



TRABAJO FIN DE GRADO
INGENIERÍA INFORMÁTICA

Aplicación web para obtener estadísticas de estrategias predefinidas de trading

Trade Analytics

Autor

Juan José Barrera Linde

Directores

Nuria Medina Medina



Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

Granada, junio de 2025

Aplicación web para obtener estadísticas de estrategias predefinidas de trading

Juan José Barrera Linde

Palabras clave: *trading algorítmico, estrategias de trading, backtesting, , microservicios, autenticación y registro de usuarios, análisis financiero, inversión*

Resumen

El análisis y la optimización de las estrategias de trading constituyen un aspecto fundamental para los traders que desean elevar su rendimiento en los mercados financieros. No obstante, gestionar y evaluar múltiples estrategias de forma eficiente puede resultar una tarea complicada. Para abordar este desafío, se ha creado una aplicación web que proporciona estadísticas detalladas sobre estrategias de trading predefinidas, lo que facilita su análisis y comparación.

La aplicación está desarrollada con *Next.js* en el frontend, lo que ofrece a los usuarios una interfaz intuitiva y dinámica. En el backend, se ha implementado una arquitectura basada en microservicios. Uno de estos microservicios se dedica a almacenar y gestionar la información relacionada con las estrategias, utilizando Prisma como ORM y PostgreSQL como base de datos relacional.

Adicionalmente, se ha configurado otro microservicio que utiliza Python para servir como intermediario entre la aplicación web y un entorno local donde opera la plataforma de trading *MetaTrader 5 (MT5)*. En este entorno, se diseñan algoritmos de trading en el lenguaje de programación MQL5, y tras completar el proceso de backtesting, los resultados se envían automáticamente a la plataforma web para su posterior análisis.

Asimismo, el sistema incluye un microservicio específico para la gestión de usuarios, que utiliza Clerk para facilitar la autenticación y el control de acceso a la aplicación.

Gracias a esta arquitectura modular basada en microservicios, la aplicación brinda a los traders la posibilidad de acceder a datos precisos y relevantes sobre estas estrategias predefinidas, lo que les permite optimizar y mejorar sus decisiones en el mercado financiero.

Web Application for Retrieving Statistics of Predefined Trading Strategies

Juan José Barrera Linde

Keywords: *algorithmic trading, trading strategies, backtesting, microservices, user authentication and registration, financial analysis, investment*

Abstract

The analysis and optimization of trading strategies are fundamental aspects for traders seeking to improve their performance in financial markets. However, managing and evaluating multiple strategies efficiently can be a complex task. To address this challenge, a web application has been developed that provides detailed statistics on predefined trading strategies, facilitating their analysis and comparison.

The application is built with *Next.js* on the frontend, offering users an intuitive and dynamic interface. On the backend, a microservices-based architecture has been implemented. One of these microservices is responsible for storing and managing strategy-related information, using Prisma as the ORM and PostgreSQL as the relational database.

Additionally, another microservice has been configured using Python, acting as an intermediary between the web application and a local environment where the *MetaTrader 5 (MT5)* trading platform operates. In this environment, trading algorithms are developed using the MQL5 programming language. Once the backtesting process is completed, the results are automatically sent to the web platform for further analysis.

Furthermore, the system includes a dedicated microservice for user management, using Clerk to enable authentication and access control within the application.

Thanks to this modular architecture based on microservices, the application provides traders with access to accurate and relevant data on these predefined strategies, enabling them to optimize and enhance their decision-making in the financial market.

D. Nuria Medina Medina, Profesora del Departamento de Lenguajes y Sistemas Informáticos

Informa:

Que el presente trabajo, titulado *Aplicación móvil para la Fundación Escuela de Solidaridad*, ha sido realizado bajo mi supervisión por **Juan José Barrera Linde**, y autorizo la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a Junio de 2025.

La directora:

Nuria Medina Medina

Agradecimientos

Quisiera dejar constancia de mi más profundo agradecimiento a todas aquellas personas que han estado a mi lado en esta dura travesía. A quienes han depositado confianza en mi labor, a quienes me han apoyado e incluso me han brindado una mano durante épocas complicadas.

Asimismo quiero acordarme y reconocer a mis compañeros de carrera con quienes he tenido el gusto de compartir momentos inolvidables que han hecho más llevadero este recorrido académico. A los docentes, tutores o profesionales que, con su entrega y conocimiento en la materia, han conseguido que surge en mí una auténtica pasión por esta materia. Y a mis amigos que la universidad me ha brindado y que, pase el tiempo que pase y viva donde viva, siempre serán una parte importante de mí.

A mi familia, a quienes debo todo lo que he logrado. Su esfuerzo incondicional, sus enseñanzas y valores han sido el pilar sobre el que he construido cada paso de este camino. Este trabajo es para vosotros, un pequeño reflejo de todo lo que me habéis dado. Gracias por vuestro sacrificio, por vuestra paciencia y por hacerme sentir siempre acompañado. No podría haber deseado una familia mejor. Os quiero con todo mi corazón.

Índice

1. Introduction	16
1.1. Motivación y contexto	16
1.2. Objetivos	17
2. Estado del arte	18
2.1. Aplicaciones web	18
2.2. Introducción al trading	19
2.3. Algoritmos de trading	20
2.4. Aplicaciones para trading similares	22
3. Planificación, metodología y presupuesto	28
3.1. Planificación	28
3.2. Metodología	29
3.3. Presupuesto	30
4. Análisis del problema	34
4.1. Descripción del problema a resolver	34
4.2. Requisitos funcionales	34
4.3. Requisitos no funcionales	37
4.4. Casos de uso	37
5. Diseño	43
5.1. Diseño arquitectónico	43
5.2. Diseño conceptual	44
5.3. Diseño de la base de datos	46
5.4. Diseño de la interfaz	47
5.5. Cuestiones de usabilidad y accesibilidad	58
5.5.1. Usabilidad	58
5.5.2. Accesibilidad	59
6. Tecnología	62
6.1. Justificación tecnológica	62
6.2. Descripción de la tecnología usada	63
7. Desarrollo	64
7.1. Frontend	64
7.2. Backend	68
7.2.1. Estructura y organización del backend	68
7.2.2. Ejemplo de implementación de la api	70
7.2.3. Sistema de gestión de usuarios CLERK	72
7.2.4. Base de datos, Prisma y PostgreSQL	73
7.2.5. Programas de intercomunicación entre la aplicación web y la plataforma Metatrader 5 en local	75
7.3. Algoritmos de trading	76
7.3.1. Estrategia basada en el Índice de Fuerza Relativa (RSI)	76
7.3.2. Estrategia en el mercado de la apertura de Nueva York	83
7.3.3. Estrategia de Acumulación, Manipulación y Distribución	88
8. Testing	97

9. Manual de uso	98
9.1. Ejecución frontend	98
9.2. Ficheros python que comunican la página web con el entorno local	99
9.3. MetaTrader 5	100
9.4. Uso final de la aplicación web	101
9.4.1. Página principal	101
9.4.2. Inicio sesión y registro	105
9.4.3. Dashboard o panel	107
10. Resultados y casos de uso	121
11. Conclusiones y trabajos futuros	123

Índice de figuras

1.	Análisis técnico	20
2.	Dashboard strategyQuant	23
3.	Estrategias predefinidas strategyQuant	23
4.	Parametros de una estrategia predefinida strategyQuant	24
5.	Historico de precios strategyQuant	25
6.	Backtesting strategyQuant	25
7.	Estadísticas del backtesting strategyQuant	26
8.	Carga de trabajo mensual	28
9.	Diagrama general de casos de uso del sistema	38
10.	Caso de uso: Acceso al panel del usuario	39
11.	Caso de uso: Selección de estrategia de trading	40
12.	Caso de uso: Ejecución del backtesting	41
13.	Caso de uso: Visualización de resultados	41
14.	Caso de uso: Gestión de estrategias guardadas	42
15.	Diseño arquitectónico	43
16.	Diagrama conceptual	45
17.	Diagrama base de datos	46
18.	Pantalla de inicio de sesión. Permite a los usuarios registrados acceder a la aplicación mediante Clerk.	48
19.	Primera vista del formulario de registro para nuevos usuarios.	48
20.	Segunda parte del proceso de registro.	49
21.	Vista general de la página principal tras iniciar sesión.	49
22.	Visualización de las estrategias disponibles.	50
23.	Detalles de una estrategia con sus parámetros.	51
24.	Interfaz para lanzar un backtesting sobre una estrategia.	51
25.	Visualización dinámica del estado del backtesting.	52
26.	Gráficos iniciales generados tras el backtest.	52
27.	Ampliación con estadísticas detalladas por estrategia.	53
28.	Comparativa entre estrategias y filtrado por rendimiento.	54
29.	Vista general de las estrategias almacenadas en la plataforma.	55
30.	Herramienta para probar estrategias predefinidas con parámetros personalizados. .	55
31.	Resultados gráficos del backtesting ejecutado.	56
32.	Zona donde el usuario puede guardar configuraciones de parámetros para futuras pruebas.	56
33.	Panel de administración donde se pueden gestionar usuarios y configuraciones avanzadas.	57
34.	Diagrama de flujo.	57
35.	Ejemplo de usabilidad en el formulario.	59
36.	WAVE accesibilidad web.	60
37.	Solución errores WAVE.	61
38.	Estructura frontend.	65
39.	Estructura dashboard detallada.	66
40.	Estructura home detallada.	66
41.	Estructura backend.	69
42.	Api PATCH.	70
43.	Api POST.	71
44.	Api GET.	71
45.	Api DELETE.	72
46.	Estructura CLERK.	73

47. Ejemplo código prisma.	74
48. Operación de venta con la estrategia RSI	81
49. Operación de compra con la estrategia RSI	82
50. Operación de venta en la apertura de nueva york	87
51. Simulación de la estrategia acumulación manipulación y distribución	94
52. Resultado del caso previsto en la simulación	96
53. Fichero enviroment frontend	98
54. Visualización página principal inicio	101
55. Visualización página principal características	102
56. Visualización página principal estrategias	102
57. Visualización formulario con un ejemplo funcional	103
58. Visualización formulario con un ejemplo funcional email	103
59. Apartado símbolos, información de la empresa de fondeo	104
60. Apartado símbolos, información de los símbolos disponibles	104
61. Inicio sesión	105
62. Registro	106
63. Registro verificación	106
64. Gestión de estrategias	107
65. Gestión de estrategias	108
66. Gestión de estrategias agregar	108
67. Gestión de estrategias modificar	109
68. Gestión de parametros	109
69. Gestión de parametros agregar	110
70. Gestión de parametros modificar	111
71. Todo el historial de simulación realizado por todos los usuarios	112
72. Todo el historial de simulación realizado por el usuario registrado	113
73. Historial de simulación guardados por el usuario registrado	113
74. Probador de estrategias	114
75. Probador de estrategias: Formulario 1	115
76. Probador de estrategias: Formulario 2	115
77. Probador de estrategias: Backtesting gráficos	116
78. Probador de estrategias: Backtesting información resultado 1	117
79. Probador de estrategias: Backtesting información resultado 2	117
80. Probador de estrategias: Backtesting información tablas	118
81. Probador de estrategias: Backtesting sección resultados	119
82. Probador de estrategias: Backtesting sección tabla transacciones	120
83. Ejemplo de resultados con las estrategias de trading gráfico 1	121
84. Ejemplo de resultados con las estrategias de trading gráfico 2	122
85. Ejemplo de resultados con las estrategias de trading estadísticas	122

Índice de cuadros

1.	Resumen del presupuesto sobre el desarrollo	32
2.	Resumen de presupuesto de equipo informático y software	32
3.	Resumen de otros gastos asociados	33
4.	Presupuesto global estimado del proyecto	33

1 Introduction

1.1. Motivación y contexto

El trading es la actividad de compra o venta de activos financieros en los mercados como puedes ser acciones(apple), divisas(EUR/USD), materias primas(petróleo) o criptomonedas(bitcoin), con el objetivo de generar beneficios y rentabilidad en un periodo de tiempo.

El interés en el mundo del trading ha aumentado considerablemente en los últimos años debido, entre otros factores, el acceso a las plataformas de inversión y por la evolución de tecnologías que permiten analizar los mercados financieros con mayor precisión. En este contexto, la optimización de estrategias de trading ha pasado a ser un aspecto importante para aquellos inversores que buscan maximizar la rentabilidad del capital del que disponen de forma eficiente y estructurada.

La motivación principal, percibida desde una visión personal, para la generación de este proyecto partió de la inquietud en el ámbito de los mercados financieros y de la inversión como un método alternativo a la inmovilización del capital en entidades bancarias sin rentabilidades.

No obstante, la gestión de las estrategias de trading, así como su eficiente evaluación, supone una problemática técnica importante, particularmente cuando es necesario manejar múltiples algoritmos bajo diferentes parámetros y condiciones de mercado. Este problema es la motivación que nos lleva a la necesidad de establecer herramientas que puedan automatizar el análisis y permitir comparaciones que tengan en cuenta la imputación objetiva de los datos.

En definitiva, este trabajo se encuentra en la intersección del desarrollo de un producto de software y del análisis financiero, con el fin de crear una solución tecnológica que ayude a la evaluación y la optimización de las estrategias de trading. Con este propósito, se ha diseñado una infraestructura de gestión de datos que es eficiente y escalable. La aplicación web desarrollada no es solo una respuesta a la necesidad personal de tomar decisiones con mejores garantías en el ámbito de la inversión, sino que también es una herramienta que es útil para traders preocupados por optimizar su rendimiento en los mercados financieros.

1.2. Objetivos

El objetivo más significativo de esta investigación es la implementación de una aplicación web que permita analizar y optimizar estrategias de trading a través de la automatización de la recolección, almacenamiento y visualización de datos relevantes. Esta es una de las herramientas que permitirá a los traders analizar rápidamente el rendimiento de diferentes estrategias de trading, ofreciendo una infraestructura escalable, modular y construida con microservicios.

Para conseguir este objetivo, se enumeran los siguiente subobjetivos específicos:

1. Investigar el ámbito y las herramientas que existen en el trading algorítmico, para después obtener las principales funcionalidades que ha de tener la propuesta de esta aplicación.
2. Analizar las necesidades de la aplicación de análisis de estrategias de trading automatizado, tanto desde el perfil de los usuarios como desde los aspectos técnicos del sistema.
3. Diseñar e implementar una interfaz en la que los usuarios puedan interactuar con aquellas estrategias de trading y de sus resultados de manera dinámica e intuitiva.
4. Desarrollar una arquitectura con microservicios que permita gestionar adecuadamente los diferentes módulos de la aplicación, garantizando flexibilidad para escalar y mantener dicha aplicación.
5. Integrar una base de datos relacional utilizando PostgreSQL y Prisma ORM para el almacenamiento estructurado y eficiente de la información relacionada con las estrategias de trading.
6. Conectar la aplicación con la plataforma MetaTrader 5 (MT5) usando un microservicio en Python que permita automatizar la ejecución de backtesting y obtener los resultados estadísticos que permitan evaluar el rendimiento de las estrategias.
7. Implementar un sistema de autenticación y gestión de usuarios haciendo uso de Clerk para que la aplicación tenga el control de acceso y, por lo tanto, seguridad que permita que los usuarios de la plataforma obtengan esa tranquilidad.
8. Realizar algoritmos de trading en MetaTrader 5 donde se obtendrán resultados y estadísticas en símbolos diferentes en base al tiempo de simulación establecido y los parámetros tanto de la cuenta como de la estrategia.
9. Optimizar el procesamiento y visualización de datos estadísticos para que los traders puedan analizar el rendimiento que tienen estas estrategias de la mejor manera posible, de forma clara y entendible.
10. Asegurarnos de la estabilidad y eficiencia del sistema a prueba y validación para que la aplicación funcione de la mejor forma en aquellos escenarios que se define.

2 Estado del arte

El estado del arte consiste en un recorrido general de los conocimientos, tecnologías y soluciones que ya han sido producidas en el ámbito de estudio del proyecto. El objetivo del estado del arte es, recopilar información del trabajo que se ha realizado, identificando apartados de trabajo previas, herramientas de relevancia y tendencias actuales, justificar el porqué de la toma de decisiones en el diseño e implementación del sistema propuesto.

El estado del arte abarca tanto aspectos relacionados con el desarrollo de aplicaciones web, el trading algorítmico, plataformas de backtesting y análisis de estrategias, o por otra parte, arquitectura basada en microservicios y gestión de usuarios en entornos de web.

2.1. Aplicaciones web

Las aplicaciones web son programas de ordenador que se ejecutan en un navegador web y permiten a los usuarios interactuar con funcionalidades avanzadas mediante una conexión a Internet. En comparación con los sitios web convencionales que ofrecen fundamentalmente contenido estático e informativo, las aplicaciones web están orientadas a la interacción dinámica, haciendo mucho más relevante en el procesamiento de datos y la ejecución de tareas complejas en el lado del cliente y en el lado del servidor.

Los sitios web estáticos que fundamentalmente ofrecen contenido informativo, las aplicaciones web están encaminadas a la interacción de una manera más dinámica en la que se vuelve mucho más importante el procesamiento de datos y la gestión de tareas en relación al servidor [2].

El origen de las aplicaciones web se sitúa a mediados de la década de los 90, con la aparición de los primeros sitios con contenido dinámico a partir del uso de tecnologías como lenguajes PHP, cuyo desarrollo se encamina en la aplicación web. Aunque el verdadero impulso lo encontramos en el primer segmento de la década de 2000 que se produce a partir de la llegada de AJAX (Asynchronous JavaScript and XML) que ofrece la posibilidad de actualizar parte de la interfaz de usuario sin tener que volver a cargar la totalidad de soporte y que fue decisiva para estrenar la nueva experiencia de navegación que estimuló aplicaciones web más interactivas y funcionales [13].

En términos de arquitectura, las aplicaciones web hoy en día dividen la aplicación web en dos partes principales:

1. Frontend de la aplicación web: esta parte es la parte visual del usuario, construida con tecnologías del tipo de HTML, CSS y JavaScript, o frameworks como por ejemplo React, Angular o Vue. Se desarrolla principalmente para poder ofrecer una interfaz sencilla, clara, funcional e intuitiva [23].
2. Backend de la aplicación web: esta parte es la lógica y funcionalidad que se tiene en un servidor. Esta parte de la aplicación web se encargará del procesamiento de datos, de conectar con bases de datos, de establecer la lógica de negocio que se requiere para la aplicación web. Esta parte normalmente se desarrolla en lenguajes como Python, Node.js, Java o PHP, también se puede usar frameworks como por ejemplo laravel, django, .net, spring boot [14].

Un modelo arquitectónico que se está utilizando ampliamente para el desarrollo de aplicaciones web es aquel que se basa en microservicios, ya que separa la funcionalidad en pequeños servicios

independientes que se comunican entre ellos. El uso de esta práctica permite entre otras características el mantenimiento, la escalabilidad e integración de diferentes tecnologías.

El desarrollo de aplicaciones web debería contemplar también otros elementos relevantes como son:

1. La persistencia de datos a través de bases de datos relacionales por ejemplo PostgreSQL, MySQL o no relacionales como MongoDB.
2. La Seguridad y la Autenticación que permiten el control de acceso a los usuarios sobre las distintas funcionalidades.
3. El Despliegue en la nube que permite la publicación y escalabilidad de los servicios de forma flexible mediante servicios como Vercel, AWS, Heroku o Docker .

Las aplicaciones web están presentes en una diversidad muy amplia de sectores como por ejemplo, el de las aplicaciones para comercio electrónico, la educación, la banca, y también en el financiero que permite la gestión de datos de mercado o análisis de datos, entre otros.

2.2. Introducción al trading

El *trading* consiste en la compra y venta de activos financieros. Dichos activos nos permiten hacer operaciones con acciones, divisas (forex), criptomonedas, materias primas, etc. A la diferencia de la inversión a largo plazo, el trading se caracteriza por su mayor actividad en sus operaciones, que tienen en presencia los movimientos de precios a corto o medio plazo [10].

El trader (persona que opera haciendo trading) ante todo pretende ir aprovechándose del movimiento del mercado: compra cuando el mercado está a un precio bajo, vende cuando el mercado está en un precio alto, en función de la estrategia que esté utilizando. Las operaciones pueden tomar desde una fracción de segundo (de alta frecuencia) hasta días o meses.

Existen diferentes estilos de trading atendiendo al periodo que dura cada operación:

- Scalping. Operaciones rápidas, de segundos a minutos.
- Day trading. Operaciones que se inician y cierran en el mismo dia.
- Swing trading. Operaciones que se mantienen abiertas varios días o incluso meses.

Las decisiones que toman los traders se sostienen en diferentes herramientas y formas de encarar el trading, pero entre ellas destacan las siguientes:

1. El análisis técnico que se basa en el estudio del comportamiento del precio en su historia utilizando gráficos y diversas formas de indicadores matemáticos (RSI, medias móviles, MACD, etc...) [27].
2. El análisis macroeconómico que estudia aquellas circunstancias económicas, políticas o financieras que impactan el valor de los activos financieros [27].
3. El análisis fundamental se usa para determinar el valor intrínseco de una empresa o activo, y luego comparar ese valor con su precio de mercado [27].
4. El análisis cuantitativo que trabaja con modelos matemáticos y estadísticos para encontrar las distintas oportunidades de mercado.



Figura 1: Análisis técnico.

En la figura 1 se muestra una gráfica de análisis técnico que será la herramienta más relevante a tener en cuenta.

La evolución de la tecnología trajo la evolución del trading, descubriendo el trading algorítmico donde los programas informáticos son capaces de ejecutar operaciones automáticas de acuerdo a una serie de reglas pre establecidas. Este último tipo de trading permite operar de una manera más rápida, más consistente y de una forma simultánea en diversos mercados.

2.3. Algoritmos de trading

El trading algorítmico es una modalidad de inversión en mercados financieros que hace uso de algoritmos de trading para ejecutar automáticamente operaciones de compraventa de activos. Los algoritmos son creados para ejecutar determinadas reglas, siguiendo una lógica concreta basada en indicadores técnicos, patrones de precios, condiciones de mercado u otras reglas definidas por el trader o programador [29].

El trading algorítmico se diferencia del trading manual en que este último hace que el trader tome la decisión de realizar o no una operación y a su vez ejecute dicha operación. El trading algorítmico ejecuta totalmente este proceso de operación, permitiendo así una mayor velocidad de ejecución, eliminación del estado emocional y capacidad de analizar varios activos a la vez.

Crear una estrategia de trading algorítmico implica tener en cuenta ciertos elementos clave:

1. Lógica de la estrategia: conjunto de reglas que indican cuándo abrir, modificar o cerrar una posición. Estas reglas suelen ser, por ejemplo, indicadores como el RSI, las medias móviles, las bandas de Bollinger, etc.
2. Lenguaje de programación y de plataforma: el algoritmo se tiene que implementar en el lenguaje que es compatible con la plataforma de trading que utilizarás. Uno de los entornos más comunes es MetaTrader 5 (MT5), la plataforma de trading que permite programar estrategias a través del lenguaje MQL5, también hay otras alternativas como python.
3. Backtesting: antes de ser utilizadas en el mercado real, las estrategias tienen que ser probadas con datos históricos para poder ver cómo es su comportamiento en situaciones reales. Este

proceso permite medir métricas como el beneficio neto, el drawdown, el porcentaje de aciertos, la relación riesgo y beneficio, etc.

4. Ejecución automática: una vez verificada, la estrategia puede ejecutarse de forma automática sin intervención humana, lo que permite operar incluso en momentos de alta volatilidad que provoca el mercado o en diferentes mercados al mismo tiempo.

Los algoritmos en el trading ofrecen las siguientes cualidades:

1. Velocidad de ejecución: Pese a que el trading manual no se puede comparar con los algoritmos, este se puede ejecutar en milisegundos.
2. Disciplina operativa: Teniendo en cuenta que las máquinas reaccionan sin la intervención de emociones o cualquier otro tipo de impulso, seguramente el número de errores por decisiones de forma errónea se reducirá.
3. Capacidad de prueba: El algoritmo puede ser optimizado y validado de antemano con simulaciones con datos históricos.
4. Multitarea: Un único algoritmo puede operar varios activos o mercados.

Los algoritmos de trading también tienen sus limitaciones, tales como;

1. Dependencia de los sistemas informáticos: Pérdidas de conexión, errores de código o bugs pueden llevar a consecuencias financieras devastadoras.
2. Overfitting: Algunas estrategias de trading pueden tener un buen rendimiento con datos históricos, pero puede ser que no lo tengan en tiempo real o tiempo futuro.
3. Condiciones de mercado en constante cambio: Un algoritmo efectivo en una situación de mercado determinada puede dejar de ser efectivo en otras etapas donde el mercado financiero es muy distinto, por ejemplo un crisis.

Los algoritmos pueden ser de diferentes tipos en función del objetivo que tenga el algoritmo, del tipo de análisis que utilice y de la lógica que ofrezca a la hora de tomar decisiones. Éstas son implementadas mediante código y ejecutadas automáticamente por plataformas de trading como MT5 o python, lo que permite operar en los mercados financieros sin que sea necesaria la intervención directa por parte del ser humano.

Los algoritmos de trading se pueden categorizar de forma general, en:

1. Algoritmos enfocados en análisis técnico: Estos emplean lo que se conoce como indicadores técnicos (entre los cuales se encuentran el RSI, las medias móviles, el MACD, etc...) para hacer señales de compra o venta, ya que al ser de los más comunes, se basan en patrones de comportamiento más comunes en el precio.
2. Algoritmos enfocados en análisis estadístico: Éstos introducen el uso de modelos matemáticos y estadísticos, con la idea de detectar oportunidades, entre las cuales podemos destacar medias móviles o por ejemplo modelos de regresión.
3. Algoritmos que funcionan por eventos fundamentales o macroeconómicos: Proceden a ejecutar operaciones en base a una serie de eventos financieros, económicos o de tipo de mercado que tengan lugar, como la noticia de un resultado financiero, una subida de tipos de interés, la publicación de determinada información macroeconómica, etc.
4. Algoritmos que funcionan por aprendizaje automático: Utilizan técnicas de inteligencia artificial para adaptarse constantemente al mercado de forma dinámica. Se alimentan con grandes volúmenes de datos y aprenden a captar patrones que no son evidentes al ojo humano humano.

2.4. Aplicaciones para trading similares

En el mercado hay pocas herramientas que permiten la optimización y evaluación de estrategias de trading a partir de backtesting y análisis estadístico. StrategyQuant es una de las más reconocidas, existiendo una solución avanzada para el desarrollo de estrategias de trading de forma predefinida.

StrategyQuant es una herramienta específica para el desarrollo de estrategias predefinidas de trading que utilizan trading algorítmico. Su propósito es el desarrollo de estrategias rentables con el menor esfuerzo por parte del usuario y sin que sea necesaria una formación muy avanzada en programación. Esta herramienta permite [30]:

- Un backtesting avanzado, facilitando la simulación de estrategias sobre datos de mercados históricos con el fin de analizar su rendimiento bajo diferentes escenarios de mercado.
- Una optimización automática, lo que permite modificar los parámetros de las estrategias para obtener un mejor rendimiento de rentabilidad y estabilidad de éstas.
- Es compatible con diferentes plataformas de trading, como MetaTrader 4, metatrader 5, NinjaTrader y MultiCharts.

Para obtener la aplicación de escritorio hay que realizar un pago único o también se tiene opción de pago mensual, voy a mostrar el caso de pago mensual, tiene 3 opciones, el plan Ultimate proporciona acceso a todas y cada una de las funciones avanzadas como soporte MQL Market, Portfolio Master, módulos premium y actualizaciones de por vida, mientras que los planes más básicos tienen acceso limitado a ciertas funcionalidades como optimización avanzada y pruebas de robustez:

- Starter: 119 \$
- Professional: 175 \$
- Ultimate: 420 \$

Casos de uso sobre la aplicación

Para obtener el uso de la aplicación se tiene que dirigir al su página web oficial, tiene que comprar una versión, una vez realizado el pago se obtendrá la aplicación de escritorio.

Al no tener el presupuesto para probar la aplicación, se ha realizado capturas de una persona que si ha comprado y realizado una usabilidad sobre dicha aplicación.

