion proporciona toda la funcionalidad necesaria requerida por la plataforma, garantizando un procesamiento, almacenamiento y recuperación de datos sin problemas, al tiempo que ofrece un compañero de negociación completo y robusto para los usuarios.

Capítulo 6. ANÁLISIS DE RESULTADOS

Este capítulo se presenta con el objetivo de entender al completo los resultados del proyecto. Con este fin se desarrolla un caso de uso completo de la aplicación web, con detenimiento en el punto más relevante de la aplicación, la interpretación de los resultados de un *backtest,*

La aplicación está ideada para que inicialmente el usuario se registre en la plataforma simplemente añadiendo un nombre de usuario que no esté ya utilizado, puesto que será su identificador único de usuario, y una contraseña. En esta demo inicial de la plataforma no existen limitaciones en cuanto a los caracteres de la contraseña y la única seguridad presente es la encriptación de la contraseña antes de ser introducida en la base de datos, aunque sería conveniente introducir una mejora de seguridad si la web pasa a un entorno de producción.

Tras el registro o inicio de sesión, el usuario será redirigido a la página principal y generalmente comprobará la librería de estrategias predefinidas para aprender y probar una de ellas. Al hacer clic en "Learn and Test Trading Strategies" en la pantalla principal, será redirigido a una vista con múltiples estrategias ordenadas alfabéticamente y con breves descripciones de las mismas. En esta vista tendrá tanto la opción de probar la estrategia como de descargar su código. Puesto que la plataforma está diseñada para usuarios sin ningún conocimiento de programación, se asume que inicialmente el usuario probará una estrategia.

Asumamos que el usuario escoge la estrategia "AverageDirectionalMovement", cuya descripción es: "*The Average Directional Movement (ADX) strategy is a trend-following strategy that uses the ADX indicator to identify strong trending markets. The strategy generates a buy signal when the ADX crosses above a threshold value, indicating the start of a new trend. It generates a sell signal when the ADX falls below the threshold, suggesting a weakening trend or a range-bound market*". Esta descripción no es muy detallada puesto que se espera que el usuario aprenda más sobre la estrategia a medida que la utilice y optimice, para no sobrecargarlo de información.

En este punto de la experiencia del usuario la funcionalidad principal de la web se empieza a hacer presente. El usuario es redirigido a la vista mostrada en la siguiente imagen.

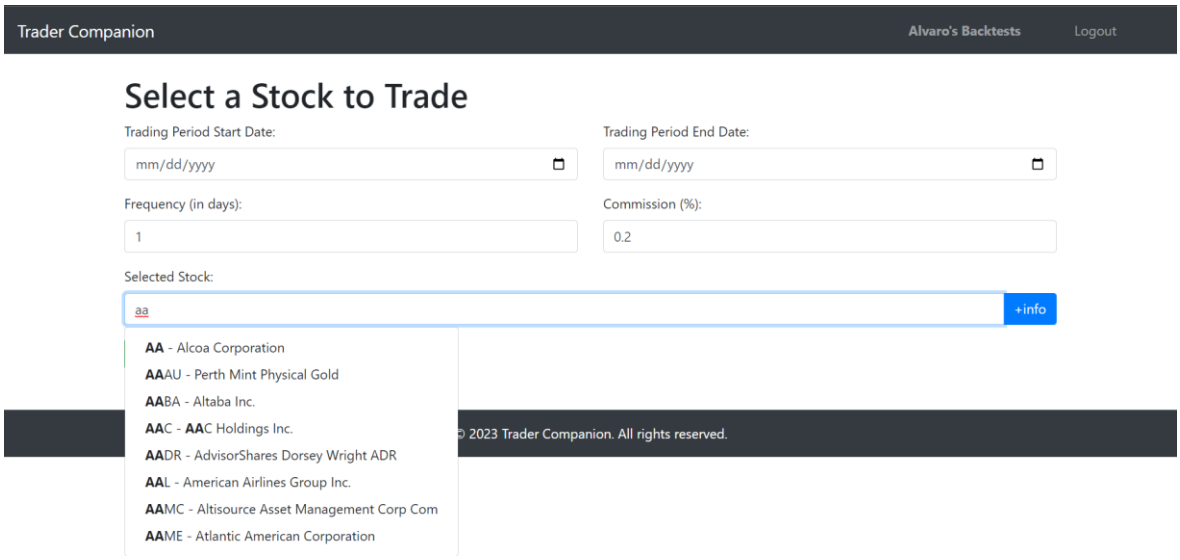


Figura 15 - Caso de uso de selección de la acción bursátil y su periodo de inversión

En esta página el usuario es capaz de seleccionar el periodo de inversión, la frecuencia y las comisiones de cada orden de su estrategia, además de la acción bursátil en la que invertirá. Se puede observar como la página web implementa una funcionalidad de búsqueda de acciones a partir de las letras que el usuario escriba del *ticker* o de la compañía. De esta forma los usuarios poco experimentados pueden simplemente buscar una compañía y la plataforma les proporciona el *ticker* que la identifica. Además, cabe recalcar que los usuarios pueden obtener más información sobre la compañía o acción bursátil concreta presionando el botón "+info". Este es un ejemplo de los datos mostrados en caso de pulsarlo.

