



ugr | Universidad
de **Granada**

TRABAJO FIN DE GRADO
INGENIERÍA INFORMÁTICA

TradingAPP

Aplicación web para trading algorítmico

Autor

Manuel Carmona Pérez

Director

José Manuel Benítez Sánchez



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍAS INFORMÁTICA Y DE
TELECOMUNICACIÓN

Granada, mes de 2021



TradingAPP

Aplicación web para trading algorítmico

Autor

Manuel Carmona Pérez

Director

José Manuel Benítez Sánchez

TradingAPP: Aplicación web para trading algorítmico

Manuel Carmona Pérez

Palabras clave: trading algorítmico, trading automático, *Wyckoff*, mercados financieros

Resumen

El volumen de datos involucrado y la velocidad a la que se realizan las operaciones de compra y venta en los mercados financieros ha hecho que la intervención de procedimientos automatizados juegue un papel fundamental en las actuales acciones de compra y venta de distintos operadores en los mercados. En estos operadores encontramos grandes inversores como entidades bancarias o instituciones.

El desarrollo de algoritmos de trading es una pieza clave en la creación de estos métodos automáticos para operar. Estos algoritmos explotan técnicas de análisis de la acción de mercados financieros ya conocidas junto con métodos basados en aprendizaje automático, entre otros, para maximizar los beneficios en las compras o ventas realizadas.

El objetivo de este proyecto es desarrollar una aplicación que implemente técnicas para realizar trading mediante algoritmos basados en métodos de análisis técnico ya conocidas. Esta aplicación tendrá una interfaz web para que sea usada por un operador humano.

TradingAPP: Algorithmic trading web application

Manuel Carmona Pérez

Keywords: algorithmic trading, automic trading, *Wyckoff*, financial markets

Abstract

The large volume of data envolved and the speed at which the buy and sell operations are done in the financial markets have made the intervention of automated procedures a vital role in how operators buy and sell in the markets nowadays. Large investors like banks or other institutions can be found among these operators.

The trading algorithms development plays a key role in these automatic methods creation. These algorithms exploit known techniques that analyze the financial markets actions and machine learning nethids, among others, to maximize benefits in the sales and purchases done.

The main purpose of this project is to develop an application that implements techniques to trade by using algorithms based on already known technical analysis methods. This application will have a web interface as it will be used by a human operator.

Yo, **Manuel Carmona Pérez**, alumno del grado en ingeniería informática de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI 17482989E, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Manuel Carmona Pérez

Granada, septiembre de 2021.

D. **José Manuel Benítez Sánchez**, Profesor del Área de Ciencias de la Computación e Inteligencia Artificial del Departamento Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada.

Informa:

Que el presente trabajo, titulado ***TradingAPP, Aplicación web para trading algorítmico***, ha sido realizado bajo su supervisión por **Manuel Carmona Pérez**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada, a 6 de septiembre de 2021.

El director:

José Manuel Benítez Sánchez

Agradecimientos

A mi familia y amigos, por aguantarme y apoyarme en esta dura etapa de mi vida.

Índice general

| | |
|---|-----------|
| 1. Introducción | 17 |
| 1.1. Motivación | 17 |
| 1.2. Objetivos | 18 |
| 1.3. Estructura del documento | 19 |
| 2. Contexto teórico | 21 |
| 2.1. Trading: conceptos técnicos y análisis de mercados | 21 |
| 2.1.1. Conceptos técnicos | 21 |
| 2.1.2. Tipos de análisis de mercados | 26 |
| 2.1.3. Principales referentes del análisis de mercados | 27 |
| 2.1.4. Análisis de Charles Henry Dow | 28 |
| 2.1.5. Análisis de Richard Demille Wyckoff | 30 |
| 2.2. Trading algorítmico | 34 |
| 2.2.1. Ventajas | 35 |
| 2.2.2. Desventajas | 36 |
| 3. Planificación | 37 |
| 3.1. Planificación temporal | 37 |
| 3.1.1. Fases del proyecto | 37 |
| 3.1.2. Diagrama de Gantt | 39 |
| 3.2. Presupuesto de ejecución | 39 |
| 3.2.1. Gastos de personal | 39 |
| 3.2.2. Gastos de recursos informáticos y energía | 40 |
| 3.2.3. Resumen del presupuesto | 41 |
| 4. Análisis | 45 |
| 4.1. Descripción de los implicados | 45 |
| 4.2. Especificación de requisitos | 45 |
| 4.2.1. Requisitos Funcionales | 45 |
| 4.2.2. Requisitos No Funcionales | 47 |
| 4.2.3. Requisitos de Información | 47 |
| 4.3. Diagramas de casos de uso | 47 |
| 4.3.1. Gestión de sesiones de la APP | 47 |

| | |
|---|------------|
| 4.3.2. Gestiones de la plataforma de trading (MT5) | 48 |
| 4.3.3. Gestión y visualización de datos de mercados | 48 |
| 4.3.4. Funcionalidades de trading | 49 |
| 5. Diseño | 53 |
| 5.1. Arquitectura del software | 53 |
| 5.2. Diseño de clases | 54 |
| 5.3. Diagramas de secuencia | 55 |
| 5.4. Primer diseño de la interfaz | 56 |
| 6. Implementación | 67 |
| 6.1. Herramientas y software utilizado | 67 |
| 6.2. Metodología de trabajo | 69 |
| 6.2.1. Issues y Pull Requests | 69 |
| 6.2.2. Paralelismo usando GitFlow | 70 |
| 6.3. Fases del desarrollo | 72 |
| 6.3.1. Guía de puntuación scrum y prioridades | 72 |
| 6.3.2. Fase previa a los sprints de desarrollo | 73 |
| 6.3.3. Sprint 1: 8 junio - 22 junio. Interfaces y modelos básicos | 75 |
| 6.3.4. Sprint 2: 22 junio - 13 julio. Desarrollo del método de Wyckoff | 80 |
| 6.3.5. Sprint 3: 13 julio - 31 julio. Funcionalidad para login y logout en MT5 | 86 |
| 6.3.6. Sprint 4: 31 julio - 14 agosto. Visualización de datos de mercados y gestión de usuarios | 87 |
| 6.3.7. Sprint 5: 14 agosto - 30 agosto. Funcionalidad para operar en tiempo real | 90 |
| 6.3.8. Sprint 6: 30 agosto - 6 septiembre. Desarrollo de la gestión de datos y últimas correcciones | 92 |
| 6.4. Diseño final de la interfaz | 92 |
| 6.4.1. Página principal y formulario de Login | 92 |
| 6.4.2. Menú principal y manual de usuario | 93 |
| 6.4.3. Menús de elección de estrategia | 93 |
| 6.4.4. Menú de backtesting y operativa en tiempo real | 93 |
| 6.4.5. Menú de gestión de datos | 93 |
| 6.4.6. Menú de visualización de datos | 93 |
| 7. Pruebas | 101 |
| 8. Conclusiones | 103 |
| 9. Bibliografía | 105 |
| 10. Anexo | 107 |
| 10.1. Manual de uso | 107 |

Índice de figuras

| | |
|---|----|
| 2.1. Representación de precios usando barras y velas japonesas. Fuente: Tipos de gráficos de trading y sus características . . . | 24 |
| 2.2. Fases de acumulación y euforia para tendencia alcista. Fuente: Teoría de Dow en Análisis Técnico — Dow Theory | 29 |
| 2.3. Ciclos del precio de Wyckoff. Fuente: El método Wyckoff . . | 31 |
| 2.4. Esquema de acumulación en el método Wyckoff. Fuente: El método Wyckoff | 33 |
| 2.5. Esquema de distribución en el método Wyckoff. Fuente: El método Wyckoff | 34 |
| 3.1. Diagrama de <i>Gantt</i> del proyecto | 42 |
| 3.2. Presupuesto de ejecución del trabajo | 43 |
| 4.1. Diagrama de casos de uso referente al sistema de gestión de usuarios de la APP | 48 |
| 4.2. Diagrama de casos de uso referente a la gestión de la pla- taforma de trading (MT5) | 49 |
| 4.3. Diagrama de casos de uso referente a la visualización de datos de mercados financieros | 50 |
| 4.4. Diagrama de casos de uso referente a las funcionalidades de trading | 51 |
| 5.1. Arquitectura general del software. | 54 |
| 5.2. Diagrama de clases de la APP | 58 |
| 5.3. Diagrama de secuencia de la creación de usuarios | 59 |
| 5.4. Diagrama de secuencia de la creación de usuarios | 59 |
| 5.5. Diagrama de secuencia de login y logout de usuarios | 60 |
| 5.6. Diagrama de secuencia de login y logout de usuarios en el bróker en MetaTrader5 | 61 |
| 5.7. Diagrama de secuencia de visualización de datos antiguos y en tiempo real | 62 |
| 5.8. Diagrama de secuencia de elección de algoritmos y trading automático y backtesting | 63 |
| 5.9. Primer diseño del menú principal de la APP | 64 |

| | |
|---|-----|
| 5.10. Primer diseño del formulario de login en <i>MT5</i> | 64 |
| 5.11. Primer diseño del menú principal (usuario identificado en <i>MT5</i>) | 65 |
| 6.1. Logos de las principales herramientas usadas en el proyecto. . | 68 |
| 6.2. <i>Issue</i> número 5, implementación de la estrategia de Wyckoff . | 70 |
| 6.3. <i>Pull Request</i> referente a la issue 6.2 | 71 |
| 6.4. Guía de puntuación scrum | 73 |
| 6.5. Datos de mercados obtenidos en las fases previas al desarrollo | 75 |
| 6.6. Media móvil con ventana de 100 días (azul claro) contra ven- tana de 30 días (azul oscuro). Fuente: investopedia.com/ . . . | 78 |
| 6.7. Creación de objetos para el modelo <i>Mercado</i> | 79 |
| 6.8. Árbol de directorios del proyecto | 80 |
| 6.9. Bandas de Bollinger en un gráfico de precios | 82 |
| 6.10. Gráficos que representan bandas de Bollinger y diferencia de sus valores | 83 |
| 6.11. Intervalos detectados con el algoritmo de detección de ten- dencias | 84 |
| 6.12. Otros ejemplos de intervalos detectados con el algoritmo de detección de intervalos | 84 |
| 6.13. Código Python para extender el modelo de Django User . . | 89 |
| 6.14. Opción a habilitar para mandar órdenes de operaciones al terminal MT5. | 90 |
| 6.15. Página principal y formulario de login, respectivamente. . . . | 94 |
| 6.16. Página principal y manual de usuario, respectivamente. . . . | 95 |
| 6.17. Menús de elección de estrategia. | 96 |
| 6.18. Menú de backtesting y trading a tiempo real, respectivamente. | 97 |
| 6.19. Menús de gestión de datos, teniendo datos guardados y sin tenerlos, respectivamente. | 98 |
| 6.20. Menús de visualización de datos antiguos. | 99 |
| 6.21. Menú de visualización de datos en tiempo real. | 100 |

Capítulo 1

Introducción

1.1. Motivación

El trading, los mercados financieros o invertir en bolsa son conceptos clásicos que hoy día están irrumpiendo más que nunca debido a los avances tecnológicos y sobre todo, gracias a la influencia de las activos digitales como las criptomonedas y otros avances de la tecnología que afectan directamente a la economía global y la centralización o no de los capitales.

En este proyecto trataremos concretamente el *trading* o comercio o intercambio comercial en español. El trading consiste en especular sobre distintos mercados financieros comprando o vendiendo activos para obtener un beneficio a corto o largo plazo. Estos mercados pueden ser de acciones, divisas, materias primas, índices o criptomonedas.

El trading es un concepto clásico, ya que el comprar o vender algo que se puede revalorizar o devaluar para buscar beneficio económico es algo que ya se ha hecho desde las antiguas civilizaciones. En algunos documentos se habla de que la antigua civilización mesopotámica de Sumer, actual sur de Irak, fue una de las primeras en practicar el trading. Hacia el año 600 a.C., el oro y la plata ya eran las primeras monedas del mundo, antes de que se creasen los sistemas monetarios.

En la actualidad, debido a la velocidad del mundo en todos los ámbitos del día a día, sobre todo en la tecnología; los procesos de compras y ventas de acciones se realizan constantemente, buscando cada operador el mayor beneficio posible y ordenando dichas operaciones a la unidad más pequeña de tiempo posible. Por esto, el trading es algo bastante avanzado y que aprovecha al máximo los recursos y conocimientos sobre la computación e inteligencia artificial actuales ya que el proceso completo se realiza de manera digital.

Una de las aplicaciones de la informática en el ámbito del trading consiste en automatizar las compras y ventas de activos financieros para obtener beneficios a corto o largo plazo. Esta aplicación es la principal motivación para la realización de este trabajo de fin de grado.

El objetivo principal de este *Trabajo de Fin de Grado* será desarrollar una aplicación que implemente un algoritmo para automatizar el proceso de comprar y vender activos para ganar dinero. El algoritmo a implementar estará basado en la estrategia clásica para operar de *Richard Wyckoff*, escritor e inversor estadounidense cofundador de *The Magazine of Wall Street*. Concretamente, a esto se le conoce como trading algorítmico. En pocas palabras, el trading algorítmico es implementar un sistema de trading que opere de forma automática.

1.2. Objetivos

El objetivo principal del proyecto es desarrollar una aplicación web para realizar trading algorítmico. El sistema implementará un algoritmo para operar basándose en técnicas de análisis de mercados financieros clásicas, en concreto, se basará en la estrategia o análisis de *Richard Wyckoff*.

El desarrollo principal de la aplicación se encontrará en la capacidad para comprar y vender de forma automática usando una cuenta comercial real tal y como lo haría un usuario humano, a través de un bróker o plataforma comercial de trading. Estas operaciones se realizarán según lo indique el algoritmo que elijamos, dentro de una lista de algoritmos que encontramos en la propia aplicación.

La aplicación permitirá al usuario identificarse con su cuenta *comercial* o *demo* (de prueba) de *MetaTrader5*, que será la aplicación externa que realizará las compras y ventas en el mercado financiero seleccionado. **Ref.: OBJ 1, OBJ 2.**

Una vez un usuario está identificado y ha iniciado sesión en *MT5*, podrá escoger un algoritmo para realizar Trading automático en tiempo real o probar las técnicas a modo de Backtesting. La aplicación también proporcionará la posibilidad de ver el histórico de operaciones realizado y el balance actual de la cuenta de *MT5* en la que se ha identificado el usuario. **Ref.: OBJ 5, OBJ 6, OBJ 7, OBJ 8.**

Además de poder realizar operaciones de manera automática, la apli-

cación dispondrá de una interfaz propia para ver los datos de mercado en tiempo real o antiguos, utilizando gráficas interactivas. **Ref.: OBJ 3, OBJ 4.**

Más formalmente, podemos definir los objetivos del producto software de la siguiente forma:

- **OBJ 1:** La aplicación tendrá un sistema de gestión de usuarios.
- **OBJ 2:** El sistema conectará con la cuenta del usuario de la plataforma de trading en cuestión.
- **OBJ 3:** El sistema permitirá a los usuarios ver gráficos en tiempo real del mercado que se quiera visualizar.
- **OBJ 4:** El sistema permitirá a los usuarios ver gráficos de datos antiguos de precios del mercado que se quiera visualizar.
- **OBJ 5:** El sistema desarrollará varios algoritmos usados para predecir el comportamiento de los mercados y hacer compras o ventas. La aplicación permitirá a los usuarios elegir entre uno de estos algoritmos para ser usado en el resto de funciones de la APP.
- **OBJ 6:** El sistema permitirá a los usuarios hacer operaciones de compra y venta de manera automatizada en un periodo de tiempo y mercado concretos, eligiendo los modelos de predicción mencionados.
- **OBJ 7:** El sistema permitirá a los usuarios probar cada uno de los algoritmos en un periodo de tiempo fijo, a modo de backtesting.
- **OBJ 8:** El sistema permitirá a los usuarios ver un histórico de operaciones realizadas así como el balance actual de la cuenta a la que se ha conectado.

1.3. Estructura del documento

Este documento sigue la siguiente estructura de capítulos con sus respectivas secciones:

1. **Introducción:** Sección que incluye la motivación o justificación que lleva a la elección del tema para la elaboración del proyecto y los objetivos que se pretenden alcanzar con el mismo.

2. **Contexto teórico:** Este apartado incluye la explicación de los conceptos específicos usados a lo largo del desarrollo del proyecto. La sección habla de la teoría básica necesaria para entender ciertas decisiones y desarrollos realizados.
3. **Planificación:** Esta sección incluye cada una de las fases de la planificación temporal del proyecto, qué se ha hecho en cada fase, durante cuánto tiempo, etc. Esta planificación viene formalizada por medio de un diagrama de *Gantt*.
4. **Análisis:** Este capítulo habla de la fase de análisis del proyecto. En esta fase se describen los implicados, los requerimientos del software a implementar y diagramas de casos de uso y comportamiento del producto.
5. **Diseño:** Este apartado incluye los principales diagramas *UML* que conforman el diseño del software del proyecto. Concretamente, encontramos un diagrama de paquetes, el diseño de clases y diagramas de secuencia. Se describe también un primer diseño de la interfaz.
6. **Implementación:** En este capítulo se exponen las herramientas y software utilizado en el desarrollo del proyecto. Se habla de la metodología del trabajo, que en este caso es la metodología ágil *scrum*; y del uso de issues y pull request de *GitHub* y *git flow*. Se describen también las fases del desarrollo y el diseño final de la interfaz.
7. **Pruebas:** Esta sección contiene la batería de pruebas realizada al producto software para comprobar su correcto funcionamiento, así como rendimiento y eficacia.
8. **Conclusiones:** En este capítulo, se describe un resumen final de lo que se ha conseguido en la realización del proyecto. En este apartado se habla de los resultados obtenidos y se expresan posibles mejoras o avances futuros.
9. **Bibliografía:** En este apartado se incluyen las referencias bibliográficas usadas a lo largo del proyecto.
10. **Anexo:** Incluye el manual de usuario de la aplicación.

Capítulo 2

Contexto teórico

2.1. Trading: conceptos técnicos y análisis de mercados

Como se ha mencionado en el apartado *Motivación* del capítulo 1, el trading es un concepto que tiene su origen en antiguas civilizaciones de Mesopotamia.

El actual concepto de trading consiste en una práctica aplicada a los mercados financieros. Realizar trading implica comprar o vender dentro de los distintos mercados financieros para vender o comprar de nuevo al cabo de un tiempo y obtener beneficio de la acción realizada. Entre esos activos financieros destacan las acciones de empresas, divisas, materias primas, índices o criptomonedas.

Estas compras o ventas, llamadas operaciones, se realizan mediante plataformas digitales que proporcionan una interfaz gráfica para visualizar precios y otra información relevante. Las operaciones que realiza el operador humano con otros operadores se hacen por medio de un intermediario o bróker.

A grandes rasgos y con un ejemplo, realizar trading beneficiosamente podría ser comprar una acción de *Apple* por 100 dólares y venderla en cierto tiempo por 150 dólares, obteniendo por tanto un beneficio de 50 dólares.

2.1.1. Conceptos técnicos

Para profundizar en términos de análisis de mercados e implementación de los mismos, debemos de presentar previamente una serie de conceptos para entender cómo funcionan los gráficos de precios y tener un glosario

más técnico sobre el tema.

Brokers y plataformas comerciales

En primer lugar, definiremos lo que son los brokers. Un bróker es un inversor; empresa o persona que participa en el proceso del trading y que recoge la oferta y demanda de un activo y hace de intermediario en el proceso. Este broker cobra comisiones por su trabajo y hace que el proceso de compra y venta en un mercado sea más rápido y eficaz.

Cuando una persona pretende operar, necesita una plataforma digital para realizar las compras y ventas de activos y esto supondrá una comunicación pasiva con un bróker. Estas plataformas de trading proporcionan una interfaz gráfica para que el usuario humano pueda operar de manera sencilla.

Entre las actuales plataformas de trading que a su vez funcionan de brokers, destacan *XTB*, *eToro*, *AvaTrade*, *XM*, *Darwinex*, *Plus500*, *IQ Option*, *OBR Invest* o *Pepperstone*.

Además de las plataformas mencionadas, existen plataformas comerciales que aportan la interfaz e información de mercados o activos financieros pero no funcionan como bróker. Éstas permiten el uso de distintos intermediarios, según uso o gustos del usuario. Es el caso de *MetaTrader*.

A continuación se muestra la lista de brókers que trabajan con MetaTrader en su última versión del software, *MetaTrader5*. Todos ellos proporcionan soporte para mercados de *Forex*, *CFD* y metales preciosos.

- XM
- FPMarkets
- Hotforex
- Alpari
- Valutrades
- Vantage FX
- ForexTime
- OctaFX
- Roboforex
- BDSwiss
- EXNess
- markets.com
- Pepperstone
- IronFX
- ICMarkets
- BlackBull Markets
- AvaTrade
- Admiral Markets

- FXOpen
- IFC Markets
- FBS
- Nordfx

Debido a la versatilidad que proporciona MetaTrader5 y la posibilidad de crear cuentas *Demo*, es decir, cuentas que usan dinero falso para usarlas con motivos de aprendizaje, será la plataforma comercial usada en el proyecto.

Representación de datos: tipos de gráficos y temporalidad

Una de las principales características de las plataformas comerciales mencionadas es que nos muestran gráficos de precios de cada uno de los activos financieros que tengan disponibles.

Estos precios pueden ser representados usando distintos tipos de gráficos. En todos los gráficos se representa el precio en función de la fecha o *timestamp*. Los tres más comunes son el gráfico de línea, el gráfico de barras y el gráfico de velas japonesas.

En el caso del **gráfico de línea**, el precio se encuentra representado como la unión de los distintos precios de cierre en un determinado periodo de tiempo. El precio de cierre es el último precio disponible de un activo en una unidad de tiempo específica. Este es el gráfico más simple y aporta menos información que el gráfico de barras o el de velas.

El **gráfico de barras** y el **gráfico de velas japonesas** representan exactamente la misma información, pero con un diseño gráfico distinto. Ambos gráficos siguen el formato de representación de precios conocido como *OHLC*. *OHLC* es la abreviatura de *Open High Low Close*. Como se ha mencionado anteriormente, este modo de representación es más complejo que el usado en el gráfico de línea e incluye más información. Los gráficos que usan el método OHLC muestran la información de los precios de apertura, máximo, mínimo y cierre para una unidad de tiempo específica. El precio de apertura es el primer dato de precio que tenemos del activo en la unidad de tiempo, el precio máximo es el máximo precio alcanzado en dicho tiempo, el mínimo es el mínimo precio alcanzado; y el cierre es el último registrado, como se ha mencionado en el párrafo anterior. En la figura 2.1 podemos ver un ejemplo de barra y de vela usados en los gráficos de barras y velas japonesas, respectivamente.

Una vez conocemos los tipos de gráficos de precios de activos financieros, podemos presentar el concepto de temporalidad. La **temporalidad** o **marco de tiempo** es la unidad de tiempo o período que elige el trader para ver representado el mercado en el gráfico y estudiarlo. Dependiendo de la temporalidad, el mercado se actualizará cada cierto tiempo, mostrando

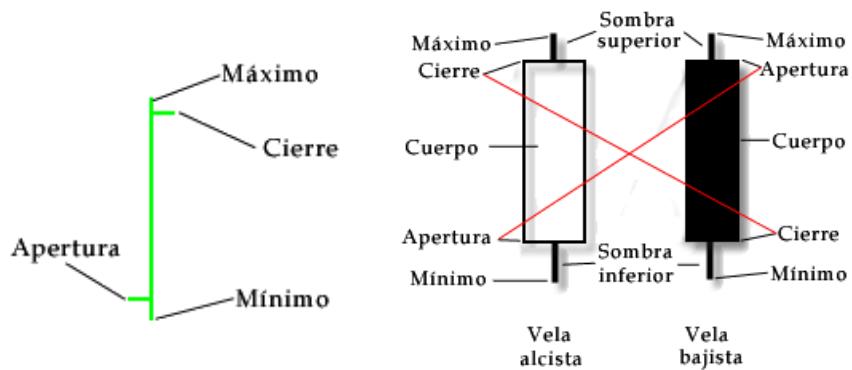


Figura 2.1: Representación de precios usando barras y velas japonesas. Fuente: [Tipos de gráficos de trading y sus características](#)

nuevas velas, barras o puntos, dependiendo del tipo de gráfico que estemos usando.

Como ejemplos de temporalidad podemos ver *M1*, que indica que el gráfico se actualizará con una nueva vela, barra o punto cada minuto; *M5*, que indica un período de actualización de 5 minutos; *H1*, que indica 1 hora; etc. Otras temporalidades son *M15*, *M30*, *H4* (horas), *D1* (días), *W1* (semanas (week)), e incluso actualizaciones mensuales.