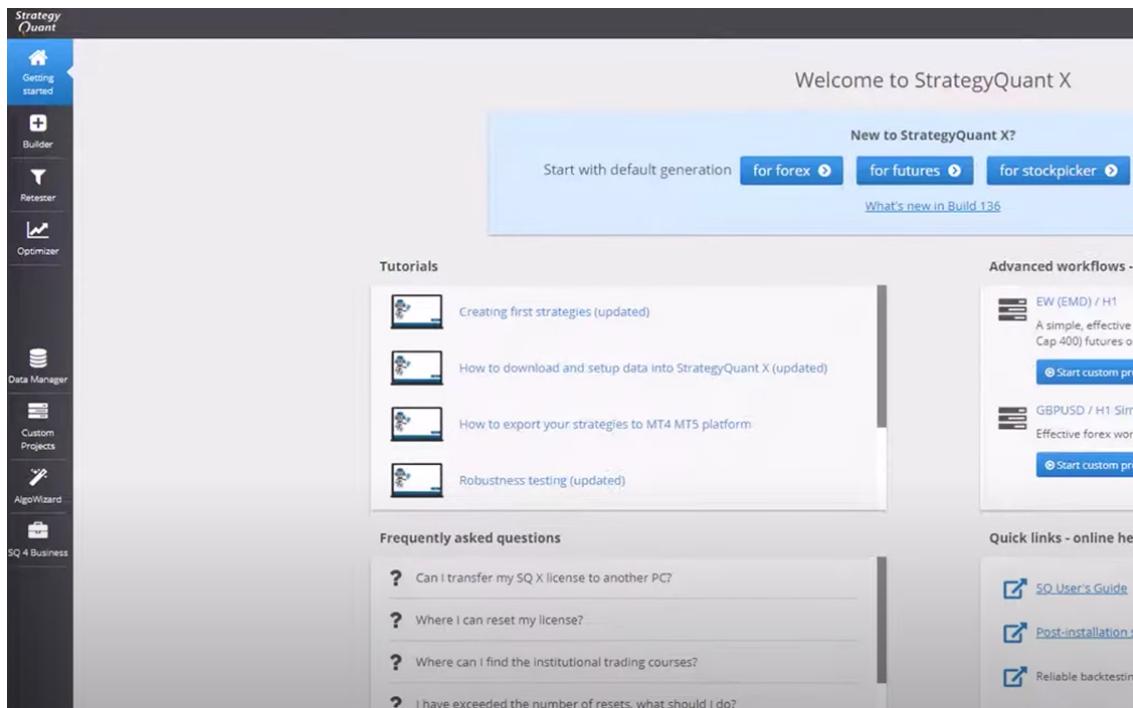


Figura 2: Dashboard strategyQuant

En la figura 2 se muestra el contenido principal, en el menu de la izquierda tenemos varios apartados, los más relevantes son "builder" donde se seleccionara la estrategia que se quiere realizar un backtesting, el ".optimizer" donde después de seleccionar una estrategia realizar una optimización de los parametros para obtener los mejores resultados, y por ultimo el "data manager" donde se obtendra el historial de precios para poder realizar el backtesting de las estrategias escogidas anteriormente con su respectivos simbolos.

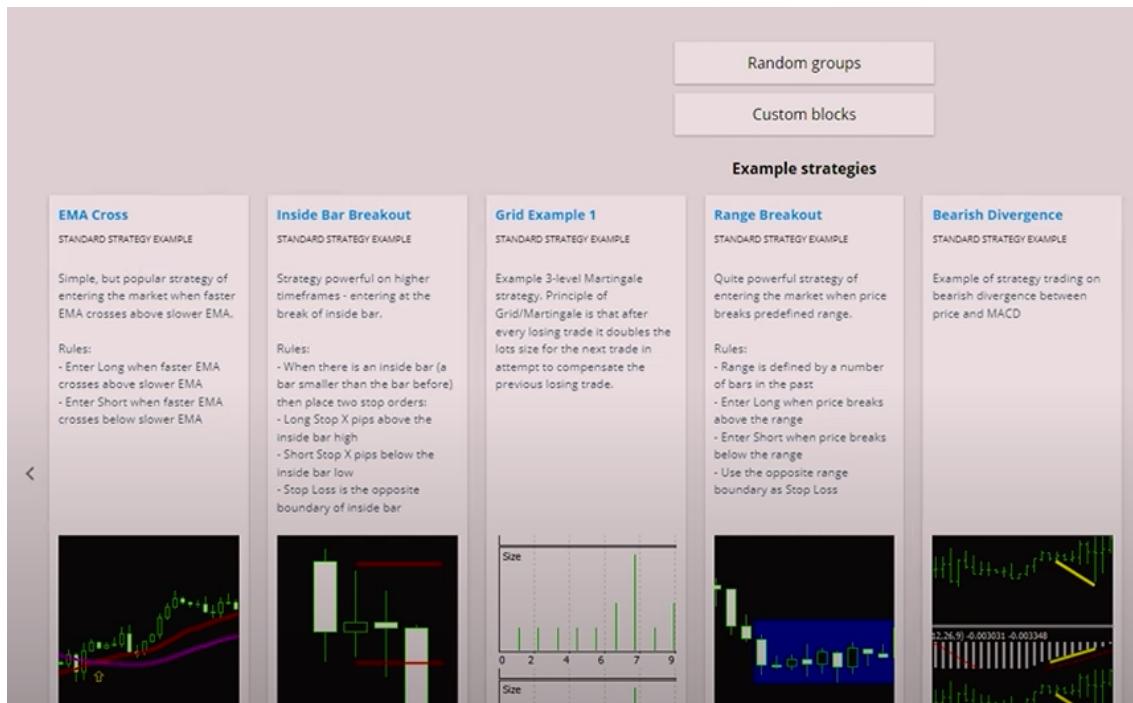


Figura 3: Estrategias predefinidas strategyQuant

En la figura 3 se puede apreciar las estrategias predefinidas que ofrece la aplicación, cada una de estas estrategias son programadas con anterioridad, se desconoce el lenguaje de programación y donde ejecuta dichas estrategias.

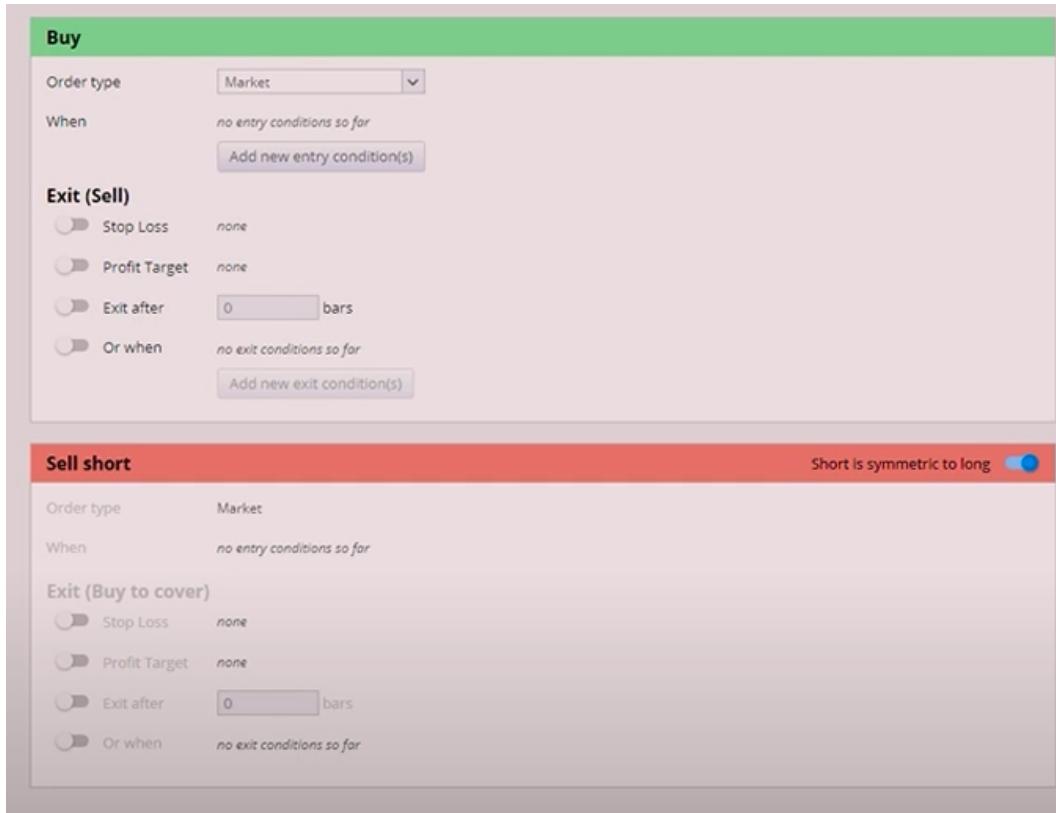


Figura 4: Parametros de una estrategia predefinida strategyQuant

Una vez seleccionada la estrategia se muestra la figura 4 la interfaz donde se realiza la configuración de parametros, los puntos más óptimos para realizar una compra o una venta, el simbolo donde realizar la estrategia y todo lo necesario para despues ofrecer un backtesting.

StrategyQuant X Ultimate Build 135 (Full license)											
Strategy Quant											
 Data Manager  Getting started  Builder  Optimizer  Custom Projects  Algo Card 	Data Manager										
	Data sources	Export	Tools	Instruments and Sessions	External indicators						
	 Dukascopy data	 TickDownloader import	 File import	 SQ Equity data	 SQ Futures data	 Darwinex Tick Data	 Crypto	 Yahoo	 Update all	 Update selected	 Mass delete
	 Data	 Instruments	 Sessions	 External indicators	 Log						
	Data are imported historical data into the program. Data manager allows you to import history from files, or to download it directly from online sources. Every data is linked with instrument that contains the definition of the imported symbol.										
	Filter items	All data sources	All data types	Records: 185							
	Symbol Name	Instrument	Underlying Symbol	Timeframe	Timezone	Data from	Date to	Total Days	Total Records	Source	Data type
	CL - NYMEX	@CL (@CL - Crude Oil West ...	M1	Exchange	2009.1.1	2023.7.11	5305	4895898	SQ Futures data	Futures	
	@CLD	@CLD (@CLD - Crude Oil We...	D1	Exchange	2000.12.29	2023.7.11	8229	5657	SQ Futures data	Futures	
	@E6	@E6 (@E6 - Euro FX - Cont...	M1	Exchange	2009.1.2	2023.7.11	5304	4967466	SQ Futures data	Futures	
	@EC	@EC (@EC - E-mini CME ...	M1	UTC-06 Centr...	2001.5.13	2023.2.9	7942	6918155	File	Futures	
	@ES	@ES (@ES - E-mini S&P 500 ...	M1	Exchange	2009.1.2	2023.7.11	5304	5028507	SQ Futures data	Futures	
	@ES.D	@ES.D (@ES - E-mini S&P 50...	D1	Exchange	2000.12.29	2023.7.11	8230	5672	SQ Futures data	Futures	
	@MF	@MF (@MF - eMicro EUR/USD ...	M1	Exchange	2009.3.23	2023.7.11	5224	3440130	SQ Futures data	Futures	
	@NM	@NM (@NM - eMicro Nasd...)	M1	Exchange	2009.1.2	2023.7.11	5304	4718848	SQ Futures data	Futures	
	@NQ	@NQ (@NQ - E-Mini Nasda...)	M1	Exchange	2009.1.2	2023.7.11	5304	4761568	SQ Futures data	Futures	
	@NQ.D	@NQ.D (@NQ - E-Mini Nasd...)	D1	Exchange	2000.12.29	2023.7.11	8230	5672	SQ Futures data	Futures	
	ADAUSDT_Binance	ADAUSDT (Binance)	M1	UTC	2018.4.17	2023.7.12	1912	2747136	Cryptocurrency	Crypto	
	ADAUSDO_Bitfinex	ADAUSDO (Bitfinex)	M1	UTC	2020.8.6	2023.7.12	1070	890575	Cryptocurrency	Crypto	
	AUDCAD_DWX	AUDCAD(D)	AUDCAD	TICK	UTC	2017.10.1	2023.7.12	2111	212033450	Darwinex	Forex
	AUDCAD_DWX_TICK	AUDCAD	AUDCAD	TICK	UTC	2017.10.1	2023.7.12	2111	212033450	Darwinex	Forex
	AUDCAD_DWX_TICK_TL...	AUDCAD	Clone of AUDCAD_DWX_TICK	TICK	EST+07	2017.10.2	2023.7.13	2111	212033450	Darwinex	Forex
	AUDCAD_Dukascopy_M1	AUDCAD	AUDCAD	M1	UTC	2006.3.10	2023.7.12	6334	5303444	Dukascopy	Forex
	AUDCAD_Dukascopy_M...	AUDCAD	Clone of AUDCAD_Dukasco...	M1	EST+07	2006.3.10	2023.7.13	6335	5303444	Dukascopy	Forex
	AUDCHF_DWX_TICK	AUDCHF	AUDCHF	TICK	UTC	2017.10.1	2023.7.12	2111	220413070	Darwinex	Forex
	AUDCHF_DWX_TICK_TL...	AUDCHF	Clone of AUDCHF_DWX_TICK	TICK	EST+07	2017.10.2	2023.7.13	2111	220413070	Darwinex	Forex
	AUDCHF_Dukascopy_M1	AUDCHF	AUDCHF	M1	UTC	2006.3.10	2023.7.12	6334	5288186	Dukascopy	Forex
	AUDCHF_Dukascopy_M...	AUDCHF	Clone of AUDCHF_Dukasco...	M1	EST+07	2006.3.10	2023.7.13	6335	5288186	Dukascopy	Forex
	AUDJPY_DWX_TICK	AUDJPY	AUDJPY	TICK	UTC	2017.10.1	2023.7.12	2111	291050389	Darwinex	Forex
	AUDJPY_DWX_TICK_TIC...	AUDJPY	Clone of AUDJPY_DWX_TICK	TICK	EST+07	2017.10.2	2023.7.13	2111	291050389	Darwinex	Forex
	AUDJPY_Dukascopy_M1	AUDJPY	AUDJPY	M1	UTC	2003.12.1	2023.7.12	7164	7336236	Dukascopy	Forex
	AUDJPY_Dukascopy_M...	AUDJPY	Clone of AUDJPY_Dukascop...	M1	EST+07	2003.12.1	2023.7.13	7165	7336236	Dukascopy	Forex
	AUDNZD_DWX_TICK	AUDNZD	AUDNZD	TICK	UTC	2017.10.1	2023.7.12	2111	200100493	Darwinex	Forex
	AUDNZD_DWX_TICK_TL...	AUDNZD	Clone of AUDNZD_DWX_TICK	TICK	EST+07	2017.10.2	2023.7.13	2111	200100493	Darwinex	Forex
	AUDNZD_Dukascopy_M1	AUDNZD	AUDNZD	M1	UTC	2006.3.10	2023.7.12	6335	5288186	Dukascopy	Forex

Figura 5: Historico de precios strategyQuant

En la figura 5 se muestra un subapartado del menu que se ha mencionado anteriormente "data manager", en esta parte se muestra el contenido de los datos que proporciona la aplicación, dicho de otra manera, el historico de precios de cada simbolo con sus respectivos timeframes, con esta informacion obtenemos las pruebas de backtesting.

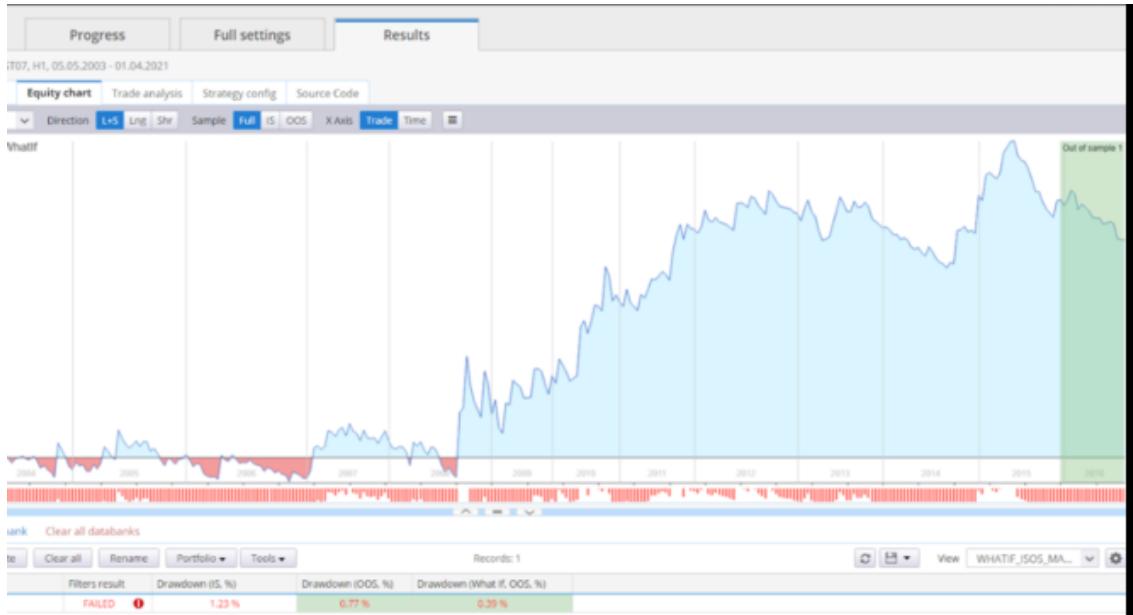


Figura 6: Backtesting strategyQuant

Si visualizamos la figura número 6 se ha realizado un backtesting a través de una estrategia predefinida y una configuración personalizada por el cliente, el gráfico muestra el proceso del balance de una cuenta tras realizar operaciones con dicha estrategia predefinida.

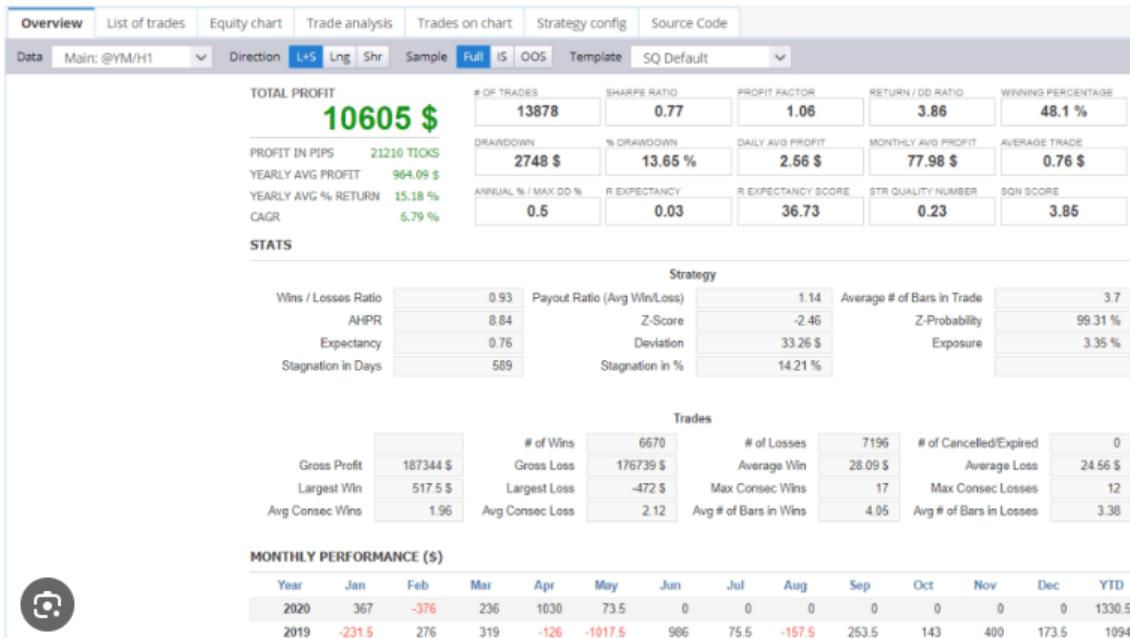


Figura 7: Estadísticas del backtesting strategyQuant

Por último en la figura 7 se muestra estadísticas obtenidas del backtesting donde se obtiene datos relevantes para determinar si una estrategia es apta para poner en producción en una operativa de trading.

Limitaciones en StrategyQuant

A pesar de su buena respuesta, hay ciertas deficiencias en el propio StrategyQuant que llevan a la creación de una alternativa como la que se propone en este proyecto:

- **Complejidad de uso inicial:** Aunque está orientado a aquellos usuarios que no programan, la plataforma tiene una curva de aprendizaje muy pronunciada para los usuarios principiantes en trading algorítmico.
- **Excesiva automatización:** Se pierde control del sistema a nivel detallado y el usuario principiante puede confundirse o malinterpretar funcionalidades del sistema.
- **Enfoque cerrado:** Las estrategias generadas o modificadas quedan encerradas en el propio entorno de StrategyQuant, obstaculizando su exportación o el análisis fuera del entorno, no se sabe con certeza qué hace la estrategia si no se visualiza una simulación detrás.
- **Coste elevado:** Los precios de licencia dificultan su uso por parte de usuarios independientes o pequeños inversores.

En comparación con StrategyQuant, la aplicación de este proyecto está orientada a evaluar y optimizar estrategias previamente desarrolladas, con StrategyQuant a parte de eso también se puede obtener optimizaciones sobre estas estrategias predefinidas.

Enfoque en la generación de estrategias:

StrategyQuant permite generar estrategias básicas desde cero utilizando inteligencia artificial, aunque el mayor uso de la aplicación son para estrategias predefinidas. La aplicación desarrollada se centra en evaluar y optimizar las estrategias previamente desarrolladas en un entorno controlado.

Accesibilidad y facilidad de uso:

StrategyQuant es una plataforma de escritorio con una interfaz técnica que está orientada a traders más experimentados. La aplicación desarrollada es una plataforma de escritorio, intuitiva y de uso simple para trades con una leve experiencia previa.

Funcionalidad de backtesting:

Ambas plataformas realizan backtesting de estrategias, aunque StrategyQuant lo hace desde un enfoque mucho más automatizado y avanzado mediante el uso de modelos de inteligencia artificial. La aplicación desarrollada permite llevar a cabo backtesting en el entorno de MetaTrader 5 con las estrategias predefinidas y entrega estadísticas detalladas sin necesidad de un tipo de configuración avanzada.

Modelo de precios y acceso:

StrategyQuant tiene un alto precio, que presenta pagos únicos que oscilan entre 1290\$ y 2900\$. La aplicación desarrollada podría tomar otro modelo de precios más flexible o gratuito dependiendo de su modelo de negocio.

Otro ejemplo de aplicación similar es **QuantConnect** [24].

QuantConnect es otra aplicación para la construcción y el backtesting de sistemas automáticos de trading algorítmico enfocados principalmente al trading de acciones, forex y criptomonedas. Sus características son la organización del desarrollo de estrategias, utilizando para ello lenguajes de programación como C y Python, la posibilidad de acceder a una gran cantidad de datos históricos que abarcan una gran cantidad de activos, la ejecución de backtesting y en la posibilidad de integrarse con brokers como Interactive Brokers o GDAX.

Las diferencias con respecto al proyecto que se ha llevado a cabo mediante este trabajo son que QuantConnect exige un nivel alto de conocimientos de programación, razón por la que su acceso estaría limitado a usuarios con un perfil técnico bajo. Además está más orientado a programadores que a traders que quieren trabajar directamente en sistemas de trading ya predefinidos, con un modelo de uso por suscripción mensual con diferentes restricciones en función de los planes que se contraten.

Por el contrario, la aplicación desarrollada en este proyecto no exige el conocimiento de programación al trabajar exclusivamente sobre estrategias de trading predefinidas y que son configurables de manera sencilla para la evaluación y optimización posterior de una estrategia. Con ello se permite un acceso más sencillo para traders que tengan escasa experiencia en programación o en sistemas técnicos complejos.

Sabiendo estos aspectos del sistema de StrategyQuant y QuantConnect podemos realizar una aplicación que resuelva estos problemas y obtener una mejor versión que gratifique la experiencia del usuario.

3 Planificación, metodología y presupuesto

3.1. Planificación

La planificación ha sido realizada de manera controlada y progresiva, equilibrando el trabajo técnico con la supervisión gracias a mi tutora que ha ido controlando el avance del mismo a través de tutorías periódicas. Desde un principio se establecieron unas fechas de entrega sobre el desarrollo cada tres semanas que facilitaban la entrega del trabajo por fases y permitir el avance de forma progresiva.

Esta planificación ha seguido un esquema temporal por bloques, es decir, que cada etapa de trabajo tenía como cometido completar una parte del Trabajo de Fin de Grado y, gracias a este control, se podían detectar a tiempo algunos posibles errores a futuro, manteniendo la línea de desarrollo planificada con relación a los objetivos definidos en el inicio del trabajo.

Las fases más relevantes en la planificación han sido las siguientes:

- Análisis y estudio previo: Estado del arte, definición de los objetivos y recogida de requisitos
- Diseño de la solución: Arquitectura del sistema, definición de los módulos del mismo y estructuración de la base de datos.
- Desarrollo iterativo: Desarrollo gradual de los diferentes microservicios y de la interfaz de usuario (UI).
- Pruebas y validación: Comprobaciones del funcionamiento del sistema, con especial atención a la automatización de backtesting.
- Redacción de la memoria: Documentación del Proyecto, explicaciones técnicas y presentación de resultados.

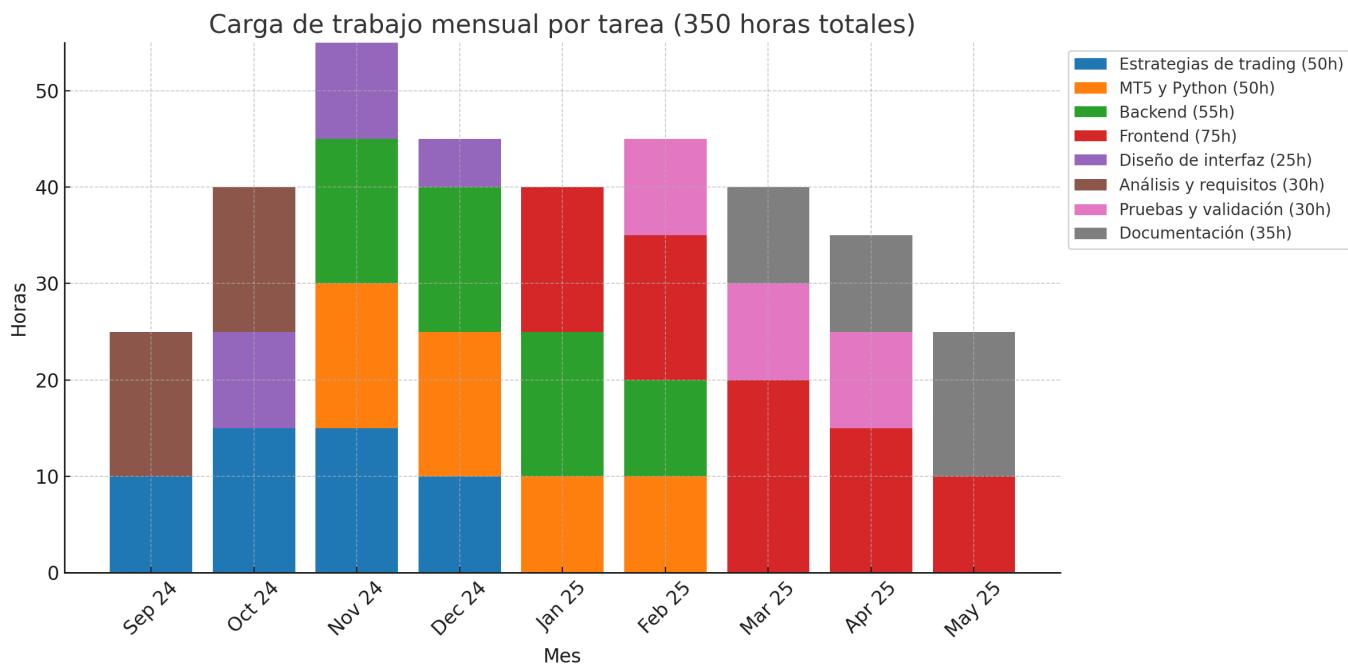


Figura 8: Carga de trabajo mensual

Como se puede apreciar en la figura 8, esta gráfica es la carga de trabajo mensual desde que se dio inicio al proyecto, estas son las horas invertidas por mes con sus respectivos roles o trabajos específicos, en los siguientes apartados se entrara más en detalle sobre esta gráfica, quiero recalcar que esta es una estimación aproximada a la real, es decir, no se tiene con exactitud las horas invertidas de cada mes pero si es un valor muy próximo al real.

3.2. Metodología

La metodología empleada en este proyecto, podría decirse, es un diseño basado en prototipos. En cada iteración del desarrollo, se buscó construir una versión funcional del sistema, esto validado con la tutora, en tutorías programadas.

Este enfoque permitió evolucionar paso a paso el diseño de la interfaz y también, la estructura técnica, haciendo mejoras en prototipos incrementales, algo muy útil para ajustar requisitos y para pulir detalles de la usabilidad.

La metodología de trabajo aplicada ha sido iterativa e incremental, el desarrollo del sistema se ha realizado en fases para poder llevar el desarrollo favorablemente. Cada una de las iteraciones ha supuesto un ciclo completo, esto es la planificación, desarrollo y revisión, lo que ha favorecido la mejora del proyecto de manera continua.