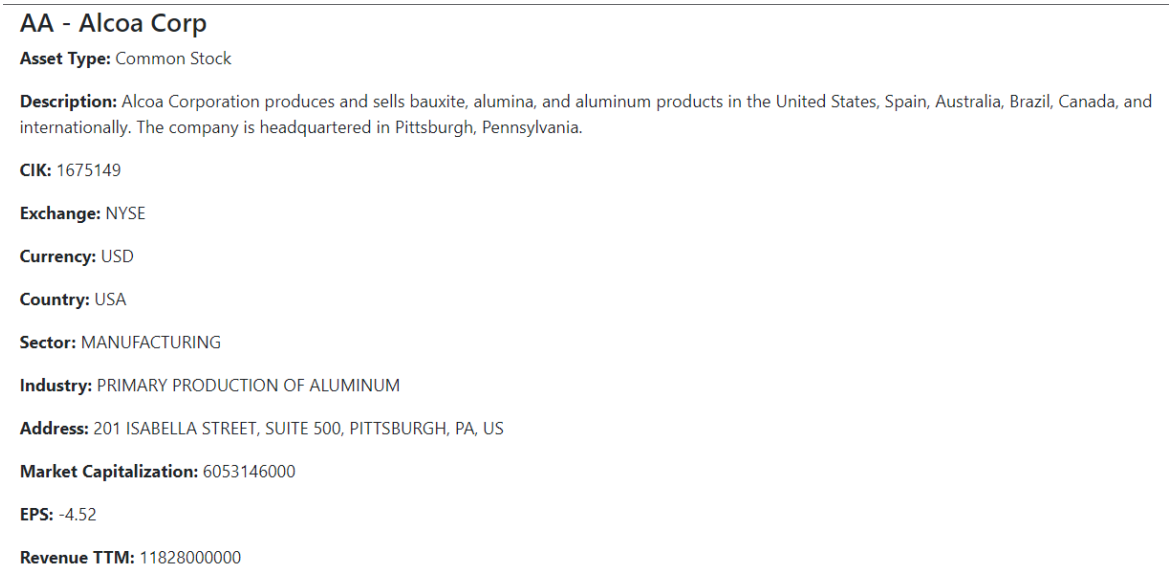


Figura 16 - Caso de uso de obtención de información financiera sobre un ticker concreto

Como se puede observar en la figura, esta funcionalidad otorga información relevante del *ticker* elegido en el momento actual. En la imagen no se muestra toda la información incluida, pero todo lo que muestra son indicadores financieros que pueden dar una idea al usuario de la salud actual de la empresa, para decidir si ha de ser el objetivo de su backtest. Incluso puede ser realmente útil a la hora de probar la estrategia sobre acciones con indicadores financieros similares, para comprobar su comportamiento sobre activos financieros similares.

Tras rellenar el formulario, el usuario ya ha realizado el *backtest* de su primera estrategia. Este proceso ha sido rápido y simple puesto que es lo que se pretende que el usuario sienta a pesar de iniciarse en un campo tan complejo. El único inconveniente es que la estrategia es probada con los parámetros asociados por defecto a la estrategia predeterminada, pero esto podrá ser modificado con la optimización de la estrategia. Por ahora el usuario verá una pantalla de carga con el texto “Running Strategy...” y a continuación se le mostrará la siguiente vista principal.

Figura 17 - Caso de uso de interpretación de resultados de un backtest

Este es el resultado que muestra Trader Companion ante cualquier *backtest*, unos indicadores claves y un gráfico interactivo del desempeño temporal. Comenzando por los indicadores clave, es relevante resaltar que se dividen en dos secciones divididas por columnas.

