UNIVERSITAT POLITÈCNICA DE VALÈNCIA

CAMPUS D'ALCOI

Desarrollo de aplicación web de soporte a la toma de decisiones para la inversión en los mercados financieros

---

MEMORIA PRESENTADA POR:
Samuel Ramón Soriano Díaz

TUTOR/A:
José Vicente Tomás Miquel

GRADO DE *INGENIERÍA INFORMÁTICA*

Convocatoria de defensa: *Julio 2021*

Resumen:

El proceso de toma de decisiones para las inversiones en los mercados financieros no supone una tarea sencilla al estar generalmente sometido a altos niveles de incertidumbre. Con el objeto de facilitar este proceso, este Trabajo de Final de Grado pretende desarrollar una aplicación web que incluya diferentes herramientas analíticas, a través de las cuales el usuario sea capaz de realizar un estudio exhaustivo del mercado bursátil.

En este nuevo software se incluyen herramientas basadas en inteligencia artificial, las cuales se basan en técnicas de aprendizaje automático y aprendizaje profundo para realizar pronósticos y en técnicas de procesamiento del lenguaje natural para analizar el sentimiento del mercado. Adicionalmente, el servicio web también cuenta con técnicas de optimización de portafolios, con el objetivo de que el usuario consiga el máximo rendimiento posible de su inversión.

En resumen, se trata de un software novedoso que permite a los inversores particulares tomar decisiones contrastadas y fundamentadas en técnicas modernas, con el fin de mejorar sus inversiones en mercados financieros tanto a corto como a largo plazo.

Palabras clave:

Aplicación Web, Análisis de los Mercados Financieros, Inteligencia Artificial, Pronóstico de Series Temporales, Análisis de Sentimiento, Optimización de Carteras.

Summary:

The decision-making process for investments in financial markets is not an easy task as it is generally subject to high levels of uncertainty. In order to facilitate this process, this Final Work Degree aims to develop a web application that includes different analytical tools, through which the user is able to carry out an exhaustive study of the stock market.

This new software includes tools based on artificial intelligence, which are based on machine learning and deep learning techniques for forecasting and natural language processing techniques for analyzing market sentiment. In addition, the web service also has portfolio optimization techniques, with the aim of the user getting the maximum possible return on their investment.

In short, it is a innovative software that allows private investors to make proven decisions based on modern techniques, in order to improve their investments in financial markets both in the short and long term.

Key words:

Web Application, Analysis of Financial Markets, Artificial Intelligence, Time Series Forecast, Sentiment Analysis, Portfolio Optimization.

Contenido

1. Introducción

Este TFG pretende abordar temas relacionados con tecnologías disruptivas como la inteligencia artificial y a su vez tratar asuntos reales como la gestión de inversiones o finanzas personales. En esta sección se hará hincapié en la problemática actual de los mercados financieros y se establecerán una serie de objetivos y tareas a seguir con la finalidad de desarrollar un proyecto innovador y práctico.

1.1 Motivación

Debido a diversos factores como la revolución tecnológica o la llegada de debacles económicas como la reciente del COVID-19, el número de inversores particulares en los mercados financieros se ha visto incrementado en gran medida. Con la aparición de intermediarios en línea (bróker), se ha facilitado el acceso a ciertas inversiones que antes estaban restringidas o limitadas. Es decir, hoy en día una persona mayor de edad con 100€ en el banco es capaz de adquirir un activo financiero. Además, la volatilidad sin precedentes que viven los mercados, sumado a las fuertes caídas, han provocado el interés general de los inversores.

La problemática que se presenta es que la mayor parte de este colectivo no cuenta con la formación y la experiencia necesaria para alcanzar una rentabilidad sostenible desde el corto hasta el largo plazo. Para solventarlo, se propone desarrollar una aplicación web que sirva de ayuda a la toma de decisiones y que incluya distintos aspectos relacionados con el análisis de los mercados financieros, con el fin de dar soporte a este tipo de inversiones.

Fundamentalmente, el proyecto está pensado para que contenga los siguientes apartados:

- Análisis de los diferentes mercados
- Pronóstico de precios a largo plazo
- Predicción de precios a corto plazo
- Análisis de sentimientos
- Optimizador de carteras de inversión

1.2 Objetivos

El desarrollo de esta aplicación web se centrará en lo siguiente:

- Ayudar al usuario a tomar mejores decisiones con respecto a sus inversiones.
- Incluir diferentes tipos de activos financieros.
- Poner a disposición de los usuarios un conjunto de herramientas que faciliten el análisis y la comprensión de los mercados.
- Permitir al usuario parametrizar y ajustar los análisis en función de sus intereses.
- Diseñar una interfaz clara y sencilla, que sea accesible tanto a usuarios inexpertos como a usuarios profesionales.

1.3 Metodología

La metodología a seguir para el desarrollo de este proyecto incluye varias fases. En primer lugar, se realizará un trabajo de investigación sobre las tecnologías, lenguajes y herramientas de programación utilizadas en el ámbito financiero. A continuación, se seleccionarán una serie de problemas a resolver con sus respectivas soluciones en formato de código, que permitan cumplir los objetivos propuestos. Tras realizar las pruebas pertinentes y confirmar los algoritmos a utilizar, se buscará la forma más adecuada de representar los resultados mediante una interfaz gráfica. Por último, se integrarán todas las soluciones en una misma aplicación web y se publicará en Internet para que cualquier usuario tenga acceso.

1.4 Estructura de la memoria

Una vez finalizada la fase introductoria, esta memoria consta de cuatro puntos clave, comenzando por el estado del arte, donde se realizará un estudio de mercado sobre los diferentes productos que existen. Seguidamente, se hará un recorrido por las tres fases del proyecto, la fase de análisis, la fase de diseño y la fase de implementación. La fase de análisis incluirá los requisitos de la aplicación y la solución propuesta para el desarrollo del proyecto. A continuación, en la fase de diseño se explicarán todas las herramientas utilizadas en el desarrollo y se detallará la arquitectura software. En la última fase, se añadirán todos los detalles de implementación de cada una de las funcionalidades de la aplicación. Por último, al final de la memoria se mostrarán los resultados obtenidos y las conclusiones.

2. Estado del arte

En la actualidad existen numerosos sitios web donde ofrecen información relevante acerca de los mercados financieros. Cabe destacar que la mayoría de ellos son muy completos e incluyen funcionalidades diversas. En esta sección valoraremos tres de los más importantes en el sector de las finanzas. Para ello, se describirán en detalle cada una de las alternativas y se realizará una comparación final, con el fin de contrastar las posibilidades que aporta cada una de ellas.

Las características básicas que podemos encontrar en este tipo de sitios son las siguientes:

- Noticias financieras
- Cotizaciones de activos
- Buscador de acciones avanzado (*screener*)
- Análisis fundamental
- Análisis técnico

Teniendo en cuenta estos puntos, lo que se intentará es proponer un nuevo tipo de análisis que ofrezca diferentes oportunidades y puntos de vista a los inversores, incluyendo para ello herramientas innovadoras.

Yahoo! Finance

El portal Yahoo!, es conocido por ser un proveedor de correo electrónico. Sin embargo, cuenta con una plataforma de noticias de diferente índole. Uno de sus apartados más relevantes es Yahoo! Finance, el cual se centra en los mercados financieros. Además de incluir noticias e información financiera, incluye las cotizaciones de numerosos activos alrededor del mundo. Además, cuenta con herramientas prácticas para la gestión de finanzas personales. Cabe destacar que no requiere de suscripción y se encuentra disponible en dispositivos móviles y tabletas.

Las funciones más destacadas se citan a continuación:

- Información financiera de diferentes activos financieros
- Datos en tiempo real
- Seguimiento de carteras
- Lista de observación (activos favoritos)
- Comparación de acciones

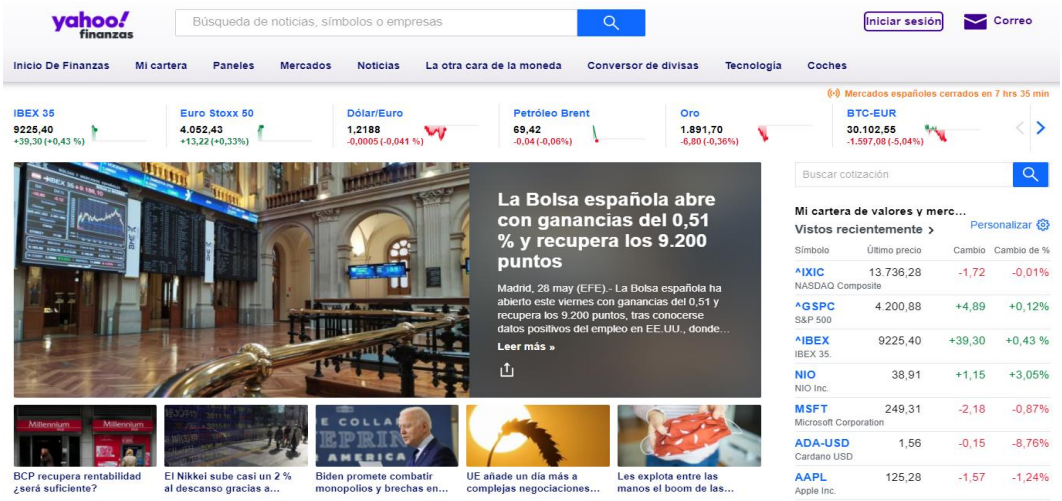


Figura 1. Yahoo! Finance. Fuente: finance.yahoo.com

Investing

La plataforma de mercados financieros Investing proporciona datos en tiempo real de más de 300.000 instrumentos financieros distribuidos en 250 mercados internacionales. Además de noticias, cotizaciones e información financiera, la plataforma ofrece acceso ilimitado a herramientas de vanguardia. La mayoría de los servicios son gratuitos y no requieren de suscripción, no obstante, cabe la posibilidad de obtener un servicio premium. Esta suscripción de pago permite acceder al área de *Insights*, donde se exponen análisis e ideas de inversión en la búsqueda de nuevas oportunidades, por parte de los mejores inversores y analistas.

Las funciones más destacadas se citan a continuación:

- Información financiera de diferentes activos financieros
- Datos en tiempo real
- Seguimiento de carteras
- Alertas personales
- Calendarios de eventos
- Calculadoras
- Buscador de acciones

Figura 2. Investing. Fuente: investing.com

Finviz

El sitio web de Finviz tiene como objetivo poner a disposición del usuario una suite de herramientas de soporte para las inversiones en mercados financieros. A diferencia de las anteriores, Finviz tiene un enfoque más técnico y muchas de las funcionalidades que ofrece están destinadas a un público más experimentado. Es importante tener en cuenta que la web está en inglés y se necesitan conocimientos sobre el sector financiero, ya que contiene conceptos avanzados. Al igual que Investing, ofrece una serie de servicios gratuitos y, además, da la posibilidad de pagar una tarifa mensual para acceder a las herramientas más avanzadas. Cabe destacar que este sitio web es famoso por su buscador de acciones avanzado, que permite añadir una infinidad de filtros.

Las funciones más destacadas se citan a continuación:

- Mapas de calor que presentan las acciones según capitalización de mercado, variación y sector
- Buscador avanzado de acciones (Screener)
- Noticias financieras
- Análisis técnico

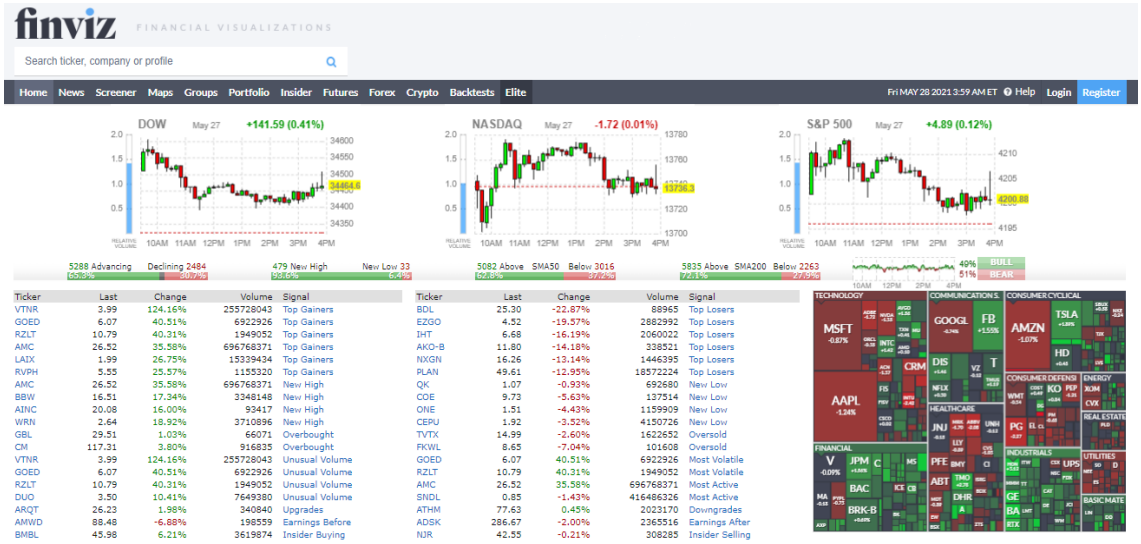


Figura 3. Finviz. Fuente: finviz.com

Resumen y Propuesta

Aunque existen numerosas páginas webs y aplicaciones que ofrecen un servicio parecido, se ha realizado un análisis exhaustivo de los tres sitios webs más referentes de la inversión en los mercados financieros. Pese a que presentan grandes similitudes, cada uno de ellos se especializa en diferentes aspectos de la inversión.

El proyecto que se va a llevar a cabo pretende aportar herramientas innovadoras, por lo que la propuesta incluye funcionalidades poco convencionales, utilizando varias tecnologías. Es por ello, que se utilizarán algoritmos de aprendizaje automático para ejecutar pronóstico de precios a largo y a corto plazo. También se analizarán los sentimientos de los diferentes activos financieros basándose en procesamiento del lenguaje natural. Por último, se propone un optimizador de carteras de inversión, el cual se ajusta mediante el parámetro de la ratio de Sharpe.

A modo de resumen, se muestra a continuación una tabla comparativa con las funciones principales de cada una de las aplicaciones, incluyendo el propio proyecto:

Tabla 1. Comparativa sitios web. Fuente: elaboración propia

Funciones	**Yahoo!**	**Investing**	**Finviz**	**Propuesta**
Diferentes activos financieros	X	X		**X**
Datos en tiempo real	X	X	X	
Seguimiento de carteras				
Screener		X	X	
Análisis fundamental	X	X	X	
Análisis técnico			X	
Pronóstico de precios				**X**
Predicción a corto plazo				**X**
Análisis de sentimiento				**X**
Optimizador de carteras				**X**

3. Especificación de requisitos

Antes de comenzar con el desarrollo de la aplicación, se ha de tener en cuenta la fase de análisis del proyecto. Esta etapa es necesaria, ya que permitirá identificar los requisitos de la aplicación y evaluar las posibles soluciones. Finalmente, se propondrá una solución y se justificará la decisión.

3.1 Análisis de requisitos

En función del escenario planteado y de los objetivos establecidos, se definirán en este apartado una serie de requisitos que servirán de apoyo a las fases de diseño e implementación.

3.1.1 Requisitos funcionales

- Sección de mercados financieros:
    - Los usuarios podrán observar la situación actual de los activos financieros que se incluyen.
    - Los usuarios serán capaces de elegir qué tipo de mercado desean visualizar y seleccionar el activo de interés.
    - Los usuarios podrán interactuar con los gráficos y datos mostrados.
- Sección de pronósticos de precios
    - Los usuarios serán capaces de visualizar pronósticos de cotizaciones a largo plazo.
    - Los usuarios podrán ajustar parámetros del pronóstico para personalizar su análisis.
    - Los usuarios serán capaces de visualizar pronósticos de cotizaciones a corto plazo.
    - Los usuarios podrán interactuar con los gráficos y datos mostrados.
- Sección de predicción a corto plazo
    - Los usuarios podrán analizar los datos más recientes del activo seleccionado.
    - Los usuarios podrán visualizar la predicción del precio para el día siguiente.
- Sección de análisis de sentimiento:
    - Los usuarios serán capaces de estudiar las opiniones de otros inversores a través de la red social Twitter.
    - Los usuarios serán capaces de evaluar la subjetividad y polaridad de los tweets relacionados con un activo a elegir.
    - Los usuarios serán capaces de distinguir el sentimiento de un activo mediante un clasificador de tweets.
- Sección de optimización de carteras de inversión:
    - Los usuarios serán capaces de seleccionar varios activos para su cartera.
    - Los usuarios podrán ver una matriz de correlación de su portafolio de inversiones.
    - Los usuarios identificarán la proporción óptima que deben poseer de cada activo para optimizar su cartera.
    - Los usuarios podrán ajustar su presupuesto para obtener información detallada sobre cómo distribuir su capital.

3.1.2 Requisitos no funcionales

- Acceso multiplataforma:
    - El usuario tendrá acceso desde cualquier dispositivo que cuente con un navegador.
- Usabilidad:
    - La interfaz debe ser clara y sencilla.
    - La aplicación permite cambiar el tema, es decir elegir entre claro u oscuro.
    - El acceso a todas las funciones de la aplicación debe realizarse en menos de 5 acciones.
    - La aplicación mostrará advertencias para comunicarse con el usuario.
    - La aplicación mostrará mensajes de error cuando se produzca algún inconveniente.
    - La aplicación incluirá mensajes de ayuda para entender los elementos de la interfaz
- Rendimiento:
    - Los algoritmos del sistema deberán ser lo más rápidos posible.
    - El tiempo de ejecución de los algoritmos que incluyen inteligencia artificial será inferior a 2 minutos.

3.3 Análisis de soluciones

Al tratarse de un proyecto software, normalmente se tienen en cuenta diferentes alternativas para dar solución a un problema. En este apartado se detallarán cuáles son las opciones a barajar para llevar a cabo el desarrollo de la aplicación. Además, se valorarán los pros y los contras de cada solución para poder tomar una decisión posteriormente. Se presentan tres posibilidades, entre las cuales distinguimos una aplicación de escritorio, una aplicación móvil y una aplicación web.

- **Aplicación de escritorio:** esta alternativa contempla el desarrollo de software de escritorio, en concreto para sistemas operativos de Microsoft Windows x64. La aplicación requerirá de una instalación y de una conexión a internet.

Tabla 2. Pros y contras aplicación de escritorio. Fuente: elaboración propia

Pros	Contras
- Tiempo de respuesta reducido	- Se requiere un proceso de instalación
- Puede ser más robusta	- Inaccesible desde otros dispositivos
- Dispone de menús y atajos adicionales	- Utiliza recursos del propio ordenador
	- Solo disponible para sistemas Windows

- **Aplicación móvil:** esta alternativa contempla el desarrollo de una aplicación móvil para dispositivos Android. La aplicación requerirá de una conexión a internet y una descarga desde la plataforma Google Play.

Tabla 3. Pros y contras aplicación móvil. Fuente: elaboración propia

Pros	Contras
- Ejecución desde cualquier lugar - Acceso fácil y rápido - Experiencia ágil y sencilla	- Se requiere un proceso de descarga e instalación - Inaccesible desde otros dispositivos - Utiliza recursos del dispositivo móvil - Solo disponible para sistemas Android

- **Aplicación web:** esta alternativa contempla el desarrollo de una aplicación web disponible desde cualquier navegador. La aplicación requerirá de una conexión a internet.

Tabla 4. Pros y contras aplicación web. Fuente: elaboración propia

Pros	Contras
- Ejecución desde cualquier dispositivo - Acceso desde cualquier lugar - No se utilizan recursos propios - Se pueden aplicar cambios y actualizaciones fácilmente	- Mayor tiempo de respuesta

3.4 Solución propuesta

Tras haber valorado las propuestas anteriormente descritas, se ha llegado a la conclusión de que la mejor alternativa es una aplicación web. A continuación, se muestra una lista de argumentos que han ayudado a tomar esta decisión:

- Multiplataforma: la aplicación puede ser lanzada en cualquier dispositivo que cuente con conexión a internet y un navegador.
- Accesibilidad: un usuario de internet puede acceder a la aplicación desde cualquier lugar y a cualquier hora.
- Ahorro de recursos: la aplicación no consume recursos locales.
- Flexibilidad y adaptabilidad: la aplicación se puede actualizar, corregir y modificar fácilmente.