Las tutorías con la tutora se han mantenido aproximadamente cada tres semanas, como punto de control del avance de la realización, de la solución de problemas y de la reordenación de prioridades, lo que ha permitido asegurar continuidad en el desarrollo y ha favorecido una correcta resolución de la metodología para abarcar proyecto final.

En términos técnicos se ha utilizado una lógica modular en la solución, aplicando principios del diseño orientado a servicios (microservicios) y la clara separación de responsabilidades entre componentes: frontend, backend, backtesting y sistema de autenticación.

En el desarrollo de la solución también se han utilizado herramientas de control de versiones, pruebas en local y simulaciones sobre datos históricos para validar el comportamiento del sistema en condiciones distintas.

Todo esto ha permitido que la forma de trabajo haya sido flexible y favorable a futuros cambios, manteniendo los conceptos claros desde las primeras fases del proyecto.

- **13 de septiembre de 2024:** Primera reunión de contacto sobre el proyecto. Se expuso la idea básica del proyecto, una aplicación web que va dedicada al análisis de estrategias de trading. Se hablaron posibles tecnologías a usar y se decidió llevar a cabo el estudio previo del estado del arte.
- **4 de octubre de 2024:** Reunión centrada en la definición de los objetivos y requisitos. Se revisaron los principales módulos del sistema, como la conexión con MT5, la gestión de estrategias, la interfaz web, se fijaron los objetivos del proyecto y se plantearon los requisitos funcionales y no funcionales.
- **28 de octubre de 2024:** Presentación del primer diseño de la arquitectura del sistema. Se definieron ciertos conceptos sobre microservicios, separación entre frontend y backend, y se decidieron herramientas como Prisma y PostgreSQL.

- **18 de noviembre de 2024:** En esta reunión se revisa el diseño de la interfaz a partir de los prototipos y mockups. Se valida la usabilidad de la interfaz propuesta, la estructura de la navegación, se incorpora pequeñas mejoras visuales para facilitar el análisis de datos por parte del usuario.
- **12 de diciembre de 2024:** Reunión dedicada a la revisión de los diseños técnicos. Se comentaron los puntos detallados de la base de datos y la estructura de los microservicios. Se propondrían correcciones sobre el modelo de datos y de la comunicación entre servicios.
- **16 de enero de 2025:** Revisión del desarrollo, el backend básico, el sistema de usuarios y el microservicio que conecta con MetaTrader 5. Se mostraban ejemplos de backtesting y las primeras pruebas en el entorno de datos reales.
- **7 de febrero de 2025:** Se comentó el desarrollo relacionado con el frontend, la visualización de la estadística y la interacción del usuario como estrategias. La tutora propuso pequeñas mejoras funcionales, se planificaron los últimos módulos que quedaban para terminar el desarrollo técnico.
- **17 de marzo de 2025:** Tutoría centrada en las correcciones finales y pruebas. Se comprobó el funcionamiento completo de la aplicación desde el backtesting hasta la visualización de los resultados. Se llevaron a cabo aportaciones para las mejoras opcionales y se proyectó el inicio de la redacción del informe.
- **9 de mayo de 2025:** Tutoría para finalizar el informe del TFG. Se revisó el informe completo periódicamente vía email, se hicieron observaciones sobre la redacción y estructura, se logró confirmar que se habían alcanzado satisfactoriamente los objetivos técnicos del proyecto.

3.3. Presupuesto

La estimación aproximada sobre el coste en desarrollar este proyecto en base a los roles de especialización de informática en España y suponiendo una previa experiencia en desarrollo fullstack y en el ámbito del trading, se podrá realizar una estimación del presupuesto. La elaboración de este proyecto permite comprender del esfuerzo invertido y permite obtener una imagen aproximada del valor que podría tener el desarrollo de un proyecto de estas características si se desarrolla en un entorno profesional, aclarar que he desempeñado todos los perfiles técnicos.

Mano de obra

Se muestra la participación de varios perfiles técnicos, cada uno de ellos cumpliendo con funciones concretas: desde la definición e implementación de las estrategias de trading, pasando por la creación del backend, el desarrollo del frontend, la integración de los servicios, el diseño de la interfaz, la validación y las pruebas, para terminar con la redacción de la documentación del proyecto. Se calcula un total de 350 horas de trabajo, distribuidas del siguiente modo:

- Diseño e implementación de estrategias de trading: 50 horas [8]
- Integración con MetaTrader 5 y desarrollo en Python: 50 horas
- Desarrollo backend (API, base de datos con Prisma y PostgreSQL): 55 horas
- Desarrollo frontend (Next.js, visualización de datos): 75 horas
- Diseño de interfaz y prototipos: 25 horas
- Análisis de requisitos e investigación: 30 horas

- Pruebas y validación del sistema: 30 horas
- Redacción de la documentación y memoria: 35 horas

Para validar el precio de la mano de obra se ha obtenido información de una página web muy popular por su autenticidad en sus datos e informes con valoraciones positivas de usuarios, se llama **glassdoor** [8].

Se ha desempeñado una asignación de cada tarea a un rol técnico especializado, con precios adaptadas a las del mercado freelance en España, considerando una tarifa mínima de 25€/hora por obtener una experiencia previa en estas tecnologías e información. Las tarifas han estado en un rango de 25€/hora (para las tareas de tipo general o de documentación) a 35€/hora (con respecto al perfil de trader algorítmico como junior). Por lo tanto, tomando en cuenta las tarifas, se calcula que el coste del total de la mano de obra es aproximadamente a 9.825€.

Equipo informático y software

El desarrollo del proyecto ha podido llevarse a cabo principalmente gracias al uso de herramientas que son gratuitas, tanto en infraestructura como en software. Aún así, se considerará como un conjunto base e inicial el equipamiento informático básico, que podría incluir en mi caso con un total de 2150€:

- Ordenador portátil de 700€ para realizar el desarrollo de la aplicación de forma remota.
- Ordenador sobremesa de 1200€ incluyendo teclado, ratón y monitor para realizar el desarrollo de la aplicación y también aprovechar el rendimiento para obtener mejores rendimientos en el backtesting de las estrategias de trading.
- En cuanto a software, se ha utilizado el editor Visual Studio Code, Prisma, MetaTrader 5, Next.js, PostgreSQL, Vercel, entre otras tecnologías, todas en versiones gratuitas, por lo que el coste en licencias ha sido nulo, por tener privilegios de estudiante y desarrollador, una estimación del precio sin estos privilegios son 250€.

Otros gastos asociados al desarrollo y despliegue

Aunque el mayor costo del proyecto está ligado a la mano de obra, un proyecto real tiene más gastos relacionados, considerando para hacer un presupuesto general de forma realista. A continuación, esta es la lista con un total de 2.775€:

- Infraestructura de servidores 600 €: Durante el desarrollo se usaron plataformas gratis como Vercel. Pero, en producción, se requieren servidores cloud escalables para alojar microservicios y ejecutar backtestings simultáneos si tenemos en cuenta que obtenemos una gran cantidad de usuarios iterando en la página web. Unos 50 € mensuales por un año, aproximadamente, unos 600 €.
- Gastos de suscripciones y servicios externos 300 €: Pese a que las herramientas de desarrollo son, en su mayoría gratis, se piensa en usar servicios externos para monitoreo, copias de seguridad o análisis de logs, unos 300 € anuales.
- Amortización del equipo informático para mantenimiento y mejoras 350 €: Los equipos de desarrollo portátil y sobremesa, en un ámbito profesional tienen una amortización estimada en 4 años si usan herramientas que necesiten gran rendimiento en sus procesos. Se calcula para un año de proyecto unos 350 €.
- Gastos generales del entorno de trabajo (500 €): Considerados costes indirectos, como conexión a internet, la electricidad, y costos por el uso del espacio de trabajo, coworkings o zona de alquiler, se espera un total de 500 €.

- Pruebas de escalabilidad y entorno cloud para usuarios (300 €): A fin de validar el rendimiento de la aplicación en escenarios con muchos usuarios concurrentes ejecutando estrategias con simulaciones para prevenir casos extremos donde se pone a prueba el rendimiento de la aplicación y los posibles problemas a futuro, un coste alrededor de 300 €.

Resumen

Tarea	Horas	Rol	Tarifa (€ / h)	Coste (€)
Diseño e implementación de estrategias de trading	50	Trader algorítmico	35	1.750
Integración con MT5 (Python y conexión local)	50	Desarrollador software Python, scripts..	28	1.400
Desarrollo Backend (API, Prisma, PostgreSQL)	55	Desarrollador Backend	30	1.650
Desarrollo Frontend (Next.js, visualización)	75	Desarrollador Frontend	27	2.025
Diseño de interfaz y prototipos	25	Diseñador UI / UX	25	625
Investigación y análisis de requisitos	30	General	25	750
Pruebas, validación y optimización	30	General	25	750
Redacción de memoria y documentación técnica	35	General	25	875
Total	350	—	—	9.825

Cuadro 1: Resumen del presupuesto sobre el desarrollo

Concepto	Coste (€)
Ordenador portátil (desarrollo remoto)	700
Ordenador sobremesa (desarrollo y backtesting)	1200
Coste estimado de licencias de software (sin privilegios)	250
Total equipo y software	2.150

Cuadro 2: Resumen de presupuesto de equipo informático y software

Concepto	Coste (€)
Infraestructura de servidores cloud	600
Suscripciones y servicios externos (monitorización, backup)	300
Amortización parcial del equipo informático (un año)	350
Gastos generales de entorno de trabajo	500
Pruebas de escalabilidad en entorno cloud	300
Total otros gastos	2.050

Cuadro 3: Resumen de otros gastos asociados

Concepto	Coste (€)
Mano de obra (desarrollo técnico)	9.825
Equipo informático y software	2.150
Otros gastos asociados	2.050
Presupuesto total estimado	14.025

Cuadro 4: Presupuesto global estimado del proyecto

Por lo tanto concluimos con los cuadros 1, 2, 3 y 4 que el presupuesto total del proyecto ronda aproximadamente de 14.025€.

4 Análisis del problema

4.1. Descripción del problema a resolver

En el escenario de las estrategias de trading algorítmico, uno de los principales problemas a los que se enfrenta la comunidad de inversores y desarrolladores de estrategias es la dificultad de poder evaluar de forma rápida, visual, centrada e integrada el rendimiento de los distintos sistemas de inversión automatizados.

Probablemente el hecho de que las estrategias de trading sean cada vez más complejas y personalizadas implica que también se haya hecho necesario contar con herramientas que nos permitan automatizar el análisis de estas estrategias, almacenar los resultados de los backtesting y obtener los datos clave como por ejemplo ganancias, pérdidas, ratios de rendimiento o drawdowns.

Históricamente esta evaluación de rendimiento se ha llevado a cabo a partir de plataformas como por ejemplo MetaTrader 5 que nos ofrece funcionalidad para poder desarrollarlas y testearlas de forma individual, también presenta limitaciones como por ejemplo no puedes evaluar varias estrategias a la vez, esta plataforma de forma optima y eficiente una estrategia que se crea con anterioridad.

Requerir interacciones constantes con la plataforma en local, la ejecución manual de los backtests y el análisis de las versiones generadas, a menudo es un problema para los traders que requieren de estas estadísticas. Tal y como se exponía anteriormente, esta situación aumenta la necesidad de tiempo en la validación de las estrategias y reduce la agilidad en la toma de decisiones por parte del trader o desarrollador.

Por lo tanto, el reto que se presenta en este proyecto es unificar y mostrar visualmente de forma simple todo el contenido, esto permite:

- Ejecutar automáticamente procesos de backtesting sobre estrategias previamente desarrolladas.
- Concentrar los resultados en una aplicación web.
- Representar estadísticas relevantes para cada estrategia de la forma más directa y ordenada posible.
- El usuario podrá acceder a diferentes estrategias y diferentes símbolos para obtener una variedad de resultados según su criterio del mercado.

El propósito es la de convertirse en una aplicación web que sea la interfaz centralizada y automatizada para gestionar, analizar y comparar estrategias de trading programadas, consumiendo menos tiempo en el aprendizaje, facilitando la representación de los resultados y potenciando la eficiencia en la toma de decisiones financieras.

4.2. Requisitos funcionales

Antes de redactar los requisitos funcionales, especificar que se ha optado por una organización basada en los módulos principales de la aplicación. Dividimos todo en: usuarios (registro, autenticación, panel), probador de estrategias (estrategias, parámetros de cuenta, parámetros de la estrategia), resultado del backtesting y el historial de configuraciones del usuario.

1. Usuarios

a) Registro de usuarios

Descripción: Los usuarios requieren de un registro para acceder al probador de estrategias.

Características:

- El usuario proporcionará:
 - Correo electrónico
 - Contraseña
 - Por ultimo verificación por correo

b) Inicio de sesión o Login

Descripción: Los usuarios deben poder iniciar sesión.

Características:

- proporcionará correo electrónico y contraseña.
- Opción iniciar sesión directamente con su cuenta de Google o github.
- Autenticación segura.

c) Panel de usuario

Descripción: Después de iniciar sesión, el usuario accede a su panel o dashboard donde puede utilizar el probador de estrategias.

Características:

- Visualización del perfil de usuario.
- Acceso al historial de estrategias guardadas.
- Acceso al probador de estrategias.

2. Probador de estrategias de trading

a) Selección de estrategia

Descripción: El usuario selecciona una estrategia de entre varias opciones disponibles, previamente debe estar registro para poder realizar esta acción.

Características:

- Listado de estrategias predefinidas disponibles.
- Botón de seleccionar para elegir la estrategia deseada.

b) Selección de símbolo

Descripción: El usuario debe poder seleccionar un símbolo de trading para después realizar un backtesting (por ejemplo, EUR/USD, BTC/USD...).

Características:

- Desplegable de selección de símbolos.
- Los símbolos disponibles se mostrara en una sección de la pagina principal y dependerá del broker y la compañía de trading en este caso empresa de fondeo.

c) Ingreso de parametros relacionados con la cuenta que se quiere probar, estos parametros son comunes para todas las estrategias

Descripción: El usuario ingresará la cantidad de dinero a simular, divisa por ejemplo en el dinero en dolares o en euros, apalancamiento para casos donde se requiera una mayor inversión, este afectara los beneficios y las perdidas, y fecha para el backtesting donde se realiza la simulación.

Características:

- Campo para ingresar el dinero inicial que desea utilizar.
- Parámetros adicionales como apalancamiento, divisa, fecha de inicio y fecha fin del test.

d) Ingreso de parámetros de la estrategia

Descripción: El usuario proporcionará parámetros específicos para la estrategia seleccionada (por ejemplo, stop loss, take profit, etc.).

Características:

- Formulario dinámico que cambia según la estrategia seleccionada.
- Ejemplos de parámetros:
 - Stop Loss
 - Take Profit
 - Hora y cierre para realizar operaciones
 - Indicadores adicionales según la estrategia.

3. Ejecución y Resultados

a) Ejecución del backtesting

Descripción: El sistema ejecuta el backtesting basado en la estrategia seleccionada y los parámetros proporcionados por el usuario.

Características:

- El backtesting procesa los datos históricos del símbolo seleccionado.
- Muestra estadísticas relevantes:
 - Beneficio bruto y neto.
 - Drawdown máximo.
 - Número de operaciones.
 - Porcentaje de aciertos en total, solo compras o solo ventas.
 - Los previamente mencionados son los más relevantes, se pondrá todos los datos que sean beneficiario para obtener un estudio sobre la estrategia.

b) Visualización de resultados

Descripción: El usuario visualiza los resultados del backtesting después de la ejecución.

Características:

- Muestra en pantalla los resultados calculados.
- Representación gráfica del rendimiento.
- Botón para guardar en el historial la configuración de la estrategia si los resultados son satisfactorios.

c) Guardado de resultados y parámetros

Descripción: Si el usuario está satisfecho con los resultados, puede optar por guardar en el historial la configuración de la estrategia.

Características:

- Opción de guardar la estrategia y parámetros personalizados.
- Se almacenarán los datos para que puedan ser utilizados en futuras ejecuciones.

4. Historial de configuraciones de estrategias guardadas

a) Carga de configuraciones de estrategias guardadas

Descripción: El usuario puede cargar configuraciones de estrategias guardadas previamente en su historial.

Características:

- Listado de las estrategias guardadas con sus configuraciones.
- Selección rápida de una estrategia para reutilizarla.
- Los parámetros guardados de la estrategia se llenarán automáticamente en el probador de estrategias.
- El usuario tiene opción de eliminar esa configuración de una estrategia guardada si ya no es de su agrado.

4.3. Requisitos no funcionales

1. Rendimiento

La ejecución del backtesting debe ser rápida y eficiente, con tiempos de respuesta razonables teniendo en cuenta que las ejecuciones serán a través de un ordenador local.

2. Interfaz de Usuario y accesibilidad

La interfaz debe ser amigable, clara y con un diseño simple e intuitivo que facilite la navegación y el uso para el usuario. Debe cumplir con las directrices de usabilidad de Nielsen, asegurando una experiencia de usuario óptima desde el diseño inicial.

También debe superar los test de accesibilidad, como los de tipo TAW, incluyendo:

- Texto alternativo en imágenes.
- Nombres de etiquetas intuitivos.
- Navegación optimizada.

3. Compatibilidad

La web debe ser compatible con diferentes navegadores y dispositivos, con diseño web específico para tablets y computadoras.

4. Seguridad y Privacidad de Datos

Se implementarán medidas de seguridad robustas para proteger la integridad y confidencialidad de los datos procesados por la herramienta. Esto incluye:

- Cifrado de los datos sensibles como la contraseña.
- Autenticación segura de usuarios.
- Protección contra accesos no autorizados.

4.4. Casos de uso

A continuación se muestran los casos de uso más relevantes que tiene un usuario a la hora de realizar las acciones en el sistema.

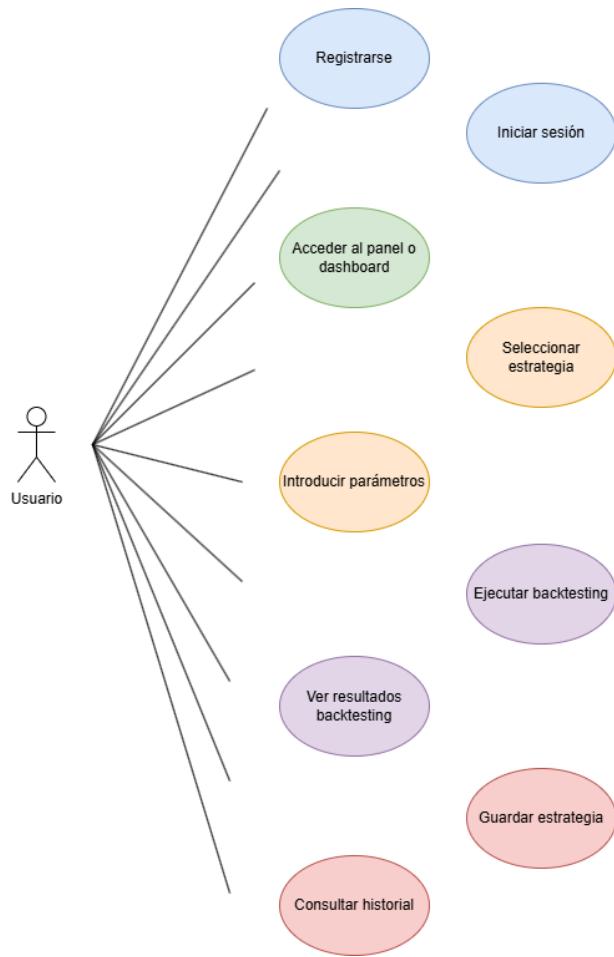


Figura 9: Diagrama general de casos de uso del sistema

En la figura 9 se muestra los casos de uso general del sistema y su explicación a continuación:

- 1. Registro de usuario:** El usuario entra a la plataforma se registra, donde ingresará el correo electrónico y la contraseña, después verifica con su correo electrónico. Al finalizar el registro, puede entrar a la plataforma en la sección del dashboard o panel con funcionalidades del sistema. En la siguiente figura se muestra el diagrama de caso de uso, color azul autenticación, color verde navegación y acceso, color naranja configuración de la estrategia, color morado ejecución y análisis, por ultimo color rojo historial y guardado.
- 2. Inicio de sesión:** El usuario inicia sesión a través del correo electrónico y la contraseña o con su cuenta de Google o Github. El sistema comprueba si es correcta la autenticación del usuario y si es así redirige al panel o dashboard.

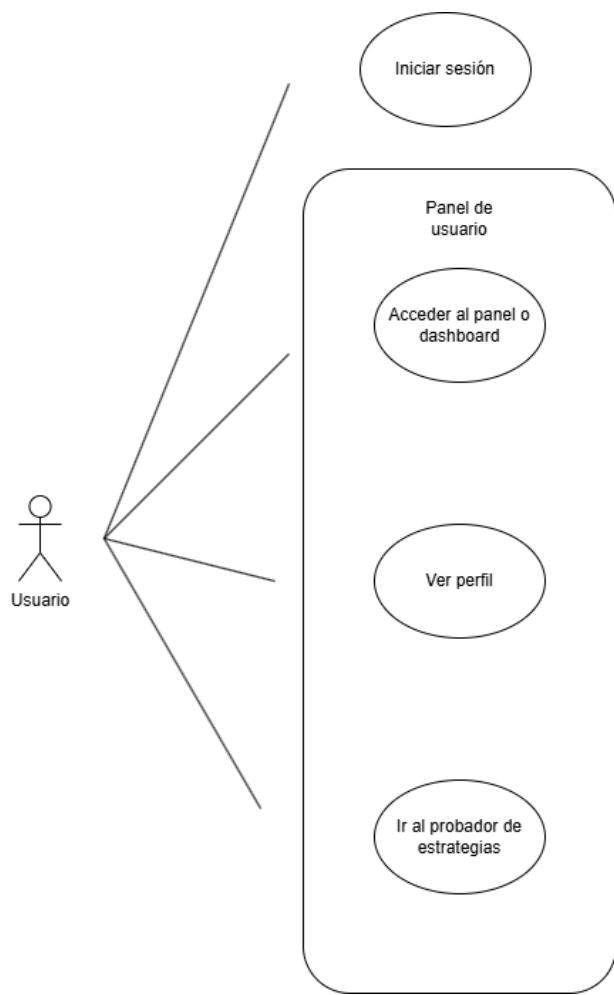


Figura 10: Caso de uso: Acceso al panel del usuario

3. **Acceso al panel del usuario:** La figura 10 muestra este caso de uso. Al iniciar sesión el usuario puede acceder al panel de usuario donde puede gestionar su perfil, ver el registro de estrategias guardadas con su configuración y entrar al probador de estrategias donde podrá realizar nuevas simulaciones. En la siguiente figura se muestra el diagrama de caso de uso.

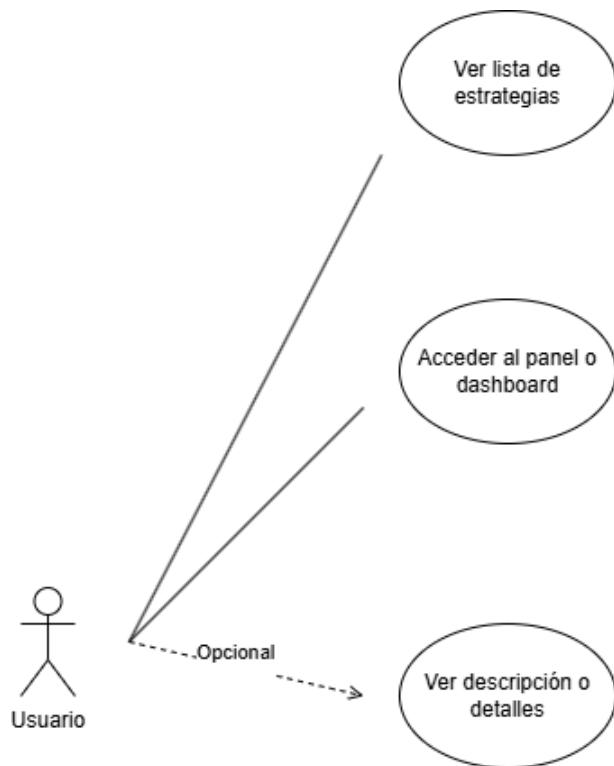


Figura 11: Caso de uso: Selección de estrategia de trading

4. **Selección de la estrategia de trading:** La figura 11 muestra este caso de uso. El usuario en el probador de estrategias puede ver un listado de estrategias disponibles. El usuario selecciona la estrategia que quiera probar. Elegirla activará el formulario para poder ingresar los parámetros que se necesitan. En la siguiente figura se muestra el diagrama de caso de uso.
5. **Formulario de parámetros:** El usuario llenará el formulario de la configuración de la estrategia para el resultado del backtesting, consta de 2 partes: la primera parte de la configuración es común para todas las estrategias e incluye parámetros como el balance de la cuenta, la divisa, fechas donde se realiza la simulación, etc. La segunda parte del formulario es específica según la estrategia seleccionada y puede incluir stop loss, take profit, ratio beneficio, entre otros.

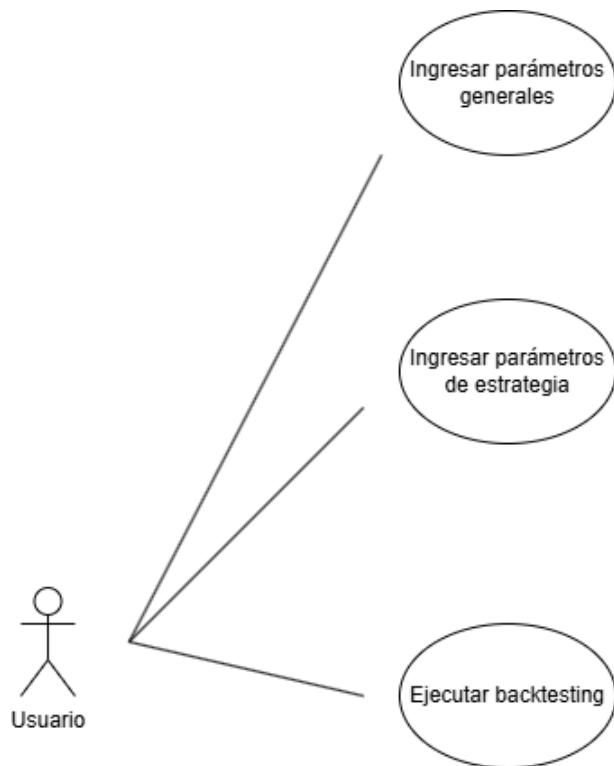


Figura 12: Caso de uso: Ejecución del backtesting

6. **Ejecución del backtesting:** La figura 12 muestra este caso de uso. Cuando el usuario ha introducido los datos, ejecuta el backtesting. El sistema procesa los datos históricos del símbolo elegido y lo simula ejecutando las operaciones con la estrategia. Se presentan estadísticas claves: beneficio neto, drawdown máximo, porcentaje de aciertos, entre otros. En la siguiente figura se muestra el diagrama de caso de uso.

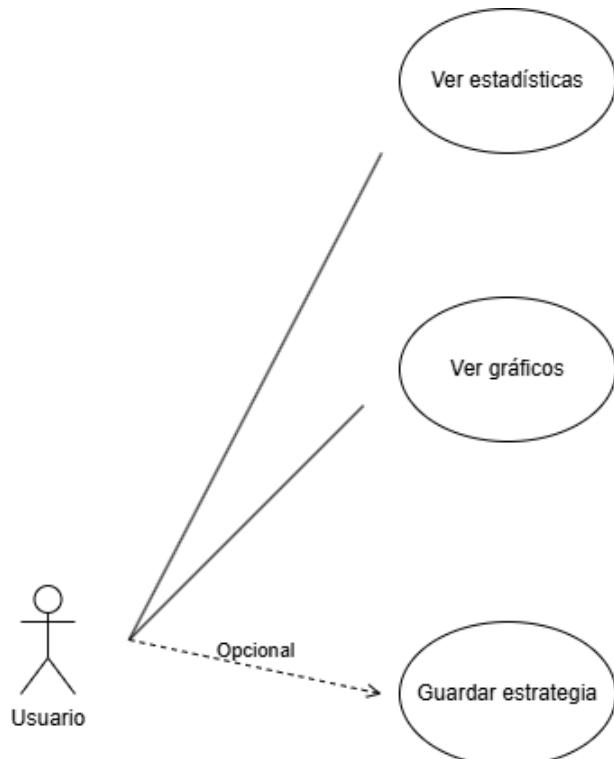


Figura 13: Caso de uso: Visualización de resultados

7. **Visualización de resultados:** La figura 13 muestra este caso de uso. Cuando acaba la simulación, el sistema le presenta los resultados del backtesting en formato de números y en gráficos. El usuario puede repasar la evolución y comprobar la eficacia de la estrategia. En la siguiente figura se muestra el diagrama de caso de uso.