Primero, a la izquierda se encuentra la sección más relevante puesto que otorga al usuario los resultados anualizados de su *backtest*. En el campo del *trading* algorítmico, el indicador más utilizado para medir la eficacia de una estrategia es el Sharpe Ratio. Este ratio no es más que la división de la rentabilidad anual entre la volatilidad anual y busca informar al usuario

de cuantos retornos obtiene de su inversión por cada punto de volatilidad. Esto es relevante, puesto que los inversores buscan activamente estrategias con poca volatilidad y por ende poco riesgo.

Por otro lado, la sección de indicadores clave ubicada a la derecha indica al usuario información general de su estrategia como el porcentaje de tiempo que ha tenido una posición abierta, ya sea de compra o de venta, o el porcentaje de retorno total a lo largo del periodo de inversión. Es importante comparar el porcentaje de retorno del backtest con el último indicador clave que es el porcentaje de retorno de comprar y mantener en cartera esa acción, puesto que si la estrategia del usuario no es mejor que simplemente tener siempre una posición compradora se puede afirmar que no es una buena estrategia para ese activo, según los datos históricos. En este caso de uso concreto se puede observar como la estrategia presenta un retorno de más del 78% desde octubre de 2006 hasta la fecha actual, una cifra mayor al 55% por mantener una posición compradora siempre, por lo que la estrategia estaría funcionando.

Tras el resultado principal de la estrategia se presenta un pequeño formulario como el siguiente.

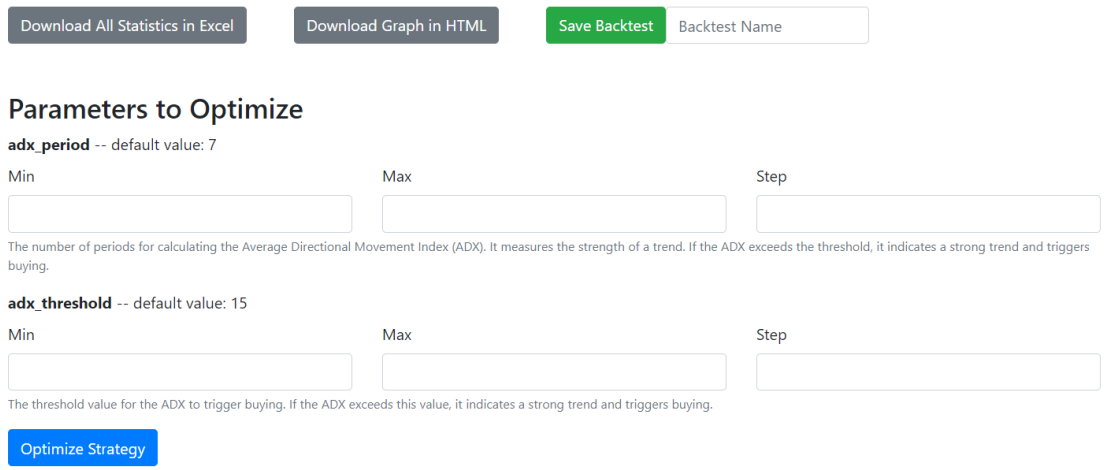


Figura 18 - Caso de uso de optimización de parámetros de un backtest

En la parte superior del formulario existen botones que permiten al usuario descargar el gráfico como HTML, descargar las estadísticas del backtest en Excel o guardar el backtest

en su librería con el nombre deseado en la plataforma, para utilizarla posteriormente. Pero la parte relevante llega con los parámetros a optimizar. El formulario muestra la lista de parámetros de esa estrategia, con su descripción y los valores por defecto que han tomado en la creación. Además, la funcionalidad real llega al rellenar cajas de texto con rangos que probar en para cada parámetro. Estos rangos son definidos por un valor mínimo, otro máximo, y un salto que se da en cada iteración. Una vez el usuario rellena los campos y pulsa "Optimize Strategy", una animación de carga aparece en la pantalla y la vista se recarga con los mejores resultados tras probar todas las posibles combinaciones

En este caso de uso se testean valores del 5 al 8 para "adx_period" y valores de 13 a 16 para "adx_threshold" y los parámetros óptimos son 7 y 16 respectivamente, que aparecen en una alerta antes de recargarse la página.

Key Indicators

Return (Ann.) [%]:	14.1360	**Exposure Time [%]:**	91.2270
Volatility (Ann.) [%]:	68.0799	**Return [%]:**	142.3331
Sharpe Ratio:	0.2076	**Buy & Hold Return [%]:**	55.0085

Figura 19 - Caso de uso de la optimización de las métricas principales tras la optimización

Simplemente observando los indicadores clave de la optimización, se puede observar claramente el funcionamiento correcto de la plataforma a la hora de optimizar valores. El retorno de la inversión ha pasado de ser un 78% en el *backtest* inicial, a un 142% en el *backtest* optimizado. Una vez obtenidos los resultados deseados el usuario puede guardar el *backtest* final en su librería para acceder a él posteriormente. La vista de la librería de *backtests* es muy similar a la vista de estrategias, con la única diferencia de que el usuario puede eliminar *backtests* si así lo desea.