Teniendo en cuenta estas ventajas, se propone desarrollar una aplicación web que cumpla con estas características y los objetivos definidos con anterioridad. Para ello, se ha realizado un estudio de posibles herramientas a utilizar y se ha seleccionado un paquete de Python de código abierto especializado en el desarrollo de aplicaciones webs interactivas. En concreto, se centra en la visualización de datos, por lo que es una opción muy viable dado que el proyecto se focaliza en el análisis de datos financieros.

Además, cabe destacar que no se necesita ningún otro lenguaje de programación ni servicio adicional, por lo que se facilita en gran medida el desarrollo y la implementación de la aplicación. Más adelante, se detallarán todos los aspectos técnicos acerca de esta herramienta y su funcionamiento.

4. Análisis y diseño de la solución

Una vez realizada la fase de análisis, se ha establecido una propuesta que cumpla con todos los requisitos y objetivos que plantea el proyecto. En este apartado, se tomarán decisiones sobre cómo se ha de llevar a cabo la solución al problema.

4.1 Diagrama de casos de uso

Se ha diseñado un par de diagramas de casos de uso con el fin de mostrar la funcionalidad de la aplicación. Además, se ha incluido una descripción detallada para cada caso de uso. Se ha tenido en cuenta por una parte la interacción del usuario y por otra parte las opciones que tiene el administrador.

Interacción usuario:

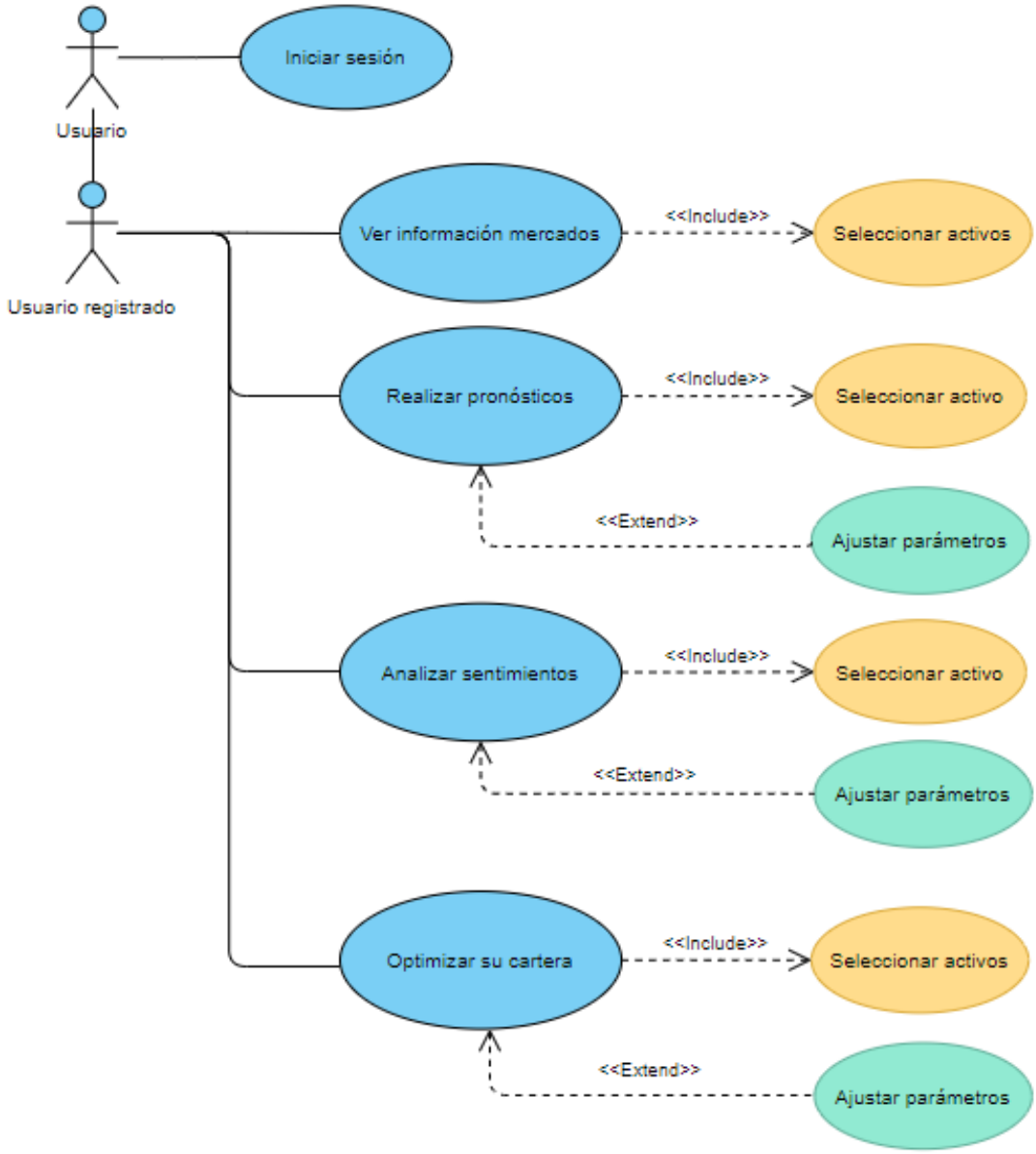


Figura 4. Diagrama de usuario. Fuente: elaboración propia

Tabla 5. Inicio de sesión. Fuente: elaboración propia

Caso de uso	**Iniciar Sesión**
Precondición	Usuario con credenciales
Postcondición	Sesión iniciada
Proceso	
Usuario	Sistema
El usuario introduce sus credenciales	
	El sistema valida el inicio de sesión

Tabla 6. Ver información mercados. Fuente: elaboración propia

Caso de uso	**Ver información mercados**
Precondición	Seleccionar activo
Postcondición	Visualizar datos y gráfico
Proceso	
Usuario	Sistema
El usuario selecciona el activo	
	El sistema muestra la información

Tabla 7. Realizar pronósticos. Fuente: elaboración propia

Caso de uso	**Realizar pronósticos**
Precondición	Seleccionar activo
Postcondición	Visualizar datos y gráfico
Proceso	
Usuario	Sistema
El usuario selecciona el activo	
	El sistema muestra el pronóstico ajustado

Tabla 8. Analizar sentimientos. Fuente: elaboración propia

Caso de uso	**Analizar sentimientos**
Precondición	Seleccionar activo
Postcondición	Visualizar análisis de sentimiento
Proceso	
Usuario	Sistema
El usuario selecciona el activo	
	El sistema muestra el gráfico de sentimientos

Tabla 9. Optimizar cartera. Fuente: elaboración propia

Caso de uso	**Optimizar su cartera**
Precondición	Seleccionar activos
Postcondición	Analizar cartera optimizada
Proceso	
Usuario	Sistema
El usuario selecciona los activos	
	El sistema muestra la información correspondiente a la cartera optimizada

Interacción administrador:

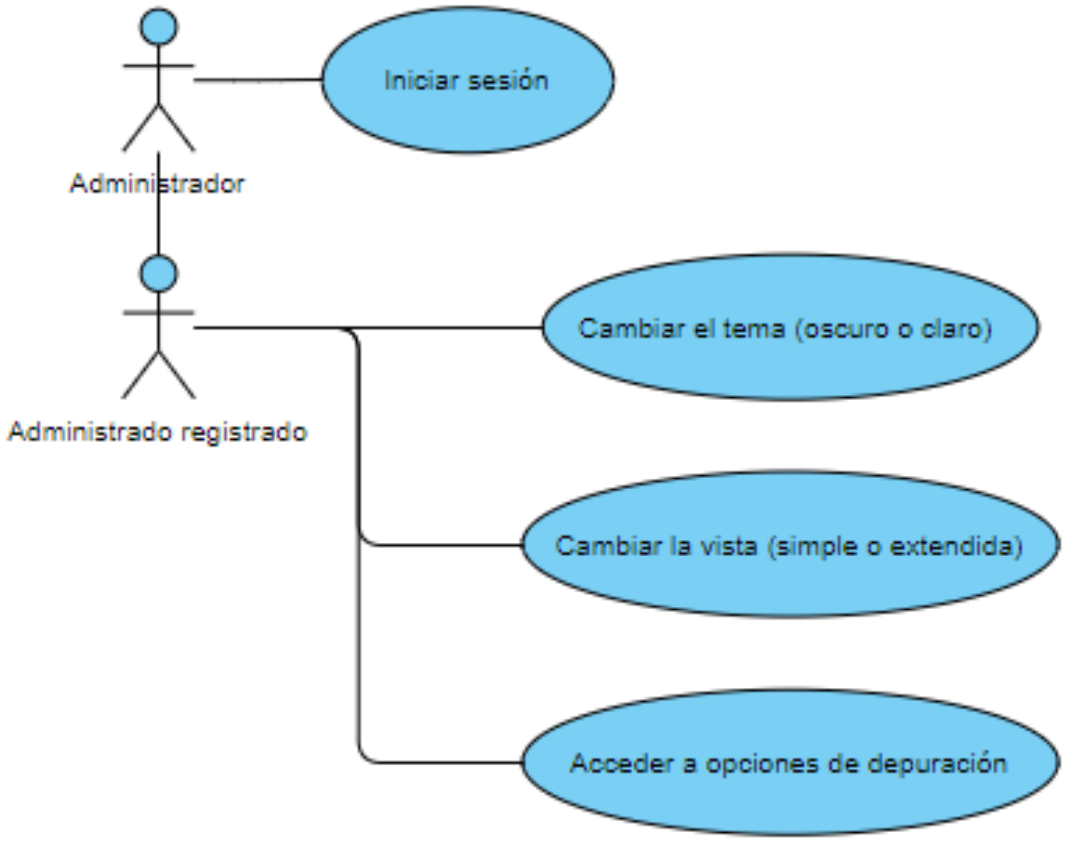


Figura 5. Diagrama de administrador. Fuente: elaboración propia

Tabla 10. Iniciar sesión. Fuente: elaboración propia

Caso de uso	**Iniciar sesión**
Precondición	Utilizar credenciales
Postcondición	Iniciar sesión
Proceso	
Usuario	Sistema
El usuario introduce sus datos	
	El sistema valida el inicio de sesión

Tabla 11. Cambiar tema. Fuente: elaboración propia

Caso de uso	**Cambiar tema**
Precondición	Permisos de administrador
Postcondición	Tema cambiado a oscuro o claro
Proceso	
Usuario	Sistema
El administrador selecciona el tema deseado	
	El sistema modifica el tema del sitio web

Tabla 12. Cambiar vista. Fuente: elaboración propia

Caso de uso	**Cambiar vista**
Precondición	Permisos de administrador
Postcondición	Vista simple o extendida
Proceso	
Usuario	Sistema
El administrador selecciona el tipo de vista	
	El sistema modifica la vista a simple o extendida

Tabla 13. Opciones depuración. Fuente: elaboración propia

Caso de uso	**Opciones de depuración**
Precondición	Permisos de administrador
Postcondición	Modificar opciones de depuración
Proceso	
Usuario	Sistema
El administrador selecciona las opciones de depuración	
	El sistema realiza la configuración correspondiente

5.2 Diagrama de clases UML

El diagrama de clases refleja por una parte, la interacción usuario sistema y por otra parte, el funcionamiento de Main, que será la aplicación principal y se explicará más adelante en el apartado de detalles de implementación.

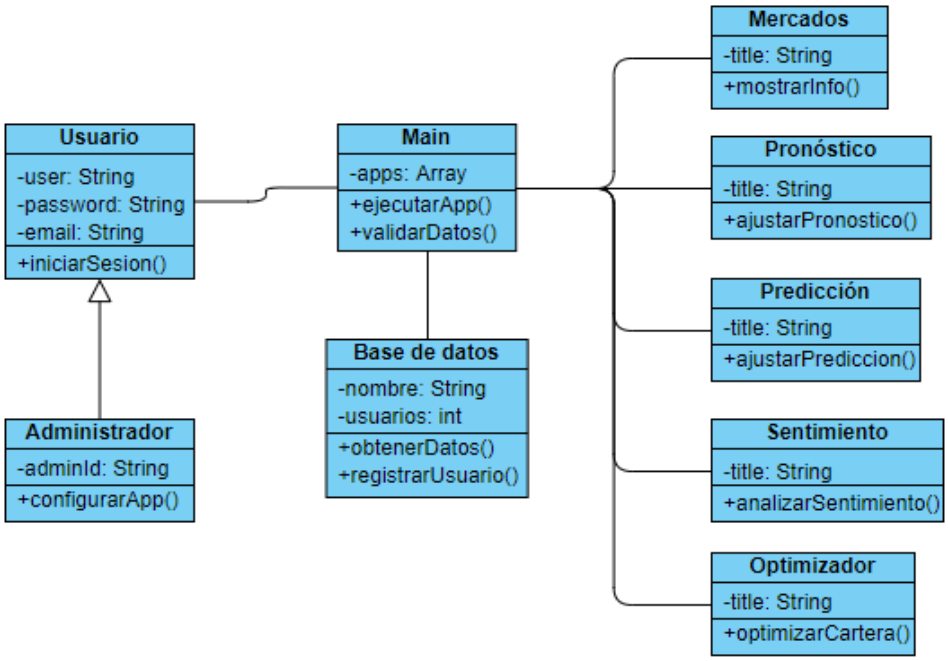


Figura 6. Diagrama de clases. Fuente: elaboración propia

5.3 Capa de presentación (diseño de la interfaz)

Esta capa tiene que ver con la interacción directa del usuario con la aplicación. Para explicarlo, se ha llevado a cabo una aproximación de lo que sería la interfaz de usuario. Como se puede observar, en primer lugar aparece el formulario de inicio de sesión, con la opción de registro. En segundo lugar, se muestra la interfaz principal. Por una parte, a la izquierda se ve un bloque de navegación en la parte izquierda desde donde el usuario puede seleccionar la aplicación que desea utilizar. Por otra parte, en el centro de la pantalla, aparece el contenido principal, donde el usuario introduce un activo y puede analizar la información que se le ofrece. Para este ejemplo se ha seleccionado la pantalla de Pronóstico a largo plazo:

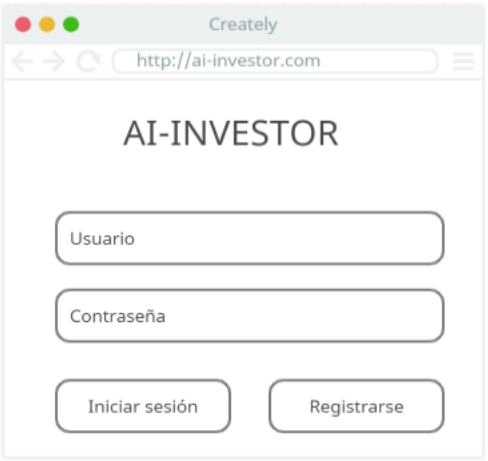


Figura 7. Diseño de inicio de sesión. Fuente: elaboración propia (app.creately.com)

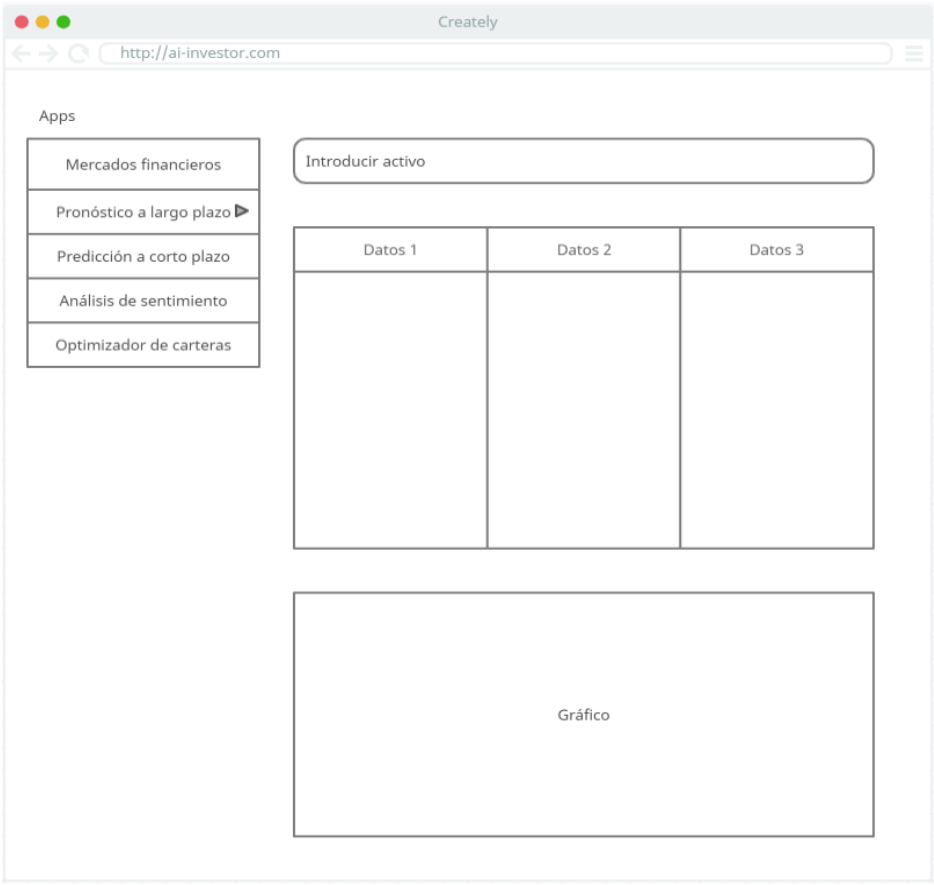


Figura 8. Diseño de interfaz. Fuente: elaboración propia (app.creately.com)

5.4 Capa de negocio

En esta capa es donde aparece la lógica de negocio y donde se ejecutan todas las operaciones que se llevarán a cabo por parte del sistema. Esta fase intermedia tiene como objetivo relacionar la capa de presentación con la capa de datos. Para ello, debe existir una interacción por parte del usuario, la cual desencadenará la ejecución de una serie de funciones, las cuales requieren de datos.

A continuación, se muestra una tabla que describe todos los procesos que realiza el sistema y la interacción con cada una de las capas:

Tabla 14. Lógica de negocio. Fuente: elaboración propia

Capa de presentación	**Capa de negocio**	**Capa de datos**
Usuario introduce credenciales	El sistema contrasta los datos introducidos con la base de datos y valida el inicio de sesión si son correctos	Base de datos de usuarios
Usuario selecciona un activo en el apartado de mercados financieros	El sistema descarga y muestra los datos financieros del activo seleccionado	API Yahoo! Finance
Usuario selecciona un activo en el apartado de pronóstico a largo plazo	El sistema descarga y muestra los datos financieros del activo seleccionado	API Yahoo! Finance
Usuario selecciona un activo en el apartado de predicción a corto plazo	El sistema descarga y muestra los datos más recientes del activo seleccionado	API Yahoo! Finance
Usuario introduce un término en el análisis de sentimiento	El sistema recopila los tweets más relevantes relacionados con el término introducido	API Twitter
Usuario forma una cartera con varias acciones	El sistema descarga los datos del precio de cada uno de los activos y muestra la información	API Yahoo! Finance

5.5 Capa de datos

En la capa de datos, podemos diferenciar tres fuentes de datos distintas:

- **Base de datos de usuarios**: almacena la información de los usuarios registrados en la aplicación. Su consulta es necesaria para validar el inicio de sesión. Para ello, se ha utilizado una base de datos Sqlite3. A continuación, se muestra la tabla que se ha creado con el software DB Browser:

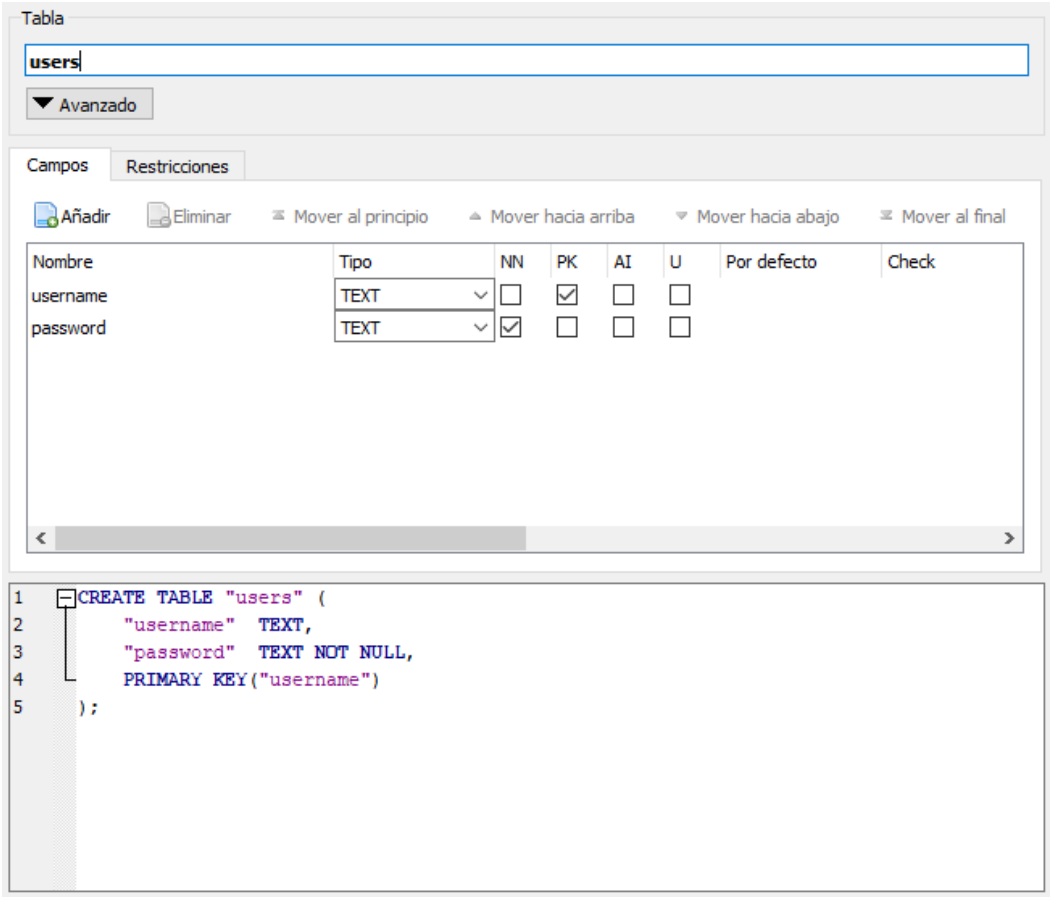


Figura 9. Base de datos. Fuente: elaboración propia (DB Browser)

Se pueden identificar dos campos: el campo username, que sería la clave primaria y el campo password, que tiene restricción no nulo.