8. **Guardar la configuración de la estrategia:** Si el usuario está satisfecho con los resultados puede guardar la estrategia y sus parámetros en su historial personal para que quede guardada y se la pueda reutilizar en el futuro.

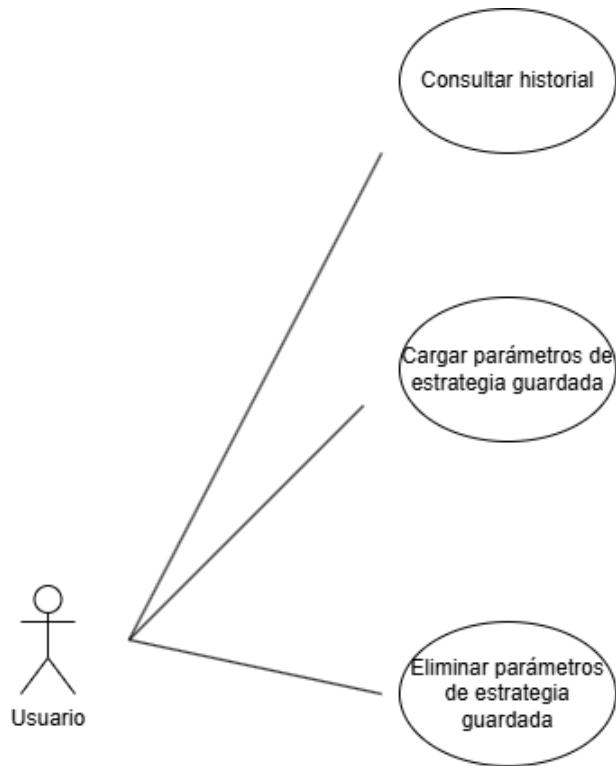


Figura 14: Caso de uso: Gestión de estrategias guardadas

9. **Gestión de estrategias guardadas:** La figura 14 muestra este caso de uso. El usuario tiene acceso a su historial y puede cargar una estrategia guardada, reutilizarla o borrarla. Puede obtener una nueva simulación modificando los parámetros a su criterio. En la siguiente figura se muestra el diagrama de caso de uso.

5 Diseño

5.1. Diseño arquitectónico

La arquitectura de la aplicación se basa en una arquitectura modular por medio de microservicios, logrando un mejor escalado y mantenimiento, además de permitir la separación de responsabilidades. En la siguiente figura se presenta un esquema representativo de la arquitectura general del sistema, en la que se puede observar la conexión de los diferentes módulos y tecnologías empleadas.

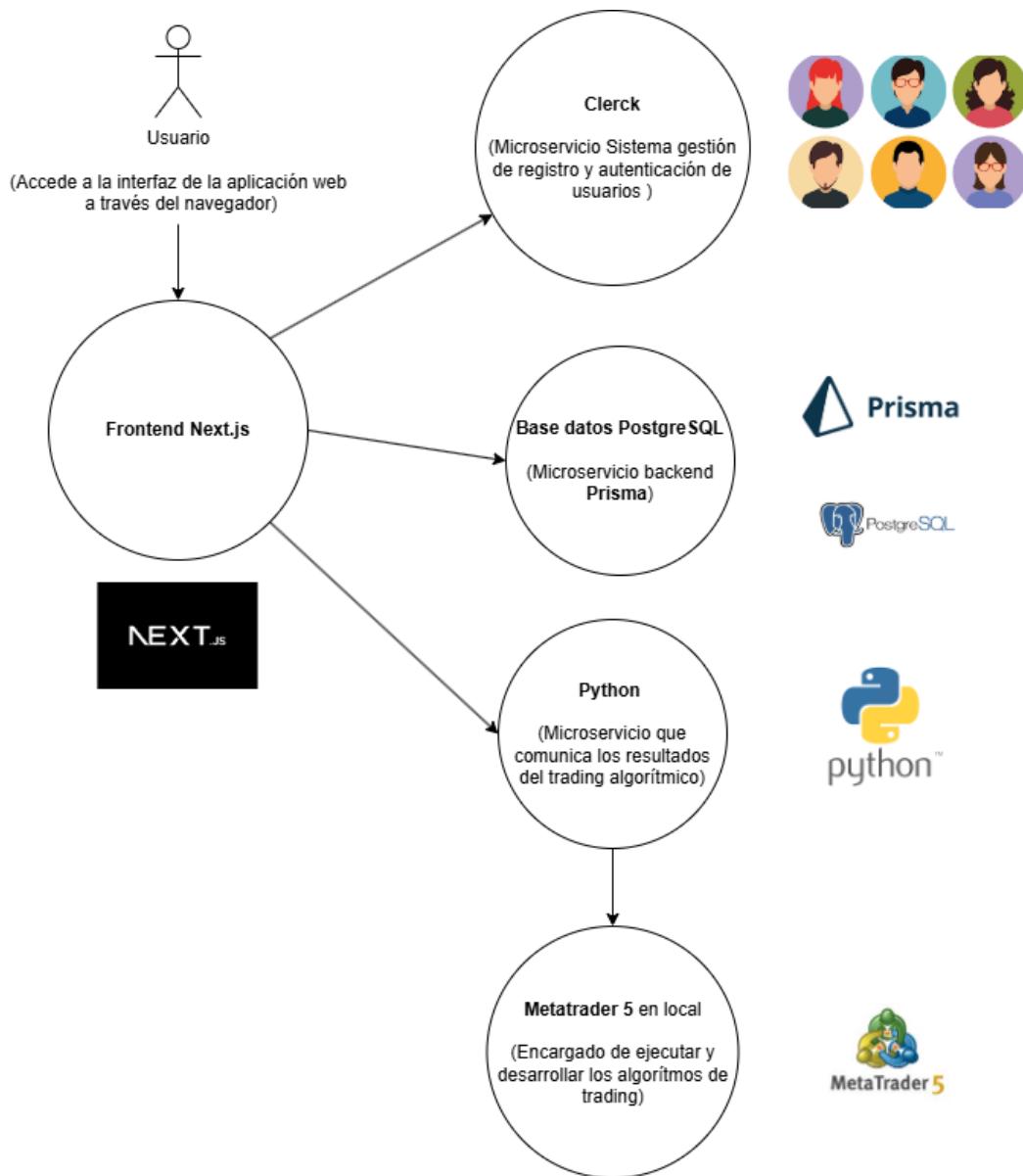


Figura 15: Diseño arquitectónico

La figura 15 muestra el diseño arquitectónico del sistema. A continuación se detalla la función de cada uno de los elementos que se representan:

- **Usuarios:** Son los protagonistas de la aplicación, se comunican con la aplicación a través del frontend, donde pueden observar estadísticas, analizar estrategias y ejecutar pruebas de

backtesting. Su acceso y permisos quedan gestionados a través del sistema de autenticación Clerk.

- **Frontend (Next.js):** Es la capa de presentación de la aplicación. Está desarrollada en el framework Next.js sobre React. Su trabajo consiste en mostrar la interfaz de usuario, gestionar el cambio de una página a otra del sistema y hacer las peticiones hacia los diferentes microservicios del backend. Es el punto de acceso para los usuarios para que accedan a través del navegador.
- **Clerk (Microservicio de autenticación y gestión de usuarios):** El microservicio Clerk ofrece los mecanismos de autenticación y gestión de registros de las sesiones de los usuarios. Este microservicio permite controlar el acceso a la aplicación y restringir funcionalidades según el estado de autenticación. Este se comunica sobre todo con el frontend para poder gestionar la identidad de los usuarios.
- **Base de datos PostgreSQL (microservicio con Prisma):** Este microservicio es el que se encarga de almacenar de forma persistente los datos asociados a las estrategias de trading, los resultados del backtesting, las configuraciones, etc... Se hace uso de la base de datos relacional PostgreSQL y del ORM Prisma para tramitar la conexión hacia la base de datos desde el backend.
- **Microservicio en Python (conexión con MT5):** Este microservicio es uno de los más relevantes y se encarga de conectar la aplicación y la plataforma de trading MetaTrader 5 (MT5). A su vez, este microservicio ejecuta scripts programados en Python que permiten iniciar el backtesting en la plataforma local y que permiten recoger los resultados generados los cuales envía al backend.
- **MetaTrader 5 (plataforma de trading):** Es una plataforma de trading profesional que permite desarrollar y ejecutar estrategias automatizadas mediante algoritmos escritos en MQL5 el cual es un lenguaje de programación específico para este caso. En este punto del proyecto MT5 se ejecuta de forma local y es manejado a través del microservicio Python que se encarga de lanzar las pruebas y de recoger los resultados de la estrategia de trading algorítmica.

Esto da lugar a una arquitectura modular que logra separar completamente cada responsabilidad en el propio sistema. El hecho de que todas las piezas estén bien separadas promueve mantener dicha arquitectura y el desarrollo de nuevas funcionalidades para el futuro, también da una protección adicional ya que si ocurre un fallo podemos saber rápidamente de qué microservicio se trata. La utilización de tecnologías modernas como Next.js, Prisma y Clerk también hace que la experiencia de usuario final sea prospero y se llegue a optimizar el propio rendimiento.

5.2. Diseño conceptual

En la figura siguiente se muestra el diagrama conceptual donde se representan las principales entidades del sistema y sus relaciones. Este modelo conceptual permite entender de forma abstracta de cómo se estructuran los datos de la aplicación, así como los tipos de vínculos que pueden existir con los distintos elementos que pertenecen en el sistema.

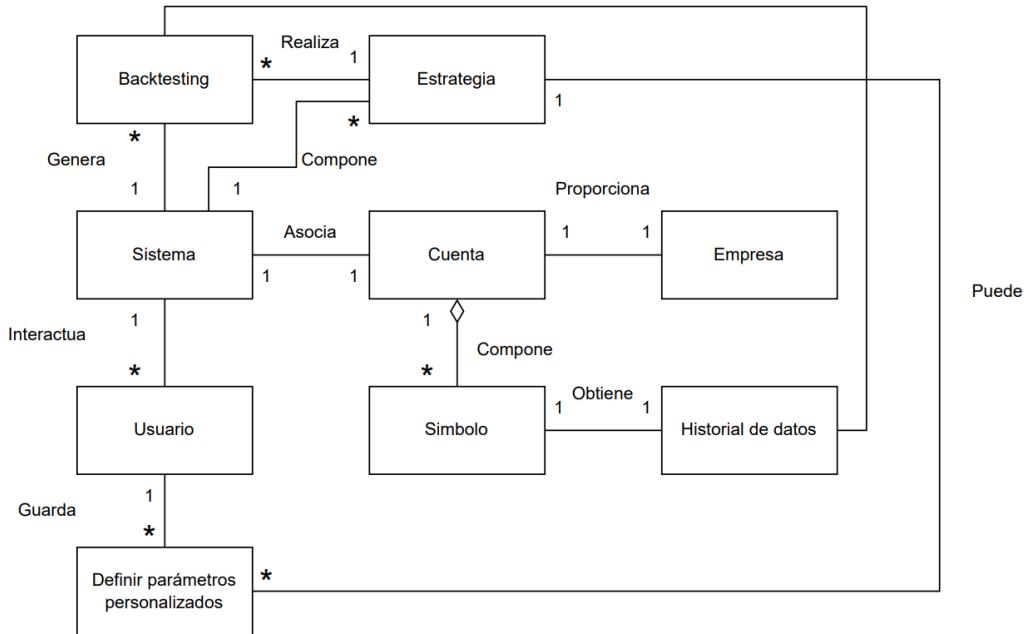


Figura 16: Diagrama conceptual.

La figura 16 muestra el diagrama conceptual. Primero de todo, el usuario es la entidad que interactúa directamente con el sistema, puede definir y guardar varios parámetros personalizados que se usarán para ejecutar estrategias de trading personalizadas. La relación que hay entre usuario y parámetros personalizados es de uno a muchos.

El sistema está vinculado a muchos usuarios (puede haber uno o más usuarios al mismo tiempo) y es responsable de gestionar y generar distintas instancias de backtesting. Las ejecuciones de prueba están vinculadas a distintas estrategias que se componen dentro del sistema y que se desarrollan previamente para poder ser evaluadas.

Las estrategias están vinculadas a las ejecuciones de backtesting de una relación uno a muchos, porque una estrategia puede ser evaluada un número indefinido de veces. Las estrategias, por otra parte, están compuestas por unos parámetros personalizados de las propias estrategias de la plataforma.

La cuenta es una única empresa, en este caso una empresa de fondeo llamada FTMO [5], que a su vez nos ofrece el servicio de conexión con el mercado de los datos. Una cuenta está compuesta por uno o varios símbolos, que son los activos financieros donde se opera por ejemplo EUR/USD, BTC/USDT, etc...

Los símbolos tienen una relación con un historial de datos, que es el conjunto de precios históricos que se utilizan para el análisis y ejecutar el backtesting. La relación es uno a uno, ya que cada símbolo tiene su propio histórico de precios.

De este modo, el modelo conceptual ilustra cómo desde el usuario se configura una estructura jerárquica donde se dejan fijados todos los componentes necesarios para la ejecución del análisis de estrategias de trading, de modo que se incorpora tanto la lógica del negocio como la trazabilidad de los datos utilizados.

5.3. Diseño de la base de datos

El modelo de datos presentado está fundamentado en una estructura relacional que tiene como propósito principal la posibilidad de almacenar y gestionar la información correspondiente a las estrategias de trading que se encuentran definidas, así como de los resultados de las simulaciones o procesos de backtesting asociados a las mismas. La siguiente figura muestra el diseño de la base de datos.

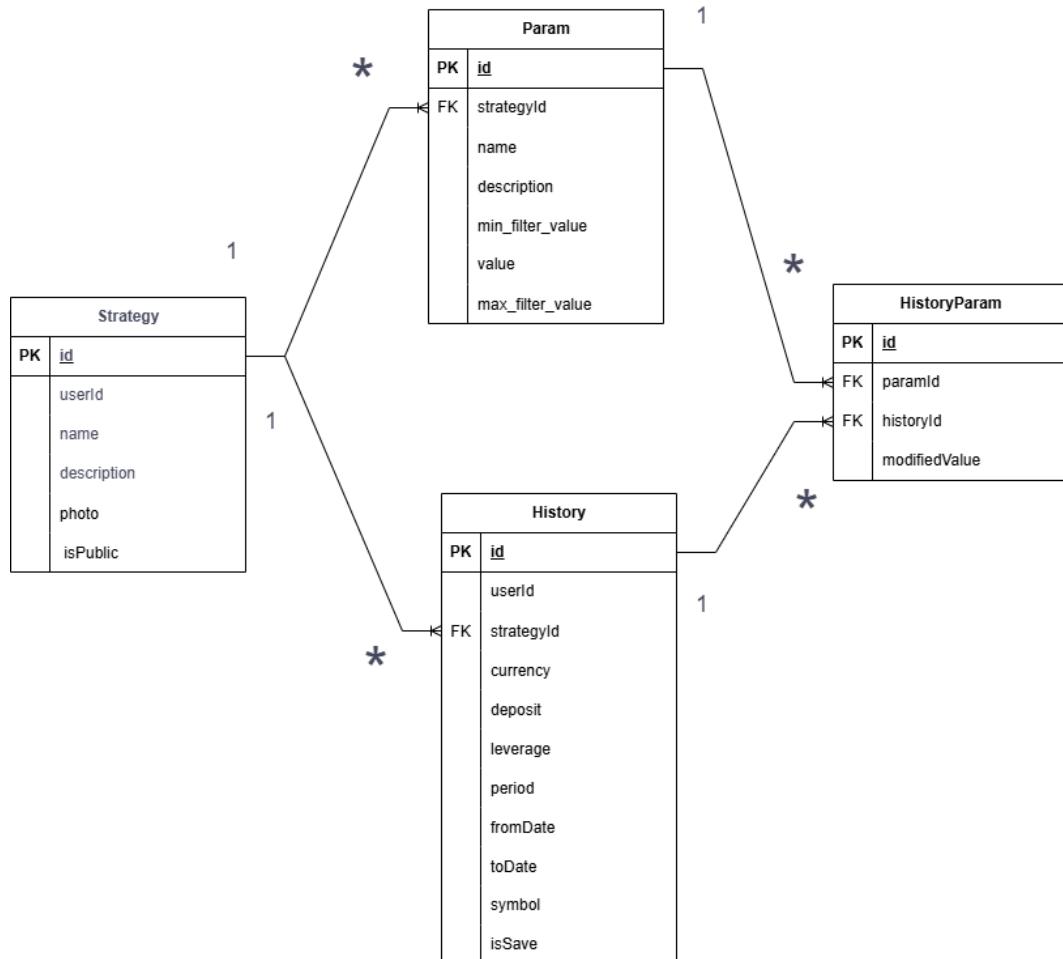


Figura 17: Diagrama base de datos

La figura 17 muestra el diagrama de la base de datos. A continuación se redacta la descripción de las tablas representativas y principales junto con su función en el sistema planteado.

Strategy

La tabla Strategy es aquella que representa las estrategias predefinidas de trading en el sistema. Una entrada de la tabla contiene información básica (name, description, photo, isPublic) que quiere decir si es pública la estrategia para que la vean los usuarios) de la estrategia. En particular, el atributo que hace referencia al usuario autenticado en Clerk es el campo userId, el cual permite identificar la relación existente entre la estrategia guardada y el usuario. La tabla Strategy establece relaciones con los parámetros de la estrategia (Param) y con los históricos de ejecución de la estrategia (History).

Param

La tabla Param establece los parámetros de una estrategia determinada. Un parámetro posee un nombre (name), un valor (value) y de forma opcional un rango de valores filtrables (min_filter_value, max_filter_value). Los parámetros de la tabla significan que están asociados a una única estrategia, la cual está identificada mediante el campo strategyId. Los valores de los parámetros pueden estar modificados por cada proceso de backtesting.

History

Contiene el resultado de un backtest que se ha realizado sobre una estrategia. Contiene información de la operación, como son: el par de divisas (symbol), el depósito inicial (deposit), el apalancamiento (leverage), el período de prueba (fromDate y toDate) y el período temporal del gráfico a analizar (period). Contiene la marca (isSave) que nos informa si el resultado ha sido guardado por un usuario. Un history siempre hace referencia a una estrategia y a un usuario asociado (tales como strategyId y userId).

HistoryParam

Nos permite llevar los valores de parámetros específicos de una estrategia que se han modificado para un backtest concreto. Cada entrada apunta a un history (historyId) y a un parámetro concreto (paramId) y, también incluye el valor modificado (modifiedValue).

Gestión de usuarios con Clerk

No se ha implementado una tabla User en la base de datos porque el manejo de usuarios y la autenticación se realiza de forma externa, es decir, un microservicio asociado a la plataforma Clerk [4]. Así, la identificación userId que queda almacenada en las entidades como Strategy o History corresponde a la ID que Clerk devuelve para un usuario que se ha autenticado. De esta forma, se puede garantizar la seguridad y la escalabilidad del sistema sin la necesidad de tener que manejar esto de manera interna guardando contraseñas o gestionando sesiones.

5.4. Diseño de la interfaz

A continuación se presentan los distintos prototipos de la interfaz de usuario diseñados para la aplicación web. Estas pantallas reflejan las principales funcionalidades de la herramienta, y permiten visualizar cómo será la interacción del usuario con el sistema.

Para la realización de dicho diseño se ha usado mockitt [16] una aplicación web especializada en el diseño e implementación de mockups, con un disponibilidad de herramientas y librerías que agilizan altamente el trabajo de este diseño.

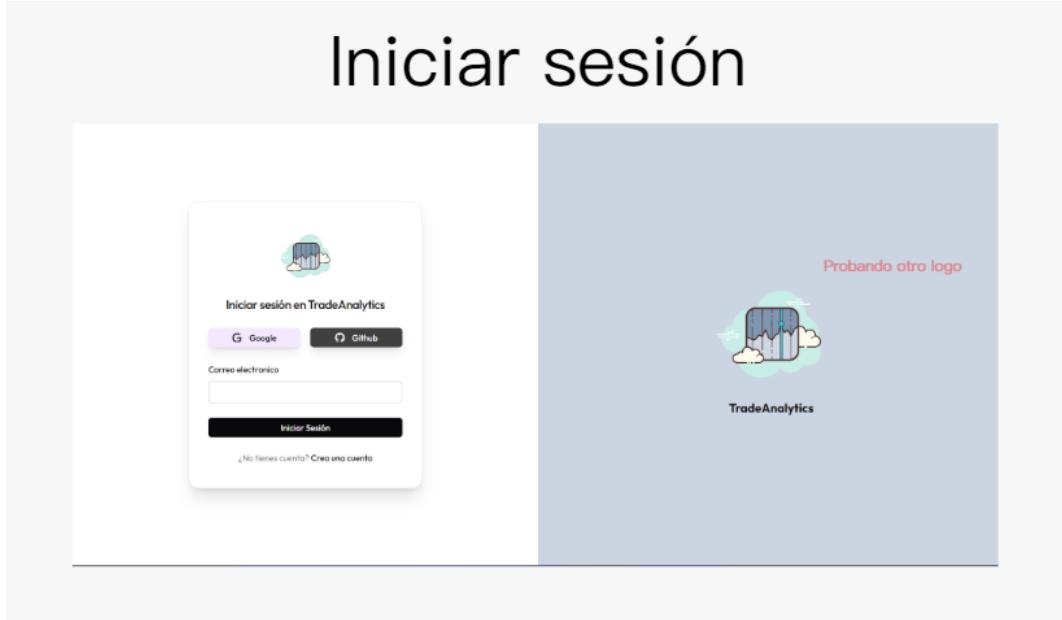


Figura 18: Pantalla de inicio de sesión. Permite a los usuarios registrados acceder a la aplicación mediante Clerk.

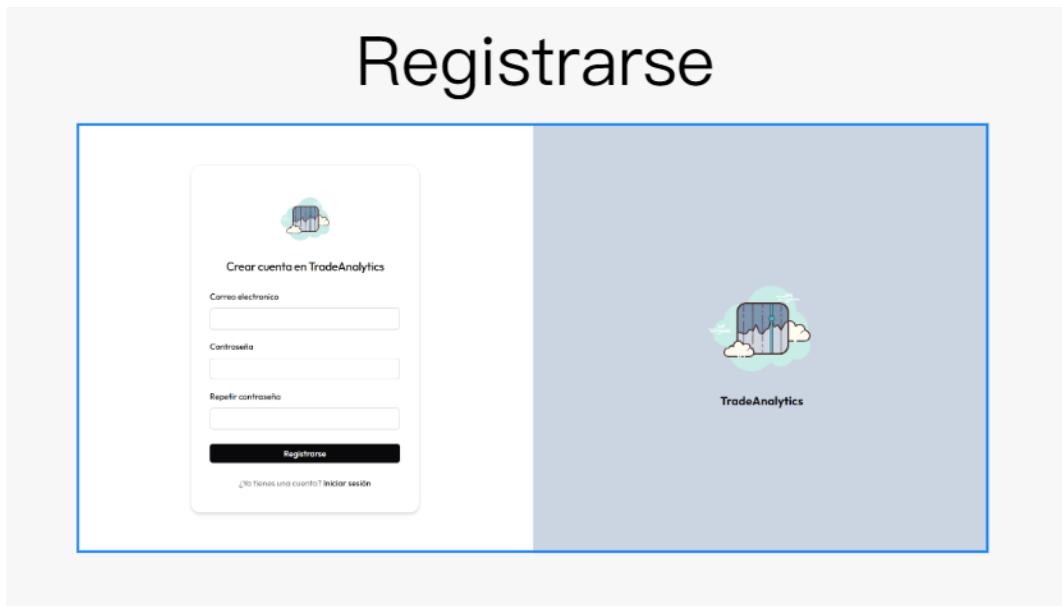


Figura 19: Primera vista del formulario de registro para nuevos usuarios.

Registrarse verificación

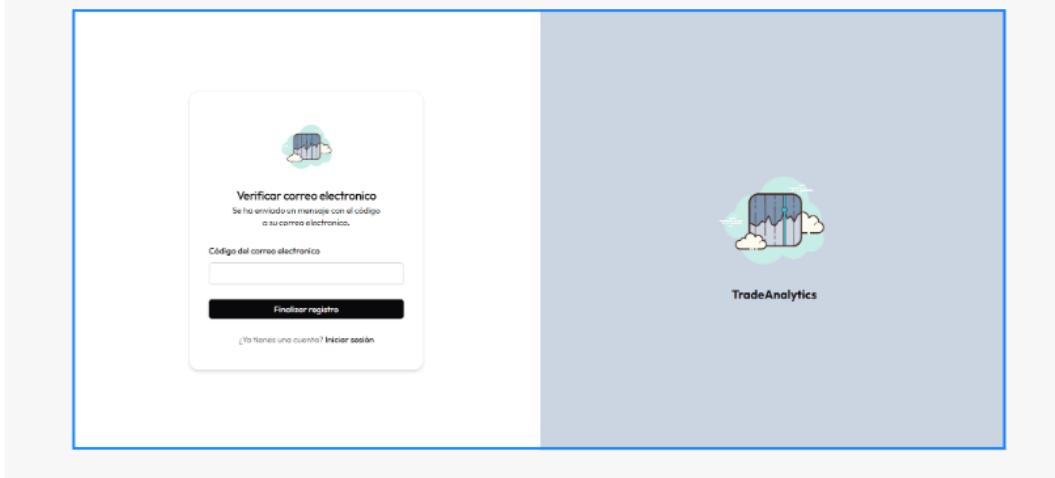


Figura 20: Segunda parte del proceso de registro.

Página principal

A screenshot of the main dashboard of the TradeAnalytics platform. At the top, there is a header with the "TradeAnalytics" logo, a "Probador de estrategias" button, and an "Iniciar sesión" button. The main content area features a large dark blue button with a circular graphic containing a blue arrow pointing upwards, with the text "TRADEANALYTICS" at the bottom. To the left of this button, there is a section with the heading "Pruebe nuestras estrategias en TradeAnalytics" and a detailed description of the platform's features. Below this, there are four icons: "EUR/USD" with two gold coins, "S&P 500" with a green money bag, "Nasdaq" with its logo, and "bitcoin" with its logo. A red arrow points from the text "Imagenes de simblos moviéndose" to the "EUR/USD" icon.

Figura 21: Vista general de la página principal tras iniciar sesión.



Características principales

Nosotros ofrecemos el mejor servicio para nuestros clientes, con la mejor calidad y precio.

Disponible las 24 horas	Soporte técnico disponible	Seguridad garantizada	Servicio al mejor precio

Figura 22: Visualización de las estrategias disponibles.

Estrategias principales

No dejes escapar la oportunidad de probar nuestras estrategias, elige el que mas se adapte a tus necesidades.

Animaciones
de textos e imágenes
al abrir la pagina



[Probar estrategias](#)

Figura 23: Detalles de una estrategia con sus parámetros.

Se un trader profesional

Regístrate y explora nuestras estrategias

[Regístrate aquí](#)



Figura 24: Interfaz para lanzar un backtesting sobre una estrategia.

Mandanos un mensaje

Nombre

Apellidos

Numero de telefono

Numero de telefono

Mensaje

Enviar

Figura 25: Visualización dinámica del estado del backtesting.

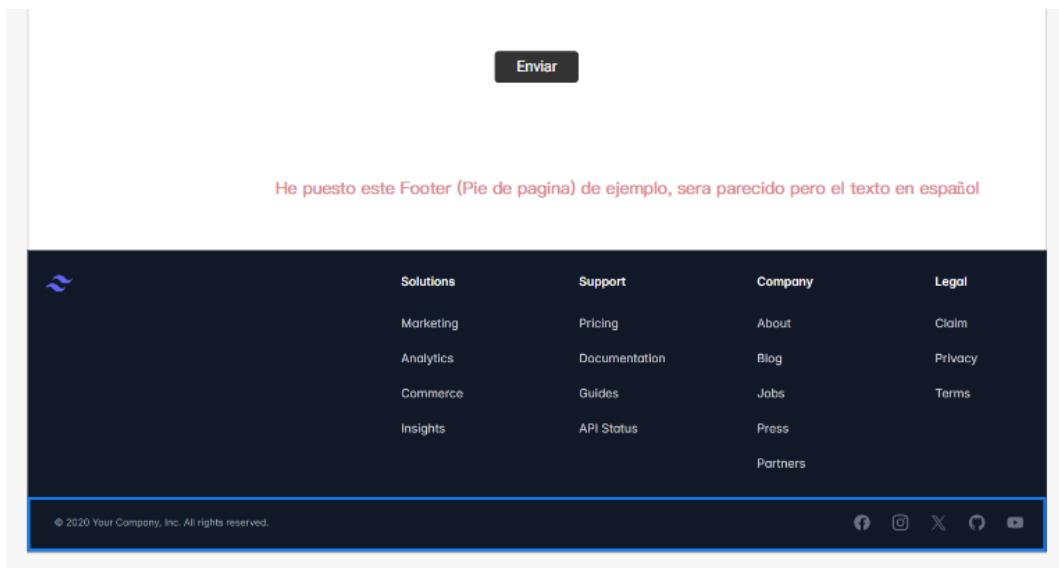


Figura 26: Gráficos iniciales generados tras el backtest.