Finalmente, al última funcionalidad que presenta la plataforma es para usuarios más avanzados que quieren dar el salto a una mayor personalización de las estrategias, la creación de nuevas estrategias. La web está ideada para que el usuario pueda utilizar de plantilla para sus estrategias una de las estrategias predeterminadas, por ello el código de todas las estrategias está comentado con todo detalle e incluye descripciones tanto de la estrategia

general como de todos los parámetros que la definen. A continuación, se muestra un ejemplo del código que obtendría el usuario de la estrategia que se estaba utilizando en el caso de uso.

```python
# Backtesting library documentation: https://kernc.github.io/backtesting.py/doc/backtesting/backtesting.html#backtesting.backtesting.Strategy
from backtesting import Strategy
# Technical Indicators library documentation: https://technical-analysis-library-in-python.readthedocs.io/en/latest/ta.html#
import talib as ta

# Strategy Description:
    # The Average Directional Movement (ADX) strategy is a trend-following strategy that uses the ADX indicator
    # to identify strong trending markets. The strategy generates a buy signal when the ADX crosses above a threshold
    # value, indicating the start of a new trend. It generates a sell signal when the ADX falls below the threshold,
    # suggesting a weakening trend or a range-bound market.

class AverageDirectionalMovement(Strategy):
    # ADX Period:
        # The ADX period determines the number of bars used to calculate the ADX indicator. A shorter period makes
        # the strategy more responsive to recent price changes, while a longer period provides a smoother ADX line.
    adx_period = 7

    # ADX Threshold:
        # The ADX threshold is the level above which the ADX value must cross to generate a buy signal. It determines
        # the strength of the trend required to trigger a trade. A higher threshold filters out weaker trends and
        # reduces the frequency of trades, while a lower threshold allows more trades but may result in entering
        # weaker trends.
    adx_threshold = 15

    def init(self):
        high, low, close = self.data.High, self.data.Low, self.data.Close
        # Calculate Average Directional Index (ADX) using TA-Lib
        self.adx = self.I(ta.ADX, high, low, close, self.adx_period)

    def next(self):
        # Check if ADX value is above the threshold
        if self.adx[-1] > self.adx_threshold:
            # Check if the previous ADX value was below the threshold
            if self.adx[-2] < self.adx_threshold:
                # Generate a buy signal when ADX crosses above the threshold
                self.buy()
        else:
            # Generate a sell signal when ADX falls below the threshold
            self.sell()
```

Figura 20 - Ejemplo de código de estrategia comentado disponible para su descarga por el usuario

Es importante resaltar las referencias a las dos librería utilizadas para construir estrategias, puesto que serán la guía para cualquier usuario que deseen personalizar al completo las estrategias, Además, la descripción tan detallada de las estrategias permite a los usuarios hacer pequeños retoques en ellas y subirlas como estrategias propias que aparecerán en su librería de estrategias. Esto se hace desde la vista de creación de estrategias accesible desde la página principal al hacer clic en el botón “Upload your own Strategy”. Esta acción llevará al usuario a la siguiente vista.