A la hora de estudiar los gráficos, la elección del marco de tiempo dependerá de si estamos operando a corto o largo plazo principalmente. Por ejemplo, no tendría mucho sentido estudiar un mercado en *M1* si queremos ordenar una operación para una semana vista, ya que no estaremos viendo realmente las tendencias del mercado.

Precios del mercado: ask y bid

En esta sección, presento los conceptos de *ask* y *bid*. El *ask* y el *bid* se corresponden con los dos precios que presenta un activo en todo momento. El *bid* es el precio que se ofrece, es decir, el precio que pone el mercado para comprar tus acciones. Por otro lado, el *ask* es el precio que se pide, es decir, el precio de venta del mercado para que el trader compre la acción del activo.

No importa cuántos vendedores o compradores haya en el mercado en el período a comprar o vender, los precios *ask* y *bid* serán el más bajo de todos los ofertantes y el más alto de todos los demandantes, respectivamente. En otras palabras, como traders, venderemos siempre la acción al que más

dinero ofrece y compraremos la acción al que más barato la venda.

Spread

El *spread* o separación es la diferencia en un momento dado entre los precios *ask* y *bid*.

En el trading, el spread debe de ser pequeño ya que sólo con entrar al mercado estamos perdiendo dinero por esta diferencia de precios. Siempre tendremos que salir a un precio peor con lo que minimizar el spread minimizará las pérdidas.

Este valor es muy alto en ocasiones donde participa muy poca gente en el mercado. En un ejemplo práctico, si sólo hay dos personas en un mercado a un determinado tiempo, y una de ellas vende una acción a 100 € y la otra quiere comprar a 60 €, el spread es de 40 €; probablemente no haya trato.

El spread es una medida de *liquidez*: a menor spread, mayor liquidez. La *liquidez* es en la práctica la facilidad que tienes para comprar o vender al precio que quieras. Esto se ve fácilmente en el anterior ejemplo, donde la liquidez sería baja al tener un spread alto.

En resumen, cuantos más participantes y más activos sean en el mercado, habrá más liquidez ya que el spread será menor; y los precios de compra y venta serán tan parecidos que el spread apenas tendrá impacto en la operativa.

Volatilidad

Cuando hay volatilidad en el trading, el precio se acelera. Si estamos usando velas o barras (véase figura 2.1)), éstas aparecerán más grandes y bruscas en el gráfico.

Este fenómeno provoca negociaciones más impulsivas en el gráfico, lo que aumenta el spread. Esto ocurre porque las compras y ventas se realizan de forma más rápida y menos eficientes.

Cotización

En el trading, la cotización es el precio de la última operación de un activo o bien el precio al que se vende o compra en el momento actual. En este último caso, la cotización es igual al precio de cierre en el instante.

2.1.2. Tipos de análisis de mercados

Dada esta introducción a lo que es el trading, vemos obvio que lo que buscamos es maximizar los beneficios cuando ordenamos una compra o una venta.

¿Cuándo se debe comprar o vender un activo? Cuando un operador humano piensa en realizar una operación en el mercado, analiza el activo en cuestión siguiendo una serie de criterios basados en distintos tipos de análisis existentes.

- **Análisis fundamental:** El análisis fundamental consiste en operar en base a las distintas noticias que ocurren en el mundo diariamente. Estas noticias sirve de fundamento para actuar de una forma u otra en el mercado.
- **Análisis técnico:** En este tipo de análisis, el operador o trader opera puramente en base al precio: su acción y movimientos. En este caso, no se tienen en cuenta las noticias. Es este el tipo de análisis interesante a la hora de automatizar. Esto ocurre porque en este método entran en juego cálculos matemáticos como medias, medias móviles, zonas de probabilidad estadística, etc. A partir de este análisis surge el trading algorítmico.
Este análisis, aunque se pueda automatizar, no te asegura que el precio vaya en la dirección resultado del análisis. A pesar de esto, si trabajamos con probabilidades y obtenemos probabilidades de ganar superiores al 50 %, obtendremos un sistema que a largo plazo conseguiría grandes beneficios.
- **Análisis de sentimiento:** En este análisis cobran importancia las opiniones de analistas, prensa u observadores de mercados. Este tipo de análisis ocurre mayormente en el *Forex* (mercado de intercambio de divisas).
- **Análisis macroeconómico:** Análisis en el que se tiene en cuenta el comportamiento y desarrollo de la economía de un país. Tiene relación directa con el *Análisis fundamental* ya que las noticias que más afectan a los mercados suelen ser las de corte económico.
- **Análisis cuantitativo:** Este análisis usa lo que se conoce como matemáticas financieras, concepto que deriva de la física y de la estadística.

2.1.3. Principales referentes del análisis de mercados

A lo largo de la historia moderna, economistas, periodistas y demás personas dedicadas a la inversión han desarrollado una serie de estrategias o métodos de análisis de mercados propios.

En concreto, destacamos los cinco principales referentes del análisis de datos de activos financieros: *Charles Henry Dow, Richard Demille Wyckoff, William Delbert Gann, Ralph Nelson Elliott y Arthur Alexander Merrill*.

Estas importantes figuras destacaron por sus avances en el ámbito de la economía y todos fueron importantes analistas de mercados. A continuación, cito a cada uno de dichos economistas en orden descendente de antigüedad con sus principales aportaciones a la materia que nos ocupa.

Charles Dow (1851-1902) fue cofundador del periódico *Wall Street Journal*. Entre sus avances en destaca la creación del índice de mercado *Promedio Industrial Dow Jones*.

En el caso de *Richard Wyckoff* (1873-1934), se destaca que fue uno de los pioneros en el análisis técnico de mercados de acciones.

El economista *Elliot* (1871-1948) descubrió que el mercado se basaba en principios sociales subyacentes. A principios de los años 20, concluyó que los mercados siguen las leyes de la naturaleza basándose en ratios de *Fibonacci*. Con esto demostró que el comportamiento de los mercados pueden predecirse.

Gann (1878-1955) destacó por sus métodos de análisis de mercados basados en geometría y matemáticas. Se consideraron técnicas innovadoras y se siguen usando a día de hoy.

Por último, el analista *Arthur Alexander Merrill* (1906-2005) destacó por sus aportes en la creación de un sistema de figuras de precios para ser aplicado en el análisis técnico.

Debido al carácter técnico de los análisis propuestos por cada uno de estos autores, y a su respectiva relevancia, en las siguientes secciones se profundiza sobre los análisis o métodos de *Dow* y *Wyckoff*, como posibles candidatos a ser implementados en la aplicación objetivo de este proyecto.

2.1.4. Análisis de Charles Henry Dow

El método de *Dow* se basa directamente en el precio, que refleja el comportamiento humano. Es el indicador más rápido para confirmar una tendencia y se basa en conceptos sencillos pero efectivos.

Este método consistía en un análisis de máximos y mínimos de las fluctuaciones de los mercados financieros, con objetivo de predecir la dirección del mercado. Estos puntos clave son comparados con los máximos y mínimos anteriores.

Dow desarrolló la siguiente teoría, compuesta de 6 principios y llamada la *Teoría de Dow*:

1. Cualquier acontecimiento externo (fundamento, noticia, etc.) se ve en el gráfico. En otras palabras, los precios son influenciados por dichas situaciones.
2. Cada mercado financiero tiene tres etapas o tendencias que se repiten:
 - **Primaria:** tendencia a largo plazo, puede durar entre 1 y 3 años.
 - **Secundaria:** tendencia a medio plazo, puede durar entre 3 semanas y 3 meses.
 - **Terciaria o menor:** tendencia a corto plazo, dura menos de 3 semanas.
3. Cada una de las etapas o tendencias primarias presentan 3 fases, que se pueden presentar de dos formas distintas, según si la tendencia es bajista (el precio tiende a bajar) o alcista (el precio tiende a subir):
 - **Tend. prim. alcista:** acumulación → tendencia → euforia
 - **Tend. prim. bajista:** acumulación → tendencia → pánico
- Estas fases se pueden apreciar en la figura 2.2.
4. El volumen de negocio confirma la tendencia. El volumen de negocio en trading es la cantidad de un activo en el que se invierte durante un periodo de tiempo. Es un indicador clave de la actividad del mercado y la liquidez.
 - Cómo confirma la tendencia alcista: si el precio sube, el volumen aumenta; si el precio baja, el volumen disminuye.



Figura 2.2: Fases de acumulación y euforia para tendencia alcista. Fuente: [Teoría de Dow en Análisis Técnico — Dow Theory](#)

- Cómo confirma la tendencia bajista: si el precio sube, el volumen baja; si el precio baja, el volumen aumenta.
5. Las tendencias se confirman con los índices *Dow Jones Industrial* y *Dow Jones Transports* (antiguo *Ferrocarril*). Si la tendencia no está confirmada por los índices mencionados tenemos una señal de debilitación de reversión de la tendencia.
 6. La actual tendencia se mantiene vigente hasta que se demuestre el cambio.

En el análisis, Dow considera que en escenarios alcistas se mantiene que tenemos máximos más altos y mínimos más altos. En escenarios bajistas tenemos máximos más bajos y mínimos más bajos. Si no ocurre ni lo primero ni lo segundo, estamos en una situación de mercado lateral o de consolidación.

Dow define en su análisis los siguientes conceptos dentro de los gráficos de precios:

- **Soporte o resistencia:** líneas (o rango) imaginarias en el gráfico de precios que indican zonas. Una vez el precio supera dichas zonas, tenemos una buena posible entrada ya que indica un cambio de tendencia. Indica puntos exactos de entrada al mercado y predice la tendencia.
- **Cascada de mínimos:** la cotización va haciendo mínimos menores. Es una señal de tendencia bajista muy útil para mercados tendenciales con poco retroceso. Se usa un *Stop Lose* más amplio, lo que nos indica un mayor riesgo.

Como conclusión, Dow propuso un análisis de mercado para detectar tendencias y períodos de consolidación de los precios. A pesar de que los mercados estén cambiando continuamente, el comportamiento de hoy día en cuanto a dichos conceptos es el mismo que Dow enunció y que recogemos en este apartado.

2.1.5. Análisis de Richard Demille Wyckoff

El aporte de *Wyckoff* fue un enfoque de análisis técnico basado principalmente en la relación entre las fuerzas de la oferta y la demanda. Cuando los grandes operadores quieren comprar o vender, llevan a cabo unos procesos que dejan huella. Dicha huella puede verse en los gráficos de precios de los mercados a través del precio y el volumen. En otras palabras, Wyckoff buscaba ver quién tiene el control del mercado con el objetivo de operar junto a dichos operadores. Para ver estas intenciones de los grandes traders, Wyckoff usaba gráficos de barras y gráficos de Punto y Figura.

Richard Wyckoff propuso en sus estudios como analista de mercados financieros, un enfoque en 5 pasos. Este enfoque destaca por su universalidad, ya que puede ser usado en cualquier mercado con suficiente liquidez y en cualquier temporalidad o marco de tiempo.

1. Determinar la posición actual y la tendencia futura probable del mercado. Consiste en realizar una evaluación del mercado para ver la tendencia futura y poder tomar decisiones a corto o largo plazo.
2. Seleccionar acciones acordes con la tendencia. En tendencia alcista, elegir acciones más fuertes que el mercado, es decir, acciones que produzcan mayores aumentos porcentuales que el mercado durante los repuntes y menores disminuciones durante las reacciones. En el caso de tendencia bajista, acciones más débiles que el mercado.
3. Elegir operaciones con una causa que iguale o exceda su objetivo mínimo. Aquí entra en juego la ley de Wyckoff de causa y efecto. Se eligen acciones que han creado una causa suficiente para su objetivo. Se utilizan proyecciones Punto y Figura.
4. Elegir acciones disponibles con respecto a la tendencia o fases de acumulación o distribución.
5. Elegir un buen activo para operar. Un buen activo será aquel que vaya en armonía con el mercado general, lo que proporcionará más probabilidades de éxito.

Uno de sus principales aportes fueron las tres leyes de su método de análisis, el método Wyckoff:

1. **Ley de oferta y demanda.** Si la demanda es mayor que la oferta, el precio sube; si la demanda es menor que la oferta, el precio baja. El trader o analista debe estudiar el equilibrio de oferta y demanda comparando precio y volumen.
2. **Ley de causa y efecto.** La causa se mide con el recuento de volumen en un gráfico. El efecto es la distancia que se mueve el precio correspondiente a dicha causa.
3. **Ley de esfuerzo y resultado.** Esta ley proporciona una advertencia de un posible cambio de tendencia en un futuro cercano.

En la figura 2.3 se puede observar cómo la ley de oferta y demanda afecta en la tendencia de un gráfico. En este caso se trata de un boceto de un gráfico de línea. La imagen comienza con una fase de acumulación. Después de esta fase encontramos una clara tendencia alcista seguida de una fase de distribución. Entre la tendencia alcista y la distribución, se vería un patrón indicando una demanda mayor que la oferta. Estos patrones se pueden apreciar estudiando volúmenes, ya que es ahí donde se ve la oferta y demanda. Tras la distribución, la oferta supera a la demanda y la tendencia cambia a bajista.

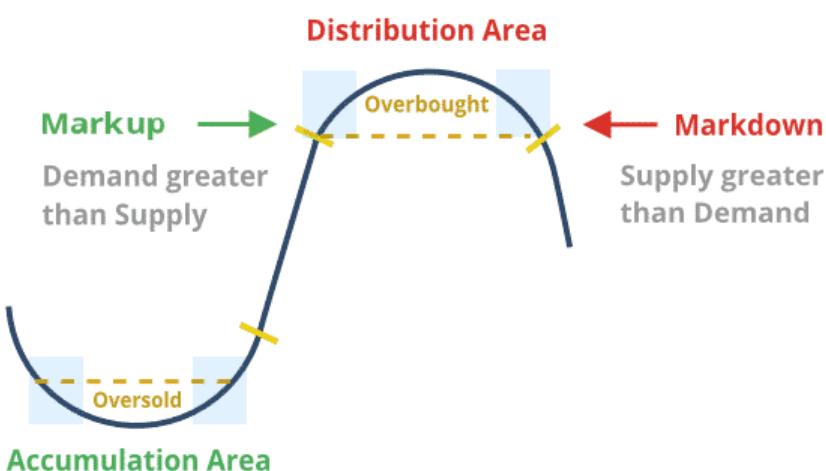


Figura 2.3: Ciclos del precio de Wyckoff. Fuente: [El método Wyckoff](#)

Tanto en este análisis como en el de Dow se ha hablado de *acumulación* y de *distribución* pero no se han introducido dichos conceptos. Tanto la fase de acumulación como la de distribución son *rangos de trading*. Los rangos de trading son valores de precios en intervalos concretos en el eje x donde la tendencia que se estaba siguiendo se detiene y hay un equilibrio relativo entre oferta y demanda. La acumulación es el período de equilibrio de oferta y demanda tras venir de tendencia bajista para continuar en alcista. La distribución es la misma zona pero proveniendo de alcista para dar lugar a bajista.

Los rangos de trading y las tendencias serán claves en el análisis de Wyckoff puesto que en su método se estudian estos conceptos y cómo son alterados por la oferta y la demanda. A raíz de estas variables y una serie de decisiones, se ordenará una operación u otra. Concretamente, las decisiones se tomarán en los rangos de trading, puesto que son las principales causas de un cambio de tendencia.

Los 5 enfoques enumerados anteriormente fueron propuestos por Wyckoff de forma teórica. A continuación expongo cómo Wyckoff propuso seguir este enfoque en la práctica. Para ello, Wyckoff describió las fases de acumulación y distribución con una serie de patrones o esquemas que siempre se repetían. Ver estos esquemas nos ayudaría a conocer la tendencia futura y operar.

Esquemas de acumulación

A continuación enumero los patrones vistos en una fase de acumulación y que darían lugar a una tendencia alcista. En este caso, si se cumplen los patrones, podríamos entonces ordenar una COMPRA.

- Vemos un *PS*, Preliminary Support. Es el patrón que da comienzo a la acumulación. Es cuando la compra de un activo comienza a provocar un soporte en el gráfico. En este momento hay aumentos de volumen y amplitud en los precios, lo que indica que la venta (tendencia bajista) llega a su fin.
- Vemos un *SC*, Selling Climax. Es el precio mínimo que se termina formando y que forma el soporte mencionado. Indica que los grandes inversores están comenzando a comprar.
- Vemos un *AR*, Automatic Rally. Hay un retroceso en el precio debido a la disminución de la presión de la oferta. Este punto marcaría el techo que serviría como soporte superior del rango de acumulación.
- Vemos *STs*, Secondary Tests. El precio fluctúa entre el AR y el SC realizando STs. Podemos encontrar múltiples secondary tests en un mismo rango. Estas fluctuaciones comprueban el equilibrio que hay entre la oferta y demanda en el rango.

- Vemos un *SOS*, Sign Of Strength. Avance del precio con rangos más amplios y un mayor volumen.
- Vemos un *LPS*, Last Point of Support. Precio mínimo resultado de un retroceso del SOS. Suele coincidir con el techo que había generado el AR, en un menor rango de precios y volumen. Puede haber más de uno aunque se llame last. En la práctica son mínimos más altos desde el AR.

En la figura 2.4 se muestran los patrones mencionados en la anterior lista.

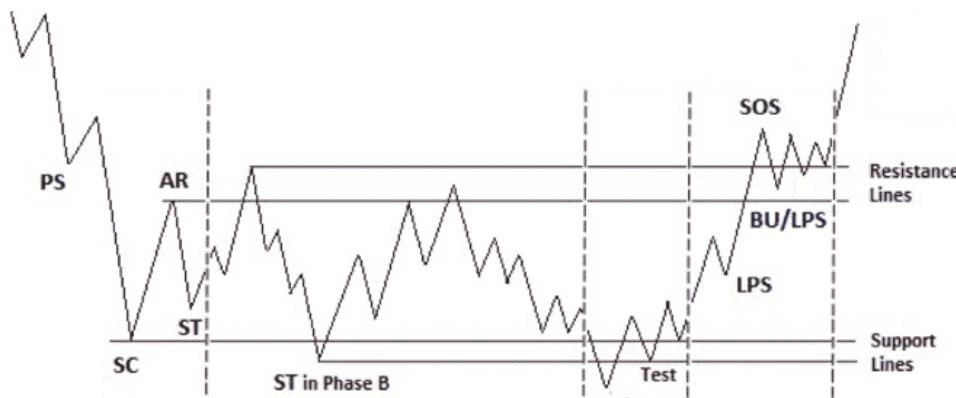


Figura 2.4: Esquema de acumulación en el método Wyckoff. Fuente: [El método Wyckoff](#)

Esquemas de distribución

En esta sección enumero las pautas seguidas en una fase de distribución y que darían lugar a una tendencia bajista. En este caso, si se cumplen los patrones, podríamos entonces ordenar una VENTA.

- Vemos un *PSY*, Preliminary Supply. Es el patrón que da comienzo a la distribución. Es cuando la venta de un activo comienza a provocar un techo en el gráfico. En este momento hay aumentos de volumen y amplitud en los precios, lo que indica que la compra (tendencia alcista) llega a su fin.
- Vemos un *BC*, Buying Climax. Es el precio máximo que se termina formando y que forma el techo o resistencia mencionado. Indica que los grandes inversores están comenzando a vender.
- Vemos un *AR*, Automatic Reaction. Hay un retroceso en la acción del precio. Este punto marcaría el suelo que serviría como soporte inferior del rango de distribución.

- Vemos *STs*, Secondary Tests. Misma definición que en el esquema de acumulación, fluctuaciones entre el AR y el BC, en este caso.
- Vemos un *SOW*, Sign Of Weakness. Se sobrepasa el suelo formado por el AR. Avance del precio con rangos más amplios y un mayor volumen.
- Vemos un *LPSY*, Last Point of Supply. Precio máximo resultado de un retroceso del SOW. Suele coincidir con el suelo que había generado el AR, en un menor rango de precios y volumen. Vienen a ser máximos más bajos, lo que indica el cambio de tendencia a bajista.

En la figura 2.5 se muestran los patrones mencionados en la anterior lista.

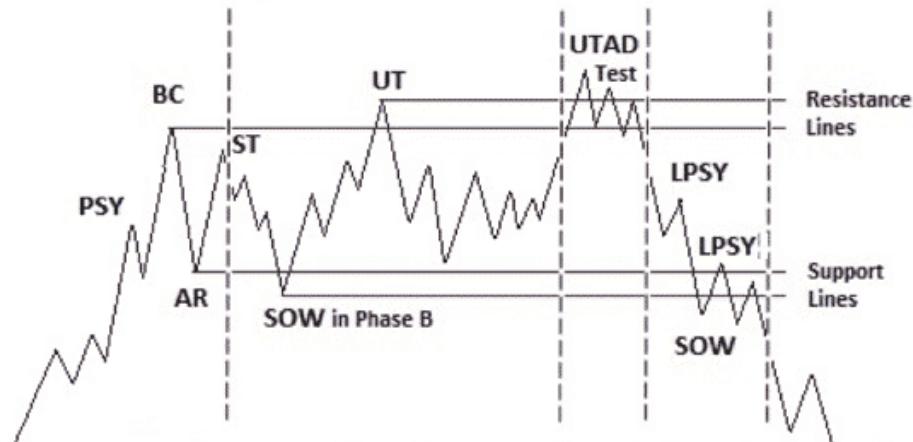


Figura 2.5: Esquema de distribución en el método Wyckoff. Fuente: [El método Wyckoff](#)

2.2. Trading algorítmico

En pocas palabras, el trading algorítmico es implementar un sistema de trading que opere de forma automática.

El trading algorítmico analiza gráficos de precios de acuerdo a unos criterios preestablecidos, que dependerán del análisis y del propio algoritmo. Cuando el mercado se ajusta a los criterios mencionados, el algoritmo que se ha diseñado para hacer trading ejecutará una acción de compra o venta de manera automática.

A parte del ahorro de tiempo y esfuerzo, que es la obvia ventaja de usar trading algorítmico, podemos encontrar otros puntos a favor que hacen de

las operaciones más eficientes si las comparamos con cómo las haría un operador humano.

2.2.1. Ventajas

- **Diversificación:** existe la posibilidad de aplicar un mismo análisis a distintos mercados financieros, aunque puede existir la posibilidad de que en ciertos mercados obtengamos beneficios con una técnica específica que no obtenemos en otro mercado distinto. Esto es más complicado para un operador humano ya que los cambios de acciones de precios en los mercados financieros hacen que un análisis técnico no sea sencillo para un trader acostumbrado a ciertos mercados financieros.
- **Evaluación de técnicas usadas:** debido a que el trading algorítmico automatiza la acción de realizar compras y ventas, podemos evaluar de forma fácil cuándo cierta técnica de análisis es más o menos eficiente en uno u otro mercado. Aquí podemos hablar de distintas horas del día, diferentes temporadas, mercados financieros, etc.
- **Evitar las emociones:** al automatizar un sistema para comprar y vender acciones, el operador humano evita dejarse guiar por las emociones. Esto puede parecer una ventaja simbólica, pero es bastante importante ya que el mercado suele estar sujeto a estadísticas, probabilidades, etc. Si el trader opera suponiendo que cierta vez ocurrirá algo distinto, acaba dejando de lado el análisis puramente técnico. En resumen, evitamos la principal razón por la cual la mayoría de personas que empiezan a dedicarse al trading fracasan, psicología y emociones.
- **Capacidad para desplegar en la nube:** al ser un algoritmo que puede ser desarrollado en un producto software, es posible desplegar o hacer deploy del mismo en un servidor, de manera que el algoritmo desarrollado siempre está conectado al mercado y aplicando reglas para comprar o vender según los criterios mencionados.
- **Precisión y capacidad para realizar compras y ventas de forma simultánea:** el programa sería siempre más preciso que un humano a la hora de realizar entradas y salidas al mercado. Además, puede hacer esto de forma simultánea y operar para distintos mercados financieros a la vez.
- **Posibilidad de aplicar aprendizaje automático:** no sólo podemos centrarnos en desarrollar un algorítmico basado en reglas o análisis del mercado actual sino que también es posible entrenar algoritmos con modelos usando datos históricos de los mercados financieros. Aquí destaca el uso de herramientas de BI y Big Data.

2.2.2. Desventajas

- **Dependencia de noticias:** como se ha mencionado en anteriores apartados de esta memoria, el trading algorítmico surge principalmente del análisis técnico. Aquí destacamos una desventaja del mismo. En el trading algorítmico es muy difícil tener en cuenta noticias que afecten o puedan afectar al comportamiento del mercado en cuestión. Si se decide implementar lógica para estudiar noticias, la complejidad aumenta bastante.
- **Dificultad para detectar la acción del precio:** debido a que el análisis del trading algorítmico es mayoritariamente técnica y basado en cálculos matemáticos, estadística, etc, es complejo detectar la existencia de patrones en el gráfico de precios. Esto sí es más sencillo de realizar por parte de un trader. Entre estos patrones podemos ver soportes, resistencias, líneas de tendencia, niveles, etc. En resumen, es complicado programar una detección de patrones ya que no es algo numérico sino que el operador los identifica a simple vista.
- **Complejidad en la programación:** es realmente complejo programar un algoritmo de trading.