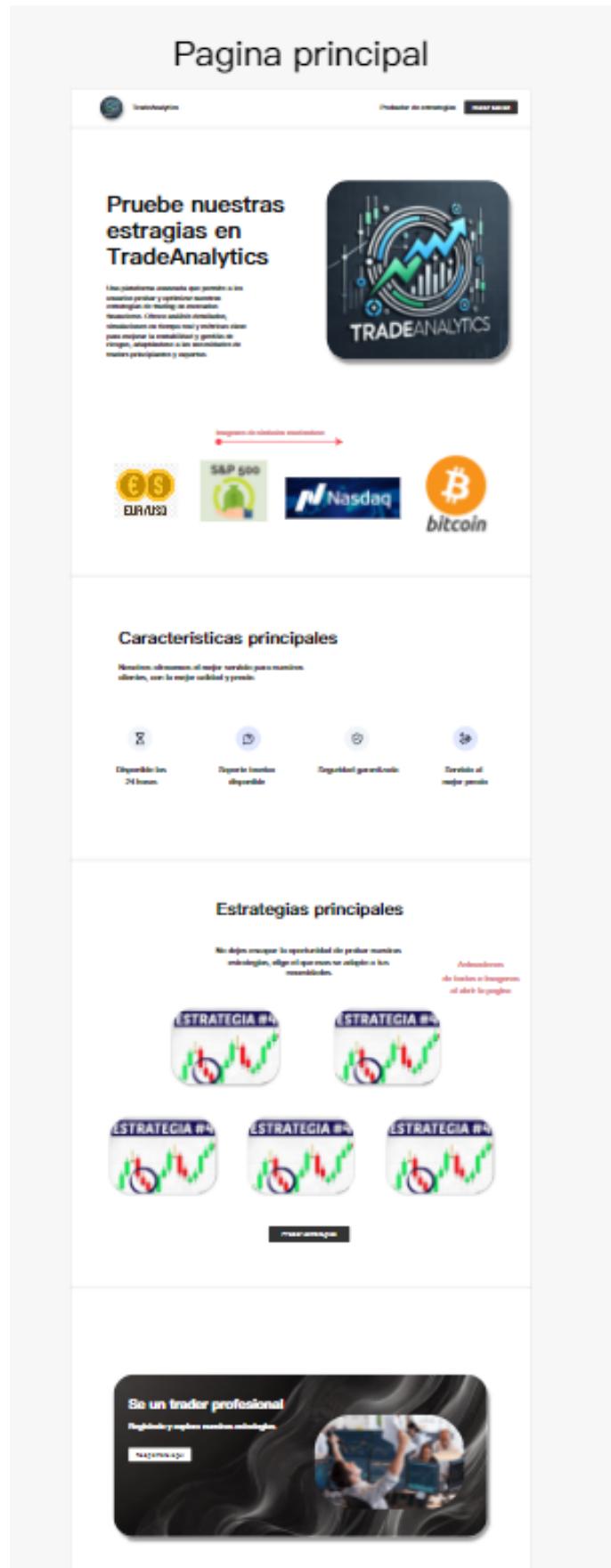


Figura 27: Ampliación con estadísticas detalladas por estrategia.

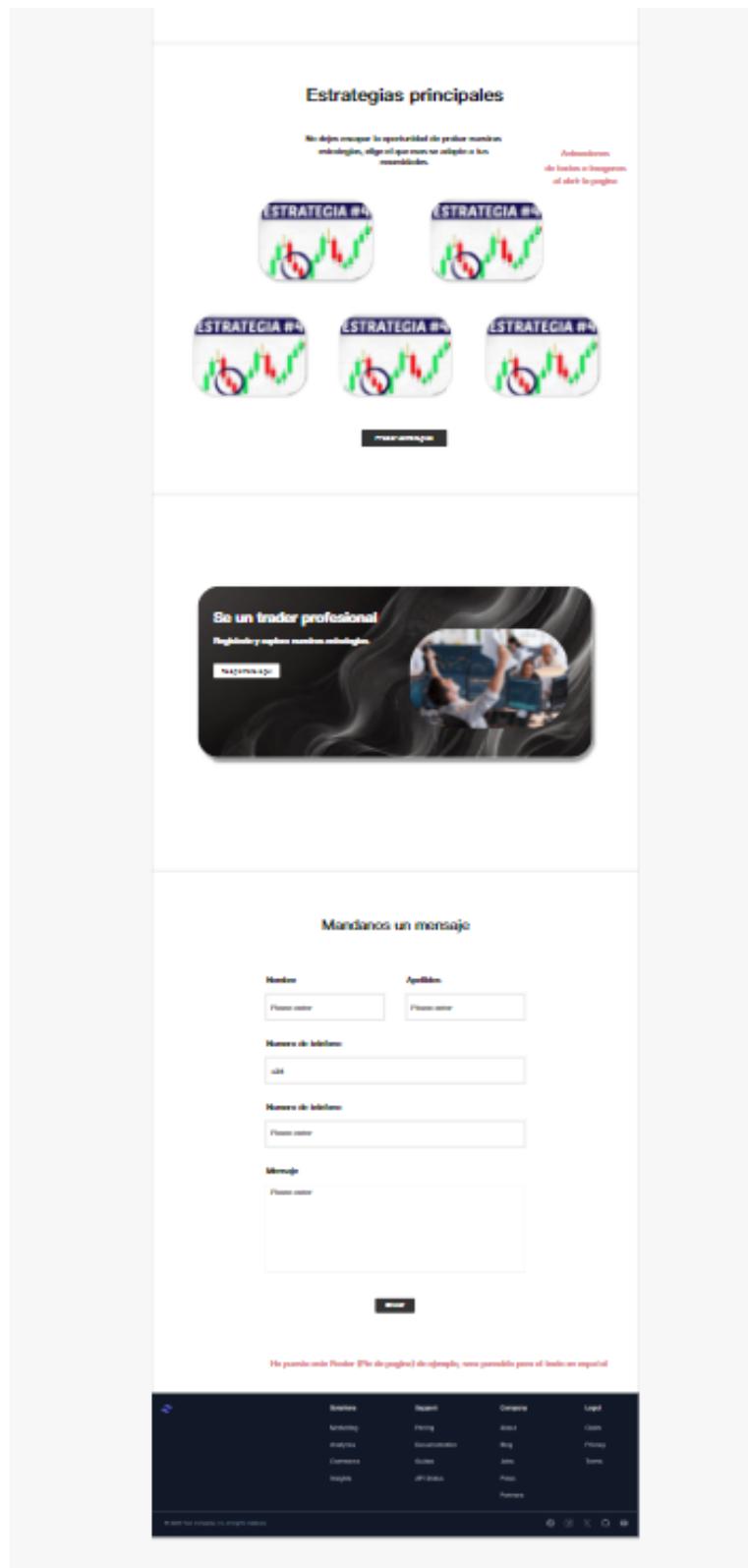


Figura 28: Comparativa entre estrategias y filtrado por rendimiento.

Dashboard

Aqui podra acceder cuando ha iniciado sesion

The dashboard interface includes a sidebar with icons for Estrategias, Parametros guardadas, and Estrategias favoritas. It also has sections for Administrador (with links to manage strategies and user parameters) and a note about administrator access. A central area displays a list of strategies with cards for 'ESTRATEGIA #4' and 'ESTRATEGIA #4' (repeated), each showing a candlestick chart, a title, data, and a 'Probar estrategia' button.

Figura 29: Vista general de las estrategias almacenadas en la plataforma.

Probando una estrategia

Aqui podra acceder cuando ha iniciado sesion

This tool interface has a sidebar with icons for Estrategias, Parametros guardadas, and Estrategias favoritas. It features sections for Symbol (set to BTC), Account (with fields for Money and Leverage), and Parameters (with fields for EMA and Parameter 2). A central area shows three dots indicating more options, and a 'Realizar backtesting' button at the bottom right.

Figura 30: Herramienta para probar estrategias predefinidas con parámetros personalizados.

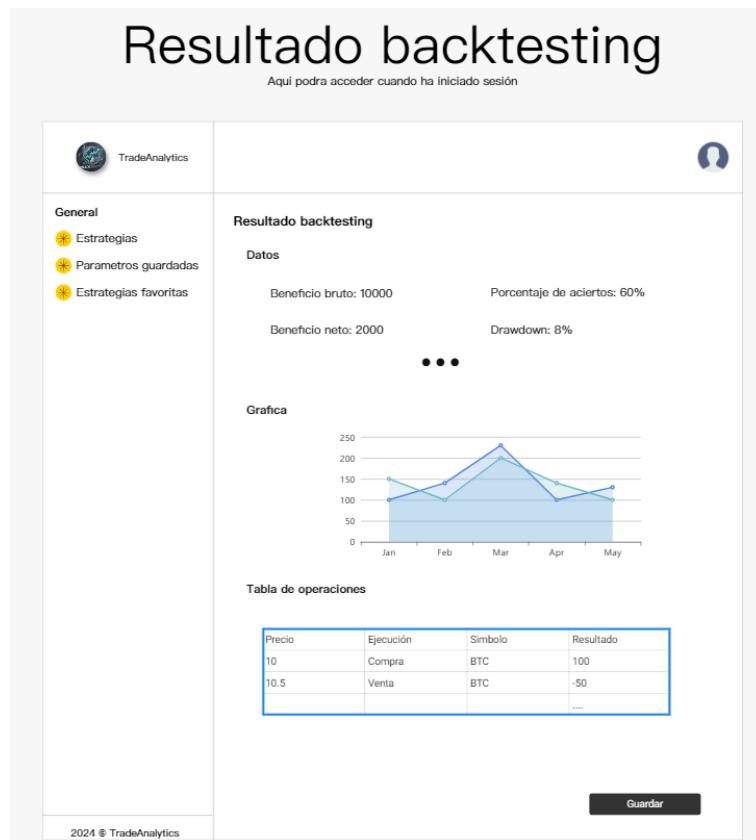


Figura 31: Resultados gráficos del backtesting ejecutado.



Figura 32: Zona donde el usuario puede guardar configuraciones de parámetros para futuras pruebas.

Figura 33: Panel de administración donde se pueden gestionar usuarios y configuraciones avanzadas.

Cada una de las figuras anteriores son explicadas en su parte de la descripción, más adelante en la sección manual de uso se detallan estas secciones con la versión y funcionalidad final. Este diseño es la versión final, es decir, se ha revisado detenidamente y mejorando constantemente con la tutora cada sección hasta obtener esta versión final.

Resaltar que las representaciones correspondientes de mockups reflejan el diseño real de la aplicación web. En este apartado, se tiene en cuenta todo aquellos aspectos visuales y de UX que condujeron al diseño, tales como una disposición clara y evidente de los elementos, una interfaz limpia y con diseño actual, la coherencia de colores y la organización intuitiva de los contenidos. También se han seguido principios de resolución responsive y accesibilidad, de manera que se dé cuenta de una experiencia de usuario totalmente integradora en diferentes dispositivos y perfiles de usuario.

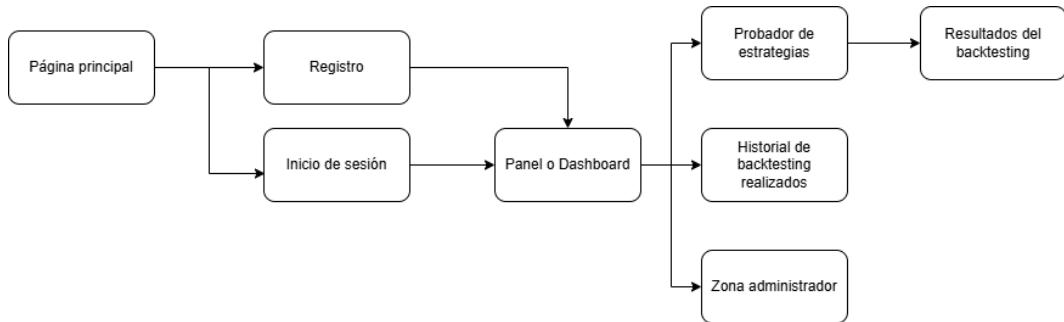


Figura 34: Diagrama de flujo.

La figura 34 muestra el diagrama flujo, esta se basa en la interfaz de la aplicación que está

formada por áreas distintas, ideadas para llevar al usuario desde que se registra hasta el momento de visualizar los resultados del backtesting, en el siguiente texto se redacta todas las figuras anteriores para finalizar el diseño el mockup:

- Acceso y registro: Se empieza entrando a la aplicación desde una pantalla de login utilizando Clerk, que a la vez deja a los nuevos usuarios registrarse con formularios sencillos y a prueba de fallos.
- Página principal: Tras entrar con sus credenciales, el usuario ve el panel de control, donde puede visualizar estrategias, empezar simulaciones, mirar resultados y cambiar su perfil.
- Busqueda de estrategias: El sistema despliega una lista de estrategias ya hechas, con sus parametros tecnicos, para ayudar en la selección basandose en el tipo de analisis que el usuario quiere hacer.
- Configuración y puesta en marcha del backtesting: En el probador de estrategias, se definen los parámetros generales (saldo, símbolo, fechas, etc.) y los específicos según la estrategia, para iniciar la ejecución del backtesting de manera automática.
- Viendo los resultados: Despues de la ejecución del backtesting, se ve gráficos con métricas importantes como ganancia, drawdown y ratio de acierto.
- Comparativa y estadísticas: La plataforma presenta una vista mucho más completa de datos estadísticos detallados permitiendo así comparar el rendimiento de muchísimas estrategias.
- Gestión de parámetros personalizados: El usuario, en su lugar, puede guardar las configuraciones de parámetros que él considere importantes para pruebas en un futuro, evitándose repetir un proceso de configuración.
- Zona administrativa: Los administradores pueden acceder a una sección privada donde pueden gestionar el sistema de gestión de estrategias.

El diagrama de flujo de pantallas, de esta manera tenemos una visión de general de como puede navegar e interactuar el usuario en el sistema.

5.5. Cuestiones de usabilidad y accesibilidad

La aplicación web que se presenta ha sido diseñada de acuerdo con una serie de principios básicos de usabilidad y de accesibilidad, de forma que aseguremos obtener por tanto una experiencia de usuario eficaz, comprensible y accesible para cualquier tipo de perfil.

5.5.1. Usabilidad

La usabilidad es la simplicidad de un usuario navegar de forma intuitiva por una aplicación o sistema. En este proyecto, se ha dado mucha relevancia la usabilidad desde el inicio buscando simplificar la interacción [17].

Como ya se ha comentado, se ha buscado desde el diseño inicial una interfaz amigable, clara y sencilla. Las heurísticas de usabilidad de Jakob Nielsen se han aplicado como un conjunto de buenas prácticas ampliamente reconocidas a la hora de evaluar y de mejorar la interacción de un usuario con el sistema. Algunos de los aspectos que se han implementado en la interfaz y que son más destacables son los siguientes:

- Consistencia y estándares: uso de los iconos, estilos y de la estructura de navegación de forma uniforme.
- Visibilidad del estado del sistema: retroalimentación inmediata al usuario a cada acción (puede ser en la carga de resultados, en errores que se produzcan en el formulario o en confirmaciones).
- Prevención de errores: validaciones en tiempo real en los formularios y confirmaciones antes de llevar a cabo operaciones que sean sensibles.

Probar Estrategia: Acumulacion Manipulacion Distribucion

Configuración base

The screenshot shows a user interface for configuring a trading strategy. It includes the following fields:

- Símbolos**: EURUSD
- Periodo**: M15
- Depósito**: Selecciona la cantidad de la cuenta (with a note: Debe seleccionar uno, highlighted in red)
- Divisa**: EUR
- Apalancamiento**: 1:33

Figura 35: Ejemplo de usabilidad en el formulario.

En la figura 35 se muestra un ejemplo de usabilidad en el formulario, si el usuario no entiende que significa la variable tiene un ícono que le muestra la información, además si no ha ingresado bien los datos o falta algún dato por llenar en un formulario donde se ha seleccionado una estrategia predefinida, se le informará al usuario con un error en color rojo y la descripción del error, de esta forma el usuario directamente sabe donde está el error y que corregir para poder avanzar en la ejecución del backtesting,

- Diseño centrado en las tareas: la interfaz con el usuario está organizada de forma que se corresponda con el flujo natural del trabajo del usuario (Nielsen identifica este principio heurístico como el de la correspondencia entre el sistema y el mundo real). Las funcionalidades se encuentran agrupadas correctamente en función de cómo la utiliza el usuario, permitiendo un acceso ágil a la estrategias, al historial y a la configuración, orientando a su objetivo dentro del entorno de trading.

El diseño se ha realizado en específico para dispositivo de escritorio, es decir, no es responsive en muchos apartados ya que la integración de gráficos y estadísticas en el backtesting es poco legible en un móvil por su formato de tamaño.

5.5.2. Accesibilidad

La accesibilidad web trata sobre qué tan bien una aplicación puede usarse por gente con habilidades distintas. Personas con problemas de vista, de oír o movilidad. Para este proyecto, se implementa con un cuidado especial esta accesibilidad [3].

Con el objetivo de asegurar una correcta accesibilidad del sistema se han seguido distintas recomendaciones que permiten el acceso a las personas con capacidades diferentes. Para tal propósito, la

aplicación ha sido sometida a las herramientas TAW, WAVE [33], cumpliendo las recomendaciones del estándar. Las siguientes mencionadas son las más relevantes:

- Descripción de texto en imágenes: todas las imágenes que son relevantes para la obtención del contenido o de la navegación tienen un atributo alt descriptivo.
- Etiquetas accesibles: formularios y controles están etiquetados para su interpretación con información adicional.
- Navegación mediante teclado: todas las funcionalidades principales pueden utilizarse sin ratón.
- Estructura semántica: uso correcto de encabezados y contenido simétrico para facilitar la interpretación por parte del usuario.

Destacar otros más secundarios como contraste de colores, el uso de tamaños de fuente escalables, secciones que produzcan molestias visuales por una mala carga del contenido y la presencia de etiquetas adecuadas.

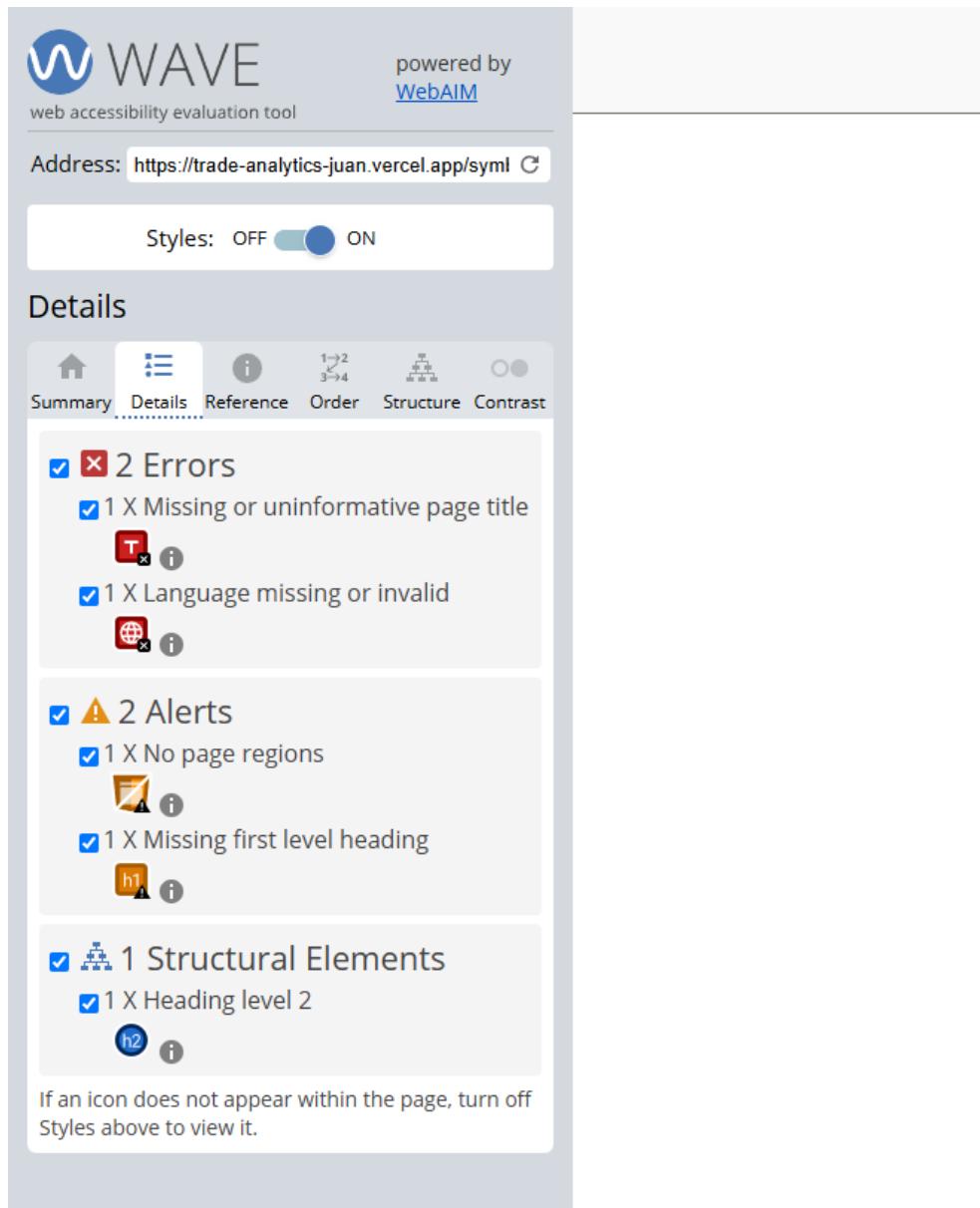


Figura 36: WAVE accesibilidad web.

En la figura 36 se muestra un test sobre nuestra aplicación web, dicho test realizado en la página de wave obtenemos dos errores, los que nos sugiere es que se añada un lenguaje valido y un título de la página, al usar un framework llamado NEXT.JS, este genera código dinámico, es decir, al no ser estático no reconoce esta información, para demostrar que estos errores no son ciertos, voy a resaltar en el código dichas soluciones de estos errores en la siguiente figura.

```
You, 3 months ago || author (You)
import type { Metadata } from "next";
import { Outfit } from "next/font/google";
import "./globals.css";
import NextTopLoader from "nextjs-toploader";
import { ClerkProvider } from "@clerk/nextjs";
import { Toaster } from "@/components/ui/toaster";

const outfit = Outfit({ subsets: ["latin"] });

export const metadata: Metadata = {
  title: "TradeAnalytics",
  description: "Conoce más sobre TradeAnalytics y cómo optimizamos estrategias.",
  icons: {
    icon: "/logo.svg",
  },
};

export default function RootLayout({
  children,
}: Readonly<{
  children: React.ReactNode;
}>) {
  return (
    <ClerkProvider>
      <html lang="es"> You, 3 months ago • change ...
        <head>
          <title>{String(metadata.title)}</title>
          <meta name="description" content={String(metadata.description)} />
        </head>
        <body className={outfit.className}>
          <NextTopLoader />
    
```

Figura 37: Solución errores WAVE.

En la figura 37 se muestra el código que se ha mencionado anteriormente. Todas estas cuestiones buscan cumplir no solo la perspectiva legal y ética, sino también permitir la construcción de una aplicación que asegure la igualdad de acceso para cualquier persona.

6 Tecnología

En este apartado se revela la tecnología elegida para el desarrollo del proyecto, desde una perspectiva técnica y estratégica. Se describe la motivación de las decisiones tecnológicas elegidas y se presentan las herramientas que se han empleado en las diferentes capas del sistema: frontend, backend, base de datos, conexión con MetaTrader 5, servicios de autenticación y despliegue.

La correcta elección de la tecnología representa un aspecto crítico, condicionando directamente el rendimiento, la mantenibilidad, la escalabilidad y la usabilidad del sistema, pero también permite garantizar una buena experiencia al usuario final y, además, permite garantizar que la solución se pueda llevar a cabo en la fase de desarrollo, así como su posible evolución o despliegue.

6.1. Justificación tecnológica

El desarrollo de esta aplicación requería tecnologías que cubrieran varios de los aspectos básicos que se debían abordar: la interacción con plataformas de trading, un diseño actual y adaptable a cualquier tipo de dispositivo, el procesamiento de usuarios, la persistencia de datos y la facilidad de uso.

Se ha optado por un enfoque modular a partir de microservicios y separación de responsabilidades, esto favorecen la escalabilidad, la reutilización del código, y la facilidad de obtener información sobre futuros problemas. Adicionalmente a esto, esta arquitectura también permitiría integrar servicios externos, por ejemplo MetaTrader 5, además de distribuir las distintas partes del sistema en entornos separados.

En cuanto al frontend, se elige Next.js con TailwindCSS, por ser una tecnología que cumple con una buena relación coste/rendimiento, con estructuras claras y que tiene un ecosistema muy agil que nos permite construir interfaces responsivas y reactivas para proporcionar una buena experiencia al usuario, esta opción también facilita el uso de componentes reutilizables y que son necesarios para manejar la complejidad de formularios dinámicos y la visualización de resultados estadísticos, esta tecnología se ha usado para proyectos como NETFLIX, twitch, HBO, etc...

El desarrollo del backend se ha aprovechado el sistema de 'API Routes' que incluye el propio framework Next.js. Este modelo permite llevar a cabo las funciones del servidor directamente en el propio proyecto del frontend, manteniendo así la lógica del backend en el entorno del frontend, en lo que se conoce como arquitectura fullstack, se ha elegido de esta manera ya que usamos más microservicios y el desarrollo de la propia aplicación web es mejor tenerlo en un propio proyecto.

El lenguaje empleado ha sido TypeScript, que ofrece seguridad durante la compilación y autocompletado. Por el lado de la conexión a la base de datos se ha utilizado Prisma, un potente ORM(Object Relational Mapper) que permite definir modelos de datos en un esquema y a su vez realizar operaciones de acceso a datos utilizando funciones tipadas. La base de datos empleada es PostgreSQL.

Gracias a la combinación de este stack (Next.js + API Routes + Prisma) se ha podido mantener un desarrollo backend simple, seguro y muy bien integrado a la parte frontend, lo que ayuda a disminuir la complejidad del sistema en general.

Uno de los pilares del sistema es la integración con MetaTrader 5 una plataforma para poder realizar operaciones y estrategias de trading, a su vez las estrategias se implementan en un lenguaje

específico llamado MQL5 asociada a Metatrader 5. Gestionada a través de un microservicio en Python, usando librerías oficiales que permiten ejecutar estrategias y simular backtesting con datos reales de mercado. En este caso, se ha escogido Python por su gran compatibilidad dentro del ecosistema de trading algorítmico y su gran comunidad donde facilita y proporciona información sobre cualquier desarrollo complejo.

Finalmente, se han elegido servicios externos como Clerk para la gestión de usuarios y autenticación, así como Vercel para el despliegue de la aplicación ya que son muy compatibles con NEXT.JS estas tecnologías, en mi caso con planes gratuitos en entorno de desarrollo.

Esta combinación de tecnologías no solo permite cumplir los requisitos funcionales y no funcionales, sino que nos da la garantía de conseguir una solución profesional, escalable, mantenible y dispuesta para ser continuada en el futuro.

6.2. Descripción de la tecnología usada

La aplicación implementada usa las tecnologías más modernas en frontend y backend, constituyendo un entorno de trabajo en fullstack. A continuación se detallan los frameworks y tecnologías más significativas que se han implementado:

Next.js [12] ha sido el framework principal para realizar el desarrollo. Permite crear aplicaciones web en React y tiene funcionalidades como: server-side rendering (SSR), rutas automáticas, y una API integrada. En este proyecto se ha usado la versión 13 en adelante.

Para el desarrollo del frontend se han usado **React** [6] y **TypeScript** [3]. React permite crear interfaces de usuario dinámicas y TypeScript añade tipado estático a un lenguaje como JavaScript, el cual mejora la robustez del código y ayuda a evitar errores en la fase de desarrollo.

Para interactuar con la base de datos se ha usado **Prisma** [11] como ORM (Object Relational Mapping). Prisma permite definir de forma declarativa el modelo de datos y generar automáticamente consultas SQL tipadas y seguras. Por otra parte, permite realizar ciertas tareas, como la creación de relaciones, validaciones o migraciones.

Se ha optado por **PostgreSQL** [7] como sistema de gestión de bases de datos. PostgreSQL es una base de datos relacional robusta, utilizada frecuentemente en entornos profesionales, que permite almacenar y consultar información estructurada: estrategias, parámetros y resultados de backtestings.