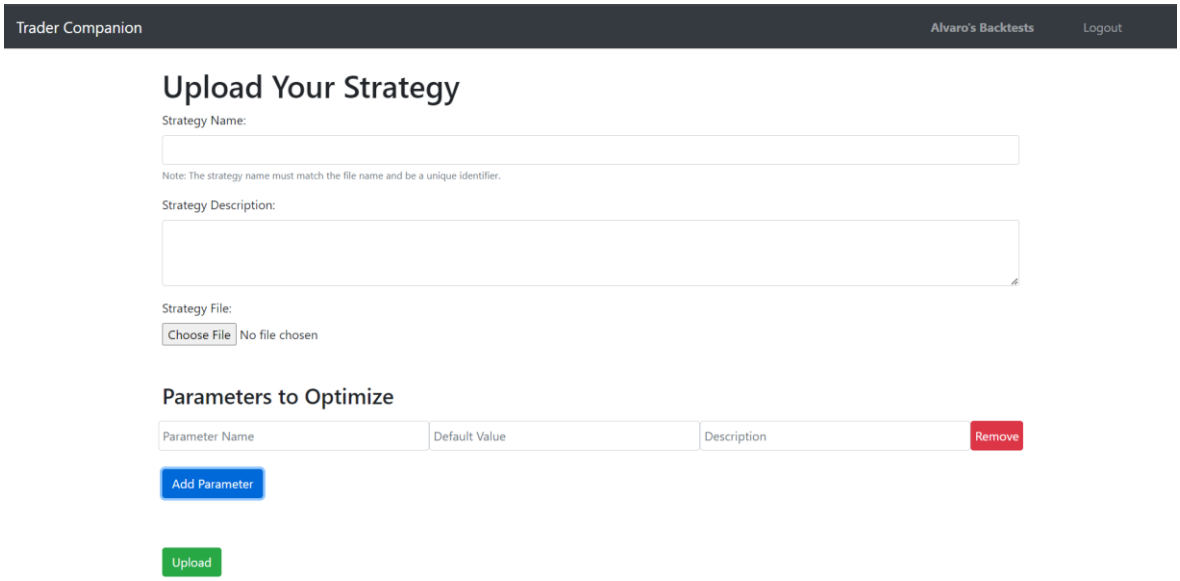


Figura 21 - Caso de uso para subir a la plataforma una estrategia personalizada por el usuario

En esta vista el usuario sube el código de su estrategia junto a un nombre único que ha de ser el mismo que el de la clase de Python que sube, y es validado por el servidor. Además, el formulario otorga flexibilidad al usuario para introducir tantos parámetros a optimizar como desee, para que la personalización de las estrategias con código sea total. Una vez subida la estrategia, si los parámetros introducidos concuerdan con los establecidos en el código, el usuario podrá utilizarla siempre que lo desee desde la librería de estrategias.

Esta última funcionalidad marca el logro de todos los objetivos de la plataforma, al acercar el código de las estrategias de *trading* algorítmico a cualquier usuario dispuesto a aprender y al servir de puente entre el conocimiento técnico y la implementación práctica mediante la programación de estrategias personalizadas e incluso propias.

Capítulo 7. CONCLUSIONES Y TRABAJOS FUTUROS

En este capítulo final, se resumirán los resultados y discutirán los logros obtenidos a lo largo del trabajo, así como se discutirán las implicaciones, contribuciones y limitaciones de la plataforma desarrollada en el campo del *trading* algorítmico.

En este trabajo, se ha identificado una necesidad en el ámbito del *trading* algorítmico: la demanda de herramientas accesibles y personalizables que permitan a los usuarios diseñar, probar y optimizar estrategias automatizadas sin requerir conocimientos avanzados de programación. Para abordar esta necesidad, se ha desarrollado una plataforma que ofrece una interfaz intuitiva, una biblioteca de estrategias predefinidas y herramientas de optimización.

Durante el desarrollo de este trabajo se han logrado todos los objetivos propuestos en la introducción en un entorno de desarrollo. La plataforma desarrollada ofrece una biblioteca de estrategias predefinidas y herramientas de optimización, lo que permite a los usuarios maximizar el rendimiento de sus estrategias y adaptarlas a las condiciones cambiantes del mercado. Estas estrategias pueden ser utilizadas con multitud de activos y se permite a los usuarios optimizar parámetros dentro de estas estrategias. Además, la posibilidad de descargar código de estrategias predefinidas y probar estrategias propias en la plataforma ayudan a que la herramienta acompañe al usuario durante todo su proceso de aprendizaje, más allá de los conceptos básicos.

A través del análisis de los resultados y pruebas realizadas en la plataforma, se ha comprobado que los usuarios, incluso sin experiencia previa en programación, pueden beneficiarse de las funcionalidades ofrecidas. La capacidad de personalizar estrategias, optimizar parámetros y evaluar su rendimiento con múltiples métricas en un entorno simulado brinda a los usuarios una ventaja competitiva a la hora de desarrollar estrategias y una mayor confianza en sus decisiones de inversión.

La plataforma también se destaca por su enfoque educativo, brindando a los usuarios la oportunidad de aprender sobre estrategias de *trading*, experimentar con diferentes enfoques y desarrollar habilidades prácticas relevantes para el campo financiero. Esto puede ser especialmente valioso en entornos académicos, como clubes universitarios, donde los estudiantes pueden beneficiarse de la aplicación práctica de los conceptos teóricos aprendidos.

7.1 LIMITACIONES Y FUTURAS MEJORAS

A pesar de los resultados obtenidos, durante el desarrollo y uso de la plataforma se han encontrado algunas limitaciones importantes que deben tenerse en cuenta. En esta sección se tratará de exponer estas limitaciones y se propondrán mejoras para el futuro desarrollo de la plataforma.