- **API de Yahoo! Finance**: se utiliza para obtener los datos financieros de los diferentes activos. Es necesario para llevar a cabo la mayoría de los algoritmos de la aplicación. Los datos diarios que se recogen para realizar las operaciones son: precio de apertura, precio máximo, precio mínimo, precio de cierre, volumen y precio de cierre ajustado.

- **API de Twitter**: se utiliza para obtener los tweets más significativos relacionados con el término que introduce el usuario. Es necesario para realizar el análisis de sentimiento, donde se analizarán los mensajes con más retweets desde la fecha indicada.

5. Detalles de implementación

5.1 Análisis de las herramientas utilizadas

Entre las tecnologías que se van a utilizar, será necesario contar con un lenguaje de programación para escribir código, junto con una serie de API's y librerías que permitirán agilizar la carga de trabajo. Además, se utilizarán entornos de desarrollo integrado, desde donde se desarrollará el proyecto software. También, se tendrá en cuenta un navegador para ejecutar las pruebas y llevar el seguimiento de la aplicación. Por último, se necesitará un software de control de versiones para gestionar los diversos cambios que se vayan produciendo.

5.1.1 Lenguaje de programación

- **Python**: es un lenguaje de programación versátil multiplataforma y multiparadigma. Destaca por la limpieza y legibilidad de su código. Se ha elegido como lenguaje debido a la flexibilidad que permite y a la enorme cantidad de librerías que lo acompañan.

  Se utilizará para escribir todo el código de la aplicación

Figura 10. Logo Python. Fuente: python.org

5.1.2 Librerías y API's

- **Streamlit**: es un paquete de código abierto que permite crear y desplegar aplicaciones web de una forma rápida y sencilla. Además, tiene compatibilidad con muchas otras librerías e incluye elementos interactivos. Se centra en el análisis y representación de los datos.

  Se utilizará para el diseño de la interfaz y para publicar la aplicación.

Figura 11. Logo Streamlit. Fuente: streamlit.io

- **Pandas:** es una librería que se utiliza en la manipulación y análisis de los datos. Incluye estructuras de datos y operaciones.

  Se utilizará para convertir el conjunto de datos en una estructura

Figura 12. Logo Pandas. Fuente: pandas.pydata.org

- **NumPy:** es una librería que ofrece soporte para la creación y el manejo de vectores y matrices, junto con una gran colección de operaciones y funciones matemáticas.

  Se utilizará en el manejo y manipulación de estructuras de datos.

Figura 13. Logo NumPy. Fuente: numpy.org

- **Matplotlib:** es una librería para la generación de gráficos a partir de datos representados en una estructura como una lista o un vector.

  Se utilizará para la visualización de los resultados.

Figura 14. Logo Matplotlib. Fuente: matplotlib.org

- **Scikit-Learn:** es una librería utilizada en el aprendizaje automático que incluye diferentes algoritmos de regresión, clasificación y análisis en general.

  Se utilizará para procesar los datos y después poder trabajar con ellos.

Figura 15. Logo scikit-learn. Fuente: scikit-learn.org

- **Keras:** es una librería de redes neuronales artificiales especializada en el aprendizaje profundo.

  Se utilizará en la predicción del precio de un activo a corto plazo.

Figura 16. Logo Keras. Fuente: keras.io

- **Yfinance:** es una librería que permite descargar datos financieros de la web Yahoo! Finance.

  Se utilizará en la mayoría de los casos para obtener los datos necesarios para trabajar.

Figura 17. Logo Yahoo! Finance. Fuente: finance.yahoo.com

- **Facebook Prophet:** es un algoritmo que permite generar modelos de predicción basados en series temporales. Resulta útil para detectar estacionalidades.

  Se utilizará para realizar pronósticos sobre la cotización de los activos financieros en el largo plazo.

Figura 18. Logo Prophet. Fuente: facebook.github.io

- **Tweepy:** es una librería que permite acceder al API de Twitter y leer tweets de usuarios, aplicando un conjunto de filtros y términos de búsqueda.

  Se utilizará para descargar los tweets relacionados con el activo seleccionado.

Figura 19. Logo Tweepy. Fuente: tweepy.org

- **TextBlob:** es una librería de procesamiento de texto, con el fin de llevar a cabo tareas relacionadas con el Procesamiento del Lenguaje Natural y extraer análisis de opinión.

  Se utilizará para obtener el análisis de sentimiento de los usuarios sobre un activo en concreto.

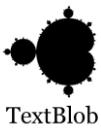


Figura 20. Logo TextBlob. Fuente: textblob.readthedocs.io

- **PyPortfolioOpt:** es una librería que implementa métodos de optimización de carteras mediante técnicas como la frontera eficiente.

  Se utilizará en el apartado de optimización de carteras.

Figura 21. Logo PyPortfolioOpt. Fuente: pyportfolioopt.readthedocs.io

- **Sqlite3**: librería para la conexión con la base de datos y la manipulación de los datos.

  Se utilizará para registrar usuarios y validar inicios de sesión.

Figura 22. Logo Sqlite. Fuente: sqlite.org

5.1.3 IDE's

- **Google Colaboratory:** es un entorno de desarrollo que permite escribir y ejecutar código desde el navegador. No necesita ningún tipo de configuración y su acceso es gratuito. Además, facilita la opción de compartir documentos y poder trabajar en equipo.

  Se utilizará para investigar y probar los diferentes algoritmos que se vayan a implementar en la aplicación.

Figura 23. Logo Colab. Fuente: colab.research.google.com

- **Anaconda Spyder:** es un entorno de desarrollo integrado para programar en lenguaje Python. Además, se incluye en la suite de aplicaciones y librerías Anaconda, que incluye el gestor de paquetes conda.

  Se utilizará para desarrollar la programación del proyecto y para gestionar todos los paquetes que se incluyen en él.

Figura 24. Logo Spyder. Fuente: spyder-ide.com

5.1.4 Navegador

- **Google Chrome:** es el navegador de Google.

  Se utilizará para llevar el seguimiento de la aplicación y hacer las pruebas necesarias.

Figura 25. Logo Chrome. Fuente: google.com

5.1.5 Control de versiones

- **GitHub:** es un sistema de control de versiones que permite alojar y gestionar proyectos software.

  Se utilizará para gestionar todos los cambios que vayan surgiendo y finalmente para desplegar la aplicación junto con Streamlit. Esta acción se detallará en el apartado de Resultados de la memoria.

Figura 26. Logo Github. Fuente: github.io

5.2 ARQUITECTURA SOFTWARE

En un principio, el paradigma en que se basa el desarrollo del software es la programación orientada a objetos. Se ha diseñado de manera que la clase principal representa una aplicación que puede contener otras aplicaciones secundarias. Por lo tanto, habrá una aplicación padre que tendrá varios hijos. De esta manera, se puede decir que el software es escalable y flexible, ya que permitirá añadir nuevas funciones a la aplicación principal. A continuación, se muestra la clase principal, donde se ve reflejada la estructura propuesta:

Tabla 15. Clase MultiApp. Fuente: elaboración propia

MultiApp()
apps #Vector de aplicaciones
__init__() #Función que inicializa el objeto de la clase con los atributos correspondientes **add_app(***title***,** *func***)** #Función que añade una aplicación al vector apps **run()** #Función que ejecuta la aplicación seleccionada

Teniendo en cuenta esta estructura, la idea será crear un objeto de la clase MultiApp() llamado main, que se corresponderá con la aplicación principal. El objeto main almacenará en la variable apps todas las aplicaciones que vayamos a incluir. Estas aplicaciones serán: markets, forecast, sentiment y portfolio. Además, cada una de las aplicaciones tendrá una función app() que contendrá todo el código y será ejecutada por la función run() del objeto main. En el siguiente diagrama se observa el diseño con más detalle:

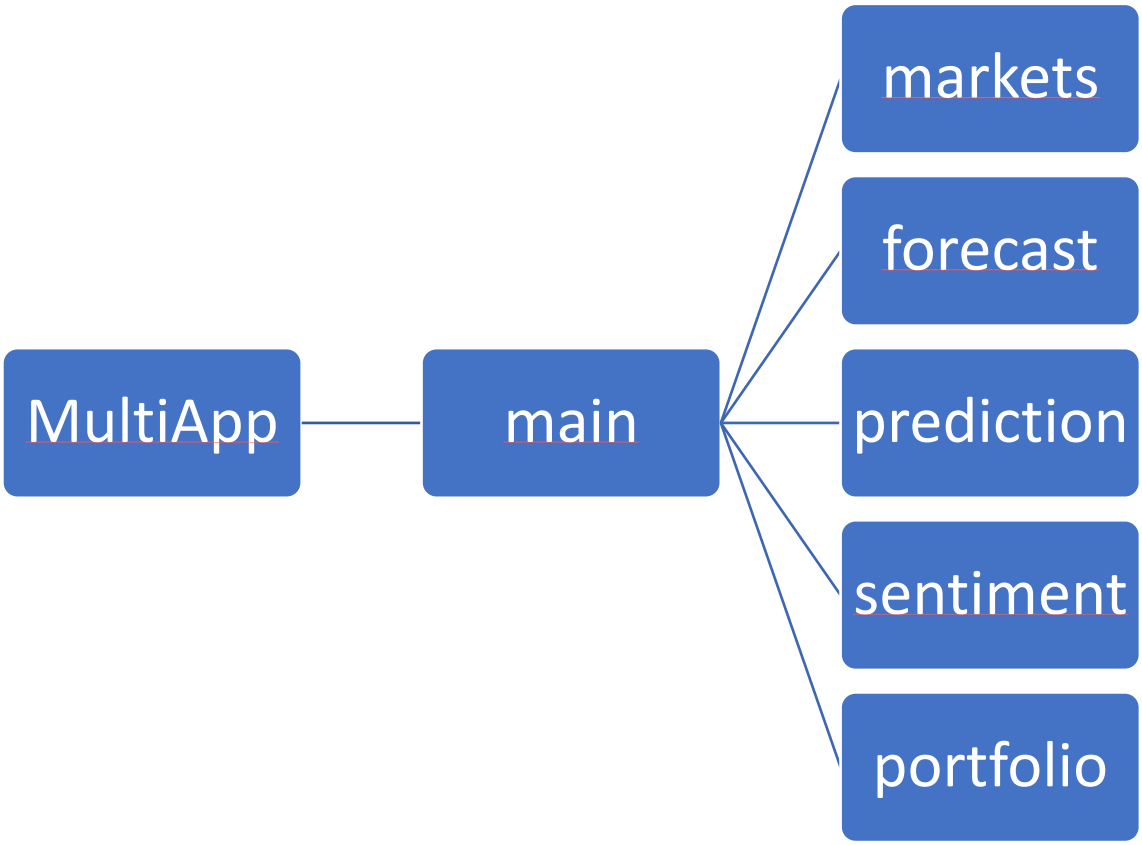


Figura 27. Jerarquía aplicaciones. Fuente: elaboración propia

5.2.1 Estructura de directorios

Siguiendo con el diseño, los directorios y ficheros se han estructurado de la siguiente manera:

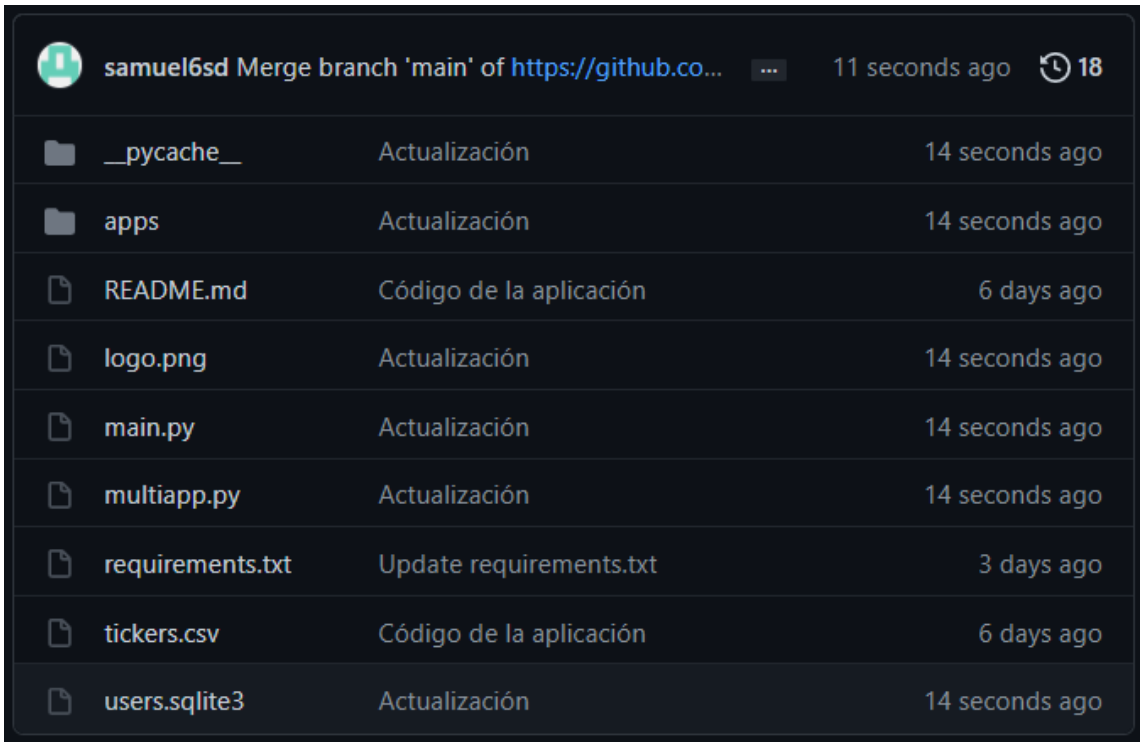


Figura 28. Estructura de directorios. Fuente: github.com/samuel6sd/ainvestor

Como se puede observar, hay dos ficheros de código, correspondientes con la clase multiapp y la aplicación principal main. Por otro lado, el fichero users.sqlite3 es la base de datos de usuarios y el archivo tickers.csv contiene una lista de símbolos bursátiles que se ponen a disposición del usuario. Además, se incluyen ficheros de soporte como son la imagen del logo y el fichero readme.md. Por último, el fichero requirements.txt que será necesario a la hora de desplegar la aplicación.

Además, la carpeta apps contiene el código de las diferentes aplicaciones que se van a mostrar en la plataforma, como podemos ver a continuación:

Figura 29. Directorio apps. Fuente: github.com/samuel6sd/ainvestor/apps

6. Implementación

La fase de implementación de un proyecto software consiste en la puesta en marcha y ejecución de las acciones previstas en el análisis y el diseño. Es por ello por lo que suele ser la más extensa y complicada. Además, es una etapa que está en constante revisión y actualización.

La estructura que va a seguir está implementación incluye la fase previa o investigación, el concepto de MultiApp y el desarrollo de cada una de las herramientas o funcionalidades que presenta la aplicación web.

Para la realización de la fase previa se va a utilizar la plataforma Google Colaboratory, que como se ha descrito anteriormente, es un entorno de desarrollo basado en bloques, que permite escribir y ejecutar código desde el propio navegador. Se ha elegido por su rapidez y facilidad a la hora de detectar y corregir fallos.

Una vez se tengan claro los algoritmos a utilizar en la aplicación, se va a configurar un entorno de desarrollo en Anaconda Python y se trabajará desde el IDE de Spyder. Dentro de este entorno se incluirán todas las librerías necesarias para el desarrollo del software.

6.1 Fase previa

Se ha decidido incluir esta fase previa dentro de la implementación, ya que antes de empezar con el desarrollo principal, se han llevado a cabo una serie de tareas de investigación, con el fin de aportar ideas y soluciones al proyecto.

La idea del proyecto es crear una aplicación capaz de dar soporte a la toma de decisiones con respecto a la inversión en los mercados financieros. Sin embargo, se pretendía utilizar herramientas poco comunes e innovadoras. Es por ello por lo que el primer paso de esta fase previa consistía en estudiar los problemas existentes y buscar las mejores soluciones.

En primer lugar, se explicará el proceso de inicio de sesión y la persistencia de los datos a la hora de registrar usuarios.

En segundo lugar, con la finalidad de añadir funcionalidad a la aplicación, se ha decidido incluir como página de inicio un apartado de visualización de mercados financieros. Desde este apartado, se podrán comparar los diferentes activos, así como analizar los gráficos y comprobar el histórico de precios.

En tercer lugar, uno de los objetivos principales ha sido incluir una herramienta de pronóstico de precios que permitiera realizar una estimación realista de cómo se puede comportar el mercado. Para ello, se han buscado diferentes algoritmos de inteligencia artificial, en concreto de aprendizaje automático. Tras comprobar y comparar varios resultados, la representación que más se adaptaba a las necesidades del proyecto ha sido el algoritmo de Facebook Prophet, que analiza series de tiempo estacionales. Además, se ha querido tener en cuenta el corto plazo y una de las opciones más interesantes era utilizar algoritmos de aprendizaje profundo. Tras un estudio de los diferentes tipos de redes neuronales existentes, se ha llegado a la conclusión de que la mejor red neuronal para este tipo de problemas era la red neuronal recurrente, en concreto la LSTM (Long Short Term Memory). Posteriormente se detallará el funcionamiento de estos algoritmos.

En cuarto lugar, otra de las funciones que está adquiriendo cada vez más relevancia en el análisis de mercados es el análisis de sentimiento. Este tipo de algoritmos también tienen que ver con el PLN (Procesamiento del Lenguaje Natural), que es un campo de la inteligencia artificial y la lingüística aplicada que estudia las interacciones mediante uso del lenguaje natural entre los seres humanos y las máquinas. Tras valorar las diferentes alternativas, se ha escogido la opción de analizar el sentimiento de los usuarios de Twitter a través de sus publicaciones. De esta manera, se puede conocer el pensamiento general de los usuarios con respecto a un activo.

Por último, se ha optado por añadir una función de optimización de carteras, ya que en otras plataformas únicamente se ofrece la acción de seguimiento. En este caso, se han realizado pruebas con la librería de Python PyPortfolioOpt, que permite analizar el precio de un grupo de activos para distribuir los pesos del portafolio con el objetivo de maximizar las ganancias. De esta manera, el inversor será capaz de decidir cuánto dinero invierte en cada activo.

6.2 MultiApp

El desarrollo de la aplicación web se ha llevado a cabo mediante el lenguaje de programación Python. Pero, para poder darle forma y funcionalidad a la plataforma, se ha utilizado una librería llamada Streamlit. Es por ello por lo que se ha decidido utilizar el paradigma de programación orientada a objetos para estructurar el contenido de la aplicación.