Clerk [4] ha gestionado el sistema de autenticación de usuarios, proporcionando funcionalidades como registro, login, control de sesiones y con el soporte de proveedores externos como Google o GitHub, así como verificación por correo, todo de una manera fácil y segura.

Vercel [32] es la plataforma en la que se ha desplegado la aplicación, está especializada en proyectos con Next.js y proporciona despliegue automático. Se obtiene una gran facilidad para subir el proyecto ya que tiene proveedores como github, por lo tanto, si se requiere realizar un despliegue y después una actualización, solo habría que actualizar el código en github y automáticamente despliega de nuevo la aplicación.

Se ha empleado el **MetaTrader 5 (MT5)** [19] para ejecutar los backtestings. Esta plataforma

profesional de trading posibilita simular operaciones financieras con datos históricos. La integración con el entorno MT5 se ha hecho a través de scripts en Python y de microservicios locales, conectando así con la información necesaria para cada una de las estrategias configuradas.

La combinación entre las distintas tecnologías ha permitido desarrollar una aplicación moderna, robusta, escalable y atendiendo al problema del análisis algorítmico de estrategias de trading.

7 Desarrollo

Para explicar el desarrollo, se detallare en las siguientes secciones los apartados fundamentales para completar este proyecto, entre ello, frontend para obtener una interfaz donde pueda navegar e interactuar el usuario de forma eficiente e intuitiva, backend para dar funcionalidad y eficiencia, desarrollo de software en python para obtener comunicación la aplicación web con la plataforma MetaTrader5 en local y, por ultimo, el desarrollo de los algoritmos de trading en MetaTrader 5.

7.1. Frontend

La construcción del frontend de la aplicación ha sido realizada con Next.js con soporte para TypeScript y una estructura de carpetas. Esta estructura emplea el nuevo modo de routing ofrecido por Next.js, a través de la carpeta app/ lo que nos permite navegar de forma declarativa y modular.

Se ha desarrollado el frontend del proyecto tanto en TypeScript como en el framework Next.js, ofreciendo un entorno fiable, seguro y moderno para la construcción de interfaces reactivas. La utilización de TypeScript da soporte a un tipado estático diferente en JavaScript, gestiona los errores en tiempo de desarrollo o una mantenibilidad del código. La estructura está basada en el sistema de rutas y de componentes del directorio app/ de Next.js, que favorece layouts anidados, segmentación de roles o carga por servidor optimizada.

Para el propósito de diseño de estilos se ha utilizado Tailwind CSS, un marco de composición de estilos que permite aplicar los mismos estilos en las clases de los elementos, lo que ha permitido obtener una interfaz visualmente limpia, responsive y coherente sin necesidad de extensos y separados archivos CSS. A su vez, el diseño de componentes de interfaz se ha visto reforzado con el uso de la moderna biblioteca shadcn/ui [28], una solución que proporciona una serie de componentes ya preestilizados con Tailwind CSS accesibles por defecto. Con esta combinación ha sido posible construir interfaces tales como formularios, tarjetas, botones o tablas con buena estética moderna, consistencia visual y mínima complejidad.

El frontend presenta un diseño modular: los diferentes elementos de la aplicación son agrupados en varias vistas, como el dashboard, la página home o las vistas de autentificación. Para conectar con el backend, se utilizan funciones fetch,async/await y axios para llamar a las diferentes rutas de las API (también definidas dentro de app/api) ofreciendo toda la comunicación necesaria entre el cliente y el servidor. Por otra parte, se ha intentado dar al usuario la mejor experiencia posible, ofreciendo feedback visual instantáneo, un paso fluido entre las diferentes vistas de la aplicación, así como acciones de guardar una estrategia o ver los resultados de forma sincrónica.

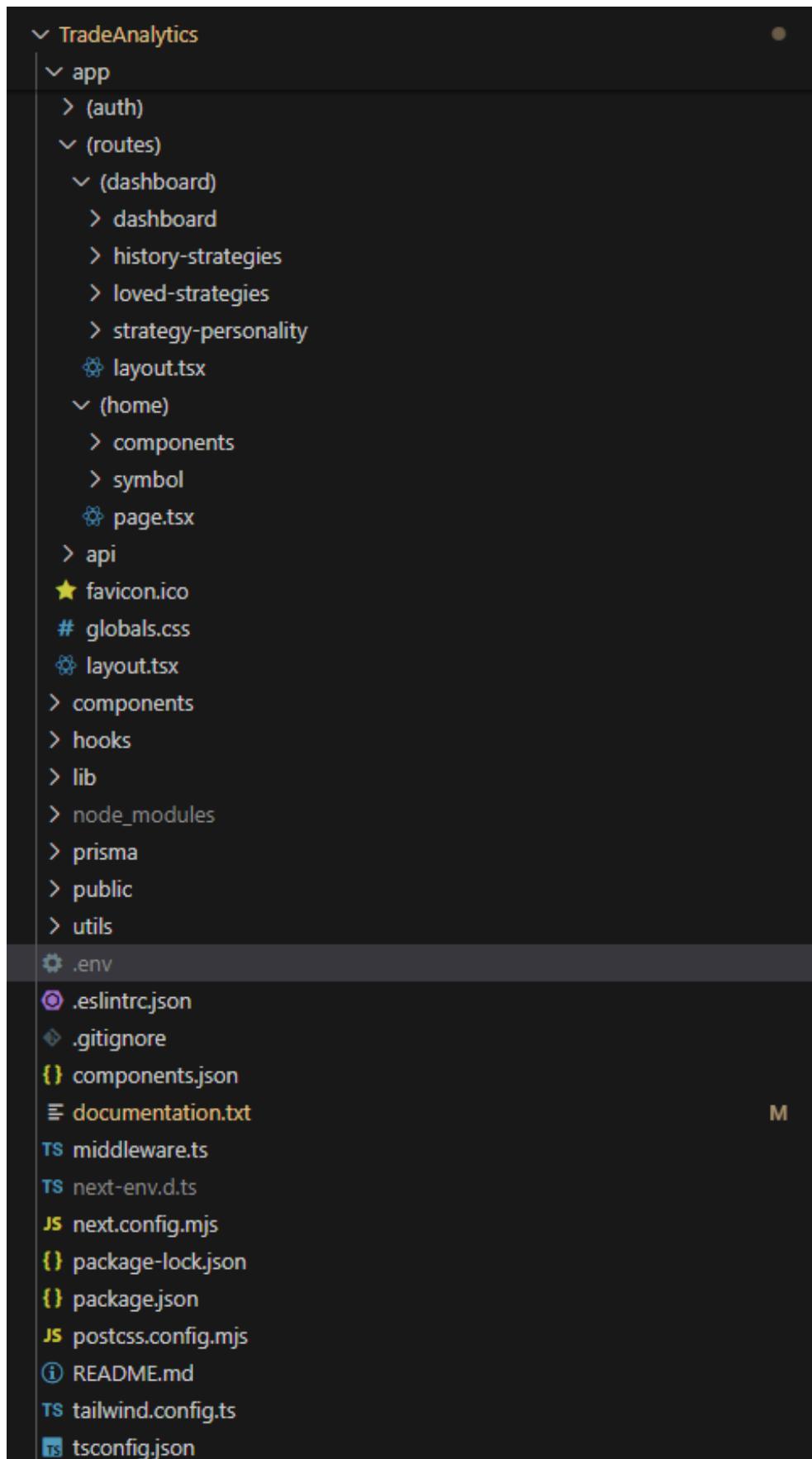


Figura 38: Estructura frontend.

```
    . . .
    ✓ (dashboard)
      ✓ dashboard
        ✓ admin
          > history-admin
          > strategy-manager
        > components
        > information
        > probe-strategy
        > result-strategy
        ⚡ page.tsx
        > history-strategies
        > loved-strategies
        > strategy-personality
        ⚡ layout.tsx
```

Figura 39: Estructura dashboard detallada.

```
    . . .
    ✓ (home)
      ✓ components
        > Contact
        > Features
        > FirstBlock
        > Footer
        > OurStrategies
        > ProbeToday
        > SliderSymbols
      ✓ symbol
        > components
        ⚡ page.tsx
        ⚡ page.tsx
```

Figura 40: Estructura home detallada.

La figura 38 muestra la estructura del frontend, la figura 39 muestra la estructura de dashboard y la figura 40 la estructura del home más detalladas que se establece de la siguiente manera:

- **app/**: Carpeta que engloba la mayor parte de la interfaz y en gran medida la parte del backend, la mayor parte del proyecto se desarrolla en esta sección.
 - **(auth)/**: rutas de autenticación (inicio de sesión y registro) usando el microservicio clerk, explicado en la sección backend que será redacta más adelante.
 - **(routes)/**: carpeta declarativa de las siguientes navegaciones en frontend, el usuario visualizara estas subrutas de la página web.
 - **dashboard/**: el usuario tendrá acceso a esta sección una vez ha iniciado sesión.
 - **dashboard/**: vista principal del dashboard, se visualiza todas la estrategias disponibles.
 - **history-strategies/**: historial de estrategias simuladas o ejecuciones backtesting.
 - **loved-strategies/**: estrategias asignadas como favoritas.
 - **strategy-personality/**: zona donde solo tendrá acceso el administrador, este puede crear, modificar y eliminar estrategias.
 - **admin/**: zona del contenido del administrador que contiene:
 - **history-admin/**: historial completo de todos los usuarios, el administrador tiene opción de borrar.
 - **strategy-manager/**: panel donde el administrador gestiona las estrategias del sistema.
 - **components/**: componentes específicos usados en las vistas del dashboard.
 - **information/**: página que ofrece explicaciones y contenido introductorio sobre cómo funciona el sistema de estrategias.
 - **probe-strategy/**: vista donde el usuario introduce los parámetros para simular una estrategia concreta.
 - **result-strategy/**: se muestran los resultados gráficos y métricos del backtesting realizado.
 - **home/**: sección donde cualquier usuario puede acceder sin necesidad de iniciar sesión.
 - **components/**: componentes de la página principal que contiene:
 - **Contact/**: formulario de contacto de la página, puede enviar un mensaje y recibira un mensaje por via email.
 - **Features/**: sección donde se listan las características importantes de la plataforma.
 - **FirstBlock/**: bloque principal de introducción.
 - **Footer/**: pie de página.
 - **OurStrategies/**: muestra un resumen de las estrategias más destacadas.
 - **ProbeToday/**: llamada a la acción para iniciar pruebas de estrategias.
 - **SliderSymbols/**: carrusel con los símbolos de trading disponibles.
 - **symbol/**: sección de símbolos de trading, muestra información de todos los simbolos disponibles y de la empresa de fondeo que nos ofrece todos los datos históricos de los simbolos.
 - **api/**: rutas API del backend, explicadas en siguiente sección.
 - **layout.tsx**: layout global, este obtiene las primeras lineas del código html.
- **components/**: componentes reutilizables como botones, tablas, formularios, etc...

- **hooks/**: custom hooks para lógica compartida, en mi caso se ha desarrollado la funcionalidad para marcar el usuario las estrategias que seleccione como favoritas.
- **lib/**: funciones para saber si el usuario es administrador y conexión a la base de datos.
- **prisma/**: definición del esquema de base de datos con Prisma, explicado en la siguiente sección backend.
- **public/**: archivos estáticos accesibles como logos, imágenes, etc...
- **utils/**: función usada para el servicio uploadthing, encargado de la gestión multimedia como el almacenamiento de imágenes.
- **.env**: variables de entorno, estas variables serán privadas y contendrán todas las contraseñas y claves de la aplicación web, por ejemplo la clave de la base de datos.
- **tailwind.config.ts**: configuración de Tailwind CSS.
- **next.config.mjs**: configuración general de Next.js.
- **middleware.ts**: control de rutas protegidas, de esta manera tendremos restricciones de acceso para el usuario, en este caso acceso al dashboard.
- **package.json**: dependencias del proyecto y scripts.
- **tsconfig.json**: configuración de TypeScript.

7.2. Backend

Para entender mejor el desarrollo del backend lo dividiremos en siguientes subsecciones desde la base hasta su desarrollo final.

7.2.1. Estructura y organización del backend

Esta aplicación cuenta con su backend desarrollado en Next.js que utiliza la arquitectura App Router. Esto permite definir rutas de cliente y de servidor en la misma estructura de proyecto. Las API Routes que ofrece Next.js sirven para controlar peticiones HTTP que son manejadas directamente desde carpetas del directorio app/api.

El código del backend se organiza en subdirectorios que corresponden a distintas funcionalidades de la aplicación. Dentro de cada módulo hay funciones asíncronas correspondientes a cada uno de los métodos de las API (por ejemplo POST, GET, PATCH) que representan un endpoint del servidor y que maneja operaciones CRUD con la base de datos.

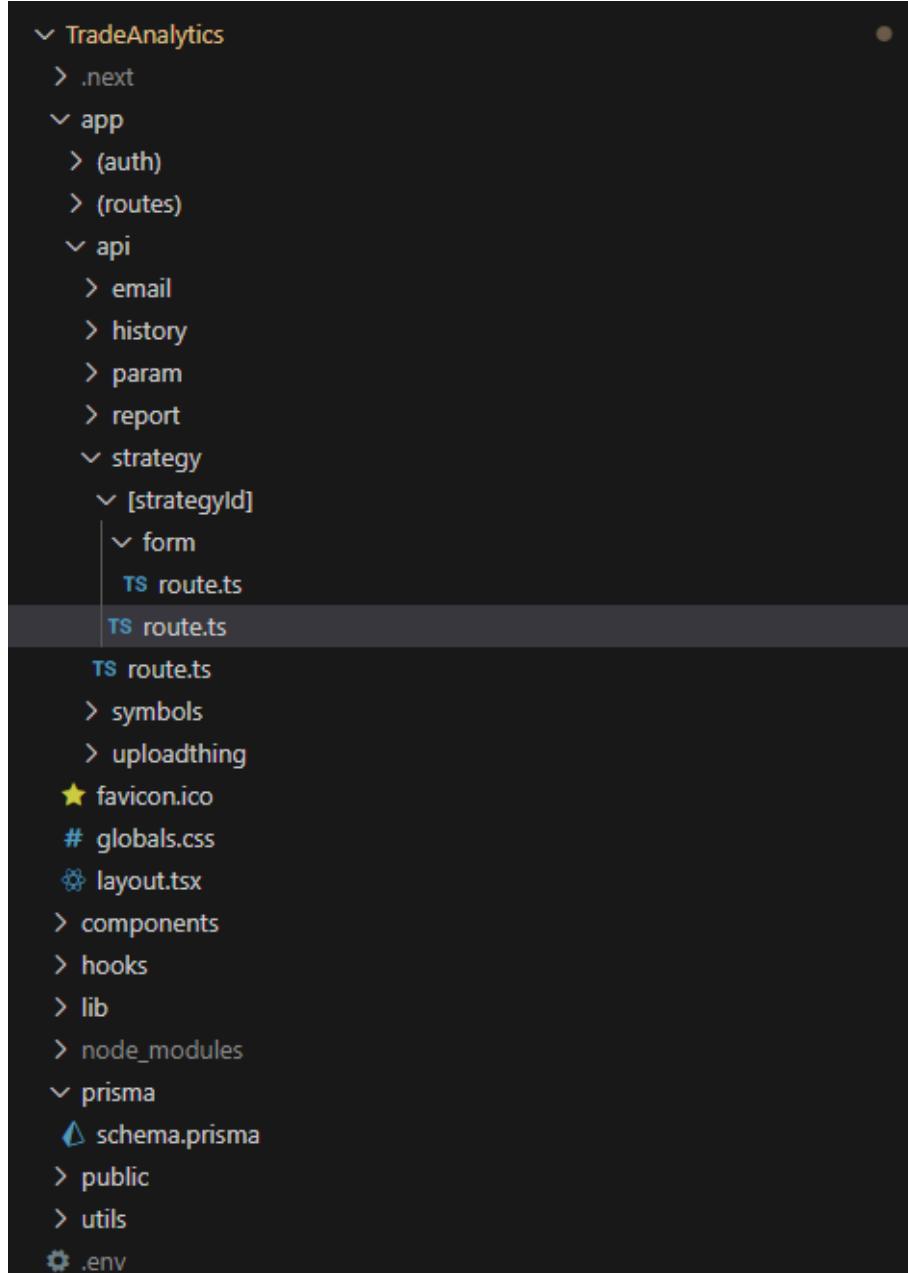


Figura 41: Estructura backend.

La figura 41 muestra una visión general de la estructura del proyecto, si nos enfocamos en las rutas del backend (API Routes) estan distribuidas por funcionalidades. Cada carpeta hace referencia a un módulo determinado del sistema:

- email/: La gestión del envío de correos, en la página principal sección de contactos donde el usuario puede mandar un mensaje por vía email.
- history/: Los históricos de backtesting que guarda los parámetros de las estrategias, realizados por el usuario.
- param/: El manejo de parámetros para las estrategias específicas.
- report/: La generación o exportación de informes relacionados con los resultados del backtesting.
- strategy/: Las estrategias de trading. Contiene subrutas dinámicas como [strategyId] para

seleccionar una estrategia en concreto, los demás directorios presenta también rutas dinámicas.

- symbols/: La gestión de los símbolos financieros disponibles (EUR/USD, BTC/USD, etc.).
- uploadthing/: La subida y gestión de archivos multimedia, en este caso para imágenes, esta librería nos proporciona almacenamientos y gestión de estos archivos [31].

Cada directorio que aparece bajo el directorio app/api/ es también un punto de acceso del backend que recibe las peticiones HTTP que corresponda. Esta estrategia modular hace que la ampliación y el mantenimiento del backend sea más fácil puesto que cada porción o parte del sistema está contenida en un contexto concreto.

7.2.2. Ejemplo de implementación de la api

Next.js permite definir rutas dinámicas en la carpeta app/api con corchetes ([]), tal como funcionan los parámetros de las rutas de su backend. En el ejemplo del archivo app/api/strategy/[strategyId]/route.ts se ha llegado a definir una ruta dinámica que en este caso consigue representar el identificador de una estrategia en el sistema: strategyId. La ruta permite realizar las operaciones básicas para trabajar con estrategias: obtener (GET), modificar (PATCH) y eliminar (DELETE), a parte tenemos la ruta general que permite crear (POST).

```
TradeAnalytics > app > api > strategy > [strategyId] > ts route.ts > PATCH
You, 5 months ago | 1 author (You)
1 import { db } from "@/lib/db";
2 import { auth } from "@clerk/nextjs/server";
3 import { NextResponse } from "next/server";
4
5 export async function PATCH(req: Request, { params }: { params: { strategyId: string } }) {
6   try {
7     const { userId } = auth();
8     const { strategyId } = params;
9     const { isPublic } = await req.json();
10
11    if (!userId) {
12      return new NextResponse("Unauthorized", { status: 401 });
13    }
14
15    const strategy = await db.strategy.update({
16      where: {
17        id: strategyId,
18      },
19      data: {
20        isPublic: isPublic,
21      }
22    });
23
24    return NextResponse.json(strategy, { status: 201 });
25  } catch (error) {
26    console.error("[strategy ID PATCH]", error);
27    return new NextResponse("Internal Server Error", { status: 500 });
28  }
29}
30
```

Figura 42: Api PATCH.

En la figura 42 muestra el código en typescript donde permite modificar una estrategia (PATCH), en la siguientes figuras muestra las funcionalidades restantes.

```

export async function POST(req: Request) {
  try {
    const { userId } = auth();
    const data = await req.json();

    if (!userId) {
      return new NextResponse("Unauthorized", { status: 401 });
    }

    const strategy = await db.strategy.create({
      data: {
        userId,
        ...data
      }
    });

    return NextResponse.json(strategy, { status: 201 });
  } catch (error) {
    console.error("[STRATEGY]", error);
    return new NextResponse("Internal Server Error", { status: 500 });
  }
}

```

You, 7 months ago • advance section admin

Figura 43: Api POST.

En la figura 43 muestra la funcionalidad POST: permite crear una estrategia.

```

export async function GET(req: Request, { params }: { params: { strategyId: string } }) {
  try {
    const { userId } = auth();
    const { strategyId } = params;

    if (!userId) {
      return new NextResponse("Unauthorized", { status: 401 });
    }

    const strategy = await db.strategy.findUnique({
      where: {
        id: strategyId,
      },
    });

    if (!strategy) {
      return new NextResponse("Strategy not found", { status: 404 });
    }

    return NextResponse.json(strategy, { status: 200 });
  } catch (error) {
    console.error("[strategy GET]", error);
    return new NextResponse("Internal Server Error", { status: 500 });
  }
}

```

Figura 44: Api GET.

En la figura 44 muestra la funcionalidad GET: obtiene los datos de una estrategia concreta.

```

export async function DELETE(req: Request, { params } : { params: { strategyId: string } }) {
    try {
        const { userId } = auth();
        const { strategyId } = params;

        if (!userId) {
            return new NextResponse("Unauthorized", { status: 401 });
        }

        const strategy = await db.strategy.delete({
            where: {
                id: strategyId,
            }
        });

        return NextResponse.json(strategy, { status: 201 });
    } catch (error) {
        console.error("[DELETE FORM ID]", error);
        return new NextResponse("Internal Server Error", { status: 500 });
    }
}

```

Figura 45: Api DELETE.

En la figura 45 muestra la funcionalidad DELETE: elimina una estrategia del sistema de forma permanente.

Se ha mostrado un ejemplo con la api strategy, las demás api estan formadas por la misma estructura GRUD.

Evidentemente, esto es una prueba de cómo se podría hacer un backend apoyado en la arquitectura frontend de Next.js de una manera modular, escalable y mantenible con rutas RESTful originadas en un solo lugar.

7.2.3. Sistema de gestión de usuarios CLERK

En la administración de usuarios, el registro y la autenticación se ha implementado un servicio externo que es Clerk. Éste permite implementar procesos completos de autenticación de una manera rápida y segura en aplicaciones construidas con Next.js

Clerk cuenta con componentes listos para su uso, los cuales se han utilizado para implementar las rutas /auth/sign-in y /auth/sign-up. Dichas rutas están definidas con el uso de archivos que permiten renderizar los formularios, sin que sea necesaria la parte del diseño que se realiza desde cero.

En el código del lado del backend, Clerk también facilita la obtención de la persona usuario autenticada, a partir de funciones como auth() que se visualiza en el ejemplo de la funcionalidad PATCH de strategy anteriormente mencionada, que devuelven por ejemplo el userId y permiten asociar cada estrategia o historial al usuario correspondiente a partir de las rutas API.

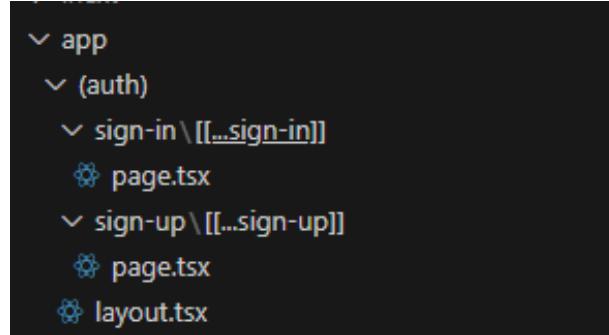


Figura 46: Estructura CLERK.

En la figura 46 se puede apreciar la estructura para integrar clerk en nuestra aplicación, este servicio nos ofrece directamente el código para solo realizar pequeñas modificaciones, las mayores modificaciones son en la parte de la interfaz ya que el idioma predeterminado es el inglés y se requiere cambiar el idioma manualmente para cumplir con el mismo idioma en toda la aplicación web.

Finalmente, Clerk maneja la parte de la seguridad, el almacenamiento de sesiones, el acceso con OAuth (Google o GitHub) y la verificación por correo, permitiendo ofrecer un sistema robusto sin tener que implementar esta lógica desde cero.

7.2.4. Base de datos, Prisma y PostgreSQL

El desarrollo de la migración e interacción de datos del sistema ha sido implementada en base a Prisma ORM, una herramienta eficaz para trabajar con bases de datos relacionales dentro del contexto de proyectos de Node.js y TypeScript. En este ejemplo el que se presenta, Prisma está asociado a una base de datos PostgreSQL, configurada en el archivo .env de la aplicación como la variable DATABASE_URL, .env sirve para alojar todas las variables que serán privadas en el despliegue de la aplicación.

Prisma permite definir el esquema de la base de datos a través de un lenguaje declarativo en el archivo schema.prisma, a partir del cual se genera un cliente que se utiliza en el código para llevar a cabo las distintas operaciones que pueden ser crear, consultar, actualizar y borrar registros de forma segura y tipada.

```

You, 6 months ago | 1 author (You)
model Strategy {
    id      String @id @default(uuid())
    userId String @db.Text
    name   String @db.Text
    description String @db.Text
    photo   String @db.Text
    isPublic Boolean @default(false)

    params Param[]
    histories History[]

    createdAt DateTime @default(now())
    updatedAt DateTime @updatedAt
}

You, 5 months ago | 1 author (You)
model Param {
    id      String @id @default(uuid())
    strategyId String @db.Text
    name   String @db.Text
    description String? @db.Text
    value   String @db.Text
    min_filter_value   String? @db.Text
    max_filter_value   String? @db.Text

    @relation(fields: [strategyId], references: [id], onDelete: Cascade)
    historyParams HistoryParam[]

    createdAt DateTime @default(now())
    updatedAt DateTime @updatedAt

    @@index([strategyId])
}

```

Figura 47: Ejemplo código prisma.

En la figura 47 se muestran dos de los modelos principales del sistema como ejemplo, los modelos completos se encuentran en el apartado diseño de la base de datos.

Strategy: representa una estrategia de trading generada por el administrador. En cada registro de esta entidad se mantiene su id; el id de la persona que la ha creado, el userId; el nombre de la estrategia; la descripción de la estrategia; la foto asociada a la estrategia; y el valor booleano isPublic que permite hacerla visible a los usuarios que nos sean administradores. También contiene relaciones hacia sus parámetros (params) y hacia su historial de pruebas (histories).

Param: contiene los parámetros que pueden configurar una estrategia. Se relaciona con un strategyId que indica a qué estrategia le corresponde el parámetro, incluyendo el nombre, el valor, los posibles límites de filtrado (min_filter_value y max_filter_value) y una posible descripción. También lo relaciona con los históricos (historyParams) donde ese parámetro se utilizó.

Ambos modelos poseen campos automáticos que registran la fecha de creación y la de última actualización (createdAt, updatedAt) lo que favorece el control del tiempo de los datos. También están especificadas las relaciones entre entidades con la directiva @relation, y se mejora el rendimiento con índices utilizando @index.

Con Prisma, la lógica de acceso a la base de datos se mantiene limpia, eficaz y coherente con el modelo de datos de la propia aplicación.

7.2.5. Programas de intercomunicación entre la aplicación web y la plataforma Metatrader 5 en local

Con el objetivo de establecer la comunicación entre la interfaz web que hemos realizado y la plataforma de trading MetaTrader 5 instalada localmente, se han trabajado unos scripts en Python que permiten automatizar la ejecución de las estrategias, capturar resultados y transformarlos a formatos legibles y reutilizables, como JSON. Dicha comunicación se lleva a cabo mediante una API REST construida con Flask que permite que la aplicación web pueda enviar datos, solicitar ejecuciones de backtesting y recibir informes de resultados. Entre las principales librerías utilizadas se encuentran:

- Flask: marco/entorno muy ligero para la creación de servidores web con Python. Se hace uso del mismo para exponer los endpoints que recibe y procesa datos desde la aplicación web.
- Flask-CORS: es utilizada para permitir que el servidor en Python pueda aceptar peticiones procedentes de dominios diferentes (por ejemplo localhost o Ngrok), de forma que se eliminan las restricciones que impone el CORS.
- MetaTrader5 (mt5): es la API oficial de MetaQuotes para interactuar con MetaTrader 5 desde Python. Permite inicializar la plataforma, acceder a los símbolos, lanzar operaciones o cerrar la sesión.
- subprocess / os / psutil: permiten lanzar MetaTrader 5 con configuraciones concretas, controlar la ejecución de la misma y acceder a los archivos que este genera.
- html2text / json: son las utilizadas para transformar los reportes en HTML que genera MetaTrader 5 a formato texto plano o estructuras JSON que se pueden enviar a la aplicación web.

Seguidamente se explican los tres scripts en los que se ha desarrollado la funcionalidad de la API local:

info-symbol.py

Es el script que contiene la API local. En concreto, usa Flask para obtener diferentes endpoints, como por ejemplo:

- /api/symbols: obtiene todos los símbolos disponibles desde MetaTrader 5.
- /api/submit_strategy: recibe los parámetros desde el frontend y genera todo el archivo .ini que configura la estrategia de MetaTrader y dependiendo de la estrategia elegida (RSI, Apertura NY, AMD), genera el archivo .ini correspondiente.

Después lanza la aplicación MetaTrader 5 con dicho archivo .ini, ejecuta el backtesting y empieza la conversión del reporte donde se mostrará toda la información y estadísticas de los resultados del backtesting, que estará disponible en el cliente web.