La primera limitación que se ha encontrado es de datos. Actualmente la plataforma permite únicamente la utilización de acciones bursátiles de los principales índices estadounidenses. Esto se debe a que la API desde la que se originan los datos históricos de precios únicamente soporta estas acciones. En futuras versiones de la plataforma se debería de cambiar a una API más completa, aunque eso conlleve el pago de alguna tasa.

La segunda limitación también está relacionada con la API de AlphaVantage, existe un límite de 5 peticiones de datos por minuto. Esto permite la utilización de la plataforma por un único usuario en el entorno de desarrollo sin ningún problema, pero sería problemático si varios usuarios se conectasen al mismo tiempo. Actualmente ya se utilizan varias conexiones a la API y se almacenan los datos más utilizados en la base de datos para evitar este problema, pero en una futura versión será necesario pasarse a la suscripción de pago o almacenar toda la información en bases de datos a medida que los usuarios trabajan con ellos.

La tercera limitación está relacionada con la forma en la que se hace el primer backtest de una estrategia. Tal y como se ha explicado en el proyecto, el primer *backtest* de una estrategia siempre se realiza con los parámetros predeterminados en el código base de

la estrategia, lo que puede llevar a que para ciertas acciones y en ciertos periodos no se genere ninguna compra o venta. De darse este caso, el programa devuelve un mensaje de error ya que no hay resultados que mostrar de la estrategia. Por lo tanto, en el caso en el que el usuario quisiera probar una estrategia con otros parámetros y optimizarla nunca podría, ya que la plataforma no te permite acceder al formulario de optimización que aparece junto a los resultados. Este error tendrá que ser corregido en futuras versiones permitiendo al usuario escoger valores concretos para las estrategias predeterminadas desde el inicio.

La cuarta limitación es la dependencia de la calidad de las estrategias predefinidas. La eficacia de las estrategias predefinidas proporcionadas en la biblioteca depende de su calidad y rendimiento histórico. Si bien se han realizado esfuerzos para seleccionar y probar estrategias sólidas, es importante que los usuarios realicen sus propias evaluaciones y ajustes, ya que puede darse el caso de que ninguna de las estrategias sea beneficiosa para una acción en concreto. La mejora necesaria en ese caso sería ampliar la biblioteca de estrategias predeterminadas.

Finalmente, la quinta limitación encontrada es las simplificación de variables y procesos. Para facilitar la accesibilidad y usabilidad de la plataforma, se han simplificado ciertas variables e incluso estrategias. Esto puede limitar la capacidad de representar completamente la complejidad y diversidad del campo de estudio en algunas situaciones. Por ejemplo, la herramienta no permite el backtesting de estrategias que dependan de más de un activo financiero por limitaciones de la librería principal. Esta es la limitación más difícil de solucionar, ya que implicaría cambiar toda la lógica detrás de la plataforma y utilizar otro *framework* de *backtesting* que aportase una mayor complejidad y nuevos conocimientos que aprender para los usuarios.

En conclusión, este trabajo ha demostrado la viabilidad y el valor práctico de la plataforma desarrollada para el diseño, prueba y optimización de estrategias de *trading* algorítmico. La plataforma ha abordado con éxito la necesidad de herramientas accesibles y personalizables en el campo del *trading* automatizado, permitiendo a los

usuarios, incluso sin conocimientos avanzados de programación, aprovechar las ventajas de las estrategias automatizadas.

Con su enfoque educativo y potencial uso en clubes universitarios, la plataforma ofrece una oportunidad única para que los estudiantes adquieran habilidades prácticas y experiencia en *trading* algorítmico, complementando su formación académica.

En general, la plataforma ha logrado contribuir al campo del *trading* algorítmico y presenta un potencial prometedor para futuras mejoras y aplicaciones. Aunque se han identificado limitaciones, estas ofrecen oportunidades para investigaciones futuras y mejoras adicionales que puedan abordarlas y fortalecer aún más la plataforma.

Capítulo 8. BIBLIOGRAFÍA

[1] Bajtelsmit, V. L., & VanDerhei, J. L. Risk aversion and pension investment choices. 1995.

[2] Barber, B. M., & Odean, T. The behavior of individual investors. In Handbook of the Economics of Finance (Vol. 2, pp. 1533-1570). Elsevier. 2013.

[3] Barberis, N., & Thaler, R. A survey of behavioral finance. Handbook of the Economics of Finance, 1, 1053-1128. 2003.

[4] Cagan, P., & Lipsey, R. E. The financial effects of inflation (No. caga78-1). National Bureau of Economic Research. 1978.