Como se ha explicado en la fase de diseño, el concepto de MultiApp tiene que ver con la posibilidad de incluir varias aplicaciones dentro del sitio web. Por lo tanto, se ha diseñado una clase que incluya la posibilidad de añadir nuevas funciones y de esta forma poder cambiar de una a otra fácilmente.

En resumen, la clase MultiApp facilita la navegación de la aplicación web, de manera que el usuario pueda elegir qué herramienta desea utilizar en cada momento. Para ello, se incluirá un desplegable con las distintas opciones a elegir, con el cual se controlará el flujo de la navegación, mediante sentencias condicionales.

A continuación, se muestra una vista previa de lo que sería la barra de navegación:

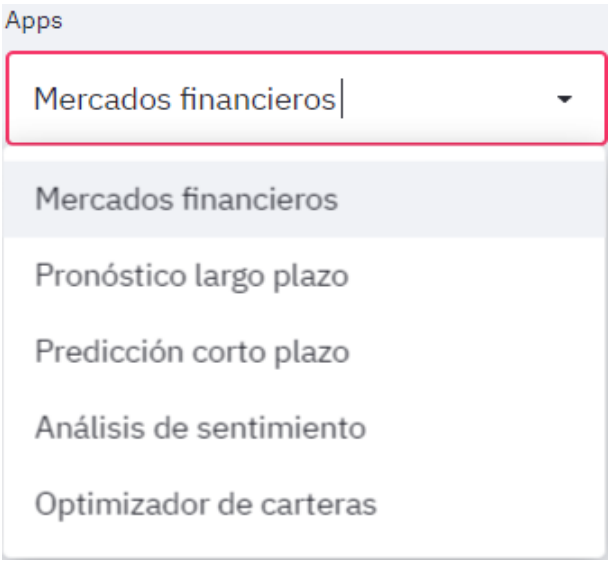


Figura 30. Navegación. Fuente: elaboración propia

6.3 Inicio de sesión y registro

Una vez se inicia la aplicación, se lleva a cabo una conexión con la base de datos users.sqlite3. En el momento que el usuario introduce sus credenciales y pulsa el botón iniciar sesión, el sistema consulta la base de datos y si la información coincide, se valida la sesión. En caso de que los datos sean incorrectos, aparecerá un mensaje de error. En cuanto al proceso de registro, si el usuario pulsa el botón registrarse y los datos introducidos son válidos, estos se registran en la base de datos, mediante una operación de inserción.

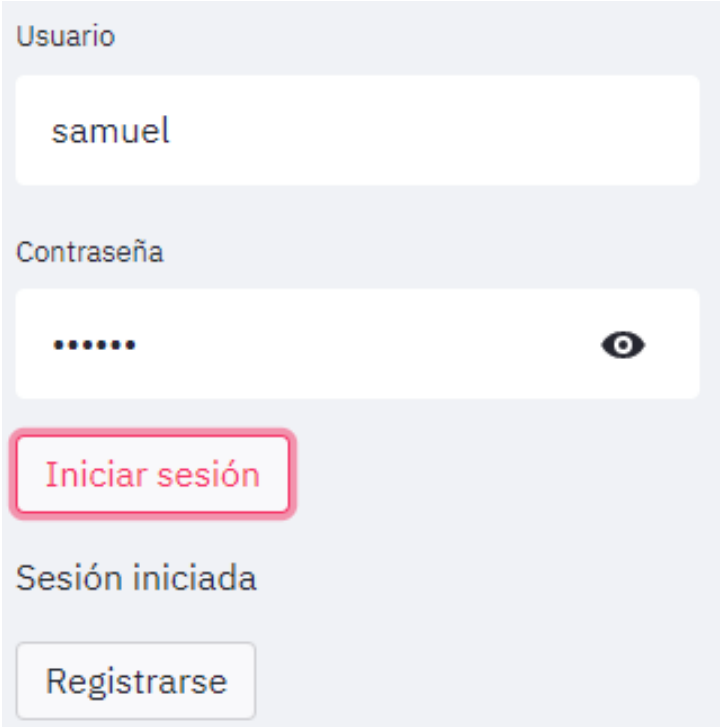


Figura 31. Formulario Login. Fuente: elaboración propia

A continuación, se pueden identificar los dos usuarios que se han registrado en la aplicación y que, por lo tanto, se han almacenado en la base de datos:

	username	password
	Filtro	Filtro
1	samuel	samuel
2	soriano	soriano

Figura 32. Tabla usuarios registrados. Fuente: elaboración propia

6.4 Mercados financieros

En cuanto a la herramienta de visualización y análisis de mercados financieros, se puede decir que presenta un diseño sencillo y acotado. Sin embargo, es suficiente para la necesidad que pretende hacer frente. La finalidad de este apartado es aportar al inversor una visión general de cómo están los mercados. De esta manera, el usuario será capaz de analizar los diferentes activos disponibles rápidamente y sin complicaciones.

En referencia a su funcionamiento, se incluye un desplegable con los tipos de mercados disponibles, en concreto: índices nacionales, materias primas, pares de divisas y criptomonedas. Una vez seleccionado el mercado que se quiere estudiar, el inversor podrá elegir entre los activos disponibles. Por ejemplo, dentro de los índices se pueden encontrar el S&P 500 o el IBEX 35, entre otros.

El diseño de las opciones viene representado por elementos de la librería Streamlit, siendo un selectbox para la elección de mercados y radio buttons para la elección de activos. Además, la selección de las diferentes opciones está controlada mediante condicionales. A continuación, se muestra gráficamente:

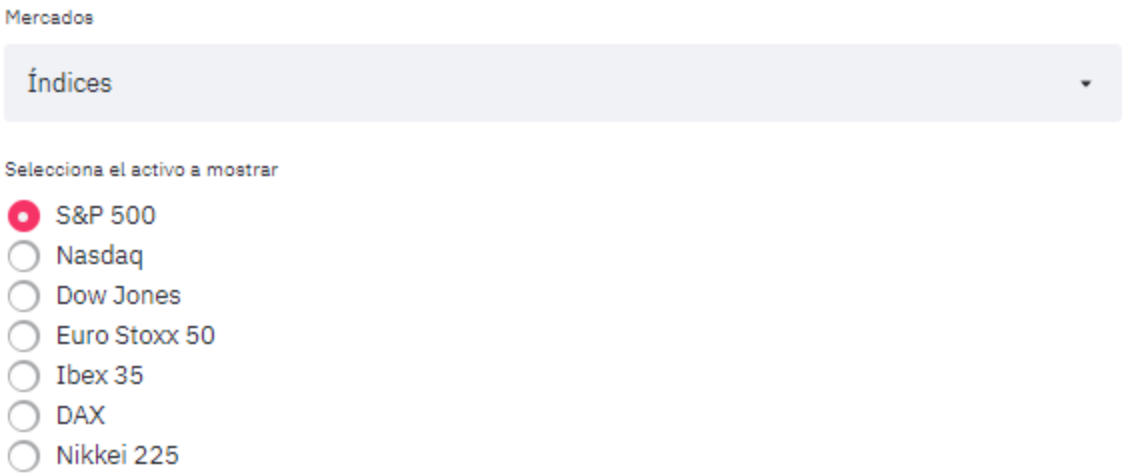


Figura 33. Selección de activos. Fuente: elaboración propia

Una vez seleccionadas las alternativas, el usuario visualizará un gráfico con la evolución del precio de cierre del activo. Además, también se mostrará una tabla con los datos históricos del activo. Cabe destacar que se recogen los registros de todos los días desde el año 2000 hasta la actualidad. Estos datos diarios hacen referencia a las siguientes variables:

- Precio de apertura
- Precio máximo
- Precio mínimo
- Precio de cierre
- Volumen de operaciones

En la siguiente captura se muestra una vista previa:

Datos históricos

	Open	High	Low	Close	Volume
1999-12-31	1,464.4700	1,472.4200	1,458.1899	1,469.2500	374050000
2000-01-03	1,469.2500	1478	1,438.3600	1,455.2200	931800000
2000-01-04	1,455.2200	1,455.2200	1,397.4301	1,399.4200	1009000000
2000-01-05	1,399.4200	1,413.2700	1,377.6801	1,402.1100	1085500000
2000-01-06	1,402.1100	1,411.9000	1,392.1000	1,403.4500	1092300000
2000-01-07	1,403.4500	1,441.4700	1,400.7300	1,441.4700	1225200000
2000-01-10	1,441.4700	1,464.3600	1,441.4700	1,457.6000	1064800000
2000-01-11	1,457.6000	1,458.6600	1,434.4200	1,438.5601	1014000000
2000-01-12	1,438.5601	1,442.6000	1,427.0800	1,432.2500	974600000
2000-01-13	1,432.2500	1,454.2000	1,432.2500	1,449.6801	1030400000
2000-01-14	1,449.6801	1473	1,449.6801	1,465.1500	1085900000

Figura 34. Visualización de mercados. Fuente: elaboración propia

6.5 Pronóstico a largo plazo

Esta aplicación permite al usuario realizar un pronóstico a largo plazo del comportamiento que podría tener el precio de un activo en el futuro. Es una de las herramientas más interesantes ya que ofrece una información aproximada y ajustada a la realidad que puede ayudar al inversor a tomar mejores decisiones. Esta opción está pensada para que el inversor sea capaz de comparar la evolución que pueden tener varios activos y decantarse por la opción más viable.

En referencia a su diseño, lo primero que se muestra es un selectbox donde el usuario deberá elegir un activo. Los activos vienen representados por un ticker o símbolo bursátil, que es un código alfanumérico que sirve para identificarlos de una forma abreviada. Para facilitar esta tarea, se han recopilado un gran número de tickers en el fichero tickers.csv, el cual se importa con la librería pandas. De esta manera el desplegable incluye todas estas opciones y el usuario podrá detectar rápidamente el activo de su interés. Por ejemplo, si queremos seleccionar la criptomoneda de Bitcoin, cuyo ticker es BTC-USD, basta con escribir btc y se mostrarán todos los símbolos que incluyen estas siglas. A continuación, se ve con mayor detalle:

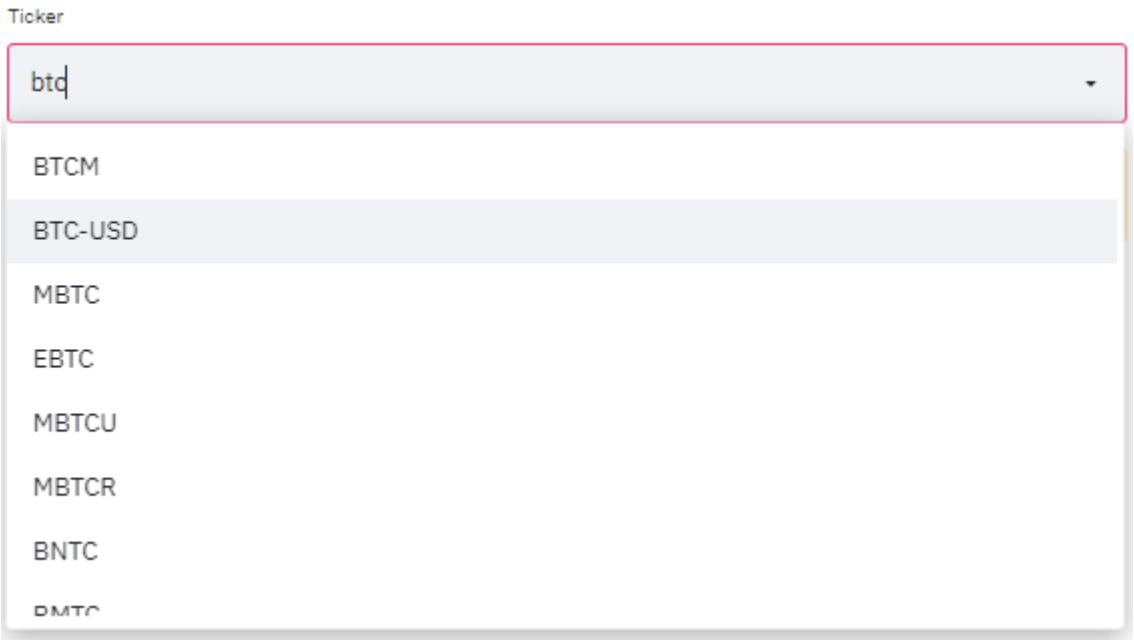


Figura 35. Búsqueda de tickers. Fuente: elaboración propia

Cabe destacar que hasta que el usuario no elija un símbolo el flujo de la aplicación no continua. Esto se controla mediante un condicional. Entonces, una vez seleccionado el activo, la aplicación mostrará la evolución y los datos relacionados con el comportamiento del precio desde 2015. Esto se representa de forma parecida a como lo hace la herramienta de mercados financieros, no obstante, también se incluye una opción de ajuste que permite decidir el horizonte temporal del pronóstico.

En la siguiente imagen se puede ver lo que se mostraría en pantalla:

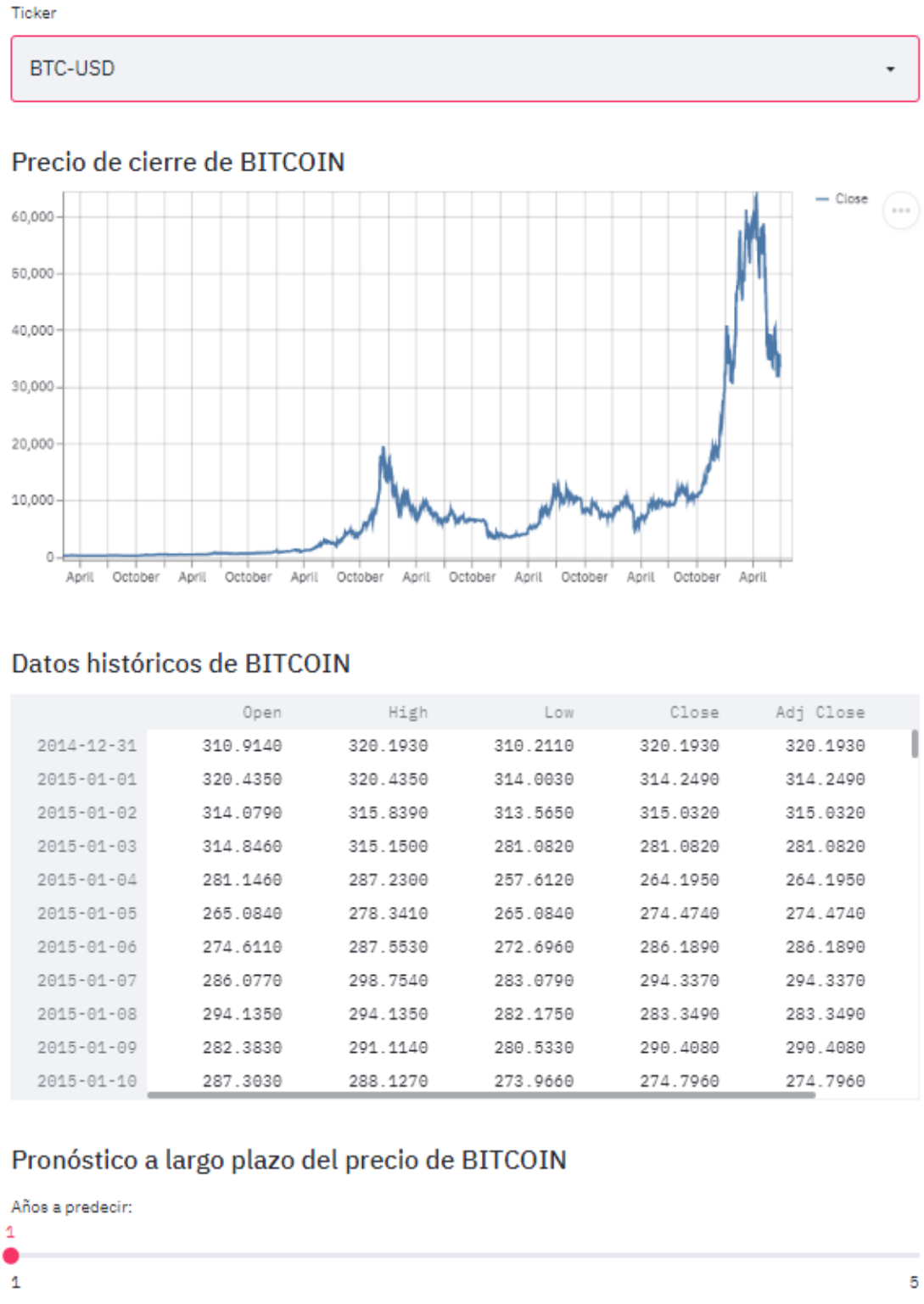


Figura 36. Datos del activo. Fuente: elaboración propia

Tras ajustar los años del pronóstico, el modelo ya puede ser ajustado. No obstante, dado que se utiliza el algoritmo de Facebook Prophet para realizar el pronóstico, se deben modificar los datos para que el ajuste sea válido. Para ello, se utilizará un conjunto de datos que incluya la fecha y el precio de cierre del activo y se nombrarán las columnas como 'ds' para la fecha e 'y' para el precio de cierre. El dataset quedaría de la siguiente manera:

	ds	y
0	2016-01-01	434.334015
1	2016-01-02	433.437988
2	2016-01-03	430.010986
3	2016-01-04	433.091003
4	2016-01-05	431.959991
...	...	...

Figura 37. Estructura de datos. Fuente: elaboración propia

Una vez ajustados los datos y realizado el pronóstico se mostraría el gráfico. En este gráfico se pueden distinguir cuatro elementos:

- Precio de cierre histórico: representado con puntos negros.
- Banda superior del pronóstico: refleja un escenario optimista y está representada en azul claro.
- Línea tendencial central: refleja el escenario más realista siguiendo la tendencia del precio y está representando con una línea de color azul.
- Banda inferior del pronóstico: refleja el escenario pesimista y está representada en azul claro.

Además, el usuario puede interactuar con el gráfico y se incluyen las siguientes funciones:

- Modo pantalla completa
- Descargar el gráfico en formato png
- Seleccionar la zona que se quiere observar en modo zoom
- Moverse dentro del gráfico
- Aplicar o quitar zoom
- Resetear la escala del gráfico
- Utilizar el punto de referencia que indica la fecha y el precio seleccionados mediante dos lineas que se cruzan
- Seleccionar temporalidad

El eje de la x representa los años y el eje de la y representa el precio del activo. En la siguiente imagen se puede ver el gráfico en detalle:

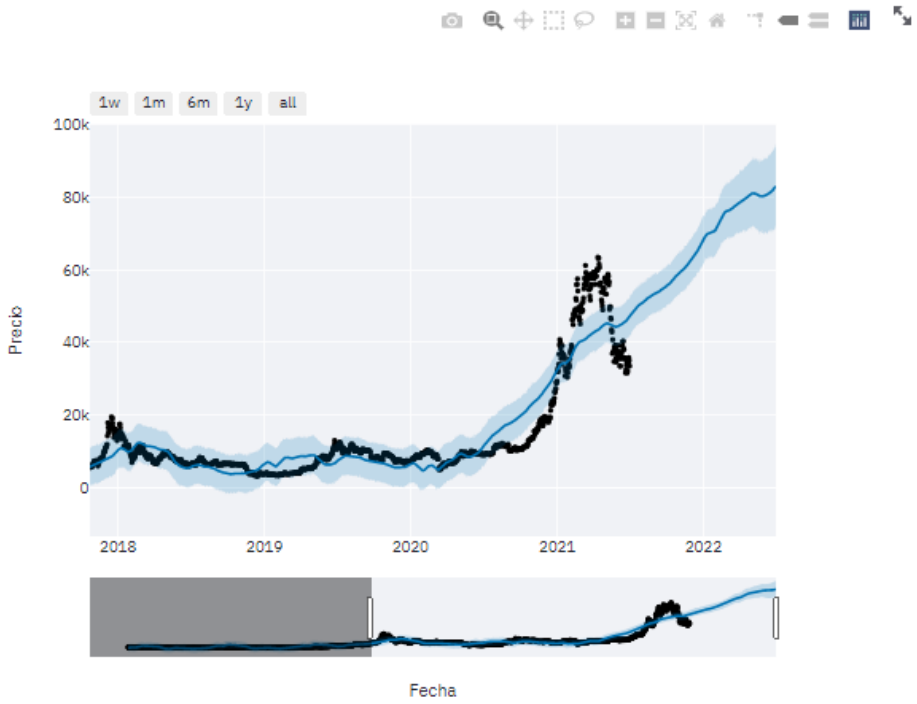


Figura 38. Pronóstico. Fuente: elaboración propia

Desde este gráfico, se puede interpretar que el precio de Bitcoin para el próximo año puede oscilar entre 75.000$ y 90.000$, según el pronóstico realizado.