También tiene endpoints para descargar:

- La imagen o gráficas generada por MT5 del backtest (/api/report_image)

- El informe en formato JSON (/api/report)

Dicho script controla tanto la lógica de la automatización como los pasos de preparación para que MT5 se ejecute de forma independiente. Al finalizar, cierra la plataforma MT5 automáticamente.

leer_report_mt5.py

Cuando MetaTrader 5 genera un archivo HTML con el resultado del backtest, este script lo transforma a texto plano, legible y estructurado. Utiliza la librería html2text para eliminar las etiquetas HTML y dejar el contenido importante del informe como las estadísticas, las operaciones realizadas y los parámetros que han sido utilizados.

Igualmente limpia la sección de transacciones de forma que cada operación quede registrada en una única línea y dispuesta para su análisis o transformación posterior. El resultado final queda guardado como reporte_convertido.txt

convertir_resultado_json_mt5.py

Este script tiene como misión, a partir del archivo .txt generado anteriormente, dar el paso de transformarlo a un objeto estructurado en formato JSON. Esto va a permitir que los resultados se puedan consumir fácilmente desde la aplicación web o almacenarlos en una base de datos.

Detallan las secciones del informe (Configuración, Resultados, Órdenes, Transacciones) y le extrae la información clave de cada uno de los apartados. Ésta correspondería a la última fase del procesamiento y da como resultado un informe limpio, ordenado y reutilizable en reporte_convertido.json.

7.3. Algoritmos de trading

Los algoritmos de trading son estrategias automatizadas que ejecutan órdenes de compra y de venta sistemáticamente y con base en unas condiciones predeterminadas. Estas estrategias hacen desaparecer el componente emocional del trading y optimizan la decisión de estas operaciones a partir de indicadores técnicos y reglas establecidas [10].

Se ha desarrollado e implementado tres algoritmos de trading automatizado, utilizando para ello MQL5 (MetaQuotes Language 5) del software MetaTrader 5. Cada estrategia es un script de trading independiente, donde según la estrategia se modulariza en más ficheros script para una mejor comprensión. Estas estrategias están introducidas dentro de un entorno de prueba (back-testing) para evaluar su rendimiento sobre los datos históricos reales. Reflejamos a continuación la lógica de funcionamiento, los parámetros que se han utilizado y un fragmento del pseudocódigo que representa la implementación funcional.

7.3.1. Estrategia basada en el Índice de Fuerza Relativa (RSI)

El Índice de Fuerza Relativa (RSI) [34] es un indicador técnico que mide la velocidad y el cambio de los movimientos del precio, oscilando entre 0 y 100. Generalmente se considera que un activo está sobrecomprado cuando el RSI supera un umbral alto (por ejemplo, 70) y el precio podría encontrarse en una zona de agotamiento, susceptible a correcciones a la baja. Por el contrario, un

RSI inferior a un umbral bajo (por ejemplo, 30) indicaría que el activo está sobre vendido, lo que podría interpretarse como una oportunidad de compra, ya que podría producirse una reversión al alza.

La primera estrategia propuesta en este proyecto se basa, precisamente, en estas zonas del RSI:

- Se ejecuta una compra cuando el RSI se encuentra en una zona sobre vendida, lo que indicaría que el precio podría rebotar al alza, si se abre una compra no se realizaran más operaciones de compra hasta que el RSI vuelva a una zona de sobre vendida.
- Se ejecuta una venta cuando el RSI se encuentra en una zona sobre comprada, lo que sugeriría una posible corrección a la baja, si se abre una venta no se realizaran más operaciones de venta hasta vuelva a una zona de sobre comprada.

Esta estrategia persigue la identificación de los mejores momentos en los cuales entrar al mercado, en base a la teoría de que los precios tienden a revertirse tras haber alcanzado extremos de sobre compra o de sobre venta, ya que el precio se mueve por oscilaciones, de aquí la famosa frase "Todo lo que sube después tiende a bajar".

Parámetros y funciones

Parámetros de configuración:

- **SL_POINTS:** Define el número de puntos (pips) que se utilizarán como nivel de Stop Loss en cada operación. Indica la máxima pérdida permitida antes de cerrar automáticamente la posición.
- **RATIO_TP:** Es el ratio de beneficio en relación al Stop Loss. Sirve para calcular el Take Profit (TP), es decir, el objetivo de beneficio en puntos. Por ejemplo, si **SL_POINTS = 2000** y **RATIO_TP = 1.0**, entonces **TP_POINTS = 2000**.
- **LOTAJE_STATIC:** Representa el tamaño fijo del lote o cantidad de capital que se arriesgará en cada operación. En el contexto de MetaTrader 5, es el volumen de la orden de compra o venta.
- **RSI_SOBRECOMPRA** y **RSI_SOBREVENDIDO:** Son los umbrales que definen cuándo se considera que un activo está sobre comprado o sobre vendido. Por defecto, los valores son 70 y 30 respectivamente. Si el RSI supera el valor de sobre compra, se considera una posible señal de venta; si cae por debajo del valor de sobre venta, se interpreta como señal de compra.
- **RsiPeriod:** Número de períodos utilizados para calcular el Índice de Fuerza Relativa (RSI). Este valor define cuántas velas o datos de cierre se consideran en el cálculo del indicador.
- **TP_POINTS:** Cantidad de puntos utilizada para establecer el Take Profit, calculado dinámicamente como **SL_POINTS * RATIO_TP**.
- **opera_compra** y **opera_venta:** Variables booleanas que evitan que el sistema ejecute múltiples compras o ventas consecutivas cuando se mantiene una misma condición de entrada.

Funciones:

- **rsi_sobrecomprado():**
 - **Argumentos:** Ninguno.
 - **Descripción:** Comprueba si el valor actual del RSI está por encima del umbral de sobre compra (**RSI_SOBRECOMPRA**).

- **Retorna:** `true` si el RSI indica sobrecompra; `false` en caso contrario.
 - **Objetivo:** Identificar un posible punto de entrada para una venta.
- `rsi_sobrevenido():`
- **Argumentos:** Ninguno.
 - **Descripción:** Comprueba si el valor actual del RSI está por debajo del umbral de sobreventa (`RSI_SOBREVENDIDO`).
 - **Retorna:** `true` si el RSI indica sobreventa; `false` en caso contrario.
 - **Objetivo:** Identificar un posible punto de entrada para una compra.
- `OnInit():`
- **Argumentos:** Ninguno.
 - **Descripción:** Función de inicialización que se ejecuta al cargar la estrategia. Inicializa el indicador RSI, configura los arrays necesarios y calcula el valor de `TP_POINTS`.
 - **Objetivo:** Preparar el sistema antes de empezar a recibir ticks del mercado.
- `OnTick():`
- **Argumentos:** Ninguno.
 - **Descripción:** Función que se ejecuta en cada nuevo tick del mercado. Evalúa las condiciones actuales del RSI y ejecuta operaciones de compra o venta con sus respectivos niveles de `Stop Loss` y `Take Profit`.
 - **Objetivo:** Evaluar condiciones de entrada y gestionar operaciones automáticamente.

Este es el pseudocódigo, contiene la clase Base donde se inicializa los parametros básicos y el archivo donde realiza las operaciones de la estrategia:

Algorithm 1 Definición de la Clase Base

```
1: Clase Base
2:
3:   Atributos privados:
4:     SL_POINTS           ▷ Pips o puntos máximos de perdida (STOP LOSS)
5:     RATIO_TP            ▷ Ratio de beneficio en base al Stop Loss
6:     LOTAJE_STATIC        ▷ Lotaje o cantidad de inversión por operación
7:   Constructor Base(sl_points, ratio_tp, lotaje_static)
8:     SL_POINTS ← sl_points (por defecto 2000)
9:     RATIO_TP ← ratio_tp (por defecto 2.0)
10:    LOTAJE_STATIC ← lotaje_static (por defecto 0.1)
11:
12:   Métodos públicos:
13:
14:   function GETSLPOINTS
15:     return SL_POINTS
16:   end function
17:
18:   function GETRATIOTP
19:     return RATIO_TP
20:   end function
21:
22:   function GETLOTAJESTATIC
23:     return LOTAJE_STATIC
24:   end function
25:
26:   function SETSLPOINTS(sl_points)
27:     SL_POINTS ← sl_points
28:   end function
29:
30:   function SETRATIOTP(ratio_tp)
31:     RATIO_TP ← ratio_tp
32:   end function
33:
34:   function SETLOTAJESTATIC(lotaje_static)
35:     LOTAJE_STATIC ← lotaje_static
36:   end function
```

Algorithm 2 Estrategia de Trading basada en RSI

```
1: INICIAR librerías y objetos necesarios
2:
3: DEFINIR variables:
4:   Parámetros estáticos:
5:     rsih                                     ▷ Donde inicializaremos el indicador RSI
6:     rsi[]                                    ▷ Array de valores RSI
7:     opera_compra, opera_venta                ▷ Para que solo realice una compra o una venta
8:   Parámetros de entrada:
9:     RsiPeriod = 14                           ▷ Periodos de velas en el tiempo para representar el RSI
10:    RSI_SOBRECOMPRA = 70                      ▷ Zona de sobrecompra
11:    RSI_SOBREVENDIDO = 30                     ▷ Zona de sobreventa
12:    SL_POINTS = 2000                          ▷ Pips o puntos máximos de perdida (STOP LOSS)
13:    RATIO_TP = 1,0                            ▷ Ratio de beneficio en base al Stop Loss
14:    LOTAJE_STATIC = 0,1                       ▷ Lotaje o cantidad de inversión por operación
15:    Base(SL_POINTS, RATIO_TP, LOTAJE_STATIC)  ▷ Inicializar con la base
16:
17: function RSI_SOBRECOMPRA
18:   return rsi[0] > RSI_SOBRECOMPRA
19: end function
20:
21: function RSI_SOBREVENDIDO
22:   return rsi[0] < RSI_SOBREVENDIDO
23: end function
24:
25: procedure ONINIT                         ▷ Función inicializadora
26:   Inicializar indicador RSI
27:   if rsih == INVALID_HANDLE then
28:     Error: No se pudo cargar el RSI
29:     return
30:   end if
31:   Configurar arrays y calcular TP_POINTS
32:   TP_POINTS = SL_POINTS × RATIO_TP
33: end procedure
34:
35: procedure ONTICK(Función inicializadora)
36:   Obtener el valor más reciente del RSI y las velas
37:   if rsi_sobrevendido() and ¬opera_compra then
38:     Ejecutar orden de compra en precio ASK con SL y TP
39:     opera_compra ← true
40:     opera_venta ← false
41:   end if
42:   if rsi_sobrecomprado() and ¬opera_venta then
43:     Ejecutar orden de venta en precio BID con SL y TP
44:     opera_venta ← true
45:     opera_compra ← false
46:   end if
47: end procedure
```

Se muestra un ejemplo de simulación para visualizar como se ejecuta cada operación:



Figura 48: Operación de venta con la estrategia RSI

En la figura 48 se muestra como se lleva a cabo la estrategia basada en el RSI (índice de fuerza relativa) a partir de un gráfico de velas japonesas, se realiza en el simbolo EUR/USD (euro vs dollar), que es un simbolo de forex. La estrategia donde se ejecuta estas operaciones está configurada con los valores por defecto; el RSI cuenta con un periodo de cálculo de 14 velas, y los niveles de referencia de sobrecompra y de sobreventa están fijados en 70 y 30, respectivamente.

En la parte inferior del gráfico, la línea del RSI se presenta en azul, junto con un área destacada en rojo que apunta un umbral clave que indica que el RSI se encuentra en el nivel de sobrecompra que es 70.

En ese momento realiza una operación de venta, en el grafico de velas se muestra dos linea horizontales en rojo, la que se situa en la parte superior es el stop loss, sirve para limitar la perdida de la operación, en este caso si el precio sube y toca la linea roja superior cerrara la posición en perdidas.

En cambio, si toca la linea roja horizontal inferior se cierra la operación finaliza con ganancias, esa linea inferior se le llama take profit que será nuestro precio objetivo de ganancias.



Figura 49: Operación de compra con la estrategia RSI

En la figura 49 se muestra otro ejemplo de simulación, en este caso en la parte inferior del gráfico, la línea del RSI que se presenta en azul, junto con un área destacada en rojo que apunta un umbral clave que indica que el RSI se encuentra en el nivel de sobreventa con valor 30.

En ese momento realiza una operación de compra, en el grafico de velas se muestra dos linea horizontales en rojo, la que se situa en la parte superior en este caso es el take profit, la linea roja horizontal inferior se cierra con perdidas ya que en este caso es el stop loss.

Todos los parametros son configurables, en este caso esta puesto con un ratio 1:1, es decir, cada operación que se ejecuta se pierde una unidad o se gana una unidad, un ejemplo con una inversión en euros sería que cada operación ganadora se obtendrá una ganancia de 100 euros y en el caso contrario si es una operación perdedora se obtendrá una perdida de 100 euros, en el caso de ratio 1:2 se pierde una cantidad de 100 euros si la operación ha sido perdedora y se gana una cantidad de 200 euros si la operación ha sido ganadora.

7.3.2. Estrategia en el mercado de la apertura de Nueva York

El mercado de nueva york se caracteriza por ser uno de los mercados más volátiles que existen en el mundo. El momento de la apertura de la cotización en el mercado provoca en muchas ocasiones grandes movimientos y es precisamente este movimiento fuerte junto con la liquidez lo que se busca explotar para intentar conseguir beneficios en el trading. Por lo que la estrategia utilizada se basa en aprovechar este tipo de volatilidad y consiste en realizar órdenes de compra/venta en función del comportamiento del mercado una vez pasados los primeros momentos de la apertura [9].

El funcionamiento de la estrategia es el siguiente:

Se determina el nivel de entrada en función de la apertura del mercado, es decir, la cotización a nivel de precio de apertura del mercado en NY.

Se colocan 2 órdenes condicionadas:

- Orden de compra si el precio es superior a un determinado nivel por encima del precio de apertura.
- Orden de venta si el precio es inferior a un determinado nivel por debajo del precio de apertura.

Es decir, cuando el precio alcanza uno de los niveles definidos anteriormente es cuando se activa la orden correspondiente y realiza la operación en la dirección del fuerte movimiento de los precios.

Con esta estrategia se gestionan el riesgo aplicando las correspondientes órdenes de stop loss y el correspondiente take profit anticipando que si el precio se mueve a favor o en contra de la operación entra en determinados niveles, se cierran automáticamente.

La estrategia busca conseguir el beneficio de los movimientos bruscos con alta volatilidad que suelen darse en los primeros minutos después de abrir el mercado de nueva york, un tipo de volatilidad que se puede aprovechar para obtener una probabilidad estadística positiva.

Parámetros y funciones

Parámetros de configuración:

- **SL_POINTS:** Número de puntos de pérdida máxima por operación. Define el nivel de Stop Loss.
- **RATIO_TP:** Ratio que define el Take Profit en relación al Stop Loss. Por ejemplo, si **SL_POINTS = 2000** y **RATIO_TP = 2.0**, entonces **TP = 4000** puntos.
- **LOTAJE_STATIC:** Tamaño del lote con el que se ejecutará la operación. Representa la cantidad fija de inversión.
- **PUNTOS_PRECIO:** Distancia (en puntos) entre el precio actual y las órdenes pendientes Buy Stop y Sell Stop.
- **HORAS_EXPIRACION:** Número de horas tras las cuales las órdenes pendientes expiran si no han sido ejecutadas.
- **time_set:** Variable booleana que se utiliza para controlar si el temporizador se ha establecido correctamente para ejecuciones posteriores.

Funciones:

- **get_segundos_hasta_NY():**
 - **Argumentos:** Ninguno.
 - **Descripción:** Calcula el número de segundos restantes hasta la apertura del mercado de Nueva York (configurada para las 15:30 UTC en MT5).
 - **Retorna:** Un número entero con los segundos restantes hasta la apertura.
 - **Objetivo:** Determinar el momento exacto en que debe activarse el temporizador inicial de la estrategia.
- **OnInit():**
 - **Argumentos:** Ninguno.
 - **Descripción:** Función de inicialización del sistema. Programa el temporizador en función de los segundos calculados hasta la apertura de Nueva York.
 - **Objetivo:** Preparar la ejecución automática de la estrategia al comenzar la sesión de Nueva York.
- **OnTimer():**
 - **Argumentos:** Ninguno.
 - **Descripción:** Función que se ejecuta cuando se activa el temporizador. Calcula los niveles de entrada para órdenes pendientes, comprueba si es día hábil, y coloca órdenes Buy Stop y Sell Stop con sus respectivos SL, TP y tiempo de expiración.
 - **Objetivo:** Ejecutar la lógica principal de la estrategia en la apertura del mercado, gestionando las órdenes de forma automatizada.

Este es el pseudocódigo, contiene la clase Base donde se inicializa los parametros básicos y el archivo donde realiza las operaciones de la estrategia:

Algorithm 3 Definición de la Clase Base

```
1: Clase Base
2:
3: Atributos privados:
4:   SL_POINTS           ▷ Pips o puntos máximos de perdida (STOP LOSS)
5:   RATIO_TP            ▷ Ratio de beneficio en base al Stop Loss
6:   LOTAJE_STATIC        ▷ Lotaje o cantidad de inversión por operación
7: Constructor Base(sl_points, ratio_tp, lotaje_static)
8:   SL_POINTS ← sl_points (por defecto 2000)
9:   RATIO_TP ← ratio_tp (por defecto 2.0)
10:  LOTAJE_STATIC ← lotaje_static (por defecto 0.1)
11:
12: Métodos públicos:
13:
14: function GETSLPOINTS
15:   return SL_POINTS
16: end function
17:
18: function GETRATIOTP
19:   return RATIO_TP
20: end function
21:
22: function GETLOTAJESTATIC
23:   return LOTAJE_STATIC
24: end function
25:
26: function SETSLPOINTS(sl_points)
27:   SL_POINTS ← sl_points
28: end function
29:
30: function SETRATIOTP(ratio_tp)
31:   RATIO_TP ← ratio_tp
32: end function
33:
34: function SETLOTAJESTATIC(lotaje_static)
35:   LOTAJE_STATIC ← lotaje_static
36: end function
```

Algorithm 4 Estrategia de Apertura de Nueva York

```

1: Definir variables de configuración:
2: SL_POINTS ← 2000                                ▷ Puntos del Stop Loss
3: RATIO_TP ← 2.0                                    ▷ Relación Take Profit / Stop Loss
4: LOTAJE_STATIC ← 0.1                               ▷ Tamaño del lote
5: PUNTOS_PRECIO ← 2000                            ▷ Distancia de las órdenes Buy Stop / Sell Stop
6: HORAS_EXPIRACION ← 2                            ▷ Horas hasta que expiran las órdenes pendientes
7: Base(SL_POINTS, RATIO_TP, LOTAJE_STATIC)        ▷ Inicializar con la base
8: Definir bandera de tiempo: time_set ← falso
9:
10: function GET_SEGUNDOS_HASTA_NY
11:     Obtener la hora actual
12:     Ajustar la hora a la apertura del mercado de Nueva York (16:30 UTC+1 → 15:30 UTC en
    MT5)
13:     Calcular la diferencia en segundos hasta la apertura del mercado
14:     if hora actual < hora de apertura NY then
15:         return segundos restantes hasta apertura
16:     else
17:         return segundos hasta la apertura del día siguiente
18:     end if
19: end function
20:
21: procedure ONINIT
22:     Programar evento de temporizador con GET_SEGUNDOS_HASTA_NY
23: end procedure
24:
25: procedure ONTIMER
26:     Obtener hora actual
27:     Convertir hora a formato de timestamp
28:     if time_set es falso then
29:         Eliminar temporizador actual
30:         Configurar temporizador cada 24 horas (para ejecución diaria)
31:         time_set ← verdadero
32:     end if
33:     if día actual es sábado o domingo then
34:         Retornar (No se ejecuta la estrategia en fines de semana)
35:     end if
36:     Obtener precios actuales:
37:         ask ← Precio de compra (ASK)
38:         bid ← Precio de venta (BID)
39:     Calcular niveles de órdenes pendientes:
40:         buy_price ← ask + PUNTOS_PRECIO * tamaño de punto
41:         sell_price ← bid - PUNTOS_PRECIO * tamaño de punto
42:     Colocar orden Buy Stop:
43:         Comprar en buy_price
44:         Stop Loss en buy_price - SL_POINTS * tamaño de punto
45:         Take Profit en buy_price + (RATIO_TP * SL_POINTS) * tamaño de punto
46:         Orden expira en HORAS_EXPIRACION horas
47:     Colocar orden Sell Stop:
48:         Vender en sell_price
49:         Stop Loss en sell_price + SL_POINTS * tamaño de punto
50:         Take Profit en sell_price - (RATIO_TP * SL_POINTS) * tamaño de punto
51:         Orden expira en HORAS_EXPIRACION horas
52: end procedure

```

Se muestra un ejemplo de simulación para visualizar como se ejecuta cada operación:



Figura 50: Operación de venta en la apertura de nueva york

En la figura 50 se muestra este ejemplo de simulación. Una vez se inicia la apertura de nueva york se colocan previamente 2 órdenes pendientes:

- Una orden de compra (buy stop) colocada por encima del precio actual en ese momento.
- Una orden de venta (sell stop) colocada por debajo del precio actual en ese momento.

Activación de la orden de venta: Como se puede ver en el gráfico, el precio ha llegado a caer y ha activado la orden de venta en el punto que hemos marcado como venta; en otras palabras, el precio ha cruzado el nivel en el que estaba colocada la orden pendiente y ha cruzado automáticamente la operación en corto.

Configuración del stop loss situado por encima del punto en el que hemos entrado en la operación y el objetivo es controlar las pérdidas en caso de que el mercado suba en lugar de bajar.

Configuración del take profit situado más abajo y marca el objetivo, es decir, si el precio llega a caer a esa line horizontal roja inferior, la operación se cerrará automáticamente con una ganancia.

7.3.3. Estrategia de Acumulación, Manipulación y Distribución

La teoría de Acumulación, Manipulación y Distribución [25] se fundamenta en la idea de que los mercados financieros son influenciados por grandes instituciones. Estas entidades realizan movimientos estratégicos en los precios con el fin de acumular activos a precios bajos y, posteriormente, distribuirlos a precios más altos. Este enfoque se desglosa en tres fases esenciales:

Acumulación: En esta etapa inicial, los grandes inversores adquieren activos de manera que no provocan fluctuaciones pronunciadas en los precios, manteniéndolos dentro de un rango lateral. Generalmente, esta fase se desarrolla en zonas de soporte clave.

Manipulación: Durante esta fase, el precio puede romper temporalmente los niveles de soporte o resistencia, creando señales engañosas que inducen a los traders minoristas a tomar decisiones erróneas. Posteriormente, el precio tiende a revertirse rápidamente en dirección opuesta.

Distribución: Una vez que las instituciones han acumulado suficientes posiciones, comienzan a vender, lo que genera un movimiento significativo, ya sea al alza o a la baja, dependiendo de la situación del mercado.

La estrategia propuesta en este proyecto tiene como objetivo identificar estas fases y aprovechar los momentos más importantes para entrar en el mercado:

- Se toman posiciones de compra cuando se detecta una fase de acumulación, seguida de una manipulación a la baja, y se observan señales de una posible reversión alcista.
- Se realizan ventas cuando se identifica una fase de distribución acompañada de manipulación al alza, junto con indicios de una posible reversión bajista.

Este enfoque permite operar en sincronía con los movimientos de las instituciones financieras, evitando caer en trampas del mercado y sacando provecho de los patrones de comportamiento donde intervienen grandes inversores.

Parámetros y funciones

Parámetros de configuración:

- **SL_POINTS:** Cantidad de puntos utilizada como nivel de **Stop Loss** en las operaciones.
- **RATIO_TP:** Relación entre **Take Profit** y **Stop Loss**. Se usa para calcular el beneficio objetivo en puntos.
- **LOTAJE_STATIC:** Tamaño fijo del lote para las órdenes de compra o venta. Define cuánto capital se arriesga en cada operación.
- **HORA_INICIO_RECOGIDA** y **HORA_FINAL_RECOGIDA**: Intervalo horario donde se identifican los niveles de liquidez, usados para definir zonas de acumulación y manipulación.
- **HORA_FINAL_OPERACIONES**: Hora a partir de la cual ya no se abren nuevas operaciones, o bien se cierran las que estén abiertas (si está activado el cierre automático).
- **USAR_HORA_CIERRE**: Parámetro booleano que determina si se deben cerrar las operaciones al finalizar la jornada.
- **timer_set**: Variable booleana que indica si ya se ha configurado el temporizador para ejecutar la estrategia diariamente.

Funciones:

- `get_max(valores[])`:
 - **Descripción:** Recorre un array de valores y devuelve el valor máximo.
 - **Argumentos:** Array de valores numéricos.
 - **Retorna:** El valor más alto del array.
- `get_min(valores[])`:
 - **Descripción:** Recorre un array de valores y devuelve el valor mínimo.
 - **Argumentos:** Array de valores numéricos.
 - **Retorna:** El valor más bajo del array.
- `get_datetime_by_hour(hora, minuto)`:
 - **Descripción:** Devuelve una marca de tiempo (`timestamp`) correspondiente a la hora y minuto especificados en el día actual.
 - **Retorna:** Objeto de fecha y hora ajustado a la hora solicitada.
- `get_liquidez_max()` y `get_liquidez_min()` (clase Liquidez):
 - **Descripción:** Devuelven respectivamente el máximo y el mínimo de la zona de liquidez identificada durante la fase de acumulación.
- `get_max_min(hora_inicio, minuto_inicio, hora_final, minuto_final)`:
 - **Descripción:** Calcula los valores máximos y mínimos del precio en el rango horario indicado. Se almacenan internamente como límites de la zona de liquidez.
- `pintar(hora_inicio, minuto_inicio, hora_final, minuto_final)` (clase Liquidez y ReentradaRango):
 - **Descripción:** Dibuja visualmente en el gráfico la zona de liquidez y las líneas verticales que indican el intervalo de operaciones.
- `cruce_compra(liquidez, velas[])` y `cruce_venta(liquidez, velas[])`:
 - **Descripción:** Detectan si el precio ha cruzado los niveles de liquidez mínimos o máximos, generando señales para entrar en compra o venta respectivamente.
- `operar(...)`:
 - **Descripción:** Ejecuta la lógica de entrada en mercado. Evalúa si se cumplen condiciones para abrir una orden de compra o venta y la ejecuta con los parámetros configurados.
 - **Parámetros clave:** Rango horario, uso del cierre automático, niveles de SL/TP, liquidez y lotaje.
- `reset()`:
 - **Descripción:** Reinicia las variables internas que controlan si una orden de compra o venta está abierta, para evitar ejecuciones duplicadas.
- `segundos_hasta_inicio_operaciones()`:
 - **Descripción:** Calcula cuántos segundos faltan hasta el inicio de la sesión de operaciones, según la hora de finalización de la recogida de datos.
- `OnInit()`:

- **Descripción:** Inicializa el sistema, programa el temporizador y reinicia el estado operativo.
- **OnTick():**
 - **Descripción:** Se ejecuta en cada nuevo tick del mercado y llama a la función `operar()` para evaluar condiciones y ejecutar operaciones si corresponde.
- **OnTimer():**
 - **Descripción:** Se ejecuta cuando se activa el temporizador. Gestiona la lógica diaria de la estrategia: calcula niveles de liquidez, los pinta y resetea el estado del sistema.