Capítulo 3

Planificación

3.1. Planificación temporal

3.1.1. Fases del proyecto

En este apartado se mencionan cada una de las fases en las que se pretende dividir el desarrollo del proyecto. Con cada fase se incluye una explicación de los temas a desarrollar. En el siguiente apartado de la sección se encuentra el *diagrama de Gantt* correspondiente.

Fase inicial o de investigación

En cuanto a la primera fase del desarrollo del proyecto, se comenzará con una investigación teórica sobre lo que es el trading, trading algorítmico y demás conceptos de economía que entran en juego en el tema. Aquí se realizará también una investigación sobre dos de los principales referentes del análisis de mercados: *Charles Dow* y *Richard Wyckoff*.

En esta fase también se harán programas de ejemplo usando las distintas *APIs* de conocidas plataformas de trading como son *MetaTrader* o *Binance* (específica para criptomonedas).

Durante este período también se estudiarán las posibles herramientas a usar para el proyecto. Se seguirán tutoriales de *Django* ya que se decidió que la APP fuera implementada usando este framework y por tanto, programada con *Python*. También se estudiará la viabilidad de realizar el proyecto en *Windows*, ya que librerías como la de *MetaTrader5* para *Python* sólo están disponibles en este sistema operativo.

Podemos llamar a esta fase inicial o de investigación y se pretende que sea la que más tiempo ocupe en el desarrollo del proyecto debido a que tener bases sólidas sobre el tema es vital para el transcurso del trabajo. Se prevee

que ocurra desde la propuesta del proyecto, noviembre de 2020, hasta principios de 2021.

Fase de recopilación de datos

En esta segunda fase de desarrollo del proyecto se pretende investigar qué volumen de datos históricos de mercados financieros podríamos recopilar de internet u otras fuentes. Esta acción es necesaria previa al análisis de lo que va a ser el producto software, ya que en todo caso, se necesitará cierto volumen de datos para poner a funcionar los algoritmos de trading, ya fuesen de análisis puro, o de aprendizaje automático.

En la primera aproximación, se implementarán una serie de scripts en *Python* para obtener datos de mercados a través de *MT5*. Con estos scripts, se busca conseguir el mayor volumen de datos posible para el mayor número de mercados financieros posibles.

Debido a la clara limitación en espacio que esto supone se estudiará la posibilidad limitar el número de mercados financieros a tratar, buscando diversidad en cuanto a tener varios tipos de mercados.

Esta fase de recopilación de datos históricos supondría desde principios de 2021 hasta principios del Q2 (abril-mayo) de 2021.

Fase de análisis

En la fase de análisis, se realizará la especificación de requisitos funcionales, no funcionales y de información. En esta fase también se describirán los implicados en la aplicación.

Esta fase ocurrirá justo al finalizar la etapa anterior y no durará más de dos o tres semanas.

Fase de diseño

En la cuarta fase del desarrollo del proyecto, se propondrá el diseño a alto nivel de la aplicación. En esta fase de diseño, se estudiará la arquitectura del software a desarrollar con un diagrama de paquetes o módulos en los que se dividirá la aplicación.

En este punto se presentará también el primer boceto de diseño de la interfaz presente en la aplicación web.

Esta fase ocupará dos semanas más y ocurrirá inmediatamente después de la fase de análisis.

Fase de desarrollo

En la fase de desarrollo de código del proyecto es donde se comenzará con la implementación del producto software.

Este período se describirá de manera más detallada en el capítulo 6. En dicho capítulo se explicará la metodología ágil usada para el seguimiento del desarrollo y cada una de las issues de *GitHub* creadas para implementar la aplicación.

Esta fase se desarrollará desde junio hasta la tercera semana de agosto de 2021.

Fase de pruebas

La última parte del proyecto recae en una fase de pruebas. En esta fase se realizarán pruebas y se incluirán posibles fixes de última hora para bugs encontrados. Esta fase también servirá como retroalimentación para las conclusiones y para ver algunos resultados del algoritmo en funcionamiento.

Este último período del proyecto ocurrirá en la última semana de agosto y primera de septiembre de 2021.

3.1.2. Diagrama de Gantt

En la figura 3.1 se muestra el diagrama de Gantt correspondiente con las fases del proyecto anteriormente mencionadas.

3.2. Presupuesto de ejecución

En este apartado menciono una aproximación de los posibles costes del desarrollo del proyecto. Detallo aquí los gastos por personal, equipos o recursos informáticos, licencias de softwares, etc.

3.2.1. Gastos de personal

En esta sección se recoge el gasto total referente a pago de personal que ha estado trabajando en el desarrollo del proyecto.

El trabajo ha sido realizado por un grupo de una sola persona o desarrollador con tutorías de un profesor.

Suponiendo que dicha persona es recién egresada, su puesto sería el de desarrollador Junior. Si echamos un ojo a páginas web externas que proporcionan datos como la media de sueldos por trabajo y ciudad, vemos que el salario base promedio de un programador Junior en Granada a tiempo completo es de 16.432 €/año, lo que sería equivalente a un total de 1.163 €/mes. Fuente: <https://es.indeed.com/career/programador-junior/salaries/Granada-Granada-provincia>.

Viendo el diagrama de Gantt descrito en el anterior punto de la memoria, preveemos que el desarrollo del proyecto dure 10 meses. Suponiendo que el trabajador trabajará a media jornada (4 horas diarias, suponiendo un total aproximado de 400 horas), el pago total que se le haría sería de $1.163 * 10 / 2 = 5.815 \text{ €}$.

A esto hay que sumarle las horas invertidas por el tutor del proyecto en las entrevistas y seguimiento de la aplicación. Aproximando el número de entrevistas previstas a 4 en la fase de investigación y 6 en la fase de desarrollo; y suponiendo una duración de 1 hora por entrevista y coste de 35 €/entrevista, tendremos una inversión de 350 €.

3.2.2. Gastos de recursos informáticos y energía

En este apartado se recoge el gasto referente a la obtención de recursos informáticos, tanto de hardware como de software necesario para el desarrollo del proyecto, así como gastos referentes a licencias y software.

- **Equipo usado para el desarrollo del proyecto:** Lenovo Legion 5 - Portátil Gaming 15.6" FullHD 144Hz (AMD Ryzen 7 4800H, 16GB RAM, 512GB SSD, Nvidia RTX2060-6GB, Windows 10).
Coste: 999 €

- **Periféricos adicionales:** Monitor extra, ratón y teclado.
Coste: 200 €

- **Licencia del IDE de desarrollo previsto, PyCharm:** Se prevee que se pueda hacer un uso gratuito del mismo, pero se contemplará la posibilidad de usar su versión Professional. Licencia gratuita con correo institucional de la universidad pero que a priori es de pago.

Coste: 8,90 €/mes = 89 €

- **Resto de software:** El resto de aplicaciones y licencias que se pre-
veen usar son de uso gratuito.

Además de los recursos informáticos, recogemos en este apartado el gasto energético supuesto en el desarrollo del trabajo. Asumiendo que se trabajará durante 10 meses a media jornada y suponiendo el gasto medio actual en España al mes (40 €) en lo que a electricidad se refiere, dicho gasto ascenderá a un total de 400 €.

3.2.3. Resumen del presupuesto

En la figura 3.2 se resumen todos los gastos vistos en las secciones anteriores.

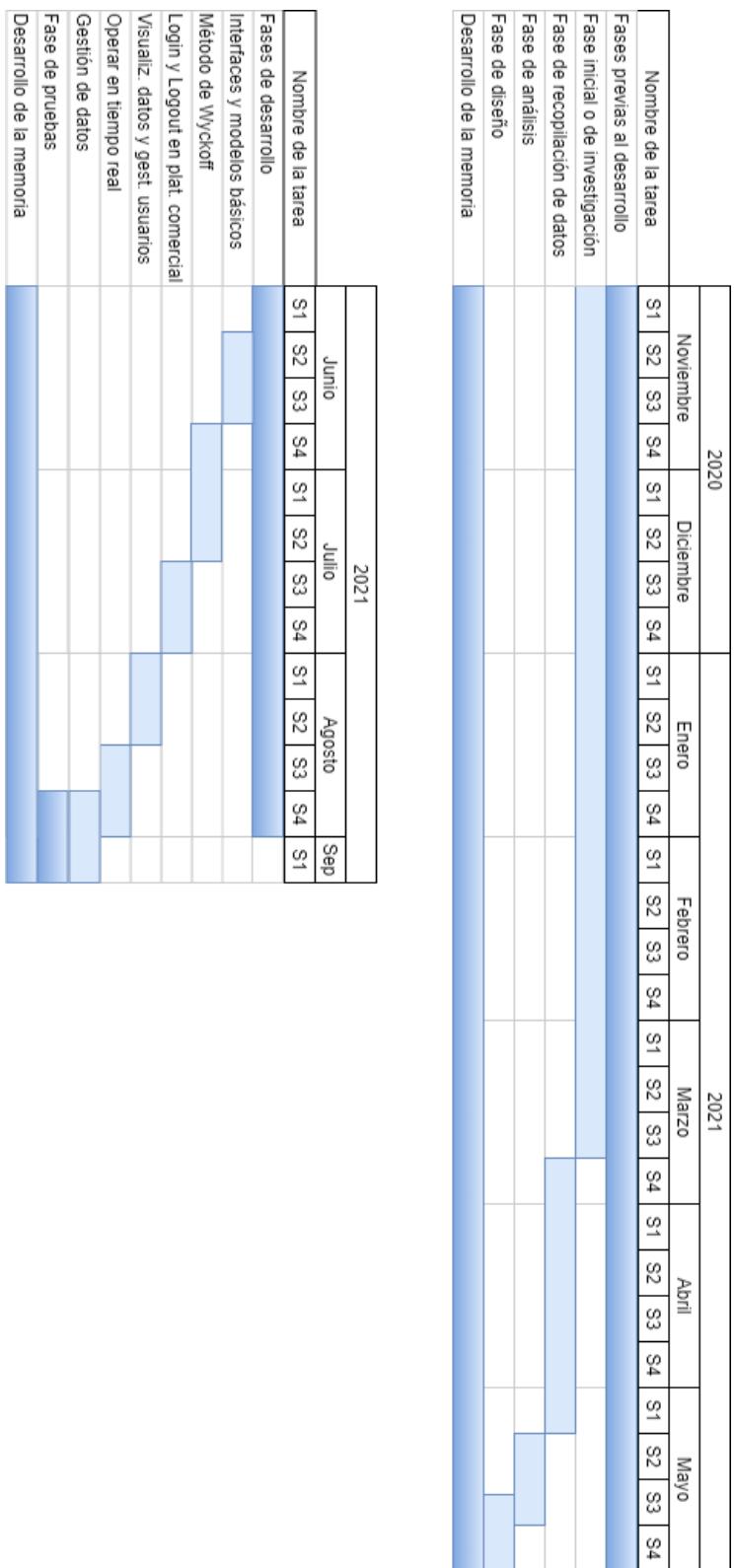


Figura 3.1: Diagrama de *Gantt* del proyecto

| Presupuesto | |
|------------------------------------|---------|
| Total sueldos desarrollador Junior | 5.815 € |
| Horas de seguimiento | 350 € |
| Equipo | 1199 € |
| Licencias de software | 89 € |
| Electricidad | 400 € |
| Total gastos previstos | 7.853 € |

Figura 3.2: Presupuesto de ejecución del trabajo

Capítulo 4

Análisis

En este capítulo se describe la fase de análisis del problema a tratar. En concreto desarrollo la descripción de los implicados en el proceso y la especificación de requisitos funcionales, no funcionales y de información que debe cubrir la solución al problema. Como refuerzo para los requisitos del proyecto, se incluyen diagramas de casos de uso en los que interviene cada uno de los implicados.

4.1. Descripción de los implicados

En esta aplicación, destacamos dos principales implicados: el administrador de la aplicación y el usuario final de la aplicación.

- **Desarrollador y administrador del sistema:** La responsabilidad del implicado será la de realizar las distintas actividades de desarrollo de la aplicación: corregir errores o añadir nuevas features, entre otras actualizaciones, que garantizan el correcto funcionamiento del sistema y su mantenimiento. Se encargará también de las tareas de gestión de la base de datos como puede ser añadir nuevos usuarios.
- **Usuario de la aplicación:** Este implicado representa al cliente que usa la aplicación. Este implicado hace uso de la aplicación como usuario final.

4.2. Especificación de requisitos

4.2.1. Requisitos Funcionales

Descripción de los requisitos más importantes a nivel de funciones que debe incluir el sistema, realizando una clasificación en categorías, a cada

uno de los requisitos se le ha asignado un código y un nombre, con el fin de identificarlos fácilmente a lo largo de todo el proyecto.

- **RF-1. Iniciar sesión en la aplicación.** El usuario de la aplicación deberá iniciar sesión en la APP para hacer uso de la misma. El sistema por tanto incluirá un **sistema de gestión de usuarios**.
- **RF-2. Gestiones de la plataforma de trading.**
 - **RF-2.1. Login en la plataforma de trading.** El usuario de la aplicación podrá conectarse con su cuenta de trading, comercial o demo, para poder hacer el uso completo de la APP.
 - **RF-2.2. Ver capital disponible.** El usuario podrá ver el capital disponible en su cuenta de trading.
 - **RF-2.3. El usuario podrá ver información de las operaciones** realizadas y de operaciones que en ese momento aún no se han cerrado.
- **RF-3. Gestión de datos.** El usuario de la aplicación podrá elegir qué datos descargar y guardar en la base de datos. Estos datos históricos comenzarán desde cualquier fecha disponible hasta la actualidad. Los datos pertenecerán a un mercado financiero específico y tendrán una temporalidad específica, siguiendo el formato *OHLC*.
- **RF-4. Visualización de datos.** El usuario de la aplicación podrá ver información de precios de un mercado financiero específico.
 - **RF-4.1. Ver datos de mercado en rango de tiempo específico.** El usuario de la aplicación podrá ver información de precios entre dos fechas específicas.
 - **RF-4.2. Ver datos de mercado con un marco de tiempo específico en tiempo real.** El usuario de la aplicación podrá ver información de precios en tiempo real con un marco de tiempo específico.
- **RF-5. El usuario podrá elegir un modelo y realizar trading algorítmico.** También podrá parametrizarlo según modelo y elegir tiempo en el que quiere dejar haciendo las operaciones automáticas.
- **RF-6. El usuario podrá elegir un modelo y realizar trading algorítmico a modo de backtesting.** De esta forma podrá probar cada uno de los modelos en un mercado y periodo de tiempo prefijados.

4.2.2. Requisitos No Funcionales

- **RNF-1.** La plataforma de trading que usará la aplicación será *MetaTrader5*.
- **RNF-2.** La aplicación permitirá el uso de cualquier bróker aceptado por *MetaTrader5*.
- **RNF-3.** Para la visualización de datos, el usuario podrá elegir un marco de tiempo de entre m1, m3, m5, m15, m30 ó m45; h1, h2, h3 ó h4; d1 (minutos, horas o días, respectivamente).
- **RNF-4.** La aplicación responderá a las peticiones de los usuarios en un tiempo determinado, mostrando un aviso de error si el tiempo de respuesta es superior al establecido.
- **RNF-5.** La aplicación deberá funcionar computadoras mediante navegador web.
- **RNF-6.** Para la implementación de la aplicación, se utilizará *Python* y su framework *Django*.
- **RNF-7.** La interfaz debe ser sencilla e intuitiva.

4.2.3. Requisitos de Información

- **RI-1.** El sistema gestor de bases de datos utilizado será *SQLite3*.

4.3. Diagramas de casos de uso

En esta sección, muestro los distintos diagramas de cada caso de uso que implementa la aplicación. Con estos diagramas se completa la fase de análisis del proyecto y con ellos vemos los roles que tienen los implicados y cómo interactúan con el sistema. Véase punto 4.1, descripción de actores o implicados.

4.3.1. Gestión de sesiones de la APP

En el diagrama 4.1 se representan los casos de uso referentes al sistema de gestión de usuarios de la aplicación.

El usuario de la aplicación deberá identificarse y cerrar sesión en *TradingAPP*. Estas acciones de Login y Logout interactúan internamente con el módulo de *Django*, que es el que implementa el sistema de sesiones.

A su vez, el administrador de la APP será quién administre los usuarios, a través del módulo de *Django*.

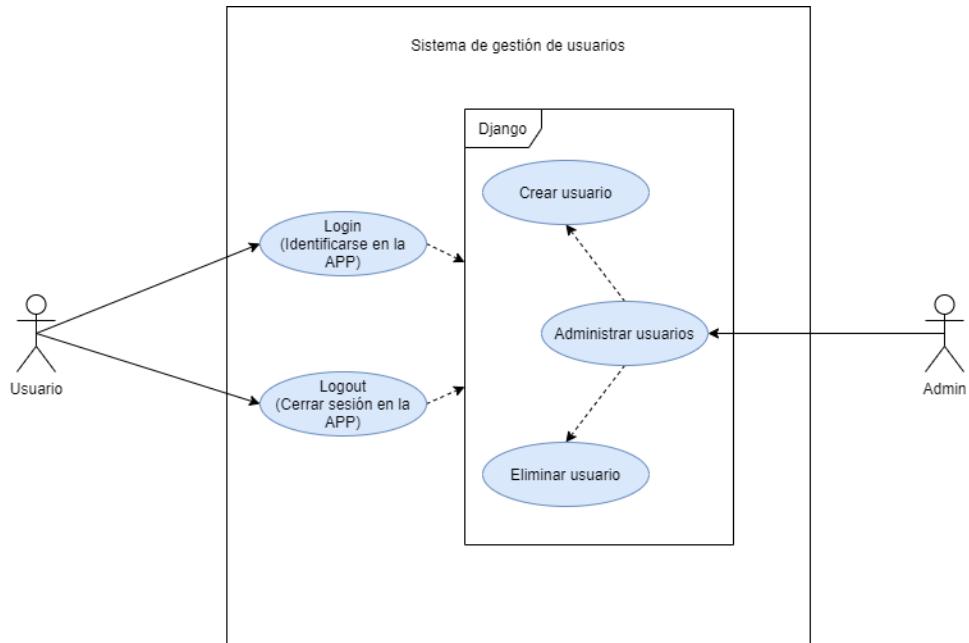


Figura 4.1: Diagrama de casos de uso referente al sistema de gestión de usuarios de la APP

4.3.2. Gestiones de la plataforma de trading (MT5)

En el diagrama 4.2 se representan los casos de uso referentes a la gestión de la plataforma de trading. Como ya hemos mencionado en los requisitos no funcionales y en otras ocasiones, esta plataforma será *MetaTrader5*. Se mencionará *MetaTrader5* a lo largo del proyecto con la abreviatura *MT5*.

El usuario deberá identificarse con su cuenta, contraseña y servidor en la plataforma en cuestión, MT5. La cuenta podrá ser comercial o demo; y podrá ser creada por cualquier bróker aceptado en MT5. A este caso de uso se le suma el de desconectarse de la cuenta en la que el usuario se ha identificado previamente.

El usuario podrá ver el capital disponible y las operaciones realizadas y en curso. Estos dos casos de uso interactúan internamente con MT5, que es el módulo que nos provee dicha información. La previa identificación del usuario con su cuenta de MT5 será obligatoria para los otros casos de uso.

4.3.3. Gestión y visualización de datos de mercados

El diagrama 4.3 muestra los casos de uso referentes a la gesstión vi-
sualización de datos de mercados financieros. Dichos datos serán mostrados

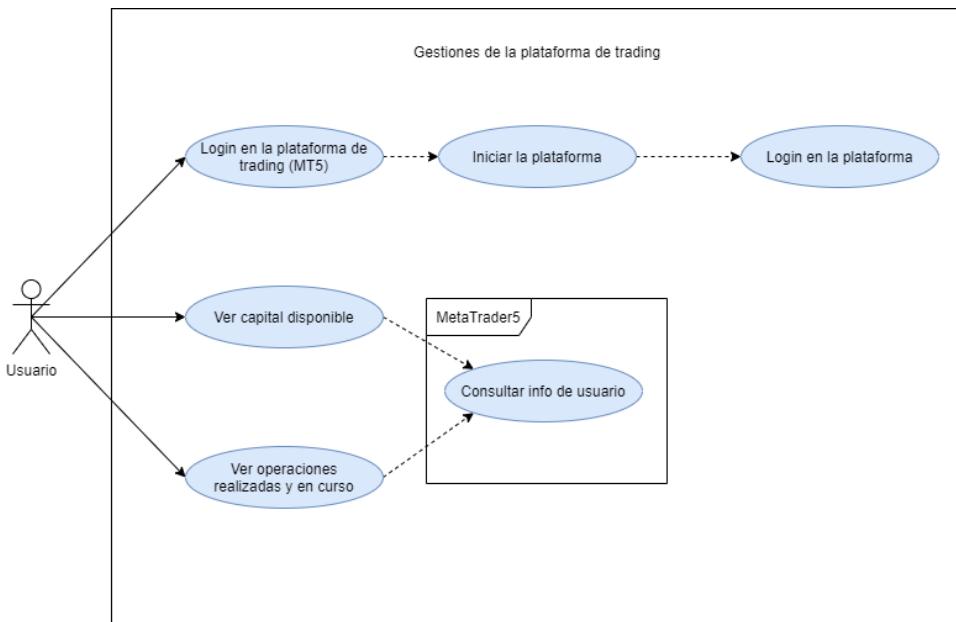


Figura 4.2: Diagrama de casos de uso referente a la gestión de la plataforma de trading (MT5)

mediante un gráfico que muestra precios por medio de velas siguiendo el formato *OHLC*. Para el insertado de dichos datos en el módulo, el usuario seleccionará los parámetros que estime oportunos para la recopilación de los datos históricos.

El usuario podrá ver datos históricos de mercado en un rango de tiempo específico. Este caso de uso interactúa con los datos proporcionados previamente por el usuario.

El usuario de la APP podrá ver datos en tiempo real con un marco de tiempo específico, es decir, con velas generadas cada minuto, cada hora, cada día, etc. Este *CU* usa el módulo de *MT5* para obtener los datos y luego procesa dichos datos para ajustarlos al formato *OHLC*.

4.3.4. Funcionalidades de trading

Finalmente, el diagrama 4.4 representa los casos de uso referentes al módulo de trading y elección de algoritmos.

El usuario podrá elegir un algoritmo de trading. A la hora de operar, el usuario podrá usar el algoritmo elegido para operar en tiempo real o realizar backtesting. El usuario también podrá operar manualmente, siguiendo su propio análisis en tiempo real, sin algoritmo.

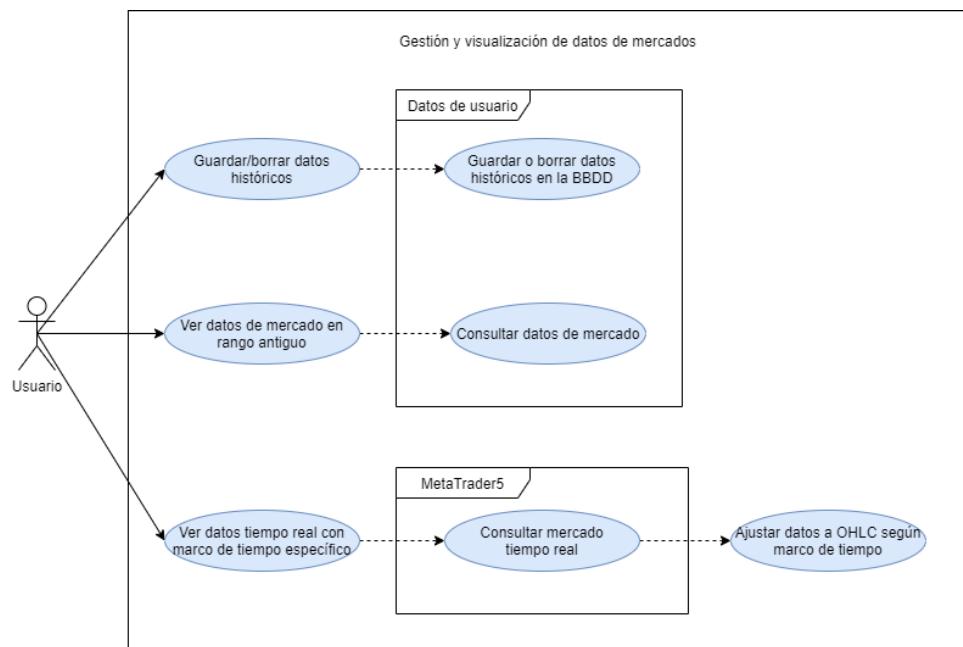


Figura 4.3: Diagrama de casos de uso referente a la visualización de datos de mercados financieros

Para los casos de uso de operar manualmente o de forma automática en tiempo real, la APP se conectará al módulo de MT5 para ordenar las operaciones de compra o venta.