Además, el pronóstico también cuenta con tres componentes que sirven para analizar la estacionalidad que sigue el activo en cuestión. Los componentes que incluye el pronóstico son los siguientes:

- Trend: refleja la tendencia que ha seguido el activo en los últimos años y la tendencia más probable que seguirá en el futuro.
- Weekly: muestra la tendencia que sigue el activo a lo largo de la semana, por lo que indica el comportamiento más frecuente que sigue el precio en cada día.
- Yearly: muestra la tendencia que sigue el activo a lo largo del año, por lo que indica el comportamiento más frecuente que sigue el activo en cada mes.

En la siguiente imagen se puede observar con detenimiento cada uno de los componentes que incluye este pronóstico:

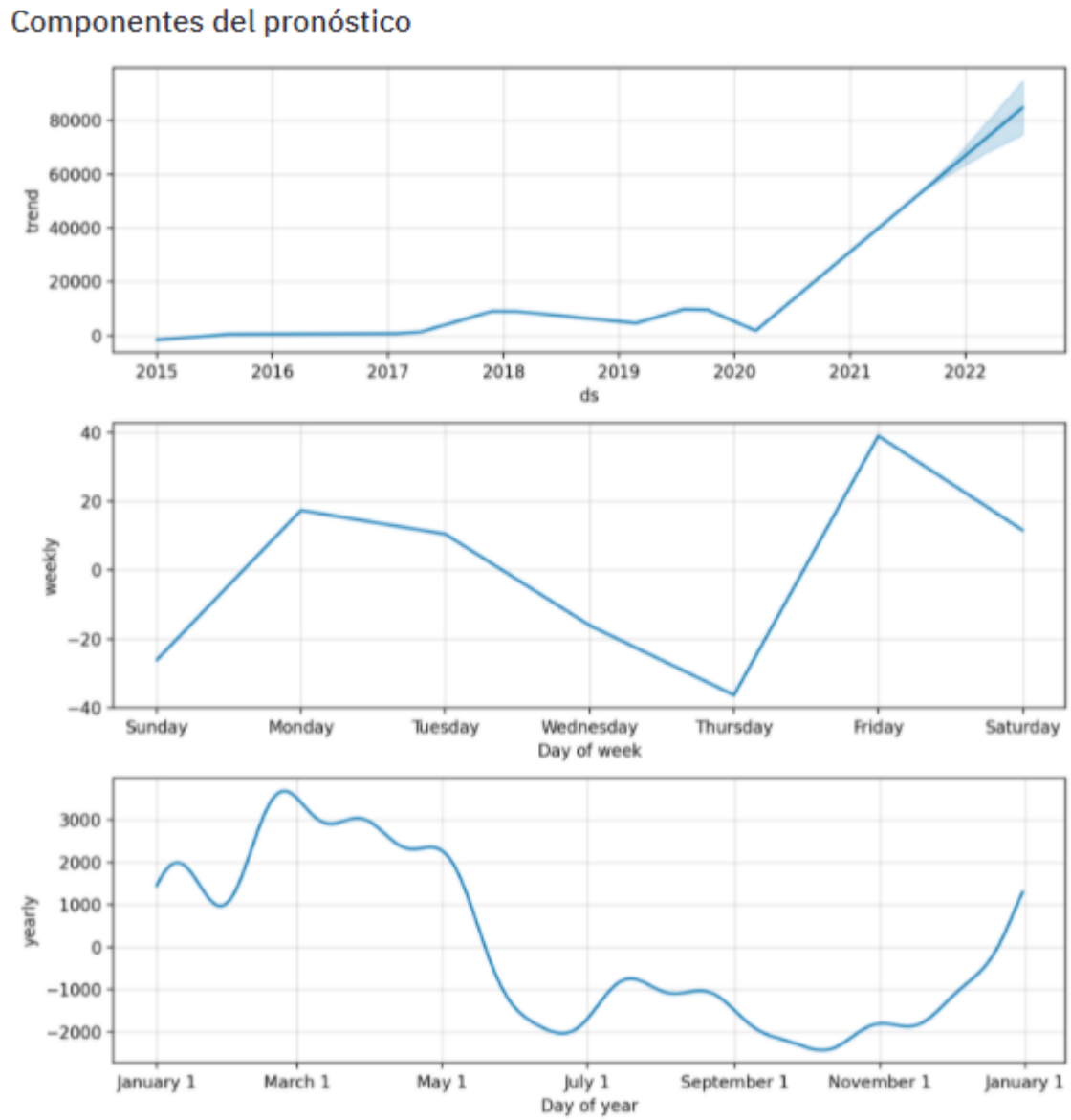


Figura 39. Componentes del pronóstico. Fuente: elaboración propia

Como se puede observar, la tendencia que sigue Bitcoin desde 2020 es muy positiva y se espera que siga al alza. Además, hay que tener en cuenta que el día que más suele bajar el precio es el jueves y el día que más alto suele estar es el viernes. Por otro lado, en cuanto al comportamiento anual, los meses de primavera y verano son los más duros, en cambio desde noviembre hasta abril, el precio experimenta una gran subida y se mantiene al alza.

Este tipo de análisis aporta información relevante sobre cuando realizar la inversión con el fin de obtener la mayor rentabilidad posible.

6.6 Predicción a corto plazo

En un principio, esta herramienta se iba a incluir dentro de la aplicación de pronóstico, sin embargo, por motivos de rendimiento se ha decidido ponerlas por separado. Aunque se trata de una observación resumida, el algoritmo de la predicción a corto plazo es el más complejo, ya que incluye un método de aprendizaje profundo basado en redes neuronales recurrentes.

Según Atriainnovation (2019), Las redes neuronales artificiales son un modelo inspirado en el funcionamiento del cerebro humano. Está formado por un conjunto de nodos conocidos como neuronas artificiales que están conectadas y transmiten señales entre sí. Estas señales se transmiten desde la entrada hasta generar una salida. Su funcionamiento viene reflejado en la siguiente imagen.

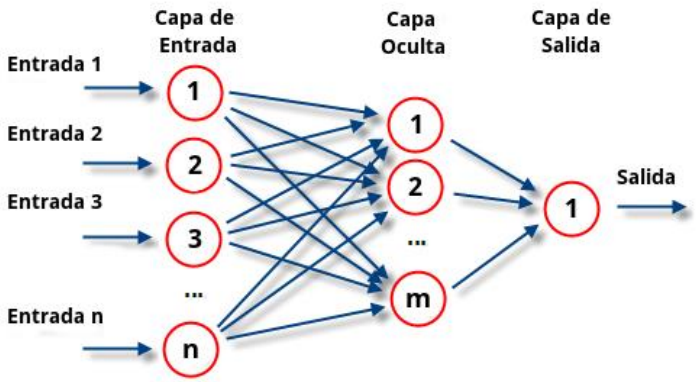


Figura 40. Redes neuronales artificiales. Fuente: atriainnovation.com

No obstante, en esta ocasión se van a utilizar redes neuronales recurrentes. Según Calvo (2018) una red neuronal recurrente no tiene una estructura de capas definida, sino que permiten conexiones arbitrarias entre las neuronas, incluso pudiendo crear ciclos, con esto se consigue crear la temporalidad, permitiendo que la red tenga memoria.

En concreto, se van a utilizar las redes LSTM (Long Short Term Memory) que son un tipo de red neuronal recurrente que da más importancia a los acontecimientos más recientes para eliminar errores acumulados en el largo plazo. Para ello, el método que emplea es aplicar una serie de filtros que mediante ciertas operaciones deciden qué información se debe almacenar y qué información debe ser desechada. A continuación se puede ver en detalle un ejemplo de una red neuronal recurrente LSTM:

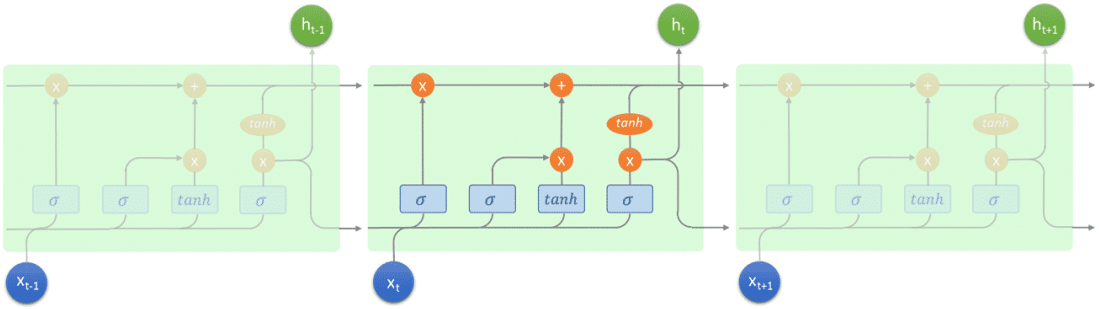


Figura 41. Redes LSTM. Fuente: diegocalvo.es

Por lo tanto, el algoritmo que se va a utilizar para realizar las predicciones consiste en analizar datos de 60 días anteriores para estimar el siguiente. Es decir, la predicción del día 61 vendrá dada por el comportamiento que haya tenido el precio los días anteriores. Para ello, lo primero es obtener el conjunto de datos que incluye lo siguiente: precio de apertura, precio máximo, precio mínimo y precio de cierre de cada día. La tabla de datos quedaría de la siguiente manera:

Date	Open	High	Low	Close
2015-01-02	27.847500	27.860001	26.837500	27.332500
2015-01-05	27.072500	27.162500	26.352501	26.562500
2015-01-06	26.635000	26.857500	26.157499	26.565001
2015-01-07	26.799999	27.049999	26.674999	26.937500
2015-01-08	27.307501	28.037500	27.174999	27.972500
...	...	...	...	...

Figura 42. Datos de entrada. Fuente: elaboración propia

El siguiente paso consiste en escalar los datos para que la red neuronal sea capaz de entrenar y ajustar el modelo de la mejor manera posible. Una vez realizado el escalado, se crea el conjunto de entrenamiento, que en este caso corresponde con todos los datos excepto los últimos 60 registros. Posteriormente, se procede a entrenar el modelo con los datos de entrenamiento y después se realiza la predicción del precio de cierre para el día siguiente, a partir de los 60 últimos días.

Cabe destacar que el modelo cuenta con dos ramas de entrada, una de ellas corresponde con las cuatro columnas de datos explicadas y la otra con una media móvil de 60 periodos para el precio de cierre. De esta manera, se aplican conceptos de análisis técnico para mejorar la eficacia del modelo. Por otro lado, la capa de salida corresponde con el precio de cierre de los días a predecir. De este modo, el precio de apertura, el precio máximo, el precio mínimo y el precio de cierre de los días 1 al 60, junto con la media móvil de 60 periodos, intentarán predecir el precio del día 61.

A continuación, se muestra la arquitectura del modelo, para ver su funcionamiento en detalle:

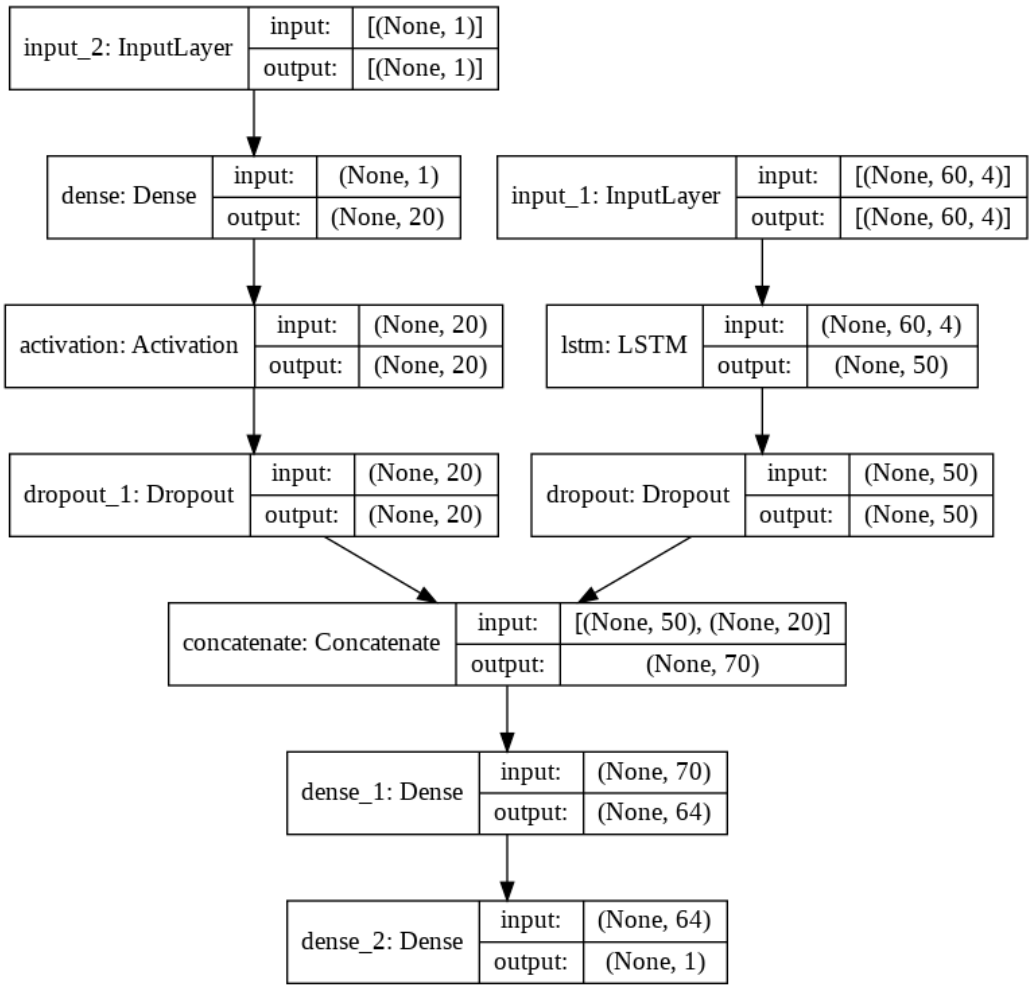


Figura 43. Modelo. Fuente: elaboración propia (Keras)

Como se puede observar en el modelo se presentan dos ramas de entrada que finalmente se concatenan para obtener el resultado o la salida mediante una capa densa. Tanto las dos entradas como la salida del modelo se explican a continuación:

- **Input 1**: corresponde a los 60 últimos datos de las 4 columnas mencionadas. Se definen 50 unidades para la capa LSTM e incluye un dropout de 20% para evitar el sobreajuste.

- **Input 2**: corresponde con la media móvil de 60 periodos. Se definen 20 unidades para la capa densa, se establece una capa de activación de tipo rectificador lineal y se añade un dropout de 20% para evitar el sobreajuste.

- **Output:** las dos entradas anteriores se concatenan y se añaden dos capas densas. La primera capa cuenta con 64 unidades y una activación de tipo sigmoide. En cambio la segunda capa de salida y última capa del modelo cuenta con una unidad y el tipo de activación es linear.

Se han realizado distintas pruebas y esta configuración ha sido la que mejores resultados ha dado. Además, se han probado diferentes optimizadores y el mejor ha sido el de Adam con la función de pérdida MSE (Mean Square Error).

En cuanto al diseño, es parecido al de pronóstico a largo plazo, aunque en este caso el gráfico presenta la evolución del último año y solo se incluyen los datos de los últimos 60 días, que son los utilizados para realizar la predicción del día siguiente. Por último, al final de la página se muestra el resultado de la predicción con el porcentaje de variación que se prevé que va a experimentar el activo.

Figura 44. Predicción a corto plazo. Fuente: elaboración propia

6.7 Análisis de sentimiento

Según Bell (2020), el procesamiento del lenguaje natural es una rama de la inteligencia artificial que permite a los ordenadores entender, procesar y generar lenguaje tal como lo hacen las personas. Su uso en los negocios está creciendo rápidamente.

En este caso, se aplicará esta técnica para elaborar análisis de sentimientos, que según Itelligent (2017), se trata de una tarea de clasificación masiva de documentos de manera automática, que se centra en catalogar los documentos en función de la connotación positiva o negativa del lenguaje ocupado en el mismo.

Por lo tanto, se analizarán los comentarios de los usuarios de internet acerca de un activo financiero, con el fin de identificar si la opinión general es positiva o negativa. Para ello, se utilizará la API de Twitter para obtener una gran cantidad de tweets y poder llevar a cabo un estudio contrastado. Además, se hará uso de expresiones regulares para formatear el texto y eliminar caracteres especiales. Por último, mediante la librería TextBlob se obtendrá la polaridad y la subjetividad de cada mensaje para posteriormente clasificarlos mediante sentencias condicionales.

Para acceder a la API de Twitter, es necesario registrar una cuenta y crear un proyecto como el que vemos a continuación. Este proyecto ofrece permisos de lectura y para autentificar el acceso y poder trabajar con la API es necesario generar keys y tokens. Estas claves se utilizarán a la hora de enviar las peticiones desde la aplicación.

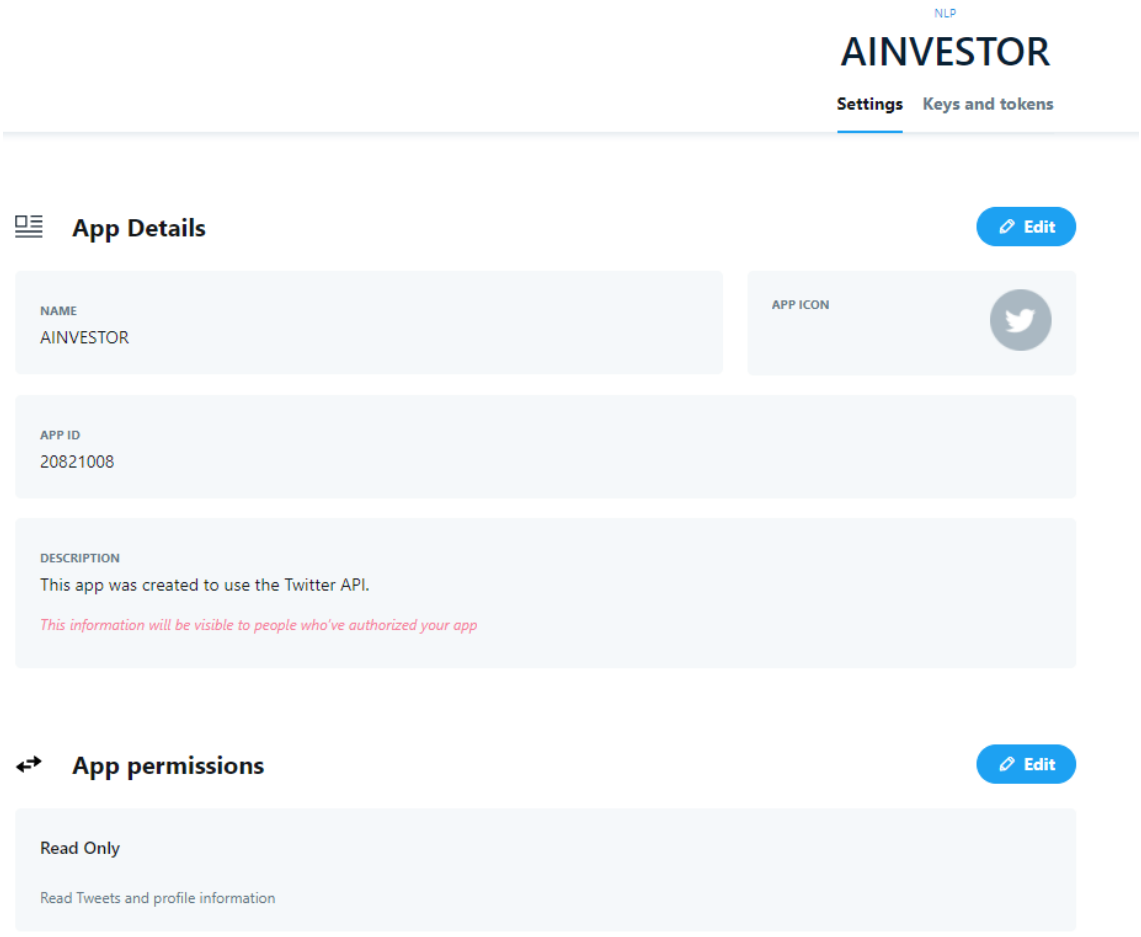


Figura 45. API Twitter. Fuente: developer.twitter.com

En este momento, el proyecto cuenta con un plan Standard sin ningún coste, pero el número de peticiones está limitado a 500.000 tweets al mes. No obstante, el plan se puede mejorar mediante una suscripción.