[5] Chan, E. Algorithmic trading: winning strategies and their rationale (Vol. 625). John Wiley & Sons. 2013.

[6] Chan, E. P. Quantitative trading: how to build your own algorithmic trading business. John Wiley & Sons. 2021.

[7] Chatterjee, S., Finke, M., & Harness, N. The impact of self-efficacy on wealth accumulation and portfolio choice. Applied Economics Letters. 2011. https://doi.org/10.1080/13504851003761830.

[8] Eurostat. Annual inflation up to 10.6% in the euro area. Euro indicators. October 2022. https://ec.europa.eu/eurostat/documents/2995521/15265521/2-17112022-AP-EN.pdf/b6953137-786e-ed9c-5ee2-6812c0f8f07f#:~:text=The%20euro%20area%20annual%20inflation,up%20from%2010.9%25%20in%20September.

[9] Fama, E. F. Efficient capital markets: A review of theory and empirical work. The journal of Finance, 25(2), 383-417. 1970.

[10] FinViz. Guided tour. 2023. https://finviz.com/help/guided-tour.ashx.

[11] Garbade, K. D., & Silber, W. L. Structural organization of secondary markets: Clearing frequency, dealer activity and liquidity risk. The Journal of Finance, 34(3), 577-593. 1979.

[12] Georgarakos, D., & Pasini, G. Trust, Sociability and Stock Market Participation. Behavioral & Experimental Finance (Editor's Choice) eJournal. 2009. https://doi.org/10.2139/ssrn.1397236.

[13] Grusky, D. B., Western, B., & Wimer, C. (Eds.). The great recession. Russell Sage Foundation. 2011

[14] Interactive Brokers LLC. IBKR Trading Platforms. 2023. https://www.interactivebrokers.com/en/trading/trading-platforms.php.

[15] Lusardi, A. Household saving behavior: The role of financial literacy, information, and financial education programs (No. w13824). National Bureau of Economic Research. 2008.

[16] Nowbutsing, B. M., & Odit, M. P. Stock market development and economic growth: The case of Mauritius. International Business & Economics Research Journal (IBER), 8(2). 2009.

[17] Oberoi, J. The 8 Best Algorithmic Trading Software & Platforms in 2023. June 19, 2023. https://www.wallstreetzen.com/blog/best-algorithmic-trading-software/.

[18] Pahlevi, R., & Oktaviani, I. Determinants of Individual Investor Behaviour in Stock Investment Decisions. AFRE (Accounting and Financial Review). 2018. https://doi.org/10.26905/afr.v1i2.2427.

[19] Pring, M. J. Investment psychology explained: Classic strategies to beat the markets. John Wiley & Sons. 1995.

[20] QuantConnect. Documentation. 2023. https://www.quantconnect.com/docs/v2.

[21] Romer, C. D. The great crash and the onset of the great depression. The Quarterly Journal of Economics, 105(3), 597-624. 1990.

[22] Seth. S. Pick the Right Algorithmic Trading Software. Investopedia. March 28, 2022. https://www.investopedia.com/articles/active-trading/090815/picking-right-algorithmic-trading-software.asp.

[23] Shefrin, H. Beyond greed and fear: Understanding behavioral finance and the psychology of investing. Oxford University Press. 2002.

[24] Trade Ideas. Features. 2023. https://www.trade-ideas.com/features/.

[25] TradeStation Group, Inc. Trading Platforms & Tools. 2023. https://www.tradestation.com/platforms-and-tools/.

[26] TradingView. Features. 2023. https://www.tradingview.com/features/.

[27] TrendSpider. Product Overview. 2023. https://trendspider.com/product/.

[28] Yashwanth K. Ticker Search Api. 2018. https://github.com/yashwanth2804/TickerSymbol

ANEXO I: ALINEACIÓN DEL PROYECTO CON LOS ODS

El trabajo encuentra una innegable conexión con el ODS número 8, relativo a el "trabajo decente y crecimiento económico". Entre las metas de este objetivo, se encuentra "8.2 Lograr niveles más elevados de productividad económica mediante la diversificación, la modernización tecnológica y la innovación, entre otras cosas centrándose en los sectores con gran valor añadido y un uso intensivo de la mano de obra". Es incuestionable que, en época actual, la innovación supone un pilar imprescindible del desarrollo económico. De la misma manera, no se entiende hoy la productividad de las sociedades sin los avances en tecnología, precisamente hablándose del sector tecnológico como un sector de "gran valor añadido".