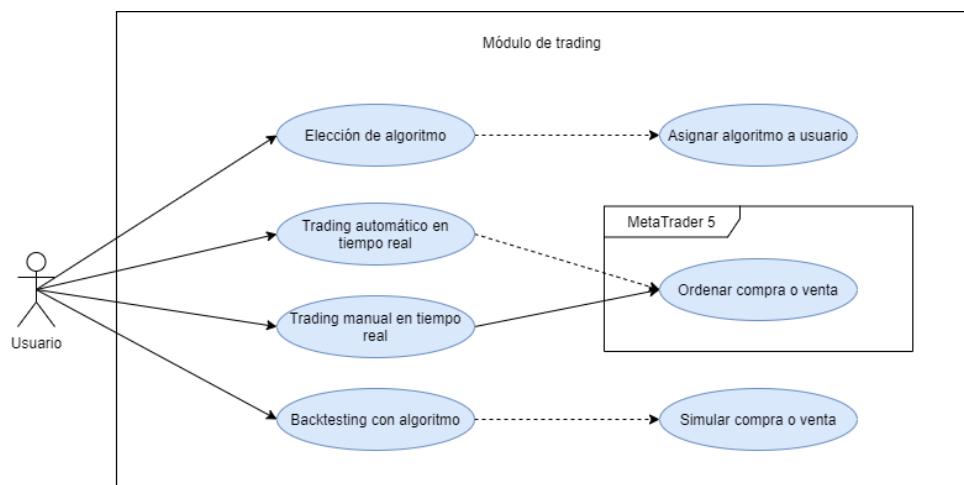


Figura 4.4: Diagrama de casos de uso referente a las funcionalidades de trading

Capítulo 5

Diseño

5.1. Arquitectura del software

En esta sección muestro un esquema que representa la arquitectura general del software mediante el diagrama de paquetes. Dicho diagrama puede verse en la figura 5.1.

Como resumen, en el esquema podemos ver cuál es la estructura principal del proyecto. El usuario de la aplicación (véase punto 4.1, segundo implicado) será el que haga un uso directo de la interfaz de la aplicación. El administrador (primer implicado, 4.1) sólo interviene en la gestión de usuarios, incluida dentro del módulo principal de la APP. A continuación presento cada uno de los módulos con sus funciones y demás utilidades:

- **Interfaz principal de la APP:** este módulo se corresponde con el controlador principal del software. Es usado por el usuario a través de una interfaz escrita en *Django* y se comunicará con el resto de módulos directamente.
- **Gestión de usuarios:** módulo proporcionado por *Django* por defecto. Implementa la página *admin* y las posibilidades de añadir nuevos usuarios, eliminarlos, editarlos, etc.
- **Datos de mercados:** este módulo se corresponderá con la base de datos y las operaciones que hagamos con dichos datos (insertado de datos, adaptaciones a cada algoritmo, etc.) Será usado por la interfaz principal de la aplicación y mandará los *inputs* a los algoritmos.
- **Algoritmos de trading o backend:** se corresponde con el backend o código fuente de la aplicación. Aquí se encontrará cada uno de los algoritmos que usemos para trading algorítmico. Recibe datos de la BBDD y se comunica con el módulo de la plataforma comercial para indicar cada una de las operaciones decididas y obtener información

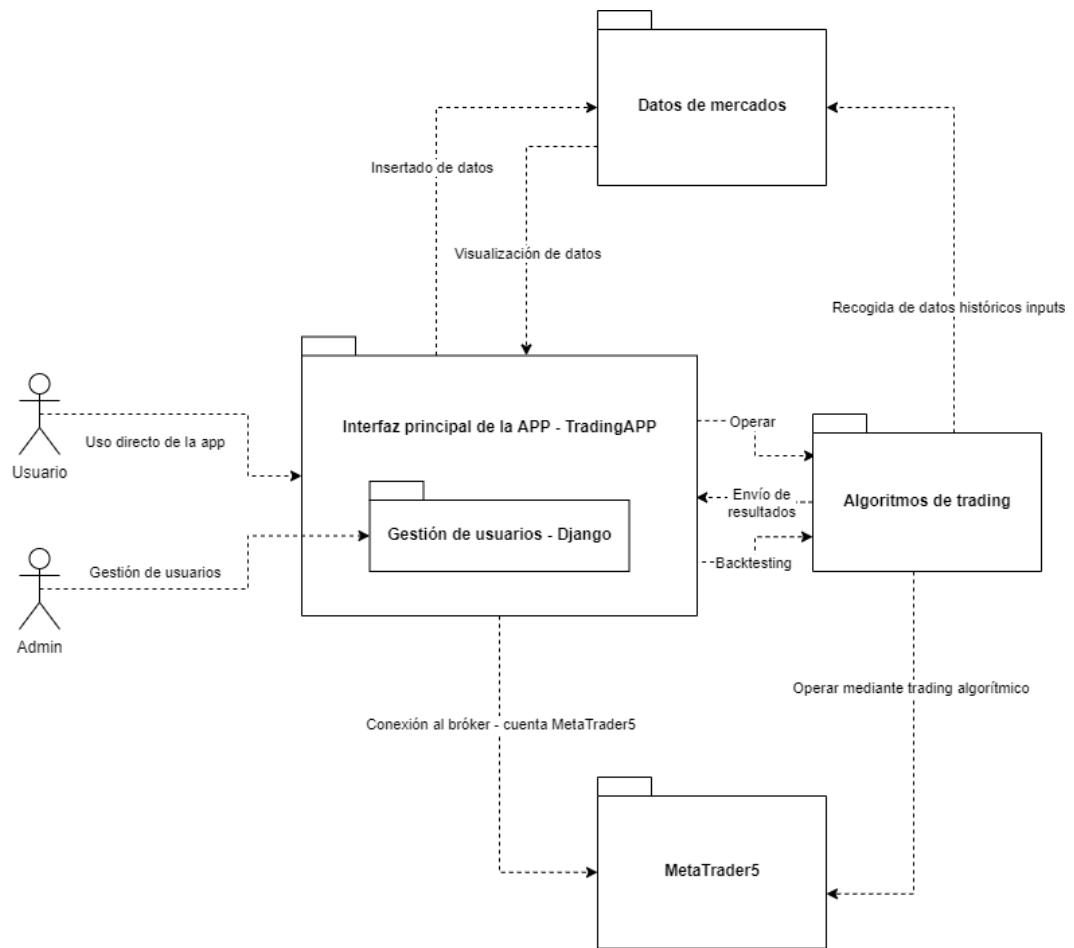


Figura 5.1: Arquitectura general del software.

en tiempo real. Se comunica con la interfaz principal para devolver los resultados de las opciones de operar y/o backtesting.

- **Plataforma comercial:** Módulo externo de la aplicación, se usa para mandar órdenes de operaciones y recibir resultados e información en tiempo real.

5.2. Diseño de clases

En esta sección se muestra el diagrama de clases del proyecto. Este diagrama se puede ver en la figura 5.2

Las clases *AbstractUser* y *User* forman parte del modelo *auth* de *Django* y proporcionan los objetos que se corresponderán con los usuarios de la

APP. He relacionado esta clase *User* con una clase personalizada *Sesion*, que es instanciada a la vez que se instancia *User*, de manera automática. Por otra parte, *ManagementUtility* es la clase que arranca el servidor web de *Django*.

AlgoritmoTrading, *Mercado* y *WyckoffTrading* son otras clases personalizadas encargadas de instanciar los distintos algoritmos de trading, los mercados de los que vamos a obtener información y el método de trading de Wyckoff a aplicar, respectivamente.

En el caso de *WyckoffTrading*, la clase es instanciada cuando operamos en tiempo real o backtesting con el algoritmo de trading. Es por ello que esta clase se usa puramente para desarrollo y no es un modelo, a diferencia de las demás que sí son modelos de Django.

5.3. Diagramas de secuencia

En este apartado incluyo los diagramas de secuencia referentes a las distintas funcionalidades de la aplicación. Las partes del sistema referentes a vistas se han marcado con cuadros de color azul; las partes referentes a los controladores se han marcado con color azul más claro; y los modelos están señalados en gris. El objetivo de esta diferenciación es dejar claro qué cosa representa cada una de las partes del patrón de arquitectura *Modelo-Vista-Controlador*.

En la imagen 5.3 se ven los pasos realizados para iniciar la aplicación.

En la figura 5.4 se pueden ver los pasos realizados por el sistema para crear un usuario y cómo lo enriquece con los detalles específicos de la sesión.

En el diagrama 5.5 se sigue el comportamiento de la aplicación cuando el usuario se identifica o cierra sesión en la APP.

En la figura 5.6 se representan los pasos de la aplicación al identificarse y cerrar sesión en el bróker de MetaTrader5.

En la imagen 5.7 se muestran los pasos de la aplicación para las funcionalidades de visualización de datos antiguos o en tiempo real.

Por último, en la figura 5.8 se muestran los pasos de la aplicación para las funcionalidades de visualización de datos antiguos o en tiempo real.

5.4. Primer diseño de la interfaz

La página web principal mostrará un menú con distintos botones que implementan funcionalidades diferentes. Previo a este menú tendremos un formulario para identificarnos como usuarios de la aplicación. A ambos lados del menú principal y durante todas las vistas de la APP tendremos hiper-enlaces con accesos directos a distintas funcionalidades de la aplicación. A continuación muestro una serie de borradores con diseños de cada una de las secciones de la interfaz.

En este primer diseño que muestro en la siguiente imagen, se puede ver el menú principal de la aplicación.

En la imagen mostrada vemos los siguientes botones: *Login*, *Logout*, *Gestión de datos*, *Ver datos de mercados*, *Operar*, *Estrategias de trading*, *Back-testing*, *Histórico de operaciones* y *Balance y operaciones en curso*. En los puntos siguientes explico qué realizaría cada botón y con qué parte del software conectaría.

- **Login:** permite conectar a nuestra cuenta del bróker que usemos en la plataforma comercial, comercial o demo. Véase la siguiente figura.
Referencia: *login*, RF-2.1.
- **Logout:** permite cerrar la sesión iniciada anteriormente con el menú de Login. Este botón no estará habilitado hasta que hayamos iniciado sesión. **Referencia:** *login*, RF-2.1.
- **Gestión de datos:** permite el insertado y procesado de datos de mercado. El usuario podrá a través del menú correspondiente introducir datos al módulo que se encarga de gestionar la BBDD, que son reconocidos automáticamente de las bases de datos de la plataforma comercial.
Referencia: *procesado de datos*
- **Ver datos de mercados:** permite ver datos de precios de un mercado elegido. Se permite ver datos en un marco de tiempo específico, en rango y de mercados específicos. **Referencia:** *visualización de datos*, RF-3.
- **Operar:** lleva al usuario a un menú donde podrá operar o usando trading algorítmico. **Referencia:** *trading algorítmico*, RF-4.
- **Estrategias de trading:** lleva a un menú en el que se puede obtener información de cada una de las estrategias de trading así como elegir una para que sea usada en las próximas operaciones. **Referencia:** *trading algorítmico*, RF-4.

- **Backtesting:** menú similar a Operar. En este caso el usuario podrá iniciar operaciones usando trading algorítmico a partir de una fecha anterior a la actual, simulando el transcurso del tiempo. **Referencia:** *backtesting, RF-5.*
- **Histórico de operaciones:** permite al usuario ver un resumen del histórico de operaciones realizadas. **Referencia:** *información de operaciones, RF-2.3.*
- **Balance y operaciones en curso:** permite al usuario ver el capital disponible, así como un balance a tiempo real de pérdidas y ganancias y el resumen de las operaciones en curso. **Referencia:** *capital disponible, RF-2.2; información de operaciones, RF-2.3.*

En la figura 5.10 muestro lo que sería el menú de **Login**. En este menú se introducirán *Login*, *Contraseña* y *Servidor* para que nuestra aplicación inicie sesión en la cuenta demo o comercial del bróker usado. Podemos ver el diseño del menú en la figura 5.11.

Una vez estemos conectados en la cuenta proporcionada por el bróker, nuestro menú principal mostrará disponibles las funcionalidades que en la figura 5.9 no estaban habilitadas. Dichas funcionalidades dependen directamente de la plataforma comercial o bróker, por lo que necesitan de conexión a la cuenta.

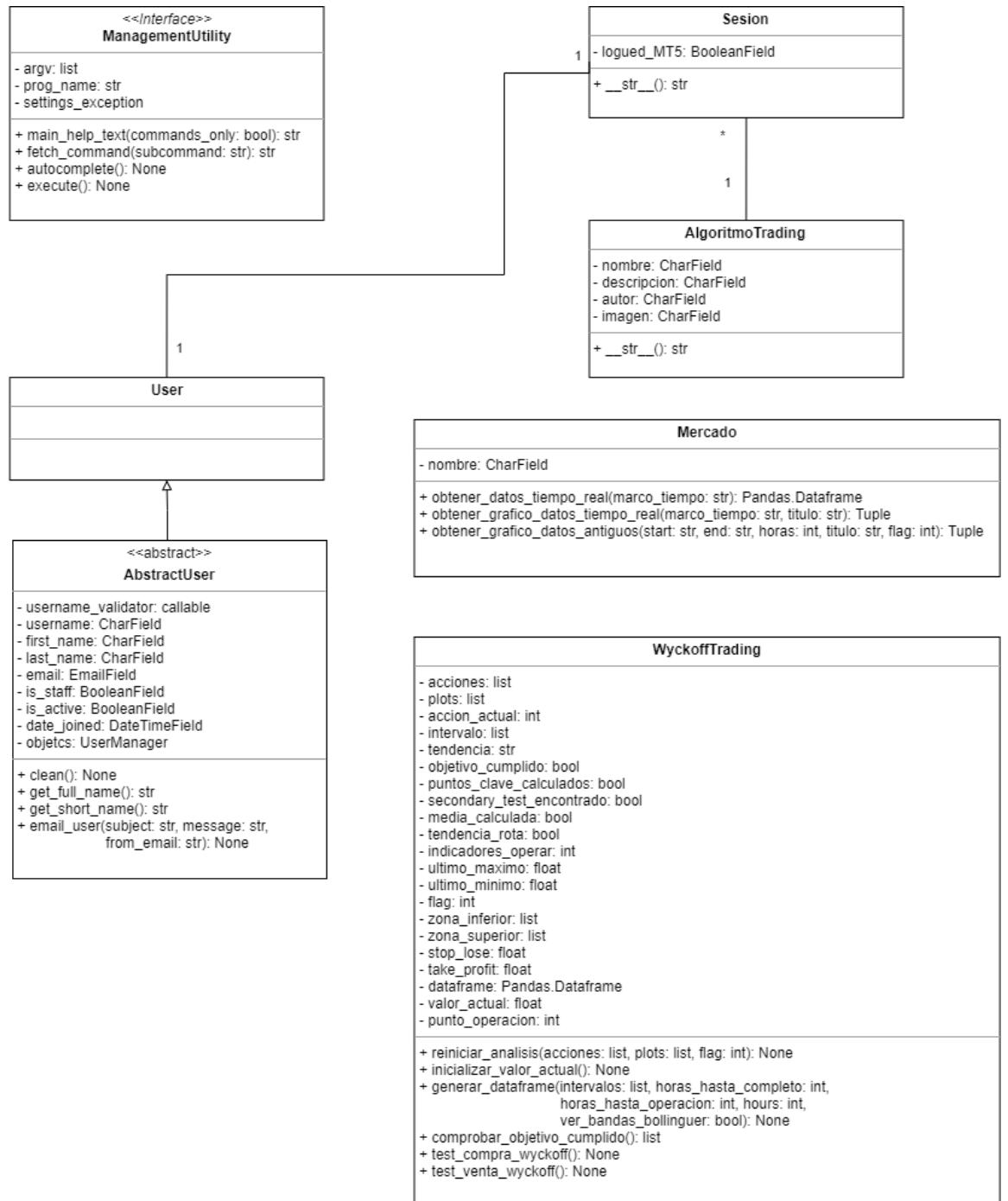


Figura 5.2: Diagrama de clases de la APP

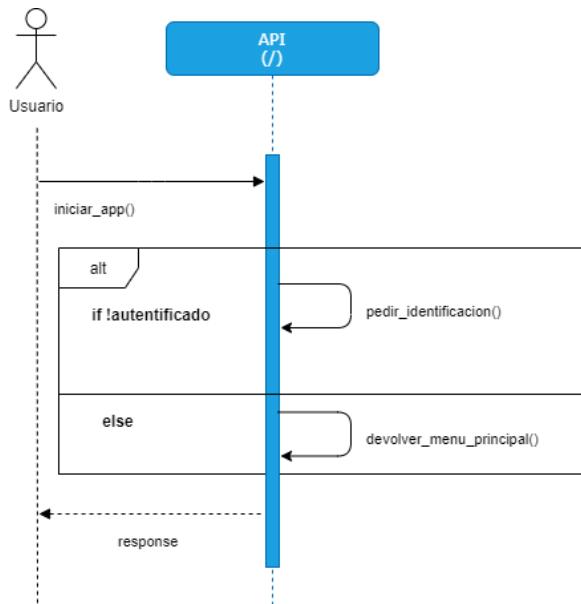


Figura 5.3: Diagrama de secuencia de la creación de usuarios

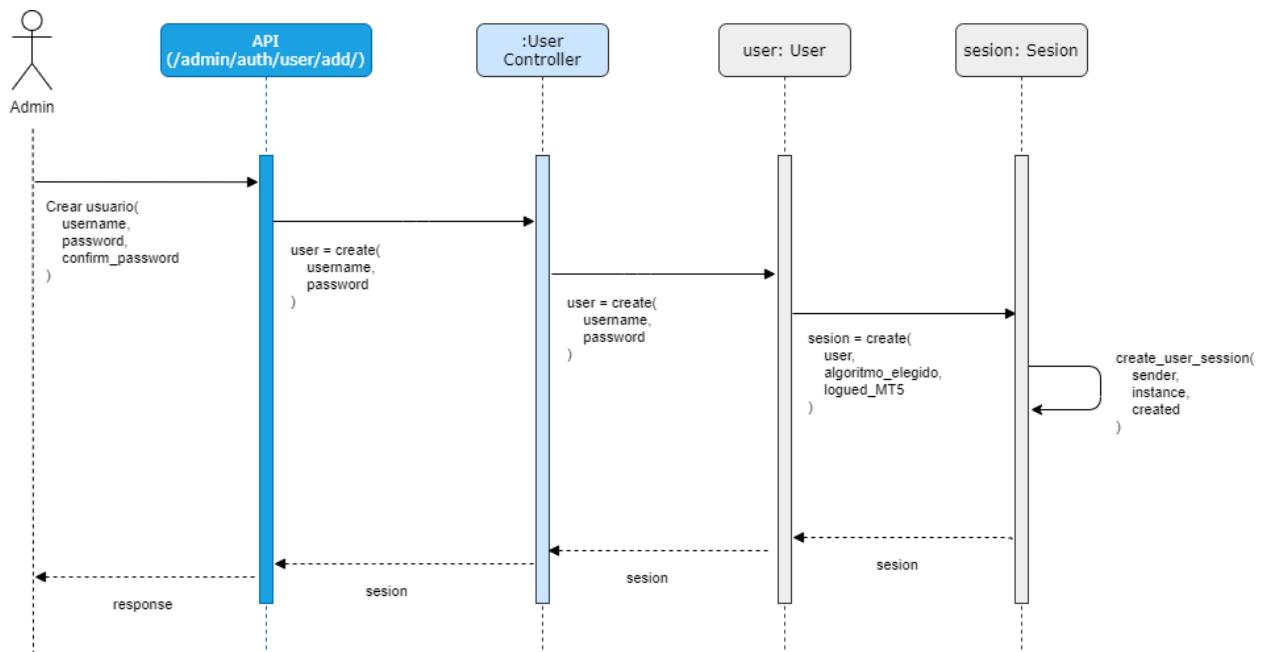


Figura 5.4: Diagrama de secuencia de la creación de usuarios

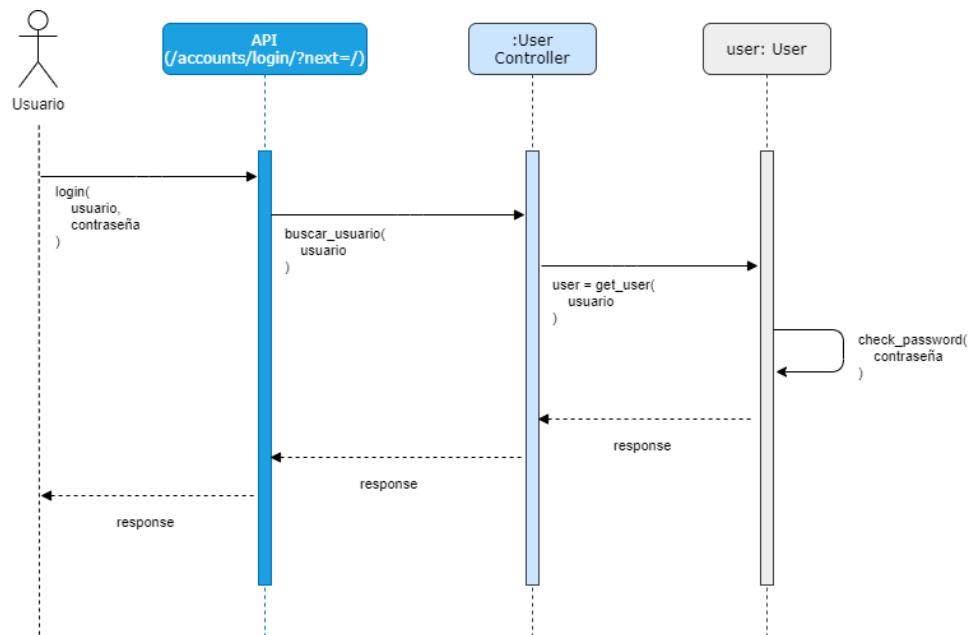


Figura 5.5: Diagrama de secuencia de login y logout de usuarios

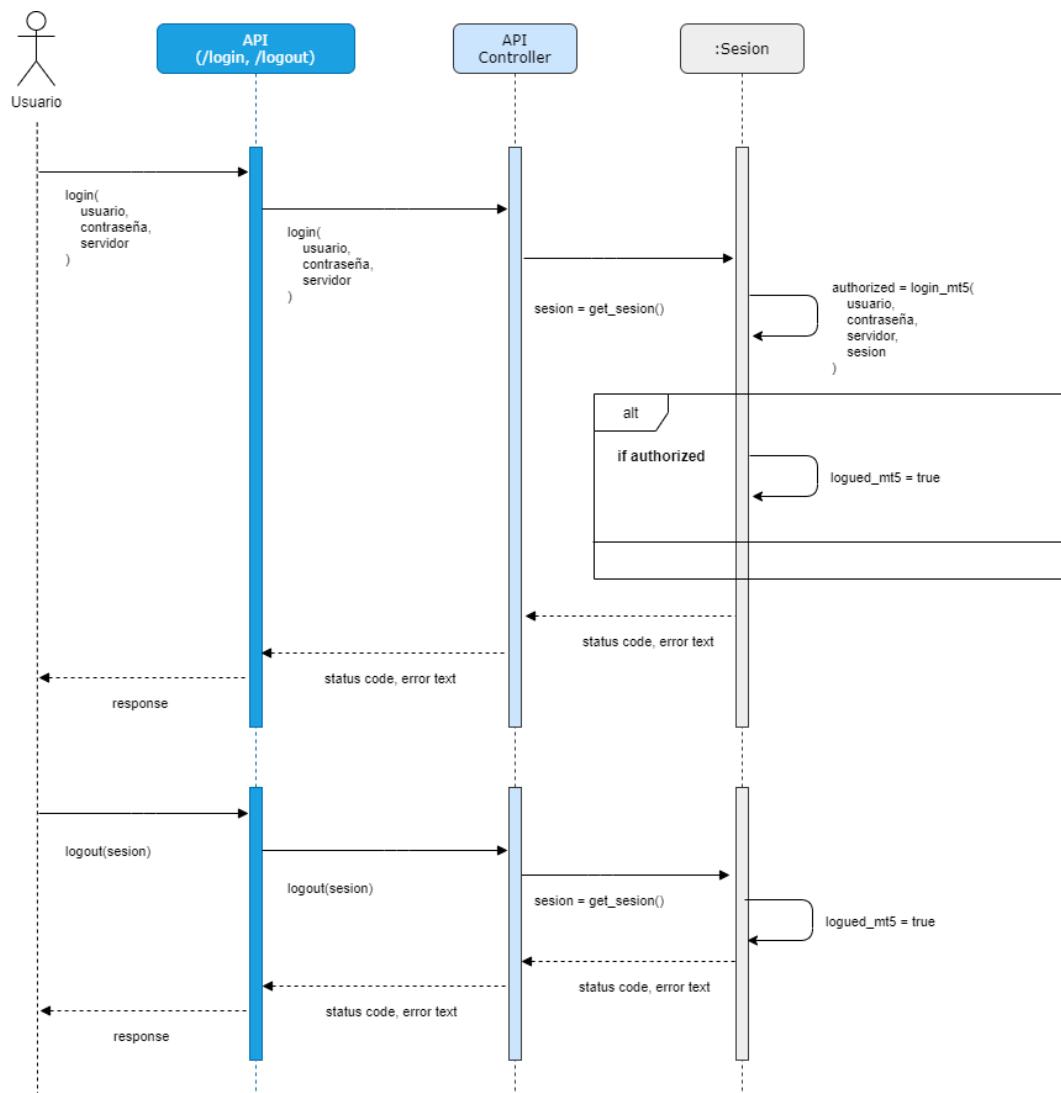


Figura 5.6: Diagrama de secuencia de login y logout de usuarios en el bróker en MetaTrader5