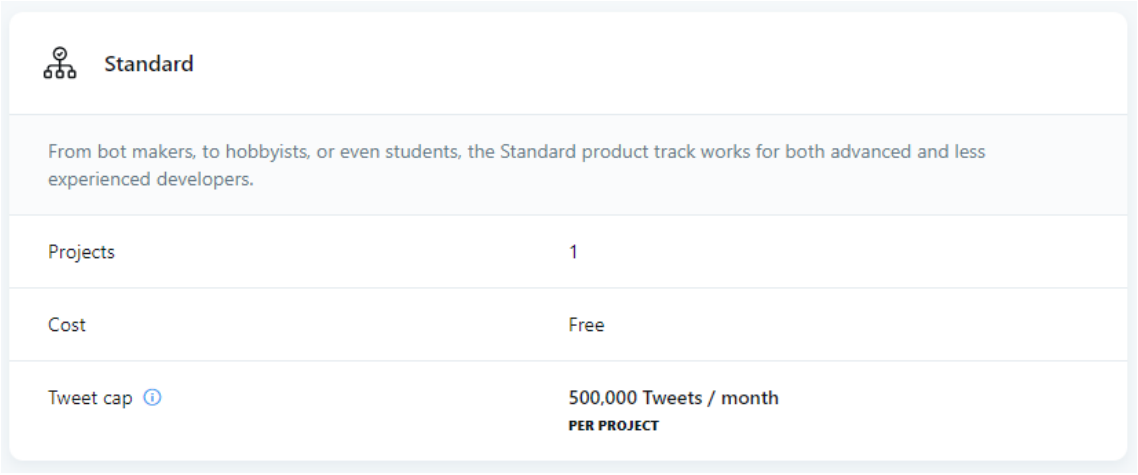


Figura 46. Plan API Twitter. Fuente: developer.twitter.com

Una vez obtenido el acceso a la API, ya se pueden realizar peticiones a la misma. Para ello, el usuario escribirá el término que desea analizar y se utilizará un filtro de Retweets, de manera que las publicaciones con más repercusión serán las que se tendrán en cuenta. Además, el usuario podrá seleccionar la fecha desde la que se recopilarán los tweets, como se puede ver a continuación:

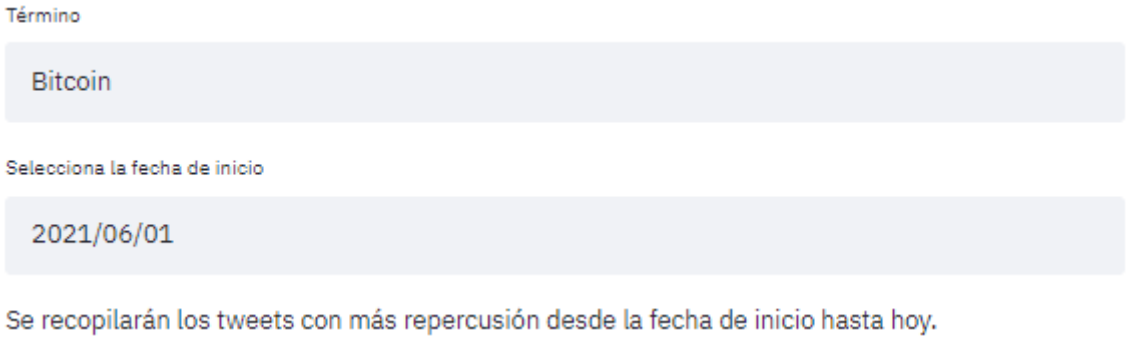


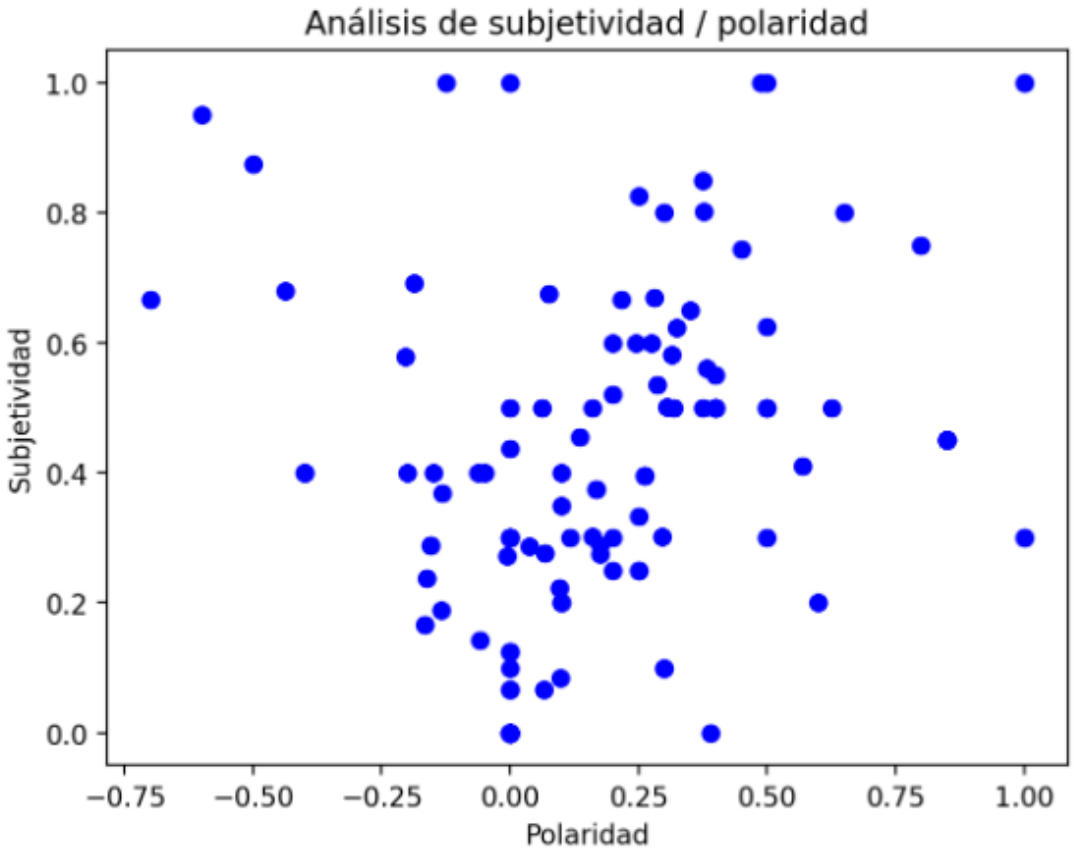
Figura 47. Término y fecha para el análisis. Fuente: elaboración propia

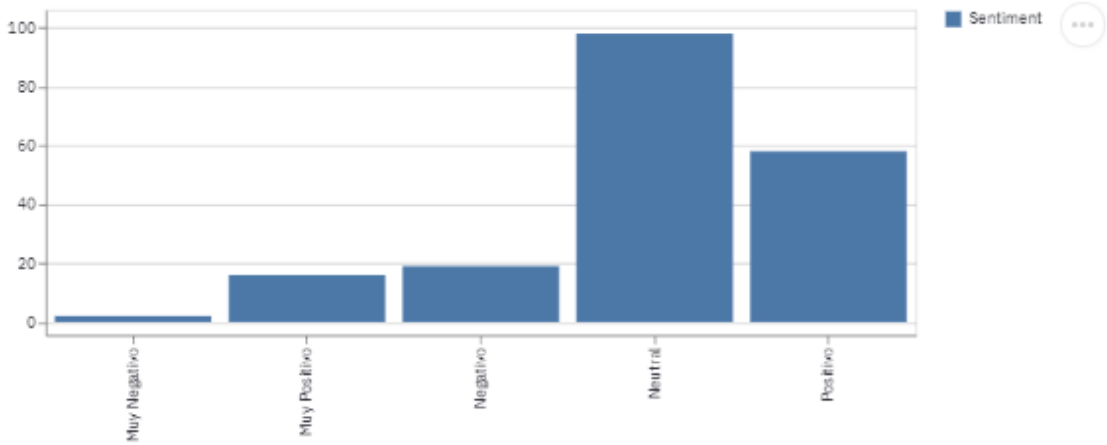
Tras ajustar la búsqueda el usuario podrá observar dos tipo de gráficos como resultado:

- **Análisis de subjetividad/polaridad**: cada punto representa un tweet y se posiciona en base a la polaridad (eje x) y a la subjetividad (eje y). Cuanto mayor sea la polaridad el tweet será más positivo y cuanto mayor sea la subjetividad el tweet se basará en emociones o intereses personales.

- **Clasificación de mensajes**: se clasifican los tweets según su polaridad en una escala del 1 al 5 representada como muy negativo, negativo, neutral, positivo y muy positivo.

A continuación se muestran los gráficos descritos:





	Sentiment
Neutral	98
Positivo	58
Negativo	19
Muy Positivo	16
Muy Negativo	2

Figura 48. Análisis de sentimiento. Fuente: elaboración propia

6.8 Optimizador de carteras

La herramienta de optimización de carteras está pensada principalmente para acciones de compañías. Por lo que el usuario será capaz de formar un portafolio de inversión con las acciones que quiera analizar. Para ello, se ha diseñado un selectbox igual que en otras aplicaciones, aunque en este se permite incluir todos los símbolos que sean necesarios. También incluye una extensa lista de tickers para seleccionar, para facilitar el trabajo al inversor. A continuación, se muestra la vista antes de introducir los valores:

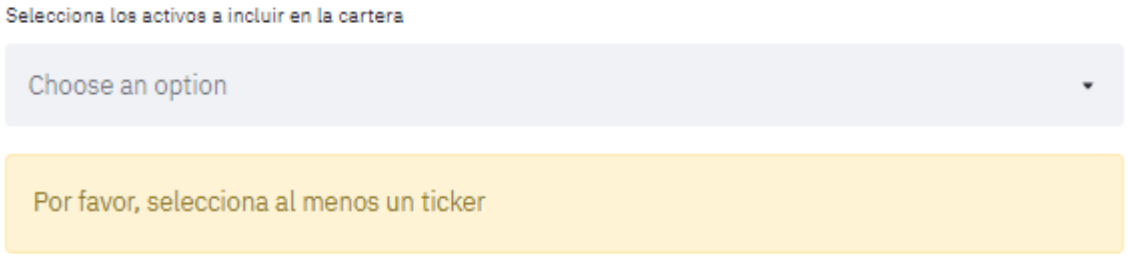


Figura 49. Selección múltiple de activos. Fuente: elaboración propia

A medida que se vayan introduciendo activos, habrá un gráfico que se irá actualizando e incluyendo el precio de las acciones para que el inversor pueda comprobar rápidamente el comportamiento que han ido siguiendo los activos en los últimos años.

Según Ennaranja (2019), la correlación es un término estadístico. Se refiere a la relación lineal que existe entre dos o más variantes. En el ámbito de la inversión, cuando dos mercados están muy correlacionados de forma positiva, su comportamiento tiende a ser similar ante los mismos estímulos. Es una de las medidas que debes valorar al estudiar la composición de tu cartera de inversión. De hecho, es una forma de asegurar una buena diversificación de tus inversiones.

Es por ello por lo que, tras seleccionar las acciones pertinentes, se muestra una matriz de correlación. Desde esta matriz el usuario puede comparar las diferentes correlaciones que existen entre las empresas y de esta manera tomar mejores decisiones a la hora de decantarse por un activo o por otro.

En la siguiente imagen se puede observar un ejemplo de cartera de inversión también conocida como las FAANMG (Facebook, Apple, Amazon, Netflix, Microsoft y Google) que se corresponden con las empresas tecnológicas con mayor capitalización de mercado. Además de poder comparar su precio, también podemos comprobar que todas presentan una alta correlación entre ellas. Esto es debido a que todas operan en el sector tecnológico y su comportamiento en el mercado es muy parecido.

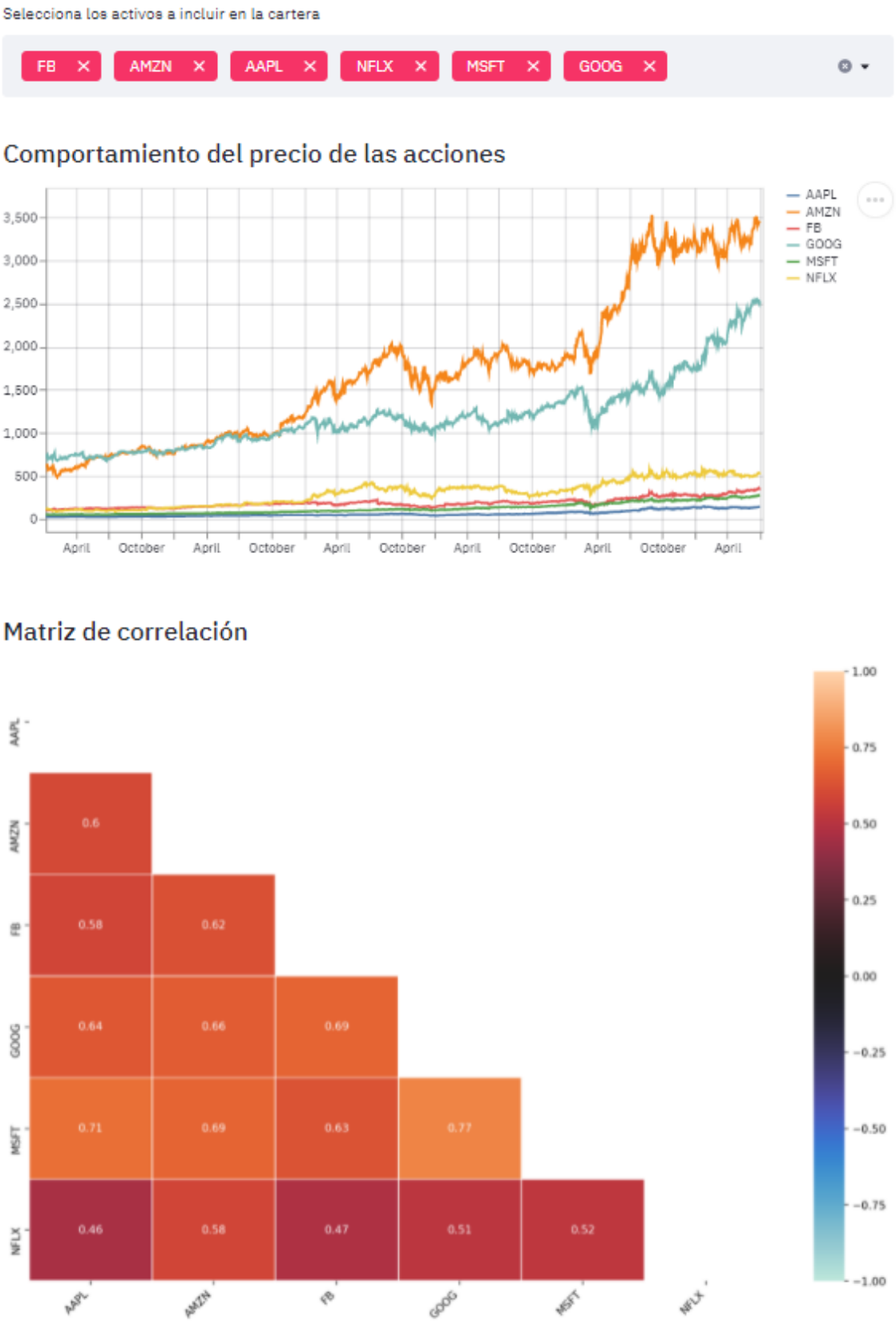


Figura 50. Información de la cartera. Fuente: elaboración propia

Una vez analizado el portafolio formado, se realizará la optimización del peso que debe tener cada activo en la cartera. Para ello, se utiliza la librería PyPortfolioOpt que incluye diferentes operaciones de optimización y visualización de resultado.

En este caso, se tiene como referencia la ratio de Sharpe, que según Arenas (2016), realiza una medición numérica de la relación Rentabilidad / Volatilidad Histórica (la cual también es conocida como desviación standard). Teniendo en cuenta que a mayor rentabilidad mayor riesgo, el algoritmo intentará maximizar esta ratio con el fin de obtener la máxima rentabilidad a costa de un riesgo moderado.

A continuación, podemos ver la distribución óptima de los pesos:

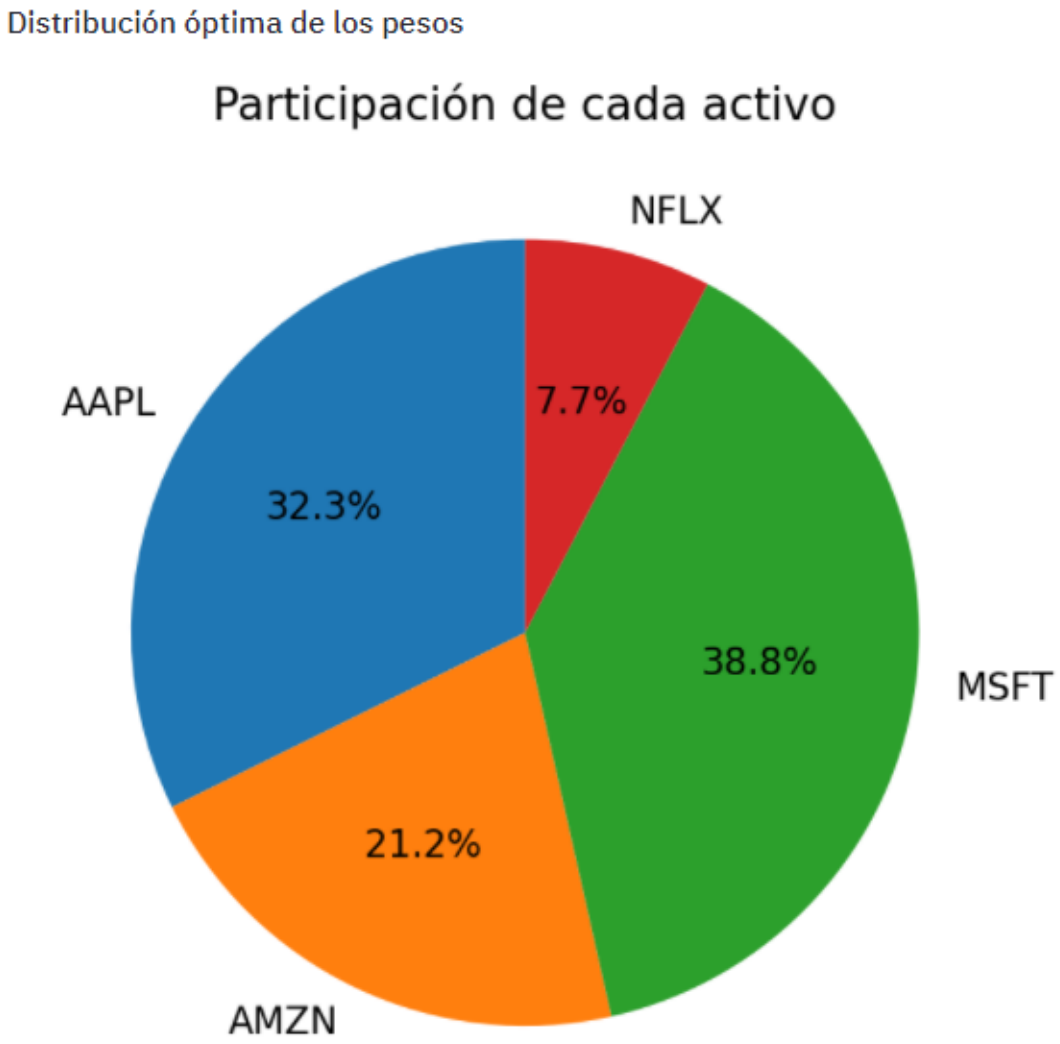
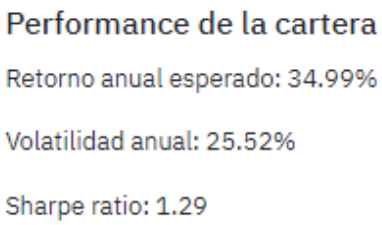


Figura 51. Distribución óptima. Fuente: elaboración propia

Por último, se ofrece información sobre el resultado esperado de la cartera y la ratio de Sharpe, con el fin de que el inversor pueda comparar diferentes carteras de inversión.



Además, una de las opciones más interesantes para el inversor es introducir la cantidad que está dispuesto a invertir con el fin de obtener lo que sería la composición de su cartera en detalle. En el siguiente ejemplo se ha seleccionado un capital de 100.000$ y acorde a los pesos optimizados y al precio actual de cada activo se especifica el número de acciones que el usuario debería adquirir.