En la relación del objetivo con el trabajo, es evidente y carente por tanto de mayor justificación el uso de herramientas tecnológicas como la base para la mejora de las herramientas de inversión. Es a través de la tecnología que este trabajo consigue acercar al ciudadano medio a la realidad de la inversión, a través de su capacidad de crear herramientas didácticas y accesibles por el fenómeno de la conectividad a Internet.

De este modo, útiles como el que defiende el presente trabajo, están asimismo en concordancia con el ODS número 9, sobre "industria, innovación e infraestructuras", en concreto con su medida 9.b, que busca "apoyar el desarrollo de tecnologías, la investigación y la innovación nacionales en los países en desarrollo, incluso garantizando un entorno normativo propicio a la diversificación industrial y la adición de valor a los productos básicos, entre otras cosas".

Este tipo de mejoras, que son creadas a través de la tecnología, y que usan esta última para llegar a la población en general, son herramientas que responden también a los parámetros de modernización e innovación que requiere la meta 8.2 de los ODS para ser cumplida.

Por otro lado, su influencia en la productividad es indudable. Este tipo de herramientas aportan seguridad al inversor, pues permiten probar o testear estrategias de inversión propias para verificarlas y poder extraer conclusiones válidas de cara a futuras inversiones, permitiéndole mayores posibilidades de éxito y reduciendo la posibilidad de fracaso o, en términos económicos, de pérdida o incluso endeudamiento. Así, el aumento de inversiones privadas seguras también fomenta la inclusión de nuevos inversores en el mercado, especialmente aquellos con mayor aversión al riesgo, aumentando por ende las fuentes de financiación existentes.

Ese aumento de las fuentes de financiación es sin ningún atisbo de duda beneficiosa para la economía de un país, máxime para aquellos que cuenten con una mayor dificultad para fomentar la inversión en sus empresas, dificultad que no se deba propiamente a las características financieras de estas últimas, sino más bien al desconocimiento del inversor de su funcionamiento.

A través de este tipo de herramientas, el inversor puede conocer qué tipo de empresas le benefician más e incluso arriesgarse a invertir en sectores que anteriormente le eran desconocidos solo por

haber podido probar con anterioridad cómo funcionan en su cotización con este tipo de simulaciones. Todo esto es altamente beneficioso para la situación financiera de la empresa, que es capaz de captar mayores fuentes de inversión y diversificar sus ingresos, lo cual tiene efectos positivos en su productividad, lo cual cumple con lo establecido en el ODS 8.

Además, cabe destacar que el proyecto tiene una vertiente didáctica clara, basada en la facilitación de conocimientos sobre *trading* algorítmico a través de la centralización de distintas estrategias, su explicación detallada, y la posibilidad de testeo o prueba, lo cual hace del mismo una herramienta perfecta para aquellos inversores amateurs que quieran dar sus primeros pasos en el mundo del *trading* algorítmico.

Sin embargo, también resulta una herramienta ideal para quienes quieran invertir sin aprender sobre las estrategias o algoritmos, sino simplemente quieran comprobar que su dinero se invierte de forma segura a través de la utilización de diferentes variables sobre los valores de las acciones de una empresa y la medición del comportamiento de estos últimos. Todo esto se halla en consonancia con los ODS 4 y 10, pues promueve nuevas oportunidades de aprendizaje, que además son accesibles para amplias capas de la población, y no permanecen restringidas para unos pocos, en un esfuerzo por democratizar el uso de este tipo de utilidades.

ANEXO II: REPOSITORIO DE CÓDIGO

Para garantizar la reproducibilidad y transparencia de nuestro proyecto, hemos creado un repositorio de código en GitHub. El enlace al repositorio completo, que incluye la implementación de la estrategia de negociación y el marco de gestión de carteras, junto con todas las librerías y dependencias necesarias, se puede encontrar en: https://github.com/Alvarogg3/TFG_Teleco.

Al compartir el acceso abierto al repositorio de código, buscamos facilitar la revisión, replicación y desarrollo de los resultados de nuestro proyecto por parte de investigadores y profesionales interesados. El examen del código y su documentación asociada permite a otros investigadores y profesionales analizar y comprender nuestro enfoque en detalle. Además, la plataforma GitHub fomenta la colaboración y el intercambio de ideas y conocimientos entre la comunidad investigadora en general.

Animamos a los usuarios a explorar el repositorio de código, ya que representa un recurso valioso para futuros análisis e investigaciones relacionadas con las estrategias de negociación en el contexto de la educación. La disponibilidad del código no solo promueve la transparencia en nuestra investigación, sino que también brinda oportunidades para mejorar y perfeccionar el enfoque implementado de manera continua.