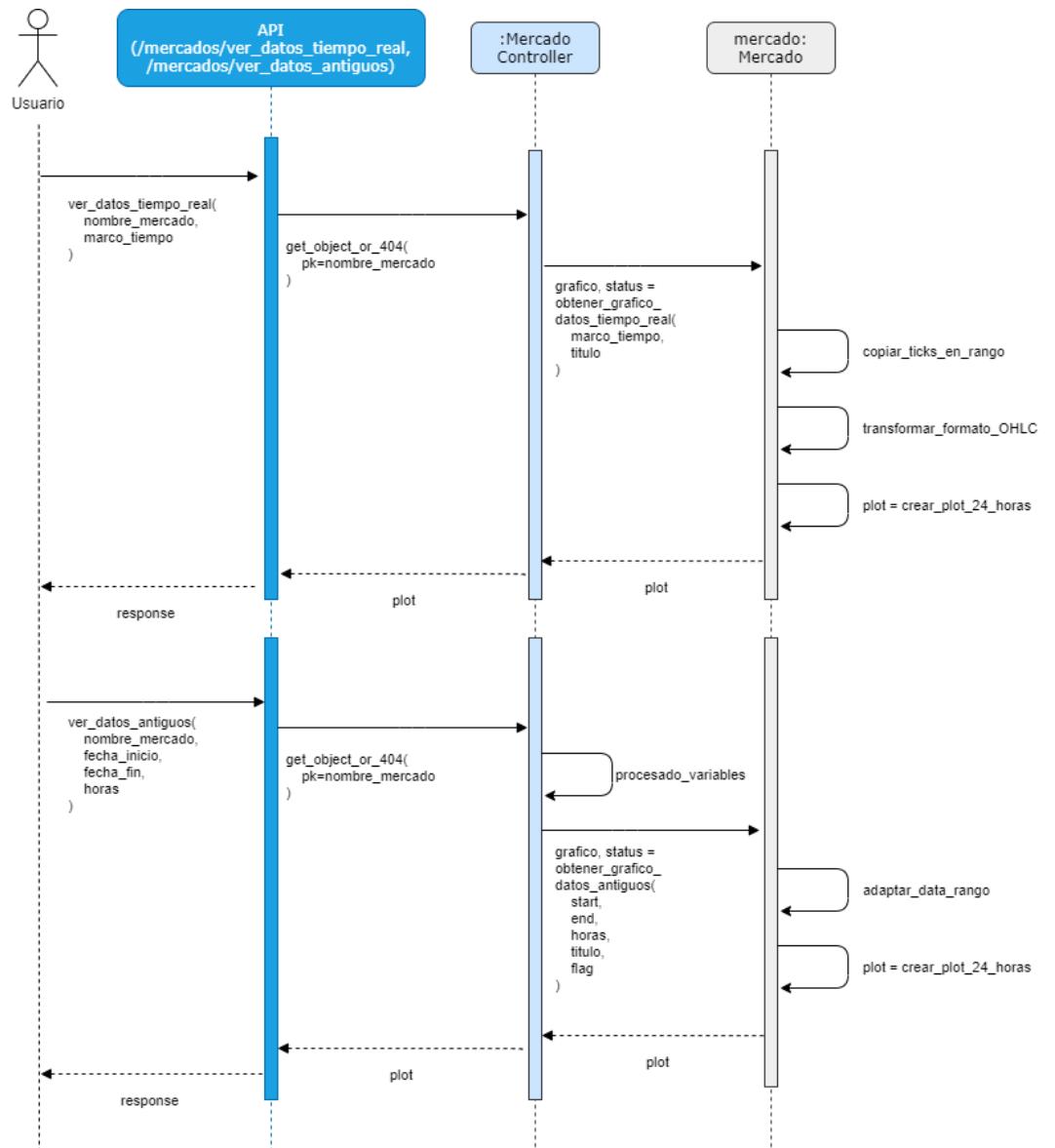


Figura 5.7: Diagrama de secuencia de visualización de datos antiguos y en tiempo real

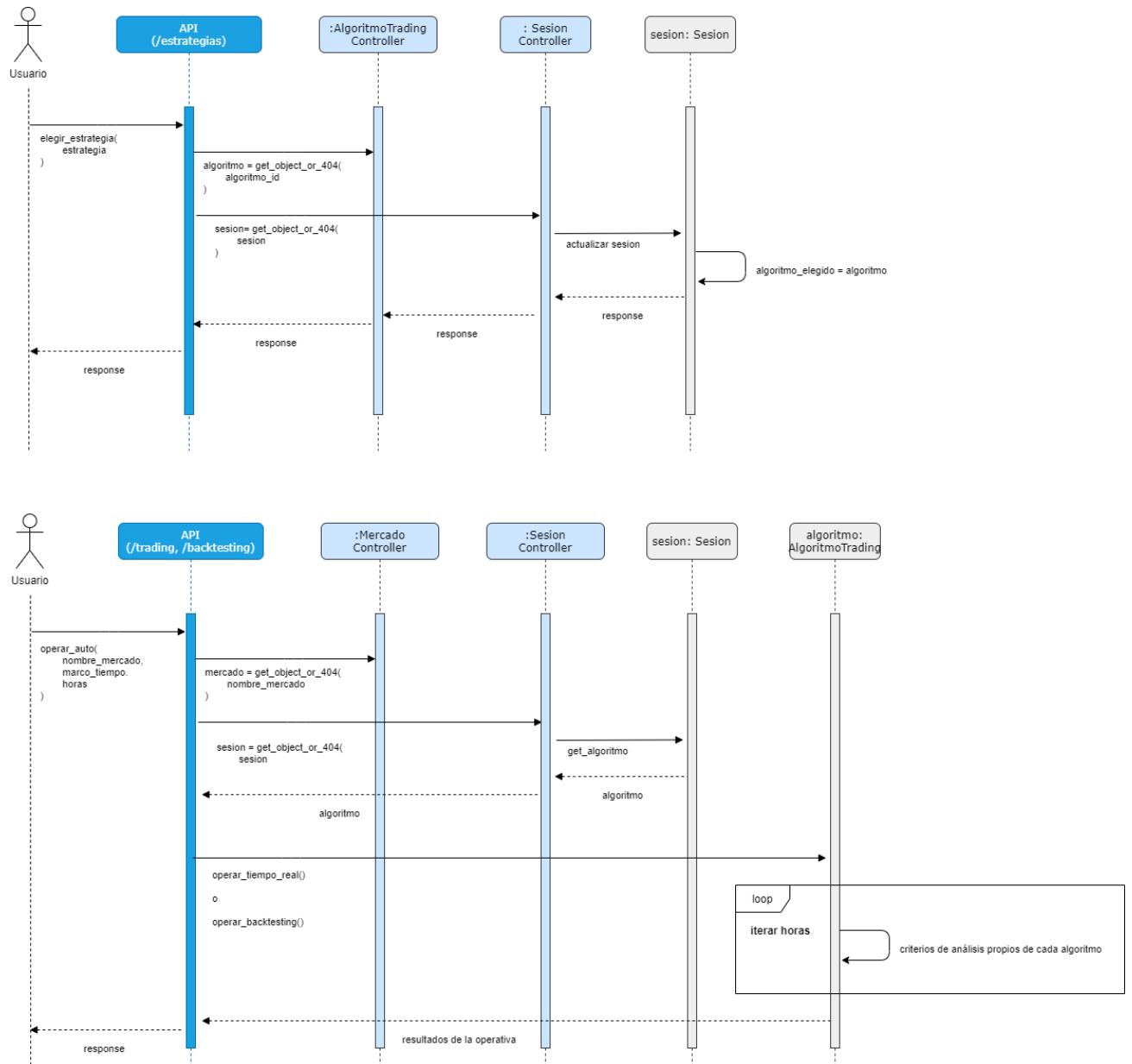


Figura 5.8: Diagrama de secuencia de elección de algoritmos y trading automático y backtesting

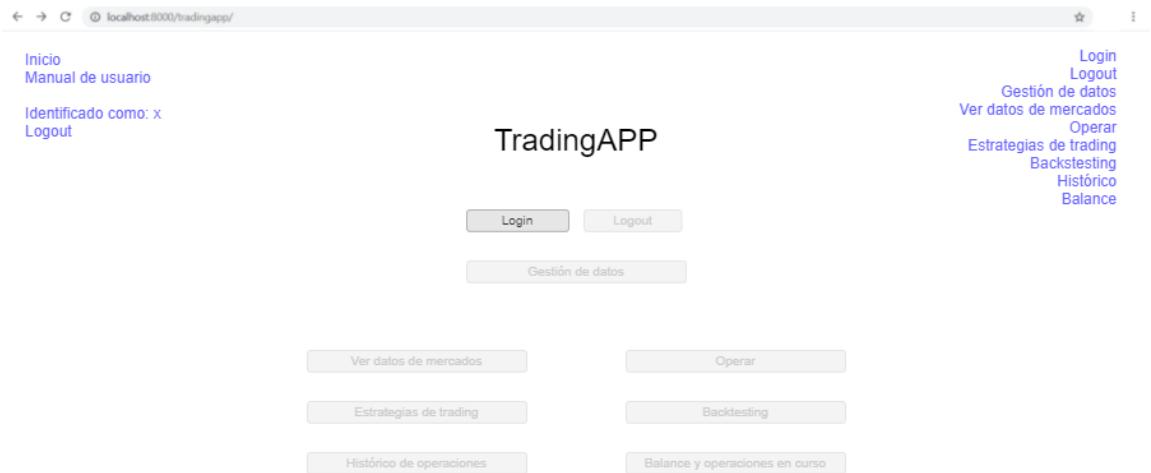


Figura 5.9: Primer diseño del menú principal de la APP

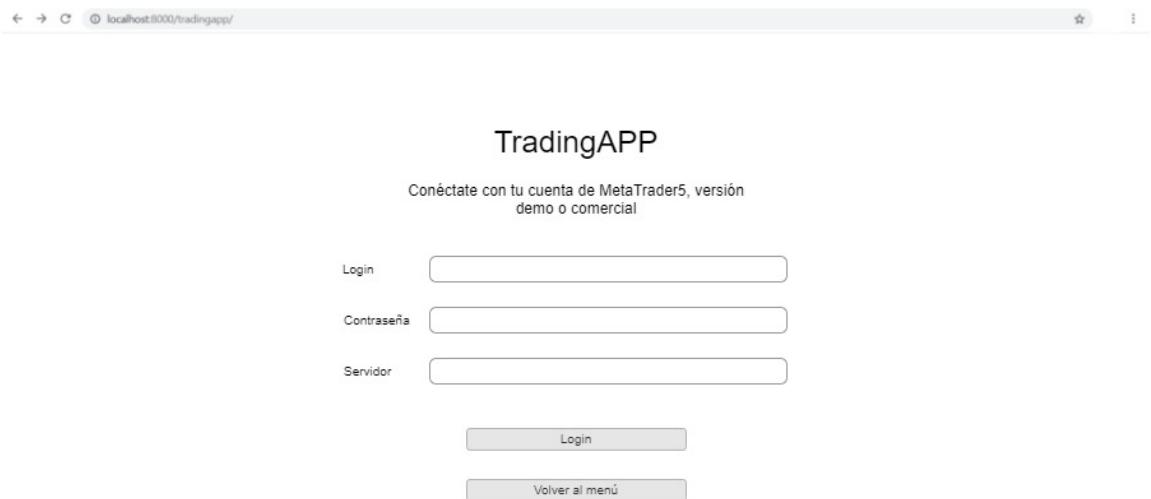


Figura 5.10: Primer diseño del formulario de login en *MT5*

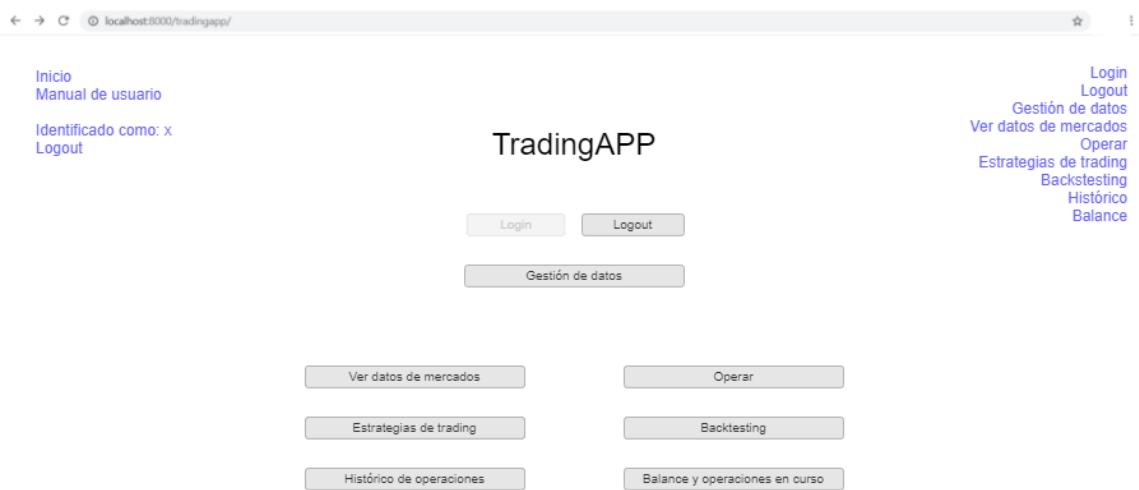


Figura 5.11: Primer diseño del menú principal (usuario identificado en *MT5*)

Capítulo 6

Implementación

6.1. Herramientas y software utilizado

En esta sección hablo de las herramientas y software utilizado en el proyecto.

El desarrollo de la aplicación web se ha hecho en el lenguaje de programación *Python*, en su versión *3.9*. Se ha usado *Django* como framework de la APP.

El entorno de desarrollo o IDE utilizado ha sido *PyCharm*. Con la cuenta institucional de la universidad tenemos un año gratis de *PyCharm Professional*. Esta licencia nos aporta ventajas a la hora de desarrollar en *Python*, como el *debugger* remoto, útil para debugar código *Python* mapeado a directorios dentro de contenedores *Docker*, por ejemplo.

Se ha usado *git* como controlador de versiones y concretamente, he usado la app *GitHub Desktop*. Esta aplicación aporta una interfaz gráfica de *GitHub* para facilitar la subida de commits, archivos al stage, ramas, etc. También permite ver las diferencias de un mismo archivo en distintas ramas o crear *Pull Requests*, entre otras cosas.

La plataforma de trading usada para el desarrollo ha sido *MetaTrader5*, como ya se ha especificado en secciones anteriores del documento. Se ha elegido esta plataforma ya que dispone de una integración o librería en *Python*. A través de esta librería podemos obtener datos de mercados financieros y conectarnos a cualquiera de los brokers que trabajan con *MetaTrader5* (véase punto 2.1.1). Como mencionan en su página, el paquete *MetaTrader* para *Python* está diseñado para obtención de datos directamente del terminal de *MetaTrader5*. Dichos datos pueden ser usados para cálculos estadísticos y aprendizaje automático. Además, la librería permite mandar órdenes de



Figura 6.1: Logos de las principales herramientas usadas en el proyecto.

compra y venta directamente desde el propio código.

Debido a que esta dependencia sólo está disponible para Windows, *Windows 10 Home* ha sido el sistema operativo elegido para el desarrollo del proyecto. Para usar Django en Windows, es necesario instalar *Visual Studio C++* en su versión *14.0* o superior.

Para la interfaz gráfica del proyecto se utiliza el lenguaje de marcas *HTML* y para aplicar estilo en estas vistas se utiliza *CSS*. Utilizo también *Javascript* para añadir scripts y lógica adicional en las vistas.

Finalmente, para gestión de bases de datos, se utiliza *SQLite*, en su versión *3*. Este es el sistema gestor de bases de datos de Django por defecto.

Para el desarrollo de este documento, se utiliza *LaTex* y más concretamente, el IDE *TexStudio*.

6.2. Metodología de trabajo

El trabajo se desarrolla siguiendo la metodología de desarrollo ágil *scrum*. Este tipo de planificación hace que las distintas tareas que se desarrollan se puntúen siguiendo un modelo de puntuación basado en dificultad. En dicha planificación, las tareas se reparten en *sprints* o iteraciones. Una iteración puede durar entre 2 y 4 semanas. En el caso de mi proyecto, los sprints han sido períodos variables de entre 2 y 3 semanas. El uso de scrum y cada uno de los sprints en los que divido el proyecto se mencionan más detalladamente en el punto 6.3 de esta memoria.

Como controlador de versiones hemos utilizado *git*, y la plataforma *GitHub*. Para la correcta y eficiente realización del proyecto, ha sido esencial *GitHub*, ya que es una plataforma que ayuda a llevar una buena estructuración del trabajo gracias a cosas como las *issues*, los *pull request* y sus respectivas revisiones; el trabajo con ramas o *branches*, etc.

El proyecto se ha desarrollado en su totalidad en el repositorio de *GitHub*:
<https://github.com/mcarmona99/TFG/>

6.2.1. Issues y Pull Requests

Cada tarea se ha representado mediante *issues*. Una *issue* es una descripción de la tarea donde se pueden incluir a miembros del grupo que estén trabajando en la misma, errores específicos de la tarea, versiones del proyecto donde hay que realizarla o aparece el error, etc.

Cada issue suele ser referenciada por un *pull request*. Los pull request son otro tipo de herramienta que sirven para incluir el código realizado en la tarea referenciada al proyecto en producción. Los *PR* explican el proceso y/o el código que se ha introducido para arreglar o implementar la tarea que se pedía. La tarea puede ser arreglar un bug, implementar nuevas funcionalidades, etc. Cada PR debe ser revisado por otro miembro del equipo antes de ser aprobado. En el PR también podemos encontrar información útil como la rama de desarrollo que se estaba usando o los commits realizados en la misma.

En el caso de este proyecto, el seguimiento y revisión de Issues y PRs ha sido hecho por el mismo desarrollador, debido a que es un trabajo de una sola persona.

En la figura 6.2 y 6.3 se pueden ver un ejemplo de *Issue* y su *Pull Request*, respectivamente.

Implementar estrategia de trading de Wyckoff #5

Closed mcarmona99 opened this issue on 10 Jun · 0 comments

Descripción

Esta issue detalla los pasos para implementar la primera estrategia de trading algorítmico de la aplicación. Se tratará de la estrategia o análisis de Richard Wyckoff. Deberá funcionar de manera similar a la estrategia base detallada en #3, es decir, mismos inputs y outputs. Cuando usamos una estrategia, se podrá operar o hacer backtesting con dicha estrategia de trading automático.

To Do

Investigar acerca de esta estrategia. Ver "apuntes TFG", donde ya se tiene una investigación detallada con información específica de análisis y ejemplos sobre gráficas. Implementar la estrategia

- mcarmona99 mentioned this issue on 17 Jun
 - Procesado de datos para obtener velas en formato OHLC #10 **Closed**
 - Añadido script para obtener datos y procesarlos en velas por hora #11 **Merged**
 - mcarmona99 self-assigned this on 17 Jun
 - mcarmona99 mentioned this issue on 20 Jul
 - Añadida estrategia de Wyckoff para backtesting #12 **Merged**
 - mcarmona99 closed this in #12 on 20 Jul

Assignees mcarmona99

Labels None yet

Projects None yet

Milestone None milestone

Linked pull requests Successfully merging a pull request may close this issue.

Notifications Customize [Unsubscribe](#)
You're receiving notifications because you're watching this repository.

1 participant

[Lock conversation](#) [Pin issue](#) [Transfer issue](#) [Delete issue](#)

Figura 6.2: Issue número 5, implementación de la estrategia de Wyckoff

6.2.2. Paralelismo usando GitFlow

GitFlow es un flujo de trabajo basado en *git* que brinda un mayor control y organización en el proceso de integración continua.

El flujo de trabajo con GitFlow se basa principalmente en dos ramas; la rama *develop* y la rama *master*. La rama *develop* es aquella rama donde convergen las ramas de desarrollo. La rama *master*, en cambio, es la rama principal y estable (*develop* también es considerada estable). Esto puede variar en los desarrollos. En mi caso, *main* es la rama que haría de *develop* y a la que se incluyen los desarrollos.

Los nuevos desarrollos parten de la rama *develop* y se mergean en ella misma una vez se han revisado. Dichos nuevos desarrollos son realizados en ramas que tienen nomenclaturas similares a:

- **feature/#issue-descripción-desarrollo:** rama que introduce una

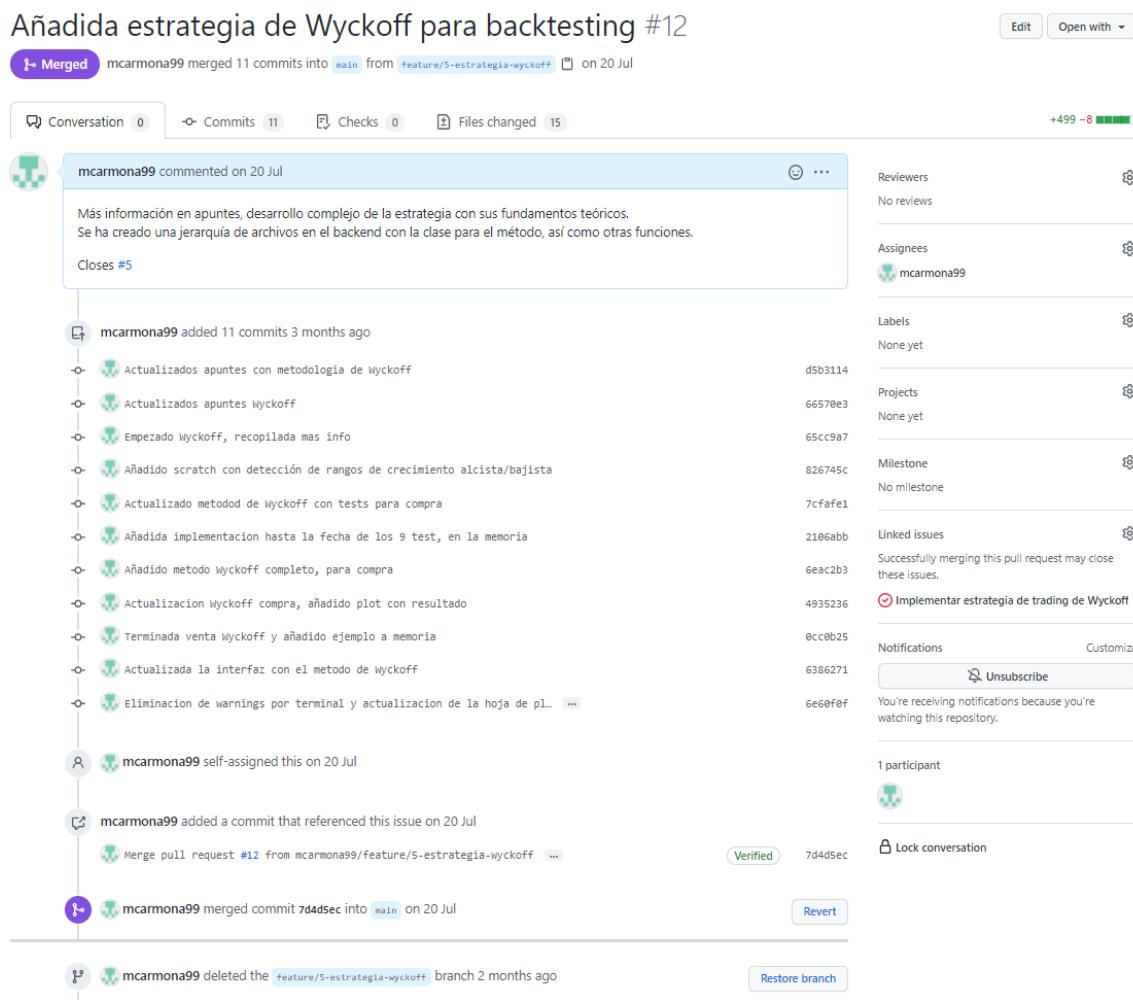


Figura 6.3: *Pull Request* referente a la issue 6.2

nueva funcionalidad.