Ejemplo:

AAPL = 236 x 137,27 = 32.395,72 → 32,3%

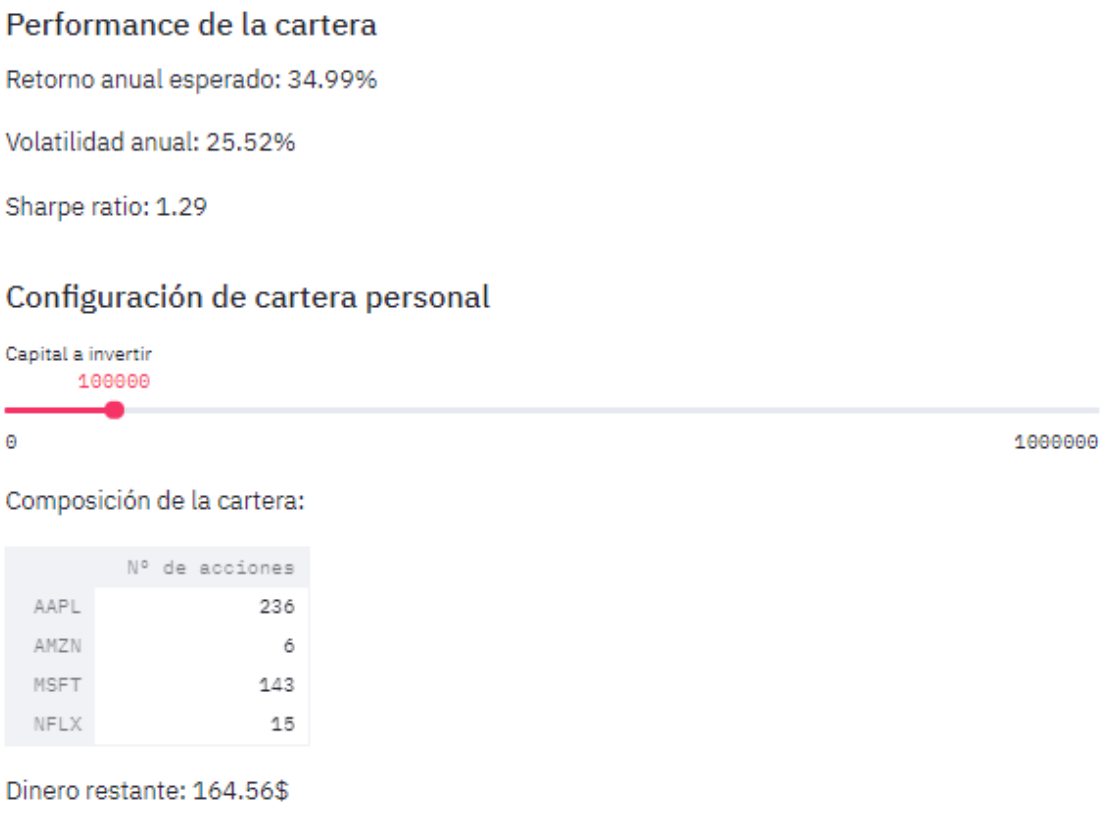


Figura 52. Resultados de la cartera. Fuente: elaboración propia

Cabe destacar que este tipo de herramienta se basa en la información pasada, por lo que no puede asegurar rentabilidades futuras.

7. Resultados

Tras haber desarrollado la aplicación, se va a mostrar la vista previa de lo que sería la aplicación desplegada. Cabe destacar que, se ha incluido el formulario de inicio de sesión y registro en el sidebar. Además, el botón de la parte superior derecha permite configurar las opciones del sitio web, como cambiar el tema o la vista.

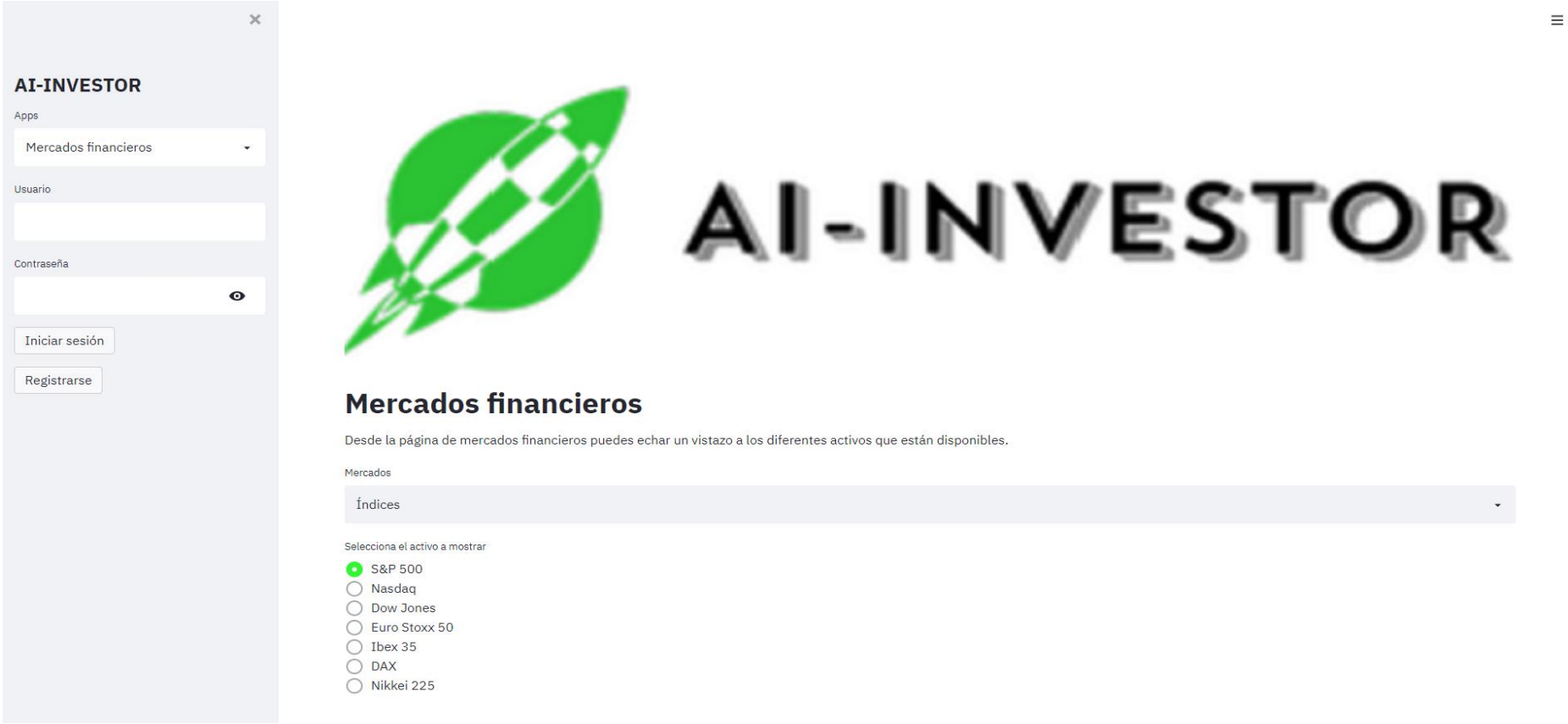


Figura 53. Página de inicio. Fuente: elaboración propia

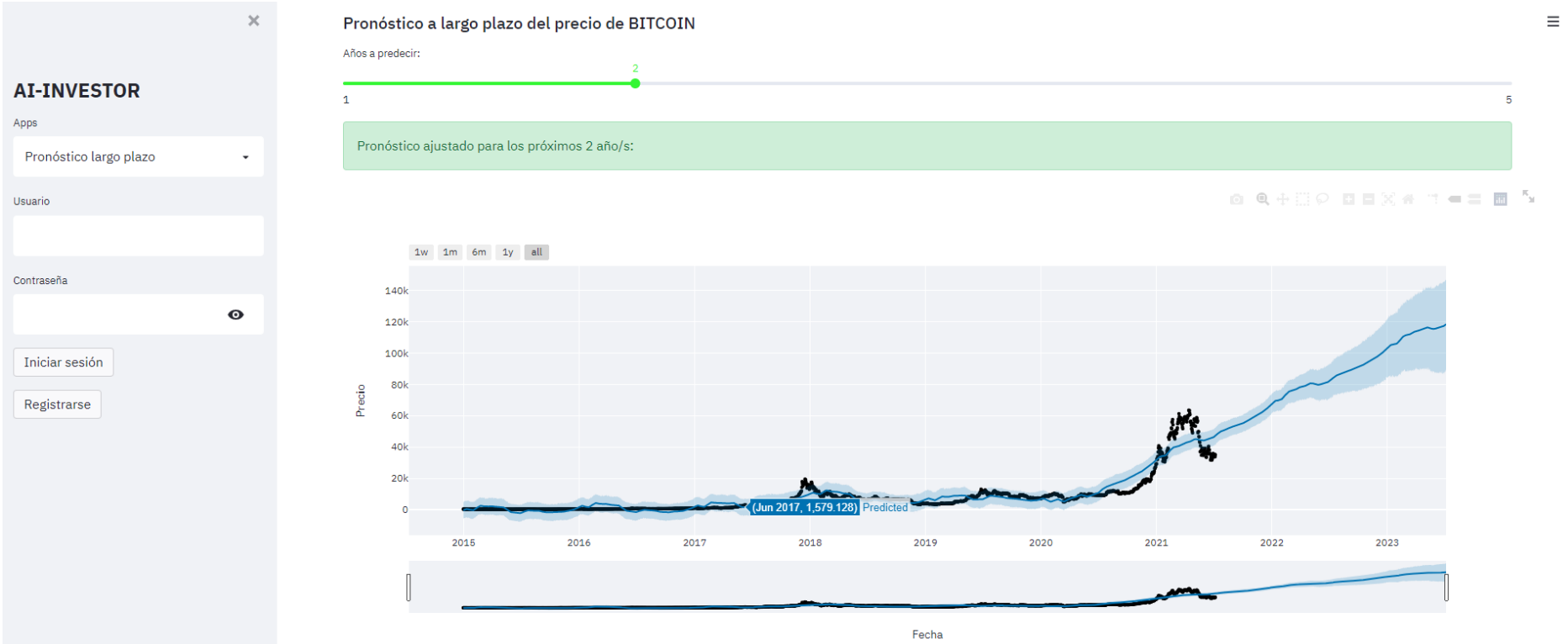


Figura 54. Pronóstico a largo plazo. Fuente: elaboración propia

Figura 55. Predicción a corto plazo. Fuente: elaboración propia

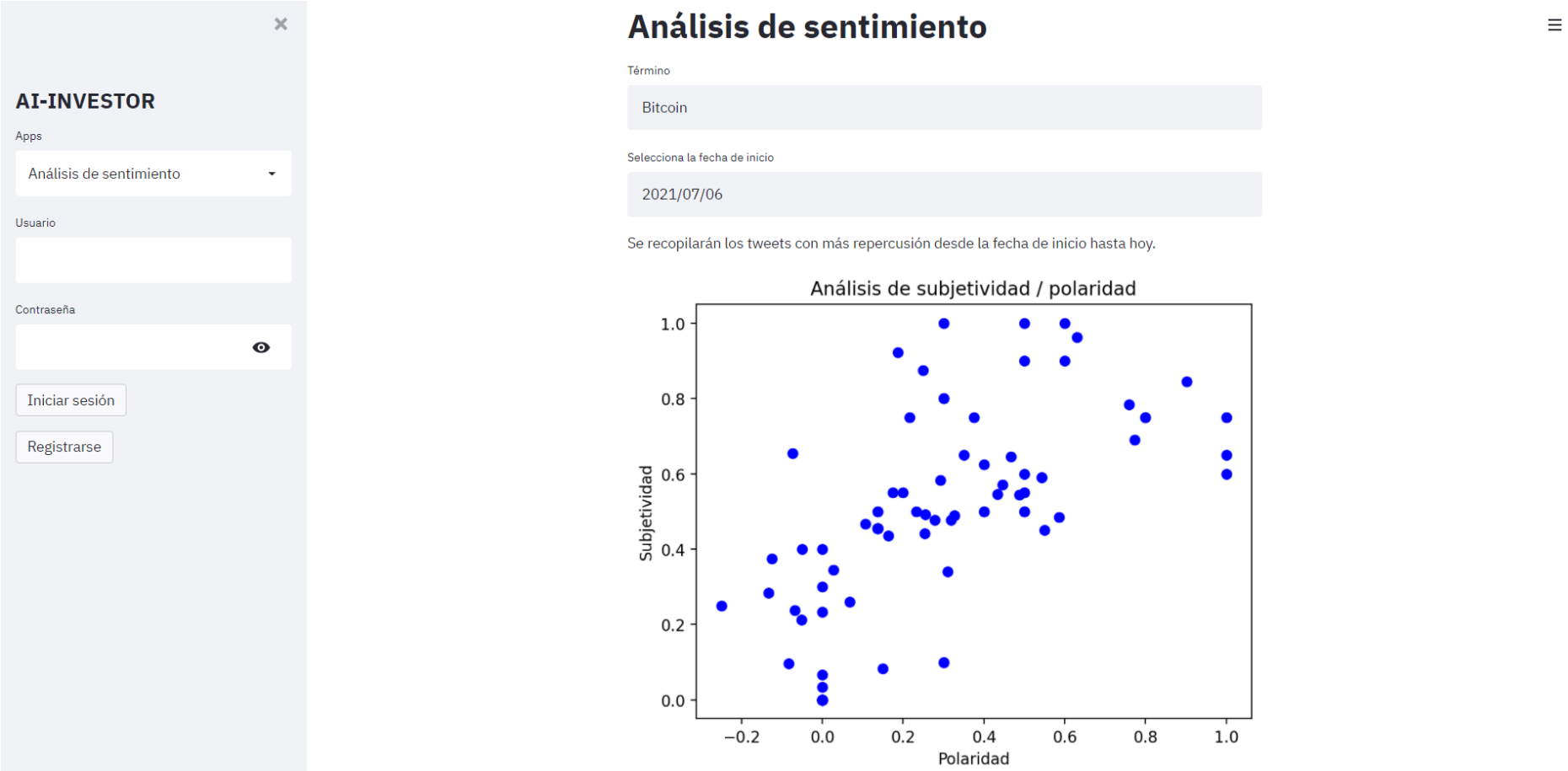


Figura 56. Análisis de sentimiento. Fuente: elaboración propia

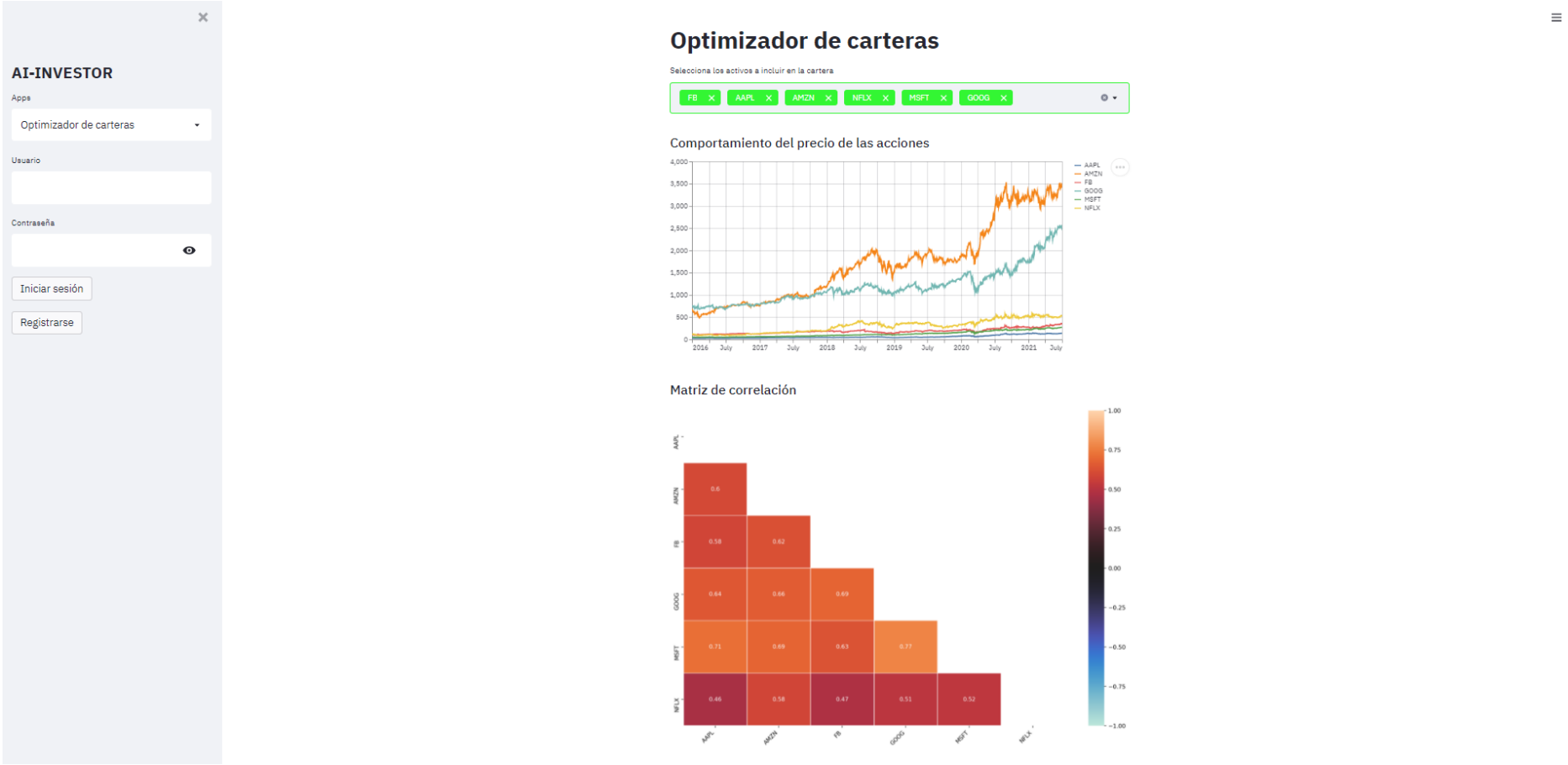


Figura 57. Información de cartera. Fuente: elaboración propia

8. Conclusiones y líneas futuras

Como conclusión, se puede decir que la aplicación desarrollada cumple con todos los requisitos propuestos al principio de la memoria. De esta manera se ofrece al usuario un conjunto de potentes herramientas de análisis con el fin de apoyar su toma de decisiones con respecto a la inversión en los mercados financieros.

Durante el proyecto se han llevado a cabo diferentes tareas dentro del ámbito del análisis, el diseño y la implementación. Es por ello por lo que a continuación se resumen todos los apartados vistos a lo largo del documento.

Una vez introducido el proyecto y desarrollada la idea, se ha analizado el mercado con el fin de estudiar el estado del arte e investigar las aplicaciones que ofrecen un servicio similar. Esto ha servido para establecer estrategias y detectar oportunidades a la hora de desarrollar la nueva plataforma.

En la fase de análisis, se han especificado todos los requisitos que la aplicación debía de cumplir y se han valorado todas las propuestas de desarrollo, decantándose finalmente por la aplicación web como solución.

Por otro lado, la fase de diseño de la solución ha servido para detectar todas aquellas herramientas y tecnologías necesarias para llevar a cabo el proyecto. Además, se ha detallado el paradigma a utilizar y la clase principal en la que se basa la aplicación. También se ha incluido un diagrama de casos de uso donde se identifican todas las acciones que el usuario puede llevar a cabo.

La fase más extensa ha sido la de implementación, la cual incluye información acerca de cómo se han construido y programado cada una de las funciones incluidas en la aplicación.

Como conclusión personal, además de poner en práctica los conocimientos adquiridos en el Grado de Ingeniería Informática, he abordado nuevos conceptos y tecnologías que me han permitido aprender y desarrollar nuevas habilidades a medida que avanzaba el proyecto. Por lo tanto, este trabajo ha supuesto todo un reto, pero ha sido realmente gratificante para mí.

Como futuras mejoras y líneas a seguir, se puede decir que el desarrollo de la aplicación a alcanzado una fase funcional. Sin embargo, faltan muchos aspectos de mejora por implementar, como por ejemplo, un apartado de perfil, desde donde el usuario pueda ver y modificar la información de su cuenta. Aun así, se pretende continuar con el desarrollo de la aplicación y seguir el plan de negocio descrito en el Trabajo de Final de Grado de Administración y Dirección de Empresas. El objetivo a medio plazo es constituir una sociedad y contratar al personal indicado para llevar el proyecto a la realidad y convertirlo en un negocio rentable. A largo plazo, el objetivo sería internacionalizar la aplicación y añadir nuevas mejoras y funcionalidades, con la finalidad de desarrollar una aplicación competente, que abarque cada vez un mercado mayor.