- **fix/#issue-descripcion-arreglo:** rama que introduce un fix a un bug o error conocido. En ocasiones tenemos hotfix o bugfix en lugar de fix.
- **support/#issue-descripcion:** para soporte de pago o soporte a usuarios.
- **release/#issue-descripcion:** para ramas de release.

Cuando estos desarrollos han terminado, como hemos dicho anteriormente, sus ramas correspondientes son mergeadas a develop, por medio de un Pull Request. Lo normal es que cada PR esté asignado a una issue y a una

rama.

Esto no ha sido de mucha utilidad en el caso de este proyecto, ya que no ha sido un trabajo en el grupo con lo que las ventajas de *GitFlow* no se han visto en gran medida.

6.3. Fases del desarrollo

En esta sección se detallan los *sprints* seguidos en el desarrollo del proyecto. Como se ha mencionado en el punto anterior, los sprints son iteraciones del desarrollo del producto que incluyen un número de tareas a realizar. Los sprints son vitales para llevar a cabo un seguimiento del trabajo hecho hasta el momento.

En el caso de mi trabajo, las fases de desarrollo se han realizado en 6 sprints o iteraciones. Cada uno de estos sprints han durado un período de entre 2 y 4 semanas.

Para llevar una correcta planificación del trabajo, los sprints e issues o tareas incluidas en cada uno de ellos han sido añadidos a una tabla u hoja de planificación. Los cambios de sprints coinciden con las entrevistas realizadas con el tutor del proyecto, ya que han servido para entrega de tareas realizadas y comentarios; e inicio de período con nuevas tareas.

6.3.1. Guía de puntuación scrum y prioridades

En la hoja de planificación he incluido una guía de puntuación de scrum que intenta añadir una equivalencia en horas a cada una de las puntuaciones disponibles en scrum para issues. **En la teoría de scrum esto se debe evitar**, ya que la puntuación de scrum es una medida de esfuerzo en el trabajo, y no de tiempo invertido. A pesar de esto, he decidido crear la equivalencia para tener un seguimiento más preciso de lo que se ha tardado en cada issue o tarea.

En la figura 6.4 se encuentra esta guía, cuando menciono días, supongo la equivalencia de un día igual a una jornada de trabajo completa, es decir, 8 horas.

Además de esta guía, en la hoja he añadido una guía sobre las prioridades que he asignado a cada issue dentro de un mismo sprint. Estas prioridades indican lo más o menos importante o vital que se considera una issue con

| Guía puntuación SCRUM | |
|-----------------------|--------------|
| Puntos | Equivalencia |
| 1 | 3 horas |
| 2 | 6 horas |
| 3 | 1 día |
| 5 | 2 días |
| 8 | 3 días |
| 13 | 5 días |
| 20 | 7 días |

Figura 6.4: Guía de puntuación scrum

respecto al desarrollo del proyecto.

Los niveles son, en orden de menor a mayor prioridad:

- **Muy baja:** funcionalidad que suele ser de diseño de la interfaz y que incluye cambios al funcionamiento principal de la aplicación. Menor prioridad.
- **Baja:** funcionalidades de diseño de la interfaz que sí que afectan aunque en poca medida al uso de la APP por parte del usuario final.
- **Media:** funcionalidad que necesita ser desarrollada pero con menor prioridad que otras que forman parte del desarrollo principal de la APP.
- **Alta:** funcionalidades pertenecientes al desarrollo principal de la aplicación.
- **Muy alta:** funcionalidades cuyo desarrollo es vital o *stopper* para otros desarrollos. Mayor prioridad posible.

6.3.2. Fase previa a los sprints de desarrollo

En este apartado hablaré a grandes rasgos de algunas de las pruebas y desarrollos que hice previos a los inicios de lo que es el desarrollo principal de la APP y por tanto, de los sprints.

En primer lugar, se invirtió tiempo desarrollando algunos scripts de Python para conectar con interfaces externas de Trading. En este punto se investigó sobre Binance y su librería para Python *python-binance*.

Fue en esta etapa cuando empecé a usar la integración de MetaTrader5 en Python, siguiendo los tutoriales oficiales de la web: Módulo MetaTrader para la integración con Python . Además de esto, estuve leyendo sobre el lenguaje de *MQL5*, que es el proporcionado por MetaTrader y su integración por Python. Este proceso es más complejo y busca conectar procesos en sockets de red, para conectar una aplicación con el terminal de MetaTrader. Debido a la facilidad de uso, me decanté por la primera opción, usar la librería de *MetaTrader* en Python.

En segundo lugar, se investigó sobre Django. En este período desarrollé los tutoriales ofrecidos en la página oficial de Django y diferencie los conceptos de proyecto y aplicación, migraciones, estructura de la base de datos, etc.

En una tercera fase previa al desarrollo, estuve investigando cómo obtener datos históricos de los distintos mercados financieros soportados por MetaTrader, tal y como se acordó con el tutor del proyecto. Estos datos históricos serían los inputs de cada uno de los algoritmos para trading desarrollados. Los datos se sacaron con script que conectaban al terminal de MT5.

En esta investigación conseguí sacar 1 o 2 años de datos de cada uno de los mercados de MetaTrader5, dependiendo de la disponibilidad. En total, conseguí obtener datos de 179 mercados. MetaTrader5 dispone de 570 mercados en total, pero la librería de Python solo soporta datos de esos 179 mercados. **Problema presentado:** a pesar de que los datos en sí no consumían una gran cantidad de memoria, MetaTrader5 cacheaba todos los mercados y los monitorizaba al haber cogido datos de los mismos. Es por esto que estuve teniendo problemas de capacidad de disco. A raíz de esto, se decidió hacer un cambio en el planteamiento y **optamos por tomar el máximo numero de años de datos posibles de 5 mercados financieros de distinta naturaleza disponibles en MT5**. Esto dio lugar a tomar 1 mercado de Forex, 2 mercados de metales; y 2 mercados de energías. Dichos mercados serían el *Euro vs Dólar*, *Euro vs Plata*, *Dólar vs Oro*, *Petróleo Brent vs Dólar* y *Gas Natural vs Dólar*.

Los datos obtenidos se pueden observar en la figura 6.5. Como se puede ver, se sacan 10 años de *EURUSD* (euro-dólar); y 5 años de *XBRUSD* (brent-dólar), *XNGUSD* (gas natural-dólar), *XAGEUR* (plata-euro) y *XAUUSD* (oro-dólar). Estos datos incluyen precios y otro tipo de información otorgado por MT5.



Figura 6.5: Datos de mercados obtenidos en las fases previas al desarrollo

6.3.3. Sprint 1: 8 junio - 22 junio. Interfaces y modelos básicos

En esta sección hablaré de las issues y demás tareas realizadas en el sprint 1. Este sprint comprende desde el 8 de junio hasta el 22 de junio de 2021.

En este sprint se realizaron las siguientes cuatro issues: #1, #3 #4, #7 y #10.

Issue 1: Creación de la interfaz principal de la APP

Esta issue es la inicial y fue catalogada con prioridad Muy alta. Como se indica en la guía de prioridades del apartado anterior, este nivel equivale a issues vitales o stoppers. En este caso, la issue detalla la primera tarea. Esta tarea es la creación de la interfaz principal de la aplicación siguiendo el diseño propuesto en la memoria del proyecto.

El diseño que menciono es aquel comentado en el punto 5.4, donde se expone el primer diseño de la interfaz.

La issue no incluye investigaciones sobre Django ya que eso se realiza en lo que hemos llamado como *Fases previas al desarrollo* y que se puede encontrar en el punto de la planificación y en el diagrama de Gantt del proyecto. En otras palabras, en esta issue se monta el esquema de la interfaz de la aplicación web en Django conociendo ya el software y cómo proseguir.

En el [pull request](#) conectado a esta issue encontramos los cambios añadidos y la interfaz creada. En dicho PR se comenzó el proyecto de *Django* y se creó la primera APP llamada *Interface*. Se desarrolló por tanto la interfaz o menú principal en una vista que se obtiene al pedir la request a /, es decir, primera página de la web. Las vistas se han creado usando *HTML* y estilos de *CSS*. Se crean también los botones del diseño sin funcionalidad.

Issue 7: Creación de la interfaz para elección de estrategias

En esta issue se pide crear la interfaz y modelos básicos para poder ver desde el botón *Estrategias de Trading* cada una de las estrategias y poder elegirlas, creando la lógica de modelos que se necesite para asignar la estrategia a la sesión actual.

Para la resolución de esta issue se crea una interfaz a la que se accede con el botón *Estrategias de trading*. Para ello se ha necesitado un modelo llamado *Sesion*. La clase *Sesion* solo tendrá en primer lugar 1 instancia y será equivalente a la sesión actual (hasta que se implemente la gestión de usuarios, cada usuario tendría la misma sesión). La sesión contiene la estrategia de trading seleccionada.

Además se crea otro modelo llamado *AlgoritmoTrading*, que representará cada uno de los algoritmos a elegir.

Se crea la relación:

Sesion — tiene → Algoritmo Trading

Estos modelos se pueden ver en el diagrama de clases del proyecto.

Esta issue y por tanto la creación de la lógica de modelos mencionada es completada con el PR #8.

Issues 4 y 3: Backtesting con estrategia y estrategia de trading algorítmico base

En la issue 4 se debe implementar la funcionalidad de backtesting. Esta funcionalidad se explica en el requisito funcional número 5.

Para esta issue era necesario el desarrollo previo de un modelo básico para realizar testing, con lo que esta issue se fusiona con la issue #3. La estrategia propuesta como base será la estrategia de medias móviles. La issue se cierra con el PR #9.

En primer lugar, para la implementación de la estrategia de medias móviles, issue #3:

Podemos empezar comentando el por qué usar una media móvil. Las medias móviles nos ayudan a eliminar el ruido en un gráfico de precios. Dichas medias pueden funcionar como soportes o resistencias (véase contexto teórico), es decir, el precio debería de ir por encima. Esto significa que las medias se comportarían como suelos, si el precio las pasa, debería de seguir bajando.

Generalmente, si el precio está por encima de una media móvil, el precio tenderá a caer; y si está por debajo, tenderá a subir. Esto obviamente no siempre ocurre. Además, el comportamiento dependerá de la ventana que estemos utilizando.

Tipos de media móvil:

- SMA (simple moving average). Media móvil simple. Un ejemplo sería una media móvil con ventana de 5 días. Dicho valor calcularía la media en un rango de 5 días.
- EMA (exponencial moving average). Media móvil exponencial. Esta media da más peso a los precios más recientes. Usa cálculos más complejos. La EMA reacciona más rápido a los cambios del precio que la SMA debido a los pesos en los cálculos.

Las ventanas del moving average son los time frame o marcos de tiempo con los que se genera la media. Una ventana de 100 días lleva una media móvil de 100 días vista. En cuanto a los tamaños de ventana, con ventanas más pequeñas la media reaccionará mucho más rápido a los cambios del precio que una media con mayor ventana. En la figura 6.6 podemos ver una ventana de 100 días (azul claro) contra una de 30 días (azul oscuro).



Figura 6.6: Media móvil con ventana de 100 días (azul claro) contra ventana de 30 días (azul oscuro). Fuente: investopedia.com/

En teoría, las ventanas más grandes benefician más a los traders que operan a largo plazo y las medias con ventanas más pequeñas ayudan más a los de corto plazo.

A la hora del desarrollo del modelo, se ha tenido en cuenta las señales de compra o venta marcadas por dos medias móviles. Cuando el precio cruza por encima o por debajo de la media móvil señala un cambio potencial de tendencia. Al usar dos medias SMA, una con una ventana más grande y otra más pequeña, tenemos el siguiente comportamiento:

- Cuando la MA a corto plazo supera a la de largo plazo → señal de compra (golden cross)
- Cuando la MA a largo plazo supera a la de corto plazo → señal de venta (dead/death cross)

Finalmente, para la implementación del backtesting con estrategia, issue #4:

Añado un atributo al modelo *Sesion* llamado *logged_MT5* para gestión de botones, ver qué botones disponibles y cuáles no en la interfaz. Por ahora será por defecto *True* para la sesión, hasta que se implemente el login y logout en MT5.

Creo la interfaz inicial para realizar backtesting.

Para operar, creo el modelo *Mercado*, que representa la divisa/mercado financiero en el que se opera, tendrá un nombre y una función para obtener el gráfico (referencia a issue de ver datos de mercado entre rango, aún no realizada).

Creo objetos para el nuevo modelo Mercado, cada uno de ellos referentes a los 5 mercados en los que estamos trabajando.

```

1 >>> from Interface.models import Mercado
2 >>> m = Mercado(nombre='XBRUSD')
3 >>> m.save()
4 >>> m2=Mercado(nombre='XNGUSD')
5 >>> m2.save()
6 >>> m3=Mercado(nombre='XAGEUR')
7 >>> m3.save()
8 >>> m4=Mercado(nombre='XAUUSD')
9 >>> m4.save()
10 >>> m5=Mercado(nombre='EURUSD')
11 >>> m5.save()
```

Figura 6.7: Creación de objetos para el modelo *Mercado*

He editado toda la estructura de carpetas de mi proyecto de Django, para usar códigos comunes a los algoritmos, los algoritmos en sí, y hasta que se implemente la gestión de datos, gestión de datos en local. La estructura mencionada se puede ver en la figura 6.8.

Issue 10: Procesado de datos para formato OHLC

En este apartado se describe la issue 10 y su desarrollo.

Esta issue explica la necesidad de desarrollar un procesado de datos para tener los precios en formato OHLC (Open High Low Close).

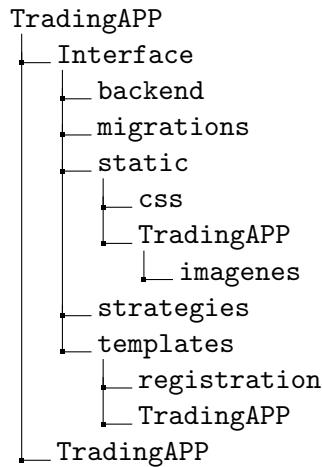


Figura 6.8: Árbol de directorios del proyecto

En el PR #11 se añaden los cambios necesarios para añadir este preprocesado de datos obtenidos de MT5. Para generar los datasets con velas y las columnas de Open, High, Low y Close, se sigue la siguiente guía en la que lo explica cómo hacerlo con pandas: Converting Tick-By-Tick Data To OHLC Data Using Pandas Resample.

6.3.4. Sprint 2: 22 junio - 13 julio. Desarrollo del método de Wyckoff

En este segundo sprint sólo se realiza una issue. La issue consiste en desarrollar la estrategia de trading de Wyckoff o *método de Wyckoff*. La issue es la número #5; y es cerrada por el PR #12.

Issue 5: Implementación del método de Wyckoff

Como se ha mencionado antes, esta issue detalla los pasos para implementar la primera estrategia de trading algorítmico de la aplicación.

Se tratará de la estrategia o análisis de *Richard Wyckoff*.

Deberá funcionar de manera similar a la estrategia base detallada en la issue #3, es decir, mismos inputs y outputs. Cuando usamos una estrategia, se podrá operar o hacer backtesting con dicha estrategia de trading automático.

Para la implementación, nos basaremos en el análisis de Wyckoff, descrito minuciosamente en el punto 2.1.5.

A partir de la teoría de Wyckoff podemos seguir 8 tests de compra y 8 tests de venta, dependiendo de si estamos en una fase de acumulación o de distribución, respectivamente.

Tests de compra para una acumulación vs Tests de venta para una distribución:

1. El objetivo bajista se ha cumplido vs El objetivo alcista se ha cumplido
2. Vemos un PS, SC, AR y STs vs Vemos un PSY, BC, STs.
3. Actividad alcista vs Actividad bajista
4. Tendencia bajista se ha roto (oferta de tendencia bajista rota) vs Tendencia alcista rota (línea de soporte ha sido rota)
5. Mínimos más altos vs Mínimos más bajos
6. Máximos más altos vs Máximos más bajos
7. La acción del precio es más fuerte que el mercado vs La acción es más débil que el mercado
8. Se crea un suelo o soporte (línea de precios horizontal) vs Se crea un techo o resistencia (línea de precios horizontal)

Si estos tests se cumplen, ya sea para compra o para venta, podemos incluir nuestra operación con un punto de parada de beneficios 3 veces superior al de pérdidas. Esto es lo que se conoce como *Take Profit* y *Stop Lose*, respectivamente. En un ejemplo práctico, si metemos una compra cuando una acción cuesta 100 €, la operativa de Wyckoff nos dice que el TP estará en 115 € y el SL en 95 €.

En cuanto a la implementación en sí, comencé buscando una forma de investigar cómo reconocer tendencias dentro de un gráfico. Estuve viendo indicadores de análisis técnico. Finalmente decidí implementar una solución que usa *bandas de Bollinger* para detección de intervalos de crecimiento y decrecimiento dentro del gráfico de precios. En otras palabras, detección de tendencias.

Las bandas de Bollinger son un indicador de volatilidad del mercado y proporcionan información de continuidad de tendencia, períodos de consolidación, períodos de amplia volatilidad y posibles máximos y mínimos del

mercado.

Dentro de lo que aportan las bandas de Bollinger, me interesa la detección de tendencia y consolidación de mercados, que viene a ser los períodos de acumulación y distribución.

El inicio de la implementación del algoritmo consiste en dibujar las bandas en un gráfico para ver cómo se ven y su comportamiento. En el gráfico de la figura 6.9 se puede observar las bandas (líneas naranja, verde y roja) y el precio (línea azul).

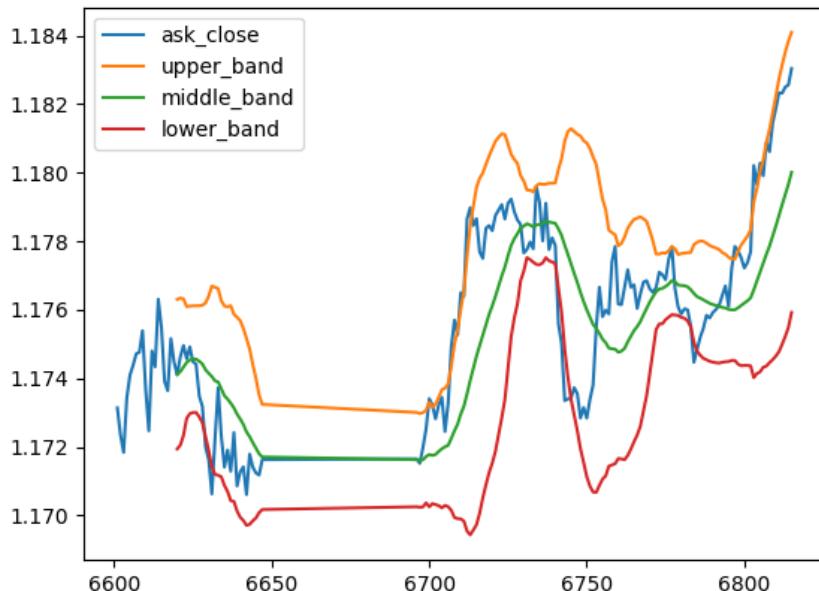


Figura 6.9: Bandas de Bollinger en un gráfico de precios

En el algoritmo de detección de tendencias que desarrollo, lo que hago es hacer un cálculo de la diferencia de bandas en cada punto de la gráfica. En la figura 6.10 se puede ver en primer lugar las bandas de Bollinger que hemos visto en la figura anterior como naranja y roja, esta vez de colores azul y naranja, respectivamente. A la derecha de dicha imagen vemos una gráfica que muestra en el eje y la diferencia de valores de cada banda de Bollinger en función del valor de x (que viene a ser un simple índice).

Como se puede apreciar, cuando tenemos tendencias, las diferencias de

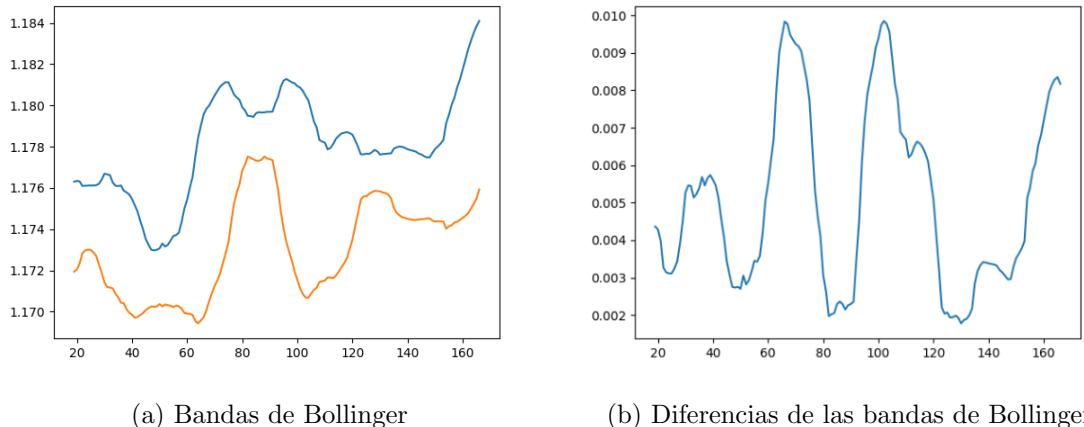


Figura 6.10: Gráficos que representan bandas de Bollinger y diferencia de sus valores

valores en las bandas son mucho más altas que cuando tenemos un período de consolidación: acumulación o distribución.

Aprovechando esto, el algoritmo detectará una tendencia cuando vea un intervalo en el gráfico de precios que las diferencias de las bandas sean consideradas **anomalías** dentro de los datos. Para que esto sea posible, considero anomalías a aquellos valores que superen la media de valores representados en la gráfica.

Al implementar esto, obtengo los resultados que se pueden ver en la figura 6.11. Los resultados vienen a ser intervalos representados con franjas de color azul en el gráfico. Cabe señalar que estas franjas se detectan en tiempo real, por lo que se han hecho simulando el paso del tiempo y generando dichos intervalos. Como podemos apreciar, vemos que se ha detectado un intervalo o tendencia bajista seguida de una acumulación; una tendencia alcista seguida de una distribución; y una tendencia bajista seguida de otra acumulación.

Una vez tenemos este algoritmo implementado, procedemos a implementar los 8 tests de compra o venta descritos anteriormente en este mismo apartado. La implementación es la misma tanto para compra como para venta, la única diferencia es cambios de signos y otros cambios menores.

- **Objetivo bajista / alcista cumplido:** detección de intervalos con bandas de Bollinger. Partimos de la base: en períodos bajistas y alcistas, tenemos mucha volatilidad en el precio, cosa que se ve en las

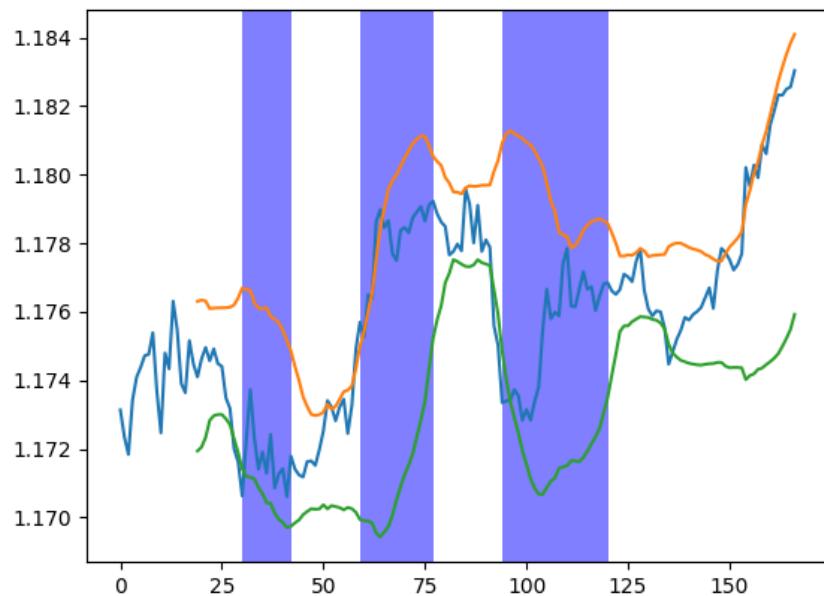


Figura 6.11: Intervalos detectados con el algoritmo de detección de tendencias

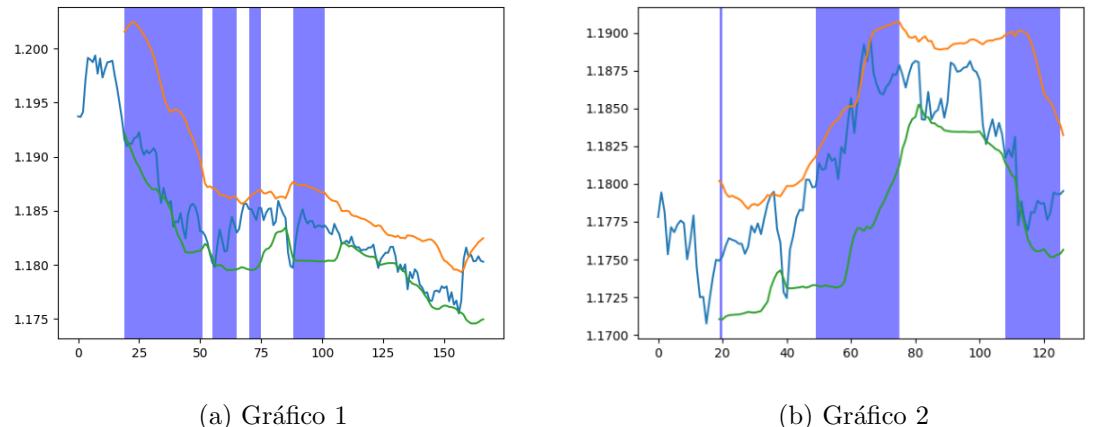


Figura 6.12: Otros ejemplos de intervalos detectados con el algoritmo de detección de intervalos

bandas (bandas más separadas). En mi implementación del algoritmo para detección de intervalos, compruebo lo siguiente, teniendo datos de precios de la última semana en tiempo real:

- Obtengo un vector de anomalías en cuanto a diferencias de valores de las bandas de Bollinger para cada unidad de tiempo. Una diferencia de valores será una anomalía si su valor es mayor que la media de diferencias.
 - El vector de anomalías indicará los tiempos en los cuales tuvimos volatilidad, esto es, indicarán intervalos.
 - Para cada anomalía, que es una lista de índices, vemos si el primer valor es mayor o menor que el último, lo que indica alcismo o bajismo.
 - En el análisis a tiempo real o backtesting, si el vector que estábamos llenando de anomalías se ha cerrado, indicará que hemos terminado un intervalo de crecimiento o decrecimiento.
- Vemos un PS, SC, AR y STs / Vemos un PSY, BC, STs:
- No es necesario comprobar que tenemos un PS o PSY, ya que nos indica que está terminando la tendencia. Que la tendencia ha terminado, lo conocemos con lo descrito en el anterior punto, el algoritmo de detección de tendencias con anomalías en las bandas de Bollinger.
 - SC → el Selling Climax indica que la amplitud del precio y presión de la oferta llegan a su clímax. A efectos prácticos el SC es el mínimo absoluto del intervalo de decrecimiento detectado. Se aplica lo análogo para detectar el BC, buying climax.
 - AR → el Automatic Rally hace subir los precios fácilmente. El máximo de este rally ayudará a definir el límite superior del siguiente rango de acumulación (mercado lateral). En mi implementación lo tomo como el máximo de los valores que hay a partir del SC hasta que llegó al fin del intervalo de decrecimiento detectado. A fin de cuentas, la teoría dice que el AR es justo eso, una subida de precios tras llegar al SC y eso lo reúne también mi algoritmo, aunque puede darse que después del SC no haya máximos muy marcados que puedan indicar un AR. Se aplica lo análogo para detectar el Automatic Reaction.
 - El ST, secondary test, se produce cuando se supera alguna de las líneas que marcan el rango de acumulación. Estas líneas son determinadas por el SC y el AR. Se determina que un punto es un ST según si está cerca del SC un tanto por ciento, cosa que podemos determinar conociendo el rango de acumulación que sería de tamaño AR - SC. Se aplica lo análogo para detectar los tests en venta.
- Tendencia bajista rota: Implementado con una media móvil. Esta tendencia no es lo mismo que el intervalo anterior, que nos indica un

bajismo mas “brusco” y directo. Esta tendencia se refiere a la actuación del precio con una perspectiva anterior. Si hemos superado a la media móvil, hemos roto dicha tendencia. Análogo para la venta.

- Mínimos más altos: Voy guardando máximos y mínimos y comparándolos en cada iteración. Añado por cada mínimo mas alto, cierto valor al indicador para operar, que será un entero y tras superar un valor, indicará la compra o venta, dependiendo de la fase en la que estemos. Análogo para la venta.
- Máximos más altos: Mismo procedimiento que en el punto anterior.
- Se crea suelo o soporte: Este suelo o techo lo determina el rango descrito anteriormente cuando he hablado de los STs. Este rango se hace entre AR y el SC. Cuando superamos el mínimo valor del suelo, salimos de la posible compra. Análogo para la venta con techo o resistencia.

Una vez descrito esto, compraremos o venderemos si se han cumplido 5 de los anteriores operadores. Cada uno suma de manera distinta al proceso y sus valores se han parametrizado según efecto en el precio. Según hemos comentado en secciones anteriores, el TP debe tener el triple de beneficio que el SL. A efectos prácticos, he reducido la relación a x2 para efectuar más compras en menos tiempo y probar el algoritmo de forma más eficaz.

Una vez implementado el análisis de Wyckoff, este puede ser usado tanto en Backtesting como en tiempo real.

6.3.5. Sprint 3: 13 julio - 31 julio. Funcionalidad para login y logout en MT5

Este tercer sprint se corresponde con el período desde el 13 de julio hasta el 31 de julio. En él se realiza la issue #2 en la que se pide implementar la funcionalidad de login y logout en MT5.

Issue 2: Implementar funcionalidad de login y logout en MT5

En la fase de especificación de requisitos, se definió RF-2.1 como:

RF-2.1. Login en la plataforma de trading. El usuario de la aplicación podrá conectarse con su cuenta de trading, comercial o demo, para poder hacer el uso completo de la APP.

En esta issue (#2) se debe desarrollar esta funcionalidad de forma que podremos usar la interfaz para conectarnos a MT5. Esta funcionalidad es la

que implementa también los botones *Login* y *Logout* del menú principal.

En esta issue se implementan dos funciones en un fichero del directorio backend para identificarse con el bróker usado en MT5 y cerrar sesión, respectivamente. La función de login necesita el usuario, la contraseña y el servidor al que se pretende conectar. Dicho servidor es proporcionado por el bróker.

Esta issue es sencilla puesto que la dependencia de MetaTrader tiene funciones para iniciar sesión en MT5. Sólo ha hecho falta adaptarlo a la APP y generar las interfaces gráficas o vistas.

Esta funcionalidad solo nos permite navegar entre cuentas para las que ya hemos estado identificados en el terminal de MT5, por tanto, para que esto funcione, en el terminal de MT5, es decir, la interfaz gráfica, tenemos que haber estado logueados con cada cuenta previamente. MT5 tiene una base de datos, con lo que la password no será necesaria si tenemos esta opción habilitada en el terminal (guardar datos de conexión, que en MT5 es *guardar los ajustes y datos personales al inicio*).

En la APP, el usuario debe de haberse identificado con antelación en el terminal del host. La opción anteriormente mencionada deberá de estar activada. De esta forma, si el usuario de la APP quiere, puede cambiar de usuario de MT5 dentro de la misma sesión de la APP.

De estas identificaciones previas al terminal de MT5 se encargará el administrador de la aplicación.

Además de estas funciones, se ha creado una función auxiliar para matar el proceso del terminal de MetaTrader5 cuando cerramos sesión en nuestra cuenta. Dicha función busca el proceso de MT5 con el módulo de MetaTrader *wmi* y le aplica un *Terminate()*.

6.3.6. Sprint 4: 31 julio - 14 agosto. Visualización de datos de mercados y gestión de usuarios

En este cuarto sprint se recogen las issues de visualización de datos de mercados financieros, en tiempo real y antiguos (issue #14); y la issue de gestión de usuarios de la APP (issue #16).

Issue 14: Funcionalidad para ver datos de mercado

En esta issue se pide ver el gráfico de precios del mercado financiero dado en tiempo real y datos antiguos. Referente a los requisitos:

RF-3. Visualización de datos.

RF-3.1. Ver datos de mercado específico. El usuario de la aplicación podrá ver información de precios de un mercado específico.

RF-3.2. Ver datos de mercado en rango de tiempo específico. El usuario de la aplicación podrá ver información de precios entre dos fechas específicas.

RF-3.3. Ver datos de mercado con un marco de tiempo específico. El usuario de la aplicación podrá ver información de precios en tiempo real o entre fechas con un marco de tiempo específico.

En el pull request asociado (PR #15), se implementa esta visualización de datos.

En el caso de ver datos de mercados antiguos, seleccionamos mercado, inicio y fecha de fin o número de horas. El módulo conecta con los datos guardados en la BBDD para mostrarlos en una gráfica de forma interactiva. En esta gráfica podemos hacer click en unos botones que nos permiten navegar en el gráfico de datos mostrados.

En el caso de ver datos en tiempo real, podemos de nuevo elegir mercado y en este caso, marco de tiempo. Se mostrará la gráfica con actualizaciones que dependerán del marco de tiempo ajustado.

Ambos desarrollos (tiempo real y antiguos) son funciones de clase de la clase Mercado. Ambas realizan preprocesado para eliminar fines de semana y mostrar los datos en formato OHLC con velas japonesas. Se utiliza *mplfinance* para imprimir gráficos de velas dados los datos ya pasados a formato OHLC.

Issue 16: Gestión de usuarios de la APP

Esta issue (#16) pide un sistema de gestión de usuarios para entrar a TradingAPP. Cabe destacar que este sistema NO es el mismo gestor de usuarios que el que se pide para identificarse en MT5. En este caso hablamos de usuarios internos a nuestra APP.

La aplicación debe pedir credenciales nada más entrar a la misma.

Cada uno de los usuarios con los que podemos entrar a la APP será

una sesión y cada sesión tendrá sus datos, historiales de compras y ventas, beneficios, etc. Como ya hemos mencionado, cuentas de MT5 aparte, como otro concepto.

La solución se aplica en el PR #17.

Se ha creado un formulario inicial para Login.

Con este formulario, se ha actualizado todo el layout principal y base de la aplicación.

Ahora el modelo *Sesion* contiene al objeto de *Django User*. En otras palabras, *Sesion* tiene un atributo llamado *User*, que viene a ser un usuario de la APP. El objetivo es que cada objeto del modelo *User* pueda tener objetos extra, como en este caso pueden ser el algoritmo elegido o el flag *logued_MT5*. Esto se ha conseguido con un mapeo de la clase *User* y *Sesion*. Cada vez que desde la vista /admin generamos un nuevo usuario, automáticamente se crea un objeto de la clase *Sesion* con las credenciales y atributos de dicho *User*.

Como he mencionado, se ha hecho un mapeo y extensión de la clase o modelo *User*. Esto se ha conseguido siguiendo el siguiente tutorial: How to extend Django User model. El código que extiende la clase User es el siguiente:

```

1 class Sesion(models.Model):
2     user = models.OneToOneField(User, on_delete=models.CASCADE,
3         null=True)
4     algoritmo_elegido = models.ForeignKey('AlgoritmoTrading',
5         on_delete=models.SET_NULL, null=True)
6     logued_MT5 = models.BooleanField(default=False)
7
8     def __str__(self):
9         return self.user.__str__()
10
11 @receiver(post_save, sender=User)
12 def create_user_sesion(sender, instance, created, **kwargs):
13     if created:
14         Session.objects.create(user=instance)

```

Figura 6.13: Código Python para extender el modelo de Django User

Finalmente, creo dos usuarios para que sean usados en la aplicación:

Usuario 1: manuel, **Contraseña:** Trading1!

Usuario 2: wyckoff, **Contraseña:** Trading2!

6.3.7. Sprint 5: 14 agosto - 30 agosto. Funcionalidad para operar en tiempo real

En este penúltimo sprint, desarrollo la funcionalidad para operar con estrategia en tiempo real.

Issue 18: Funcionalidad para operar en tiempo real con estrategia.

Esta issue pide desarrollar la funcionalidad con descripción siguiente: El usuario podrá elegir un modelo y realizar trading algorítmico. También podrá parametrizarlo según modelo y elegir tiempo en el que quiere dejar haciendo las operaciones automáticas.

En esta issue, además de lo que se pedía, he aprendido a debuggar el código de Django usando PyCharm, para arreglar varios problemas que me habían surgido.

Importante en este desarrollo, para que podamos realizar operaciones desde el script de Python, tenemos que haber habilitado previamente la opción trading algorítmico en la plataforma de MT5. Esto puede verse en la figura 6.14. Además de esto, tenemos que estar conectados con una cuenta de MT5 por lo que el login en MT5 es obligatorio para acceder a esta funcionalidad.

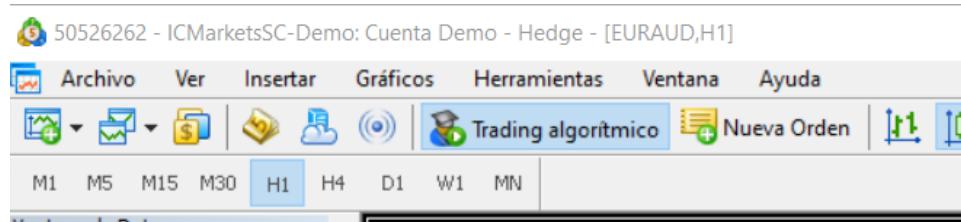


Figura 6.14: Opción a habilitar para mandar órdenes de operaciones al terminal MT5.

Nota: para comprar y vender, he creado cuentas demo en el bróker de ICMarkets. Más información sobre brokers en el punto 2 de la memoria.

El PR que completa este desarrollo es el #19.

Para realizar las compras y ventas desde nuestro script, tenemos que hacer uso de la función *order_send*, proporcionada por la librería *MetaTrader*.

```

1 order_send(
2     request      # estructura de la solicitud
3 );

```

request es una estructura del tipo *MqlTradeRequest* que describe la acción comercial necesaria. Parámetro no nombrado obligatorio.

Un ejemplo de *request* ya parametrizado podría ser el siguiente:

```

1 request = {
2     "action": mt5.TRADE_ACTION_DEAL,
3     "symbol": symbol,
4     "volume": lot if lot else 0.01,
5     "type": mt5.ORDER_TYPE_BUY if action.lower() == "buy" else mt5
6         .ORDER_TYPE_SELL,
7     "price": price,
8     "sl": stop_lose,
9     "tp": take_profit,
10    "deviation": deviation,
11    "magic": 234000,
12    "comment": "python script open",
13    "type_time": mt5.ORDER_TIME_GTC,
14    "type_filling": mt5.ORDER_FILLING_IOC,
15 }
16 order_result = mt5.order_send(request)

```

Como vemos, se utiliza la opción *mt5.ORDER_TYPE_BUY* para ordenar compras y *mt5.ORDER_TYPE_SELL* para ventas.

Al intentar usar esta función sin tener el botón anteriormente mencionado habilitado, obtendremos la siguiente respuesta:

```

1 OrderSendResult(retcode=10027, deal=0, order=0, volume=0.0, price=0.0,
2     bid=0.0, ask=0.0, comment='AutoTrading disabled by client',
3     request_id=0, retcode_external=0, request=TradeRequest(action=1,
4     magic=234000, order=0, symbol='EURUSD', volume=0.01, price
5     =1.19432, stoplimit=0.0, sl=1.1933200000000002, tp=1.19532,
6     deviation=20, type=0, type_filling=2, type_time=0, expiration=0,
7     comment='python script open', position=0, position_by=0))

```

Tras probar con el método de Wyckoff, conseguimos una operativa en tiempo real que funciona correctamente, realizamos compras y ventas según criterios en tiempo real.

6.3.8. Sprint 6: 30 agosto - 6 septiembre. Desarrollo de la gestión de datos y últimas correcciones

En este último sprint se realizan las últimas tareas del proyecto. Entre ellas, la gestión de datos que hasta ahora se hacía en local y algunas correcciones menores y mejoras de diseño.

Issue 20: Última iteración del proyecto.

Esta issue realizar los últimos desarrollos y correcciones del trabajo.

En este punto se actualiza el estilo de la interfaz de la aplicación. Se añade un logo y colores para realizar una mejora en su diseño.

Se añade aquí el sistema de gestión de datos. Cada usuario podrá seleccionar mercado, fecha de inicio de datos históricos y marco de tiempo en el que se guardarán en el formato *OHLC*. Estos datos se guardarán en la sesión del usuario, en un nuevo campo que será *TextField* y tendrá guardados los datos en formato *csv*. Esta variable o campo del modelo *Sesion* será accedido por cada una de las funciones de la APP tales como visualización de datos o backtesting para hacer uso de los datos.

Se ha hecho también una actualización en la visualización de los datos antiguos para adaptar la funcionalidad a lo comentado en el párrafo anterior.

La funcionalidad de backtesting ha sido actualizada también. Ahora, existe la posibilidad de realizar backtesting con marco de tiempo. Antes esto estaba fijado al marco *H1*.

Por último, en esta issue se mejora la salida de las gráficas de backtesting. Ahora tenemos más verbose en las gráficas y han sido formateadas.

Se añaden también las decisiones tomadas por el algoritmo en el modo de trading en tiempo real y backtesting.

6.4. Diseño final de la interfaz

En esta última sección de la implementación, añado el diseño final de la interfaz de la aplicación web.

6.4.1. Página principal y formulario de Login

Diseño en la figura 6.15

6.4.2. Menú principal y manual de usuario

Diseño en la figura 6.16

6.4.3. Menús de elección de estrategia

Diseño en la figura 6.17

6.4.4. Menú de backtesting y operativa en tiempo real

Diseño en la figura 6.18

6.4.5. Menú de gestión de datos

Diseño en la figura 6.19

6.4.6. Menú de visualización de datos

Diseños en las figuras 6.20 y 6.21.

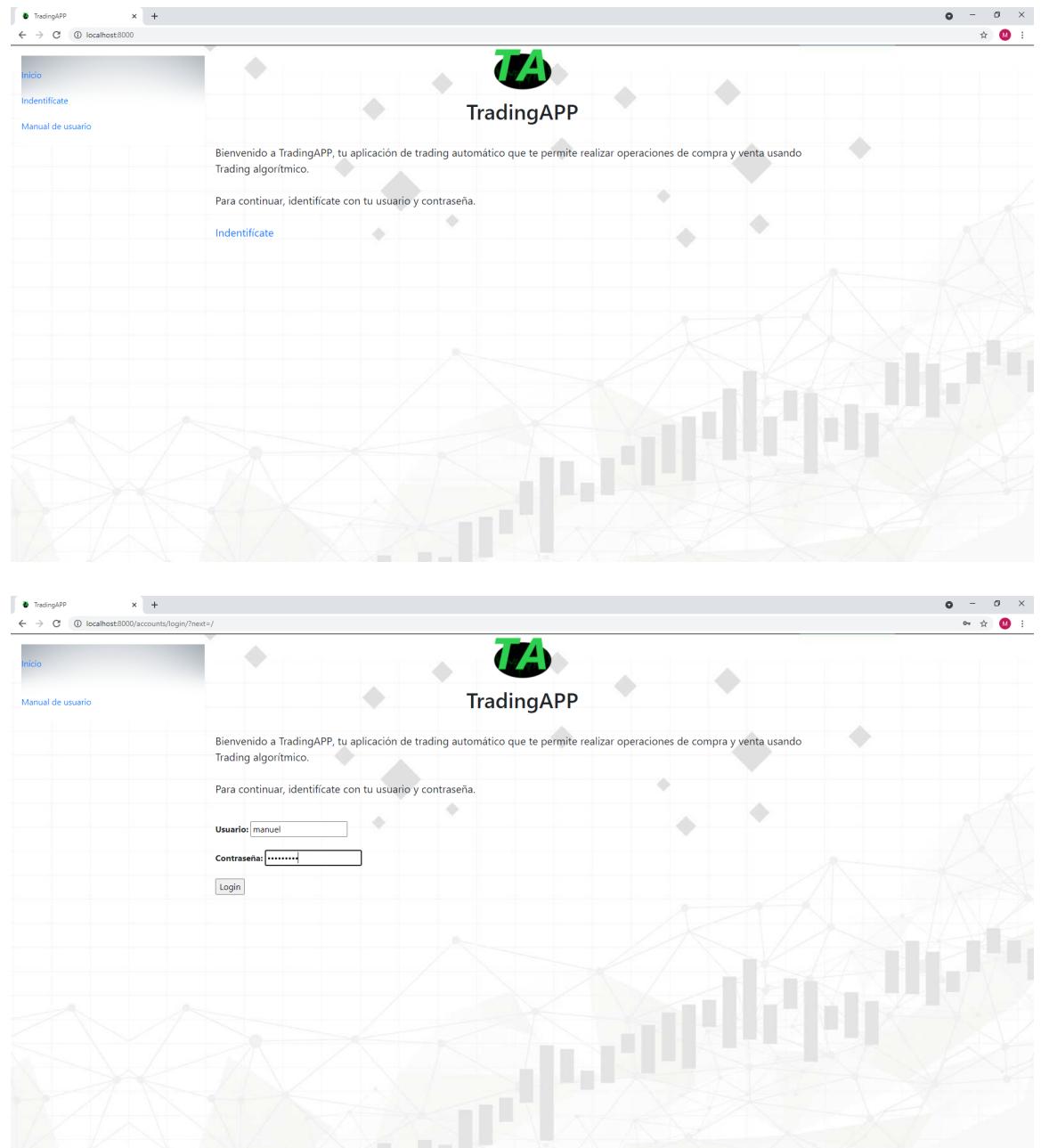


Figura 6.15: Página principal y formulario de login, respectivamente.

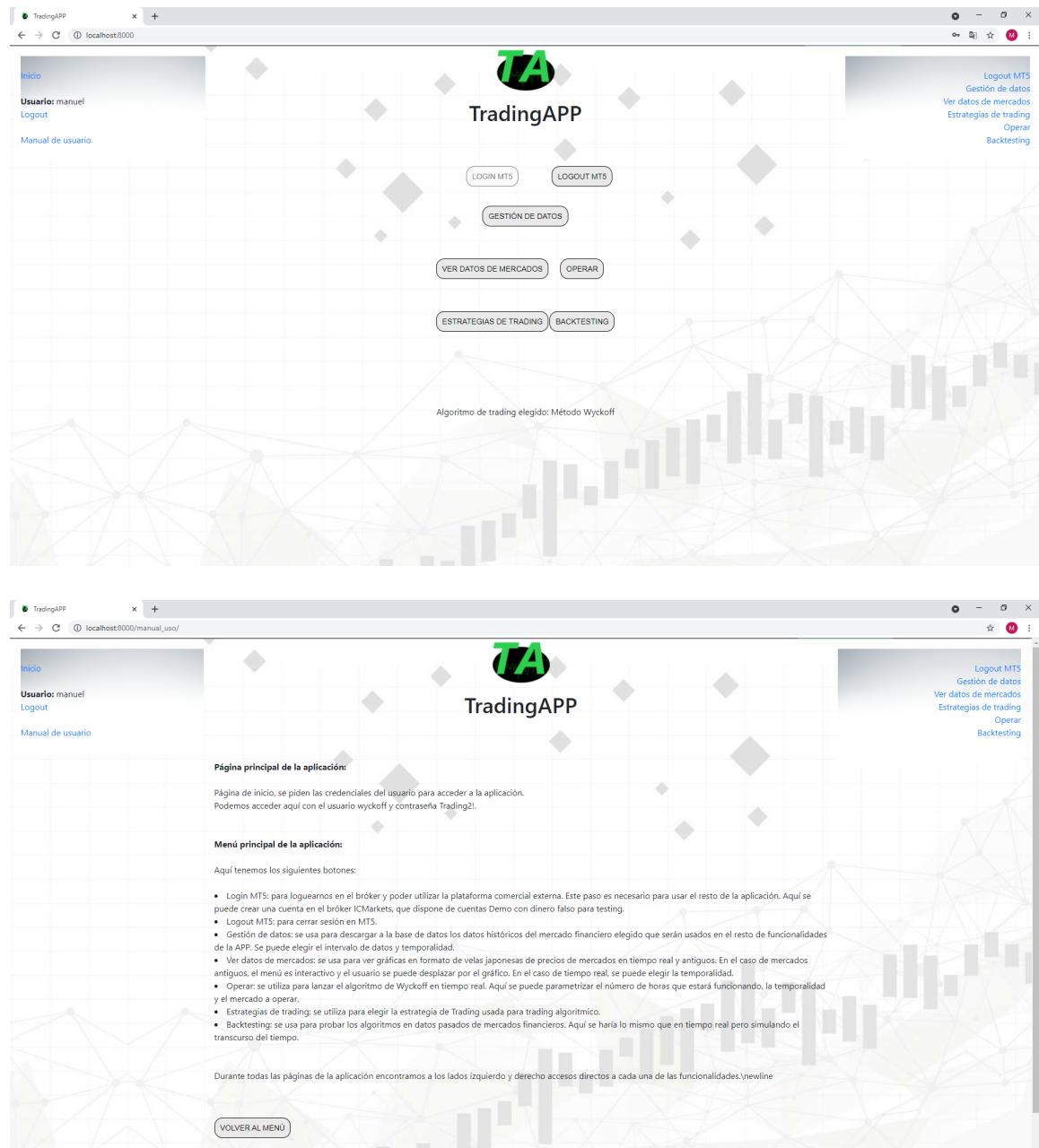


Figura 6.16: Página principal y manual de usuario, respectivamente.