9. Bibliografía

ARENAS, E (2016). *Ratio de Sharpe*. <https://www.rankia.mx/blog/como-comenzar-invertir-bolsa/3354504-que-ratio-sharpe-analiza-rendimiento-inversion> [Consulta: 5 de Julio de 2021]

ATRIAINNOVATION (2019). *¿Qué son las redes neuronales?* <https://www.atriainnovation.com/que-son-las-redes-neuronales-y-sus-funciones> [Consulta: 5 de Julio de 2021]

BELL, T (2020). *Procesamiento del lenguaje natural*. <https://www.computerworld.es/tecnologia/que-es-y-como-funciona-el-procesamiento-del-lenguaje-natural> [Consulta: 5 de Julio de 2021]

CALVO, D (2018). *Redes neuronales recurrentes*. <https://www.diegocalvo.es/red-neuronal-recurrente> [Consulta: 5 de Julio de 2021]

ENNARANJA, ING (2019). *La correlación*. <https://www.ennaranja.com/inversion/que-es-la-correlacion/> [Consulta: 5 de Julio de 2021]

ITELLIGENT (2017). *Análisis de sentimiento*. <https://itelligent.es/es/analisis-de-sentimiento> [Consulta: 5 de Julio de 2021]

10. Anexo I (código fuente)

Se ha decidido incluir el código fuente de la aplicación con el fin de que la lectura del documento sea más llevadera. A continuación, se presenta el código en lenguaje Python de todos los scripts que intervienen en el funcionamiento de la aplicación:

10.1 multiapp.py

```python
# Importar librerías
import streamlit as st

# Clase que ejecuta varias aplicaciones
class MultiApp:

    # Se crea un array donde se almacenará el título y la función de cada app
    def __init__(self):
        self.apps = []

    # Función para añadir apps al array
    def add_app(self, title, func):
        self.apps.append({
            "title": title,
            "function": func
        })

    # Función que ejecuta la app seleccionada mediante el selectbox
    def run(self):
        app = st.sidebar.selectbox('Apps', self.apps, format_func=lambda app: app['title'], )
        app['function']()
```

10.2 main.py

Esta es la aplicación principal, desde donde se ejecutan las otras y donde se incluye el apartado de login.

```python
# Importar librerías
import streamlit as st
from multiapp import MultiApp
from apps import markets, forecast, prediction, sentiment, portfolio
import pandas as pd
import sqlite3

# Objeto de la clase MultiApp()
main = MultiApp()

# Apps disponibles
main.add_app("Mercados financieros", markets.app)
main.add_app("Pronóstico largo plazo", forecast.app)
main.add_app("Predicción corto plazo", prediction.app)
main.add_app("AnÃ¡lisis de sentimiento", sentiment.app)
main.add_app("Optimizador de carteras", portfolio.app)

# Logo tÃtulo y llamada ejecución de la aplicación seleccionada
st.image('logo.png')
st.sidebar.title('AI-INVESTOR')
main.run()

# Conexión con la base de datos users
con = sqlite3.connect('users.sqlite3')
cur = cur = con.cursor()

# Formulario de login
user = st.sidebar.text_input('Usuario')
psw = st.sidebar.text_input('Contraseña', type= 'password')
# Inicio de sesión
if st.sidebar.button('Iniciar sesión'):
    statement = f"SELECT username from users WHERE username='{user}' AND Password = '{psw}';"
    cur.execute(statement)
    if not cur.fetchone():
        st.sidebar.write("Inicio de sesión fallido")
    else:
        st.sidebar.write("Sesión iniciada")
# Registro
if st.sidebar.button('Registrarse'):
    cur.execute("INSERT INTO users VALUES (?, ?)", (user, psw))
    con.commit()

# Base de datos de los tickers disponibles
tickers = pd.read_csv("tickers.csv", delimiter=';')
tickers = tickers.set_index('Symbol')
```

10.3 markets.py

En el apartado de markets se han incluido únicamente dos activos por mercado, aunque en el futuro se irán añadiendo más opciones:

```python
# Importar librerías
import streamlit as st
import yfinance as yf

# Función de la app MARKETS
def app():

    # Título y descripción
    st.title('Mercados financieros')
    st.write('Desde la página de mercados financieros puedes echar un vistazo a los diferentes activos que están disponibles.')

    # Selección de mercado
    market = st.selectbox('Mercados', ('Índices', 'Materias primas', 'Divisas', 'Criptomonedas'))

    # Función para mostrar toda la información de un activo
    def show(sym, act):
        data = yf.download(sym, '2000-01-01')
        d = data.set_index(data.index.strftime("%Y-%m-%d"))
        d = d.drop('Adj Close', 1)
        st.header(act)
        st.line_chart(data['Close'])
        st.subheader('Datos históricos')
        st.write(d)

    if (market == 'Índices'):
        ind = st.radio('Selecciona el activo a mostrar', ('S&P 500', 'Nasdaq', 'Dow Jones', 'Euro Stoxx 50', 'Ibex 35', 'DAX', 'Nikkei 225'))
        if (ind == 'S&P 500'):
            show('^GSPC', ind)
        elif (ind == 'Nasdaq'):
            show('^IXIC', ind)

    elif (market == 'Materias primas'):
        mat = st.radio('Selecciona el activo a mostrar', ('Oro', 'Plata', 'Cobre', 'Petróleo'))
        if (mat == 'Oro'):
            show('GC=F', mat)
        elif (mat == 'Plata'):
            show('SI=F', mat)

    elif (market == 'Divisas'):
        div = st.radio('Selecciona el activo a mostrar', ('EUR-USD', 'USD-JPY', 'USD-CHF', 'USD-CAD', 'GBP-USD', 'AUD-USD'))
        if (div == 'EUR-USD'):
            show('EURUSD=X', div)
        elif (div == 'USD-JPY'):
            show('USDJPY=X', div)

    elif (market == 'Criptomonedas'):
        cripto = st.radio('Selecciona el activo a mostrar', ('BTC', 'ETH', 'USDT', 'BNB', 'ADA', 'XRP', 'DOGE', 'DOT'))
        if (cripto == 'BTC'):
            show('BTC-USD', cripto)
        elif (cripto == 'ETH'):
            show('ETH-USD', cripto)
```

10.4 forecast.py

```python
# Importar librerías
import streamlit as st
import pandas as pd
import yfinance as yf
from fbprophet import Prophet
from fbprophet.plot import plot_plotly

# Función de la app FORECAST
def app():
    # Colección de tickers
    tickers = pd.read_csv("tickers.csv", delimiter=';')
    tickers = tickers.set_index('Symbol')

    # Título
    st.title('Pronóstico a largo plazo del precio de un activo')

    # Seleccionar activo
    ticker = st.selectbox('Ticker', tickers.index)
    if ticker == ' ':
     st.warning('Por favor, selecciona un ticker')
     st.stop()
    com = tickers.loc[ticker,'Name']

    # Obtener y visualizar datos históricos de la acción seleccionada
    data = yf.download(ticker, start="2015-01-01")
    # Gráfico
    st.subheader('Precio de cierre de ' + com)
    st.line_chart(data['Close'], use_container_width=True)
    data['Date'] = data.index.strftime("%Y-%m-%d")
    df = data.set_index('Date')
    # Datos
    st.subheader('Datos históricos de ' + com)
    st.write(df)
    df['Date'] = data.index

    # PRONÓSTICO LARGO PLAZO
    st.subheader('Pronóstico a largo plazo del precio de ' + com)

    # Seleccionar los años para el pronóstico
    years = st.slider('AÃ±os a predecir:', 1, 5)
    period = years*365

    # Modificar dataset para que coincida con el módulo Prophet
    df = data[["Date","Close"]]
    df = df.rename(columns={"Date": "ds", "Close": "y"})

    # Crear el modelo y ajustarlo a los datos
    m = Prophet()
    m.fit(df)
    future = m.make_future_dataframe(periods=period)
    forecast = m.predict(future)
    st.success(f'Pronóstico ajustado para los próximos {years} año/s:')

    # Mostrar pronóstico
    fig2 = plot_plotly(m, forecast, xlabel='Fecha', ylabel='Precio')
    st.plotly_chart(fig2, True)
    # Mostrar componentes del pronóstico
    st.subheader("Componentes del pronóstico")
    fig3 = m.plot_components(forecast)
    st.write(fig3)
```

10.5 prediction.py

```python
# Importar librerías
import streamlit as st
import pandas as pd
import numpy as np
import yfinance as yf
from sklearn.preprocessing import MinMaxScaler
from keras.models import Model
from keras.layers import Dense, Dropout, LSTM, Input, Activation, concatenate
from keras import optimizers

# Función de la app PREDICTION
def app():
    # Colección de tickers
    tickers = pd.read_csv("tickers.csv", delimiter=';')
    tickers = tickers.set_index('Symbol')

    # Título
    st.title('Pronóstico a corto plazo del precio de un activo')

    # Seleccionar activo
    ticker = st.selectbox('Ticker', tickers.index)
    if ticker == ' ':
     st.warning('Por favor, selecciona un ticker')
     st.stop()
    com = tickers.loc[ticker,'Name']

    # Obtener y visualizar datos históricos de la acción seleccionada
    data = yf.download(ticker, start="2016-01-01")
    # Gráfico
    st.subheader('Precio de cierre de ' + com)
    mes = data['Close'].tail(365)
    st.line_chart(mes, use_container_width=True)
    data['Date'] = data.index.strftime("%Y-%m-%d")
    df = data.set_index('Date')
    # Datos
    st.subheader('Datos de los últimos 60 días de ' + com)
    st.write(df.tail(60))
    df['Date'] = data.index

    # Preparar los datos
    df = df[['Close', 'Open', 'High', 'Low']]

    # Escalar los datos (sc para el conjunto de entrenamiento y price_sc para la predicción)
    close = np.array(df['Close']).reshape(-1, 1)
    train_sc = MinMaxScaler()
    price_sc = MinMaxScaler()
    price_sc.fit(close)
    df_train = train_sc.fit_transform(df)

    # X1 : Datos (Close, Open, High, Low) - 60 valores predicen el siguiente
    x1_train = np.array([df_train[i : i + 60].copy() for i in range(len(df_train) - 60)])

    # X2 : Media del precio de cierre para una ventana de 60 días
    x2_train = []
    for h in x1_train:
      sma = np.mean(h[:,0])
      x2_train.append(np.array([sma]))
    x2_train = np.array(x2_train)
```

```
    for h in x1_train:
      sma = np.mean(h[:,0])
      x2_train.append(np.array([sma]))
    x2_train = np.array(x2_train)

    # Y : Valores a predecir ; close
    y_train = np.array([df_train[:,0][i + 60].copy() for i in range(len(df_train) - 60)])

    # Comprobar que todos los conjuntos tienen el mismo nº de filas
    assert x1_train.shape[0] == x2_train.shape[0] == y_train.shape[0]

    # Definir el formato de las dos entradas de la red neuronal
    lstm_input = Input(shape=(x1_train.shape[1], x1_train.shape[2]))
    dense_input = Input(shape=(x2_train.shape[1],))

    # Primera rama
    x1 = LSTM(50)(lstm_input)
    x1 = Dropout(0.2)(x1)
    lstm_branch = Model(inputs=lstm_input, outputs=x1)

    # Segunda rama
    x2 = Dense(20)(dense_input)
    x2 = Activation("relu")(x2)
    x2 = Dropout(0.2)(x2)
    tech_ind_branch = Model(inputs=dense_input, outputs=x2)

    # Combinar ramas
    comb = concatenate([lstm_branch.output, tech_ind_branch.output])
    y = Dense(64, activation="sigmoid")(comb)
    y = Dense(1, activation="linear")(y)

    # Compilar y entrenar el modelo
    model = Model(inputs=[lstm_branch.input, tech_ind_branch.input], outputs=y)
    opt = optimizers.Adam(lr=0.0005)
    model.compile(optimizer=opt, loss='mse')
    model.fit(x=[x1_train, x2_train], y=y_train, batch_size=32, epochs=50, shuffle=True)
    st.success('Modelo entrenado')

    # Se realiza una predicción para el día actual y el día siguiente
    # Esto permitirá comparar las predicciones y clasificar el movimiento del precio
    df_pred = df_train[len(df_train)-61:, :]
    x1_pred = np.array([df_pred[i : i + 60].copy() for i in range(2)])
    x2_pred = []
    for h in x1_pred:
      sma = np.mean(h[:,0])
      x2_pred.append(np.array([sma]))
    x2_pred = np.array(x2_pred)

    fut = model.predict([x1_pred, x2_pred])
    fut = price_sc.inverse_transform(fut)

    # Previsión de si el precio va a subir o a bajar en función a los 60 últimos días
    dif = ((fut[-1] - fut[-2]) / fut[-2]) * 100
    if fut[-1] > fut[-2]:
      st.markdown(f'_El modelo predice que el precio subirá mañana {dif.round(2)}%_')
    else:
      st.markdown(f'_El modelo predice que el precio bajará mañana {dif.round(2)}%_')
```

10.6 sentiment.py

```python
# Importar librerías
import streamlit as st
import tweepy, re
import pandas as pd
from textblob import TextBlob
import matplotlib.pyplot as plt

# Función de la app SENTIMENT
def app():
    # Título
    st.title('Análisis de sentimiento')

    ticker = st.text_input('Término')
    if ticker == '':
     st.warning('Por favor, selecciona un término')
     st.stop()

    # Claves de acceso para la API de twitter
    consumerKey = 'f0y99pKjB4x2SNBHtK5fiD4IB'
    consumerSecret = 'jJXTcEuriPbjEzDvrBsXQ3eBeAOl2oqKaiwoUlhpRKPzHbcyAu'
    accessToken = '1350780188142039041-3NZlpno82LwUqlxL1SFTrHwzLs9luY'
    accessTokenSecret = 'B3KCfjNhCUR5glKMdqkK9Sjp5CEIOxGEBxQ4yJfkPMD4h'

    # Autenticar el acceso a la API
    auth = tweepy.OAuthHandler(consumerKey, consumerSecret)
    auth.set_access_token(accessToken, accessTokenSecret)
    api = tweepy.API(auth, wait_on_rate_limit=True)

    # Seleccionar fechas
    start = st.date_input("Selecciona la fecha de inicio")
    st.write("Se recopilarán los tweets con más repercusión desde la fecha de inicio hasta hoy.")

    # Obtener n tweets relacionados con term y filtrar por nº de RT
    term = '#'+ ticker + ' -filter:retweets'
    get_tweets = tweepy.Cursor(api.search, q=term, lang = "en", since=start, tweet_mode='extended').items(100)
    tweets = [tweet.full_text for tweet in get_tweets]
    df = pd.DataFrame(tweets, columns=['Tweets'])

    # Función para formatear los tweets
    def cleanTwt(twt):
      twt = re.sub('#'+ticker, ticker, twt) # Eliminar '#' del ticker
      twt = re.sub('#[A-Za-z0-9]+', '', twt) # Eliminar cualquier otro '#'
      twt = re.sub('\\n', '', twt) # Eliminar los '\n'
      twt = re.sub('https?:\/\/.*[\r\n]*', '', twt, flags=re.MULTILINE) # Eliminar hyperlinks
      return twt

    # Se crea una lista con los tweets formateados
    df['Cleaned_Tweets'] = df['Tweets'].apply(cleanTwt)

    # Función para obtener la subjetividad
    def getSub(twt):
      return TextBlob(twt).sentiment.subjectivity
    # Función para obtener la polaridad
    def getPol(twt):
      return TextBlob(twt).sentiment.polarity

    # Crear las columnas Subjetividad y Polaridad
    df['Subjectivity'] = df['Cleaned_Tweets'].apply(getSub)
    df['Polarity']  = df['Cleaned_Tweets'].apply(getPol)

   # Función para clasificar tweets en base a su sentimiento
   def getSen(pol):
     if pol == 0:
       return 'Neutral'
     elif (pol < -0.5):
       return 'Muy Negativo'
     elif (pol < 0 and pol > -0.5):
       return 'Negativo'
     elif (pol > 0 and pol < 0.5):
       return 'Positivo'
     elif (pol > 0.5):
       return 'Muy Positivo'

   # Crear la columna de sentimiento
   df['Sentiment'] = df['Polarity'].apply(getSen)

   # Gráfico de puntos para mostrar subjetividad y polaridad
   fig1 = plt.figure()
   ax1 = fig1.add_subplot()
   for i in range(0, df.shape[0]):
     ax1.scatter(df['Polarity'][i], df['Subjectivity'][i], color='blue')
   ax1.set_title('Análisis de subjetividad / polaridad')
   ax1.set_xlabel('Polaridad')
   ax1.set_ylabel('Subjetividad')
   st.pyplot(fig1)

   # Gráfico de barras
   count = df['Sentiment'].value_counts()
   st.bar_chart(count)
   st.write(count)
```

10.7 portfolio.py

```python
# Importar librerías
import streamlit as st
import pandas as pd
import yfinance as yf
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from pypfopt import expected_returns
from pypfopt import risk_models
from pypfopt.efficient_frontier import EfficientFrontier
from pypfopt.discrete_allocation import DiscreteAllocation, get_latest_prices

# Función de la app PORTFOLIO
def app():
    # Título
    st.title('Optimizador de carteras')

    # Colección de tickers
    tickers = pd.read_csv("tickers.csv", delimiter=';')

    portfolio = st.multiselect('Selecciona los activos a incluir en la cartera', tickers['Symbol'])
    if not portfolio:
     st.warning('Por favor, selecciona al menos un ticker')
     st.stop()

    # Crear el dataframe con el precio de cierre ajustado de todas las acciones de la cartera
    df = pd.DataFrame()
    df = yf.download(portfolio, data_source='yahoo', start='2016-01-01')['Adj Close']

    # Visualizar comportamiento del precio de cada acción
    st.subheader('Comportamiento del precio de las acciones')
    st.line_chart(df, use_container_width=True)

    # Calcular retornos esperados anuales y matriz de covarianza
    mu = expected_returns.mean_historical_return(df, compounding=False)
    S = risk_models.risk_matrix(df, method='sample_cov')

    # Visualizar matriz de correlación a partir de la matriz de covarianza
    st.subheader('Matriz de correlación')
    risk_models.cov_to_corr(S)
    fig1, ax1 = plt.subplots(figsize=(15,10))
    mat = np.triu(risk_models.cov_to_corr(S))
    sns.heatmap(risk_models.cov_to_corr(S), annot=True, linewidths=.5, ax=ax1, vmin=-1, vmax=1, center= 0, mask=mat)
    ax1.set_xticklabels(ax1.get_xticklabels(),rotation=45,horizontalalignment='right');
    st.pyplot(fig1)

    # Optimizar cartera
    ef = EfficientFrontier(mu, S)
    weights = ef.max_sharpe()
    cleaned_weights = ef.clean_weights()
    cw = pd.DataFrame(cleaned_weights.values(), cleaned_weights.keys(), ['Pesos'])
    cw = cw.drop(cw[cw['Pesos'] <= 0].index)

    # Visualización de pesos optimizados
    st.subheader('Distribución óptima de los pesos')
    fig2 = plt.figure()
    ax2 = fig2.add_subplot()
    ax2.pie(cw['Pesos'], labels=cw.index, autopct='%1.1f%%', startangle=90)
    ax2.set_title('Participación de cada activo')
    st.pyplot(fig2)

    # Resultados del portafolio optimizado
    st.subheader('Performance de la cartera')
    performance = ef.portfolio_performance()
    ret = performance[0]*100
    volatility = performance[1]*100
    sharpe = performance[2]
    st.write(f'Retorno anual esperado: {ret.round(2)}%')
    st.write(f'Volatilidad anual: {volatility.round(2)}%')
    st.write(f'Sharpe ratio: {sharpe.round(2)}')


    # Seleccionar capital a invertir
    st.subheader('Configuración de cartera personal')
    capital = st.slider("Capital a invertir", 0, 1000000, step=1000)
    if capital == 0:
        st.stop()

    # Nº de acciones a comprar de cada compañía y dinero sobrante
    latest_prices = get_latest_prices(df)
    da = DiscreteAllocation(weights, latest_prices, total_portfolio_value=capital)
    allocation, leftover = da.lp_portfolio()
    allocation = pd.DataFrame(data=allocation.values(), index=allocation.keys(), columns=['Nº de acciones'])

    # Resultados
    st.write('Composición de la cartera: ')
    st.write(allocation)
    st.write(f'Dinero restante: {leftover.round(2)}$')
```