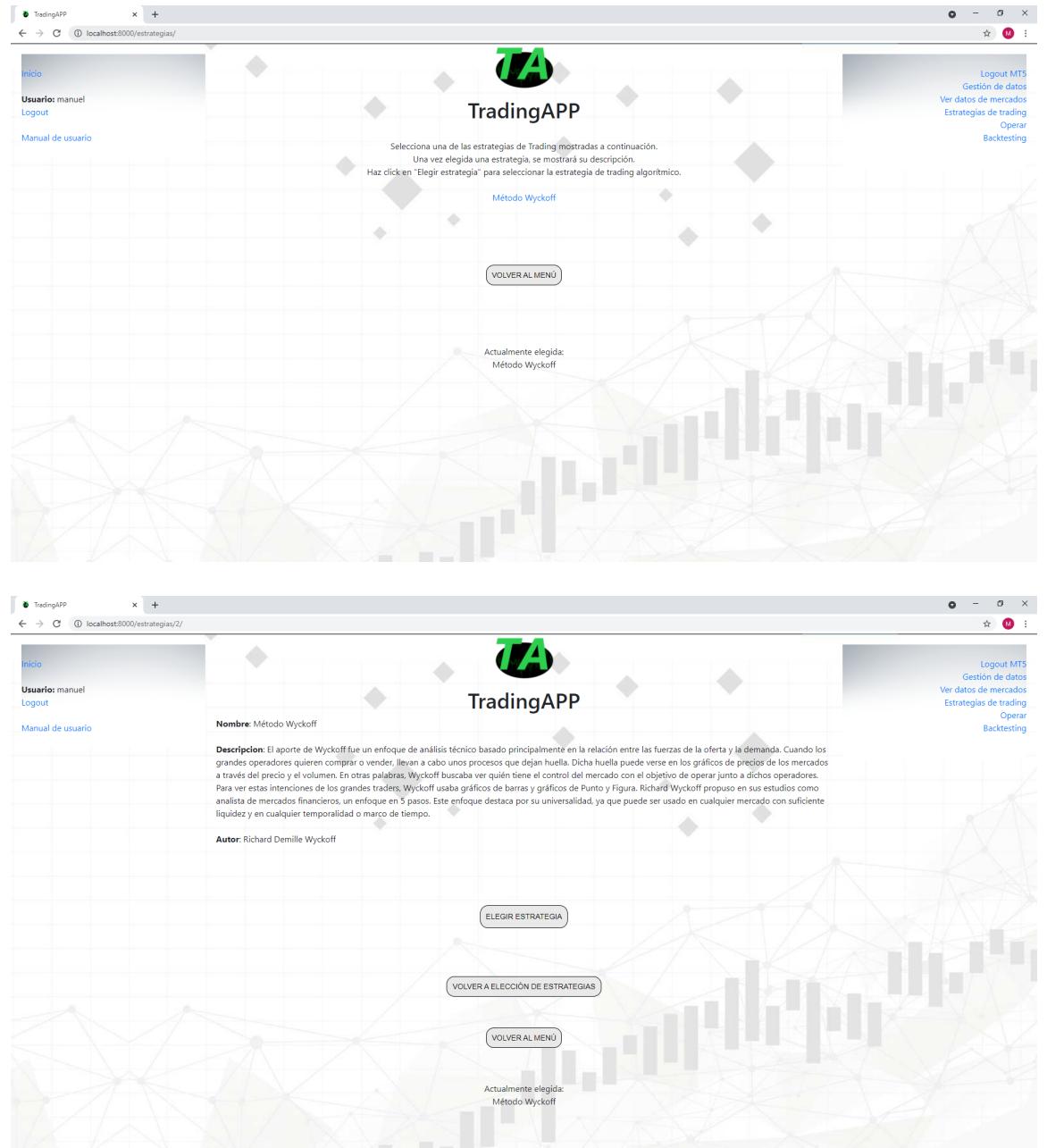


Figura 6.17: Menús de elección de estrategia.

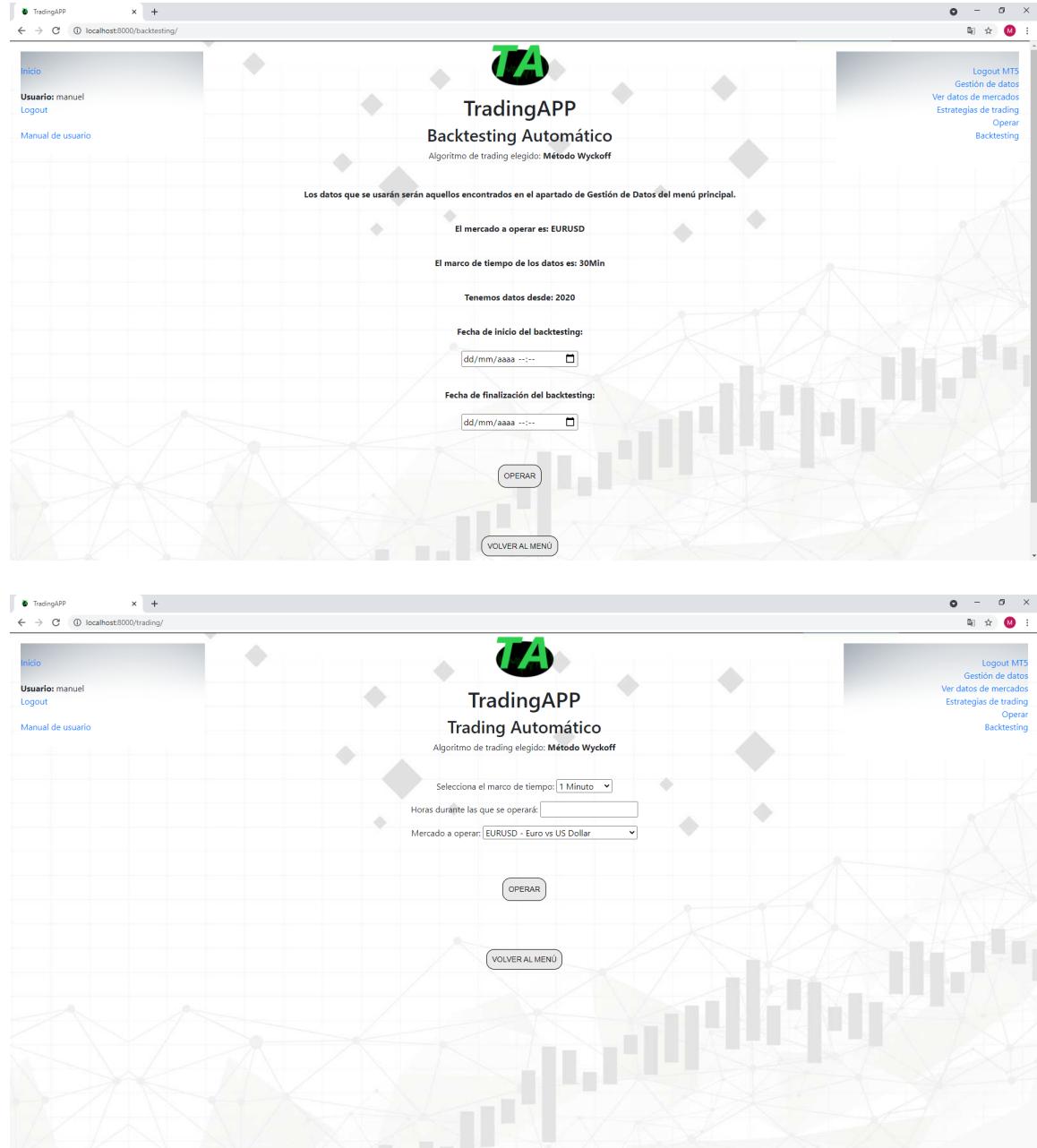


Figura 6.18: Menú de backtesting y trading a tiempo real, respectivamente.

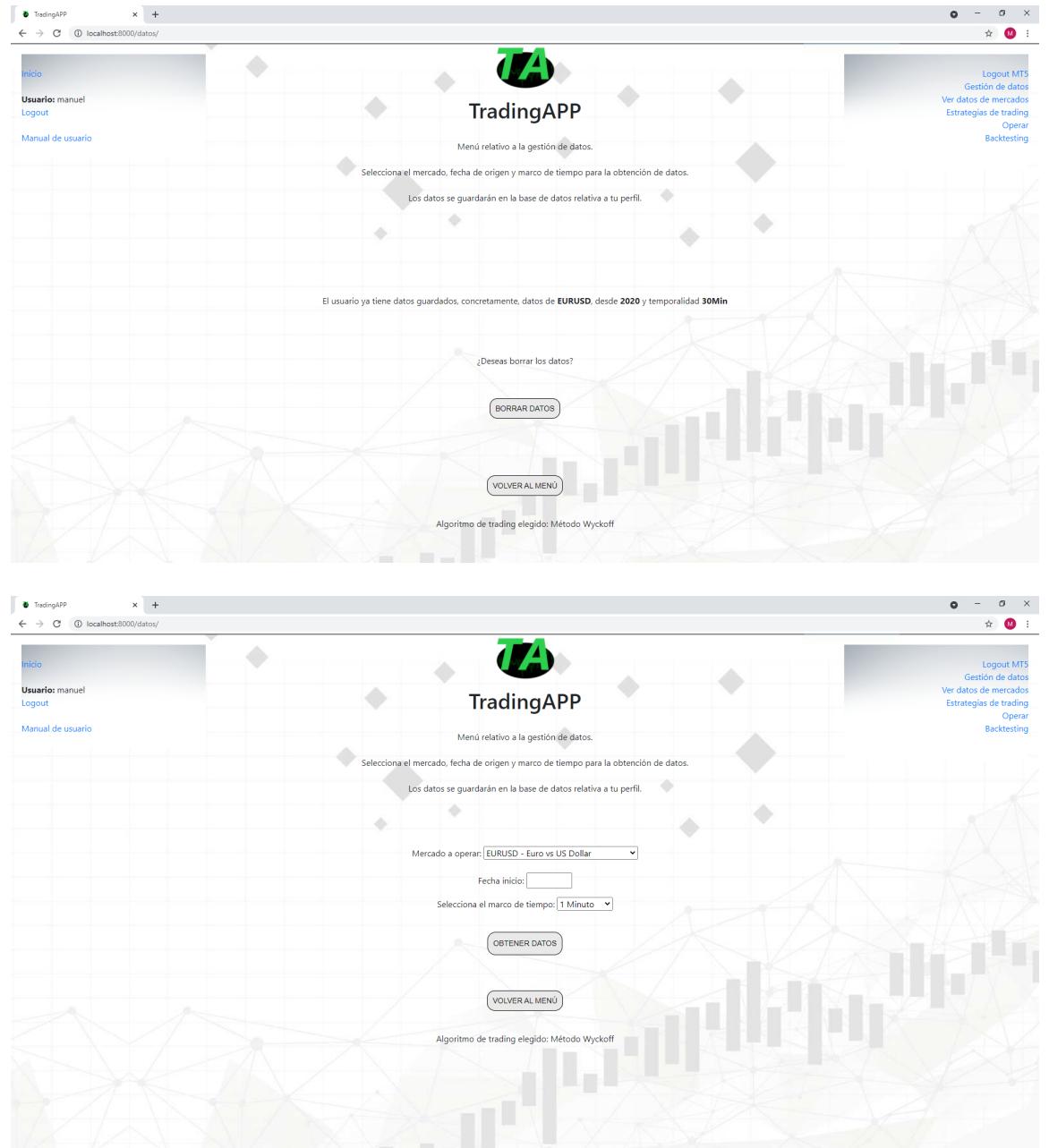


Figura 6.19: Menús de gestión de datos, teniendo datos guardados y sin tenerlos, respectivamente.

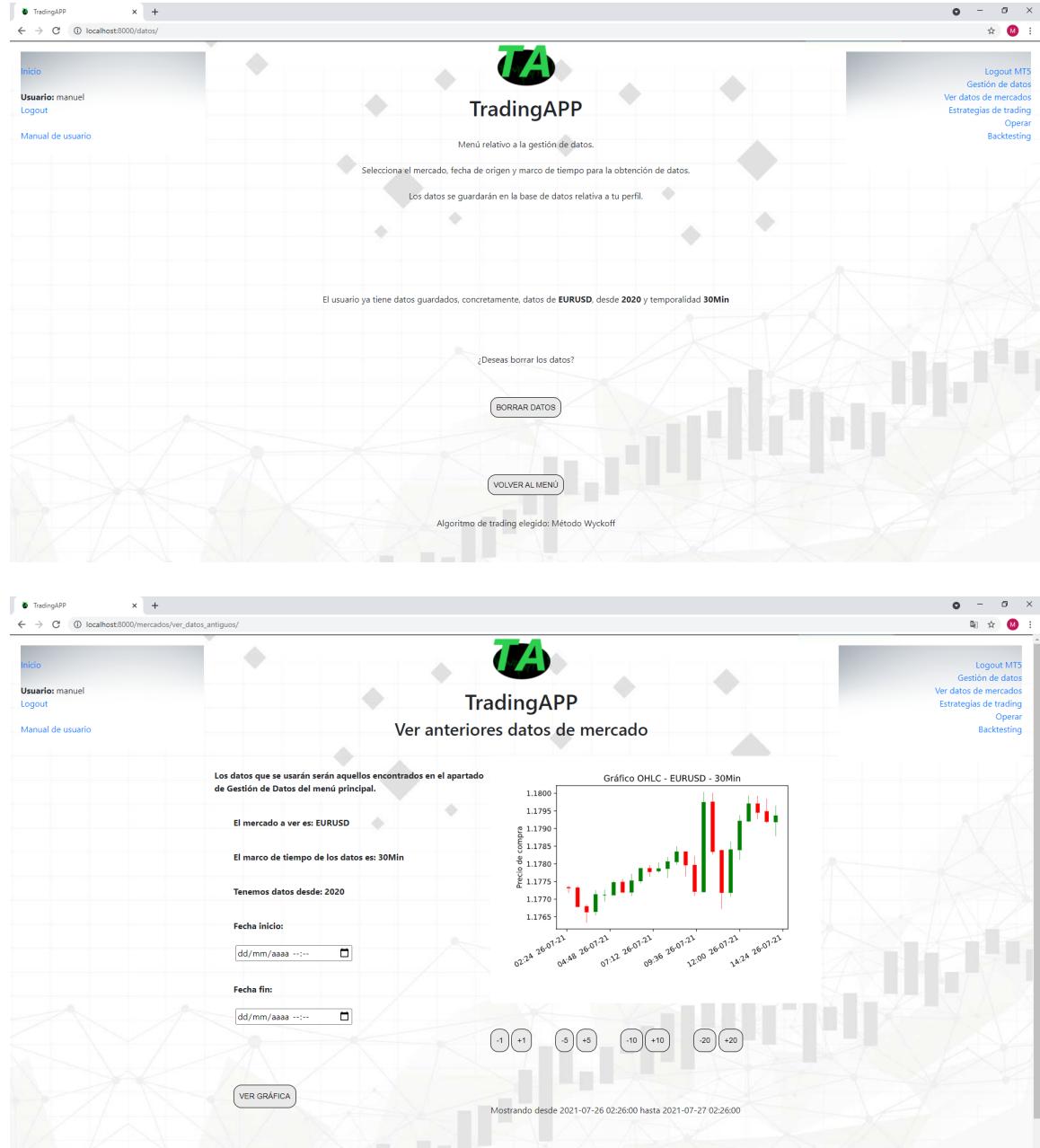


Figura 6.20: Menús de visualización de datos antiguos.

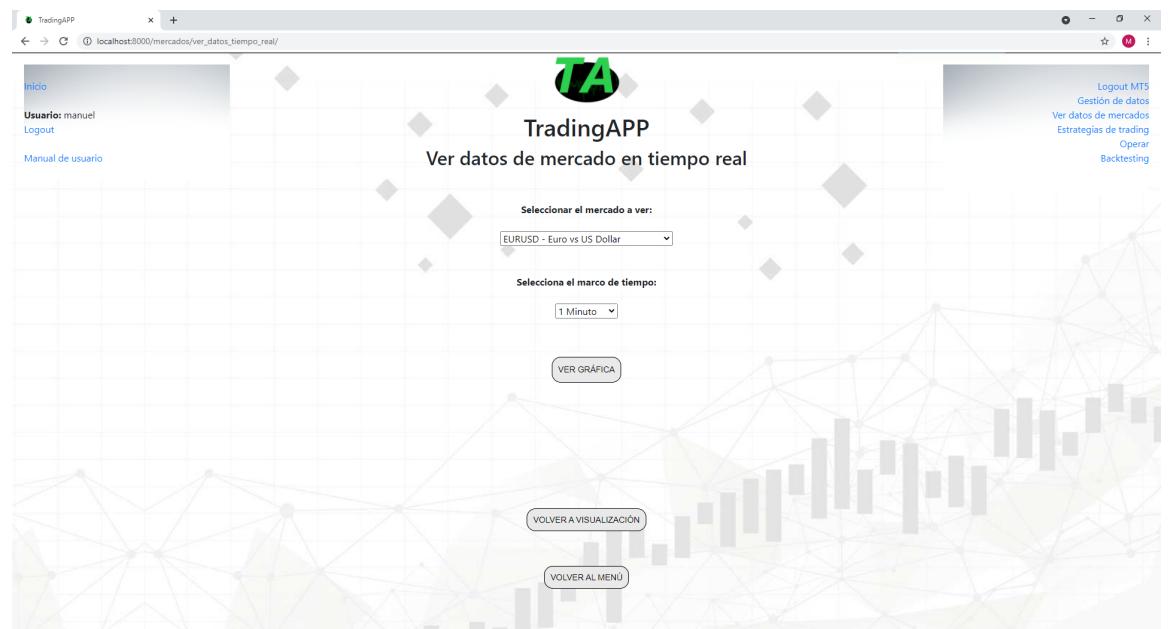


Figura 6.21: Menú de visualización de datos en tiempo real.

Capítulo 7

Pruebas

Capítulo 8

Conclusiones

Capítulo 9

Bibliografía

En este último capítulo de la memoria del proyecto se presentan las referencias bibliográficas usadas. Para citar dichas referencias, hago uso del manual de estilo *APA*, concretamente, de su séptima edición.

- ¿Qué es el volumen de negocio?. Debitoor.es. (2021). Retrieved 4 September 2021, from <https://debitoor.es/glosario/definicion-volumen-de-negocios>.
- ¿Qué son los Spreads? Bid, ask, liquidez y perrerías de los brokers. Novatos Trading Club. (2021). Retrieved 4 September 2021, from <https://www.novatostradingclub.com/formacion/spread-bid-ask-liquidez-brokers/>.
- (2021). Retrieved 4 September 2021, from <https://plataformasdetrading.com/>.
- (2021). Retrieved 4 September 2021, from <https://tradennials.com/evolucion-del-trading/>.
- Armagafx.com. (2021). Retrieved 4 September 2021, from <https://armagafx.com/metodo-wyckoff-leyes/>.
- Capitales, C., & Capitales, C. (2021). 3 tipos de análisis del Mercado Forex que debes conocer. Club de Capitales. Retrieved 4 September 2021, from <https://clubdecapitales.com/educacion/3-tipos-analisis-del-mercado-forex/>.

- Documentation on MQL5: Integration / MetaTrader for Python. Mql5.com. (2021). Retrieved 4 September 2021, from https://www.mql5.com/en/docs/integration/python_metatrader5.
- El Método Wyckoff. Wyckoff Analytics. (2021). Retrieved 4 September 2021, from <https://www.wyckoffanalytics.com/wyckoff-method-spanish/>.
- Indicadores de amplitud. NH-NL — Novatos Trading Club. Novatos Trading Club. (2021). Retrieved 4 September 2021, from <https://www.novatostradingclub.com/analisis-tecnico/indicadores-de-amplitud/>.
- Integración de MetaTrader 5 y Python: recibiendo y enviando datos. Mql5.com. (2021). Retrieved 4 September 2021, from <https://www.mql5.com/es/articles/5691>.
- León, D. (2021). HTML tutorials and reference. Htmlquick.com. Retrieved 4 September 2021, from <https://www.htmlquick.com/>.
- Ruiz, R. (2021). Desarrollando Trading algorítmico - HobbieCode. HobbieCode. Retrieved 4 September 2021, from <https://www.hobbiecode.com/trading-algoritmico/>.
- Theory, T. (2021). ¿Que es la teoría de Dow y cómo aplicarla?. Admirals. Retrieved 4 September 2021, from <https://admiralmarkets.com/es/education/articles/forex-indicators/teoria-dow>.
- Trading algorítmico. IG. (2021). Retrieved 4 September 2021, from <https://www.ig.com/es/plataformas-trading/trading-algoritmico>.
- Writing your first Django app, part 1 — Django documentation — Django. Docs.djangoproject.com. (2021). Retrieved 4 September 2021, from <https://docs.djangoproject.com/en/3.1/intro/tutorial01/>.

Capítulo 10

Anexo

En este capítulo extra de la memoria añado un manual de uso de la aplicación, para poder usarla desde cero.

10.1. Manual de uso

El proyecto se ha desarrollado en su totalidad en el repositorio de *GithHub*:
<https://github.com/mcarmona99/TFG/>

Una vez el proyecto sea entregado, este enlace será público y el proyecto se podrá descargar desde la web de *GithHub*.

Para iniciar el proyecto, necesitaremos de un entorno de *Python3*.

Desde *Windows*, podemos crear un entorno virtual de Python de la siguiente forma. Se incluye también la orden para activar dicho *venv*.

```
1 >python3 -m venv .\tradingapp_venv  
2 >.\tradingapp_venv\Scripts\activate
```

A este entorno tendremos que instalar las dependencias. El *virtual environment* creado estará presente en los archivos del proyecto como un entorno embebido, con lo que no será necesario crear uno.

Para iniciar la aplicación web, tendremos que usar el siguiente comando, desde el directorio padre *TradingAPP*:

```
1 >python3 manage.py runserver
```

Tras esto, la aplicación estará escuchando en local en la dirección <http://localhost:8000/>.

Página principal de la aplicación:

Página de inicio, se piden las credenciales del usuario para acceder a la aplicación.

Podemos acceder aquí con el usuario **wyckoff** y contraseña **Trading2!**.

Menú principal de la aplicación:

Aquí tenemos los siguientes botones:

- Login MT5: para loguearnos en el bróker y poder utilizar la plataforma comercial externa. Este paso es necesario para usar el resto de la aplicación. Aquí se puede crear una cuenta en el bróker ICMarkets, que dispone de cuentas Demo con dinero falso para testing.
- Logout MT5: para cerrar sesión en MT5.
- Gestión de datos: se usa para descargar a la base de datos los datos históricos del mercado financiero elegido que serán usados en el resto de funcionalidades de la APP. Se puede elegir el intervalo de datos y temporalidad.
- Ver datos de mercados: se usa para ver gráficas en formato de velas japonesas de precios de mercados en tiempo real y antiguos. En el caso de mercados antiguos, el menú es interactivo y el usuario se puede desplazar por el gráfico. En el caso de tiempo real, se puede elegir la temporalidad.
- Operar: se utiliza para lanzar el algoritmo de Wyckoff en tiempo real. Aquí se puede parametrizar el número de horas que estará funcionando, la temporalidad y el mercado a operar.
- Estrategias de trading: se utiliza para elegir la estrategia de Trading usada para trading algorítmico.
- Backtesting: se usa para probar los algoritmos en datos pasados de mercados financieros. Aquí se haría lo mismo que en tiempo real pero simulando el transcurso del tiempo.

Durante todas las páginas de la aplicación encontramos a los lados izquierdo y derecho accesos directos a cada una de las funcionalidades.