Este es el pseudocódigo, contiene la clase Base donde se inicializa los parametros básicos, la clase donde se definen y desarrollan las funciones de la estrategia, y por ultimo el fichero donde realiza las operaciones de la estrategia:

Algorithm 5 Definición de la Clase Base

```
1: Clase Base
2:
3: Atributos privados:
4:   SL_POINTS           ▷ Pips o puntos máximos de perdida (STOP LOSS)
5:   RATIO_TP            ▷ Ratio de beneficio en base al Stop Loss
6:   LOTAJE_STATIC        ▷ Lotaje o cantidad de inversión por operación
7: Constructor Base(sl_points, ratio_tp, lotaje_static)
8:   SL_POINTS ← sl_points (por defecto 2000)
9:   RATIO_TP ← ratio_tp (por defecto 2.0)
10:  LOTAJE_STATIC ← lotaje_static (por defecto 0.1)
11:
12: Métodos públicos:
13:
14: function GETSLPOINTS
15:   return SL_POINTS
16: end function
17:
18: function GETRATIOTP
19:   return RATIO_TP
20: end function
21:
22: function GETLOTAJESTATIC
23:   return LOTAJE_STATIC
24: end function
25:
26: function SETSLPOINTS(sl_points)
27:   SL_POINTS ← sl_points
28: end function
29:
30: function SETRATIOTP(ratio_tp)
31:   RATIO_TP ← ratio_tp
32: end function
33:
34: function SETLOTAJESTATIC(lotaje_static)
35:   LOTAJE_STATIC ← lotaje_static
36: end function
```

Algorithm 6 Estrategia de Acumulación, Manipulación y Distribución (Parte 1)

```
1: Definir funciones auxiliares
2: function GET_MAX(valores[])
3:     result  $\leftarrow$  -1
4:     for cada i en valores do
5:         if valores[i]  $>$  result then
6:             result  $\leftarrow$  valores[i]
7:         end if
8:     end for
9:     return result
10: end function
11: function GET_MIN(valores[])
12:     result  $\leftarrow$  INT_MAX
13:     for cada i en valores do
14:         if valores[i]  $<$  result then
15:             result  $\leftarrow$  valores[i]
16:         end if
17:     end for
18:     return result
19: end function
20: function GET_DATETIME_BY_HOUR(hora, minuto)
21:     Obtener hora actual y ajustarla a la hora deseada
22:     return Timestamp correspondiente
23: end function
```

Algorithm 7 Estrategia de Acumulación, Manipulación y Distribución (Parte 2)

1: **Clase Liquidez:**

2: **Atributos:**

3: max \leftarrow 0

4: min \leftarrow 0

5: **function** GET_LIQUIDEZ_MAX

6: **return** max

7: **end function**

8: **function** GET_LIQUIDEZ_MIN

9: **return** min

10: **end function**

11: **procedure** GET_MAX_MIN(hora_inicio, minuto_inicio, hora_final, minuto_final)

12: Obtener los valores máximos y mínimos entre los tiempos indicados

13: Guardar los valores en max y min

14: **end procedure**

15: **procedure** PINTAR(hora_inicio, minuto_inicio, hora_final, minuto_final)

16: Dibujar un rectángulo en el gráfico desde d_inicio a d_final, marcando la zona de liquidez

17: **end procedure**

18: **Clase ReentradaRango:**

19: **Atributos:**

20: compra_abierta \leftarrow FALSO

21: venta_abierta \leftarrow FALSO

22: **function** CRUCE_COMPRA(liquidez, velas[])

23: **return** Verdadero si el precio ha cruzado por encima del nivel de liquidez mínima

24: **end function**

25: **function** CRUCE_VENTA(liquidez, velas[])

26: **return** Verdadero si el precio ha cruzado por debajo del nivel de liquidez máxima

27: **end function**

28: **procedure** OPERAR(hora_inicio, minuto_inicio, usar_hora_cierre, hora_final, minuto_final, puntos_sl, puntos_tp, liquidez, lotaje_static)

29: Obtener el tiempo de inicio y fin

30: **if** el tiempo actual es mayor a la hora de cierre y usar_hora_cierre es verdadero **then**

31: Cerrar todas las posiciones abiertas

32: **return**

33: **end if**

34: **if** el tiempo actual está fuera del rango de operación **then**

35: **return**

36: **end if**

37: Obtener precios de las últimas dos velas

38: **if** cruce_compra(liquidez, velas) y compra_abierta es FALSO **then**

39: Abrir orden de compra con SL y TP definidos

40: compra_abierta \leftarrow VERDADERO

41: **end if**

42: **if** cruce_venta(liquidez, velas) y venta_abierta es FALSO **then**

43: Abrir orden de venta con SL y TP definidos

44: venta_abierta \leftarrow VERDADERO

45: **end if**

46: **end procedure**

47: **procedure** RESET

48: compra_abierta \leftarrow FALSO

49: venta_abierta \leftarrow FALSO

50: **end procedure**

51: **procedure** PINTAR(hora_inicio, minuto_inicio, hora_final, minuto_final)

52: Dibujar líneas verticales marcando el inicio y fin del rango en el gráfico

53: **end procedure**

Se muestra un ejemplo de simulación para visualizar como se ejecuta cada operación:

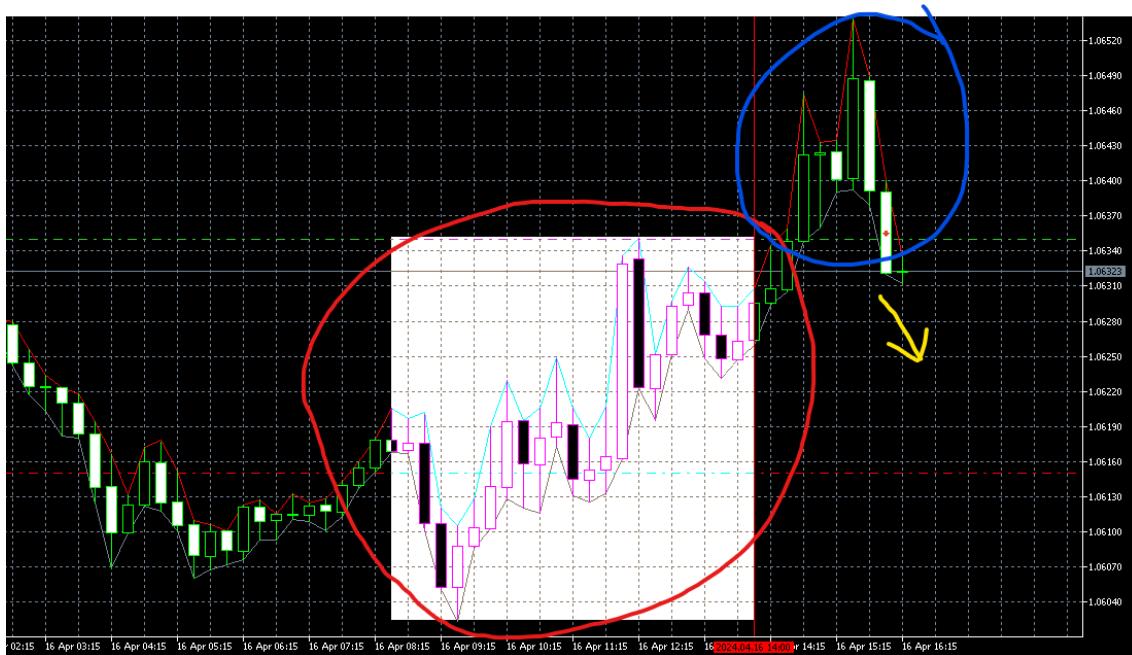


Figura 51: Simulación de la estrategia acumulación manipulación y distribución

En la figura 51 se muestra este ejemplo de simulación. Tenemos que identificar tres zonas:

- Acumulación una zona donde se forma una indecision entre compradores y vendedores, en la simulación se visualiza como una caja blanca, en la figura 5 se ha marcado con un circulo rojo para tener mas claridad, en esta zona obtenemos los máximos y mínimos segun las horas que se visualizan estas zonas de acumulación, por ejemplo en el simbolo EUR/USD (euro contra el dolar), se forma de 9 a 11, entonces ponemos ese rango horario.
- Manipulación cuando rompe este máximo o mínimo de la zona de acumulación, en este caso ha roto el máximo, este movimiento llama la atención a los traders a creer que se inicia una tendencia alcista, esta zona esta señalada con color azul.
- Distribución cuando los traders creen que es una buen oportunidad de comprar, nosotros esperamos a que el precio en este caso baje de nuevo, y si toca el máximo de la zona de acumulación es cuando se realiza una venta, por lo que hemos deducido que el precio ha sido manipulado ya que la dirección del precio ha cambiado, tras la venta aprovechamos una distribución deduciendo que la tendencia sera bajista dibujado con una flecha de color amarillo, en este caso en la figura 6 se aprecia como fue efectiva en este caso la estrategia.

Algorithm 8 Ejecución de la estrategia

```

1: Definir parámetros de configuración:
2: HORA_INICIO_RECOGIDA ← 8:30                                ▷ Hora de inicio de recopilación de datos
3: HORA_FINAL_RECOGIDA ← 14:00                                 ▷ Hora de fin de recopilación de datos
4: HORA_FINAL_OPERACIONES ← 22:00                               ▷ Hora límite para operaciones
5: USAR_HORA_CIERRE ← Verdadero                            ▷ Determina si cerrar operaciones al final del día
6: SL_POINTS ← 2000                                         ▷ Stop Loss en puntos
7: RATIO_TP ← 1.0                                           ▷ Relación Take Profit / Stop Loss
8: LOTAJE_STATIC ← 0.1                                       ▷ Tamaño de lote fijo

9: Definir instancias de clases:
10: Base ← (SL_POINTS, RATIO_TP, LOTAJE_STATIC)
11: Liquidez l                                              ▷ Objeto para gestionar niveles de liquidez
12: ReentradaRango r                                         ▷ Objeto para gestionar operaciones

13: Definir bandera de temporizador: timer_set ← Falso
14: function SEGUNDOS_HASTA_INICIO_OPERACIONES
15:   Obtener la hora actual
16:   Ajustar la hora a la hora final de la recogida de datos
17:   Calcular la diferencia de tiempo hasta ese momento
18:   if hora actual < hora programada then
19:     return segundos restantes hasta inicio de operaciones
20:   else
21:     return segundos hasta el día siguiente a la misma hora
22:   end if
23: end function
24: procedure ONINIT
25:   Programar evento de temporizador con SEGUNDOS_HASTA_INICIO_OPERACIONES
26:   Reiniciar el estado de reentrada en rango
27: end procedure
28: procedure ONTICK
29:   Ejecutar la lógica de operación en base a los datos recogidos
30:   Llamar a OPERAR de la clase ReentradaRango
31:   Parámetros: horario, uso de cierre automático, niveles de liquidez y lotaje
32: end procedure
33: procedure ONTIMER
34:   if timer_set es Falso then
35:     Eliminar temporizador actual
36:     Configurar temporizador para ejecutarse cada 24 horas
37:     timer_set ← Verdadero
38:   end if
39:   Obtener la hora actual en GMT
40:   if día actual es sábado o domingo then
41:     Retornar (No se ejecuta la estrategia en fines de semana)
42:   end if
43:   Reiniciar la reentrada en rango
44:   Obtener los niveles máximos y mínimos de liquidez en el período seleccionado
45:   Dibujar la zona de liquidez en el gráfico
46:   Dibujar las líneas que indican el período de operación
47: end procedure

```

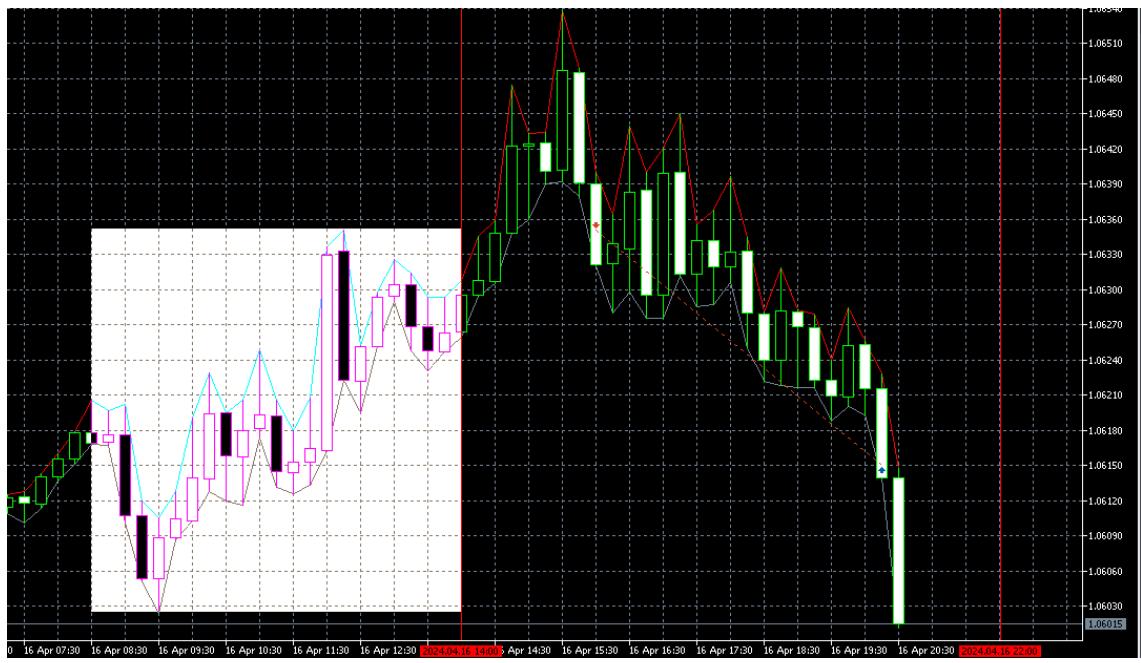


Figura 52: Resultado del caso previsto en la simulación

En la figura 52 en este caso en concreto la estrategia ha sido efectiva en dicha operación, esta estrategia está reconocida por traders profesionales con alta competencia en el mercado, incluso también se muestra información de grandes fondos de inversión donde mueven grandes cantidades de dinero que tienen en cuenta esta estrategia, ya que ellos mismo pueden manipular el mercado con estas grandes cantidades de dinero.

8 Testing

El testing (o pruebas de software) es un apartado relevante a tener en cuenta en esta aplicación. Este tipo de pruebas tienen como objetivo comprobar que todas las funciones desarrolladas funcionen correctamente, cumpliendo así los requisitos previstos y asegurándose de que los errores o fallos se encuentran antes del deploy final. Para este proyecto se han tenido en cuenta distintos tipos de pruebas, que se describen a continuación, aplicando un enfoque de ingeniería de software.

Principalmente, se han desarrollado pruebas de unidad para comprobar ciertas funciones concretas, como aquellas que se ocupan de disponer de los parámetros que ha introducido el usuario en el probador de estrategias y validarlas. Se han realizado pruebas para comprobar que la lógica que sirve para comprobar el funcionamiento de estas unidades de código lo hace correctamente.

A continuación, se procedió a ejecutar un conjunto de pruebas de integración, que permitieron asegurar que los diferentes módulos del sistema se conectaban entre sí de manera correcta. En particular, se examinó la integración entre el frontend desarrollado en Next.js, las API routes que hacían de backend y la interfaz con la base de datos a través de Prisma. También se verificó que la integración se daba correctamente con un servidor local, que ejecuta las operaciones de trading en MetaTrader 5 a través de scripts de Python.

Así mismo, se llevaron a cabo las pruebas de sistema, también conocidas como pruebas “end-to-end”, consistentes en la simulación de flujos completos de un usuario dentro de la aplicación. Por ejemplo, se examinó que el usuario pudiera registrarse, seleccionar una estrategia, configurar los parámetros, ejecutar el backtesting y visualizar los resultados. Este tipo de pruebas permitieron validar el sistema de forma global.

Finalmente, el reconocimiento de que existían diversas funcionalidades en la interfaz de usuario que incluía elementos visuales también llevó a realizar una serie de pruebas manuales, que incluía comprobar la correcta disposición que tenían los diferentes componentes, como pueden ser tablas, botones, la correcta carga de formularios y validación de errores, los mensajes a usuarios y la coherencia de las distintas secciones de la interfaz de usuario.

Por otro lado, el enfoque que se tomó para la testeo de la aplicación no fue la de utilizar herramientas automatizadas de testing, sino el de realizar pruebas manuales de forma iterativa, lo que permitió entender el desarrollo de la aplicación junto con la validación que las funcionalidades principales respondan correctamente a los flujos esperados, con este proceso se ha mejorado algunas funcionalidades y aspectos de la aplicación para realizar el correcto deploy.

Destacar que se detectó múltiples problemas que se fueron resolviendo de forma iterativa. Por ejemplo, se identificaron errores en la validación de los parámetros de formulario del probador de estrategias, donde ciertos valores erróneos pasaban desapercibidos y generaban errores en el servidor. Igualmente, se solucionaron problemas de sincronización del frontend con las respuestas del backend, sobre todo en el tratamiento de errores de la API de MetaTrader 5. Estas incidencias favorecieron la mejora de los flujos de usuario, la optimización de los usos y la robustez de la propia aplicación antes de realizar el deploy.

9 Manual de uso

A continuación se detalla paso a paso como poder ejecutar el proyecto de forma local, sin tener en cuenta la página web desplegada, importante todo el proyecto se ha realizado en el sistema operativo windows y hay una librerias de python que solo se pueden ejecutar en windows, por lo tanto, se recomienda la ejecución en este sistema operativo para evitar futuros problemas.

9.1. Ejecución frontend

Para ejecutar el frontend de la aplicación de manera local (sin que sea necesario el despliegue en Vercel), es necesario seguir unos pasos para configurar correctamente el entorno de ejecución del frontend.

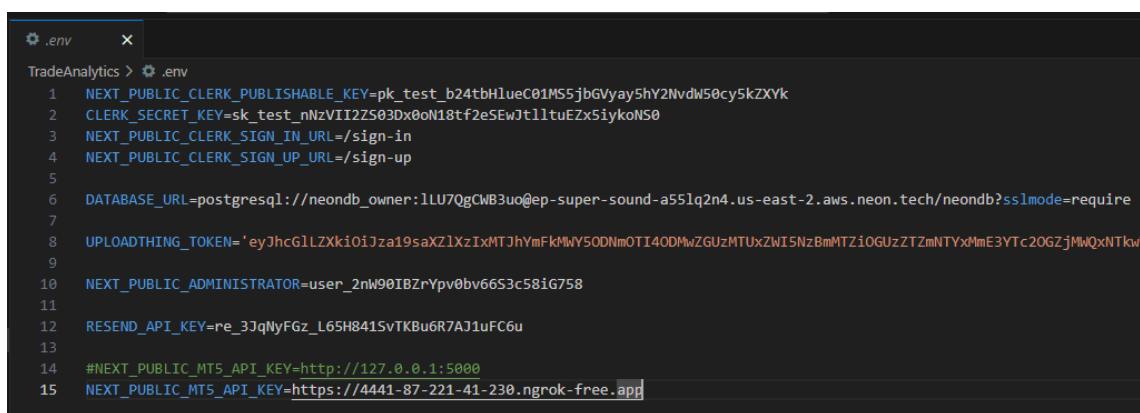
Para ello, hay que descargar el proyecto desde la página donde se suele subir todo el contenido del proyecto. Una vez descargado, hay que moverse a la ruta raíz del proyecto a partir de la consola del gestor de terminal.

Antes de la primera ejecución de la aplicación habrá que tener instaladas en el sistema ciertas herramientas, como son Node.js [20] (versión 18 o superior) y npm [21] (el gestor de paquetes que se incluye con Node). Ambas permiten instalar y ejecutar el frontend sin problemas.

Dentro del proyecto será necesario instalar todas las dependencias. Estas incluyen librerías como Next.js (el framework del frontend), Tailwind CSS (para los estilos), TypeScript (para el tipado estático del código) o Shadcn/UI (una colección moderna de componentes para React) y servicios externos como Clerk y UploadThing para la autenticación y carga de archivos, respectivamente, para ello se usa el siguiente comando para instalar todas la dependencias:

```
npm install
```

La siguiente etapa a realizar, una vez ya cargadas en Node.js y gestionadas las dependencias usando npm, es la de crear el archivo .env, el cual debería encontrarse en la raíz del proyecto, si no se encuentra en la raiz del proyecto se crea manualmente. En este archivo se almacena información que el proyecto requiere para poder ejecutarse y que no debería estar disponible públicamente. Este archivo requiere las especificaciones de las variables de entorno que podemos encontrar las claves de los servicios.



```
TradeAnalytics > .env
1 NEXT_PUBLIC_CLERK_PUBLISHABLE_KEY=pk_test_b24tbHlueC01MS5jbGVyay5hY2NvdW50cy5kZXyK
2 CLERK_SECRET_KEY=sk_test_nNzVII2ZS03Dx0oN18tf2e5EwJt1ltuEZx5iykoNS8
3 NEXT_PUBLIC_CLERK_SIGN_IN_URL=/sign-in
4 NEXT_PUBLIC_CLERK_SIGN_UP_URL=/sign-up
5
6 DATABASE_URL=postgresql://neondb_owner:1LU70gCWB3uo@ep-super-sound-a551q2n4.us-east-2.aws.neon.tech/neondb?sslmode=require
7
8 UPLOADTHING_TOKEN='eyJhcGlZXXkiOiJza19saXZlXzIxMTJhYmFkMjY5ODNmOTI4ODMwZGUzMTUxZWI5NzBmMTZiOGUzZTzmNTYxMmE3YTC20GZjMWQxNTkw'
9
10 NEXT_PUBLIC_ADMINISTRATOR=user_2nW90IBZrYpv0bv66S3c58iG758
11
12 RESEND_API_KEY=re_3JqNyFGz_L65H8415vTKBu6R7AJ1uFC6u
13
14 #NEXT_PUBLIC_MT5_API_KEY=http://127.0.0.1:5000
15 NEXT_PUBLIC_MT5_API_KEY=https://4441-87-221-41-230.ngrok-free.app
```

Figura 53: Fichero enviroment frontend

En la figura 53 se muestra un resumen de todas las variables utilizadas en el archivo `.env`, necesarias para el correcto funcionamiento del proyecto en entorno local.

- Crear una cuenta en Clerk [4] donde hay que crear un proyecto, en este proceso se generan dos claves, una pública y otra privada, las cuales deben ser introducidas en el archivo `.env`, los siguientes pasos son intuitivos y te muestran las claves sign-in y sign-up necesarios para el correcto funcionamiento.
- La dirección URL de la base de datos en PostgreSQL, también es un servicio externo donde se tendrá que registrar y obtener una clave, es un porceso sencillo y rápido [22].
- La API de UploadThing lo mismo, registrarse y obtener una clave [31].
- Para en envio de mensajes por email se ha optado por el servicio resend [26], especifico en react y nextjs, se tiene que registrar en la página web oficial y obtener la clave, solo es usado en la página principal en la sección de contactos.
- Así como la dirección del servidor para conectar la página web con la plataforma local, en este caso uso ngrok, un servicio capaz de desplegar un servidor de forma temporal para la correcta comunicación entre la aplicación web y los programas de python.

Una vez se añaden todas estas configuraciones, ya se puede ejecutar la aplicación web de forma local con un sólo comando que inicia el servidor de desarrollo de `http://localhost:3000`. Desde allí se puede acceder a toda la funcionalidad que ofrece: login, ver las estrategias, cargar los parámetros, hacer backtesting, ver los resultados y gestionar las configuraciones personales, este es el comando para ejecutar el proyecto:

```
npm run dev
```

9.2. Ficheros python que comunican la página web con el entorno local

Para permitir la interconexión entre la aplicación web y la plataforma MetaTrader 5 que se encuentra instalada en el sistema local, se ha desarrollado un pequeño servidor en Python, usando la librería Flask, de esta forma obtenemos una api local.

Entorno virtual y dependencias

Se ha creado un entorno virtual específico con Conda [1] para poder disponer de las diferentes dependencias en el proyecto, conda es opcional para no tener que instalar versiones diferentes de python, ya que las librerías de MT5 solo están disponibles con la versión de python 3.11. Para ello, se describen a continuación los pasos que se realizaron para la creación y ejecución del entorno:

```
conda create -name mt5python python=3.11 conda activate mt5python pip install pandas pip install MetaTrader5 pip install requests pip install pyarrow pip install psutil npm install -g ngrok
```

Ejecución

Por último estando en el entorno virtual en mi caso `mt5python`, se ejecuta el servidor usando el ejecutable de python.

```
python info_symbols.py
```

Tenemos el servidor activo en local, hay que hacer un despliegue remoto del servidor, para agilizar y ejecutar de forma rápida y eficaz, se ha usado ngrok, este nos permite levantar el servidor de forma remota el tiempo que tengamos nuestro ordenador local activo. En primer lugar hay que obtener un token para obtener acceso a este servicio, para ello, se tiene que registrar y de esta manera se obtiene esta clave [18].

```
ngrok config add-authtoken TU_TOKEN_DE_AUTENTICACION ngrok http 5000
```

Por ultimo, en nuestro frontend nos dirigimos al archivo .env y cambiamos la clave, de esta forma ya tendremos la comunicación completa entre el frontend y la api local, por ejemplo en mi caso el siguiente comando:

```
NEXT_PUBLIC_MT5_API_KEY=https://4441-87-221-41-230.ngrok-free.app
```

9.3. MetaTrader 5

Para empezar hay que tener instalada la plataforma MetaTrader 5 (MT5), que se puede descargar de su página oficial [15]. Su instalación es fácil: bastará con ejecutar el instalador y hacer clic en "siguiente" dura unos segundos hasta que finalice el proceso. Con MT5 ya instalado, se puede hacer uso de la plataforma, abriéndola desde un acceso directo en el escritorio o desde el menú de inicio.

A continuación se necesitará crear una cuenta de práctica gratuita en una empresa de fondeo. En este caso, se usará FTMO, que es una de las empresas más populares del sector del trading y que ofrecen la posibilidad de abrir cuentas demo con capital ficticio. Para ello se tendrá que acceder a su página oficial [5] y registrarse utilizando un correo electrónico. Durante el registro, se podrá acceder al área de usuario, donde se podrá solicitar una cuenta demo gratuita. Durante la creación de la cuenta, el usuario podrá decidir el capital de la simulación (por ejemplo: 10.000, 25.000, etc.), la moneda (euros, USD, etc.), y la plataforma que se desea utilizar, eligiendo a MetaTrader 5 (MT5) como la plataforma que hemos instalado anteriormente.

Una vez que se haya creado la cuenta, FTMO enviará un correo con todas las credenciales requeridas: número de cuenta, contraseña y el servidor. Hay que introducir esos datos en MT5. Lo primero que hay que hacer es abrir MetaTrader 5 e ir al menú "Archivo" → "Conectarse a la cuenta comercial" y allí introducir las credenciales. Una vez que hacemos clic en "Aceptar", la cuenta quedará conectada, con esto ya tendremos acceso al histórico de precios de los simbolos.

El siguiente paso consiste en compilar los algoritmos de trading que se encontrar con todos los archivos disponibles para ejecutar el proyecto, con extensión .mq5. Abrimos MT5 y desde el botón que aparece en la barra superior vamos a "MetaEditor". En el MetaEditor hay que pegar los archivos .mq5 en la carpeta Experts.

Dentro del editor, se abre cada uno de los archivos de extensión .mq5 y se hace clic en el botón "compilar". De este modo, se generarán los .ex5 que son los ejecutables necesarios para el probador de estrategias.

Una vez compilados los algoritmos, la configuración queda terminada. El entorno local quedará preparado para recibir las solicitudes desde la web, ejecutar backtestings de las estrategias configuradas y devolver los resultados obtenidos para poder ser analizados visualmente mediante la

interfaz del frontend.

9.4. Uso final de la aplicación web

Para explicar en detalle todas las funcionalidades disponibles de esta aplicación web, se detallara todas las secciones disponibles.

9.4.1. Página principal

En esta página principal se presenta de una forma clara y accesible toda la información general del proyecto y además se complementa con algunos elementos visuales como los botones destacados que permiten realizar las primeras acciones clave: registrarse o bien ir directamente al probador de estrategias (dentro del dashboard). Esta estructuración permite navegar y acceder inmediatamente a las funciones más relevantes del sistema.



Figura 54: Visualización página principal inicio

Característica principales

Las mejores herramientas, garantizando calidad y adaptabilidad para el desarrollo del proyecto.



Figura 55: Visualización página principal características

Estrategias principales

No dejes pasar la oportunidad de probar las estrategias, elige la que más se ajuste a tus objetivos.

Scalping Day Trading Swing Trading Position Trading Momentum Trading Algorithmic Trading



Figura 56: Visualización página principal estrategias

En la figuras 54,55 y 56 se muestran secciones de esta página principal. Una de las funcionalidades principales es el formulario de contacto, que está completamente funcional y permite a cualquier visitante enviar un mensaje directamente al equipo encargado del proyecto. Este formulario se basa en un sistema de envío de emails que permite mantener la comunicación sin tener que abrir un cliente de email externo.

Contacta conmigo

Envíame un mensaje, te responderé por vía email

Ana

Sanchez

juanjosebl2@gmail.com

666555444

Esto es un mensaje de ejemplo para ver si su funcionalidad

Enviar

Figura 57: Visualización formulario con un ejemplo funcional

Gracias por contactar conmigo ➔ Recibidos x



Juan Barrera <trade-analytics-info@resend.dev>
para mí ▾

Gracias por contactar conmigo estimado Ana, en breve me pondré en contacto contigo.

Responder

Reenviar



Figura 58: Visualización formulario con un ejemplo funcional email

En la figura 57 y 58 se muestra un ejemplo de su funcionalidad. En la parte superior de la página se encuentra un header o barras de navegación que permite al usuario desplazarse por distintas secciones las cuales tienen un papel importante en la aplicación. Estas secciones son:

- **Símbolos:** donde se proporciona información acerca de los símbolos de trading que forman parte de la aplicación (por ejemplo, EUR/USD, BTC/USD, etc.). También se explica que los datos históricos de precios han sido obtenidos de una empresa de fondeo real, en este caso FTMO.

Figura 59: Apartado símbolos, información de la empresa de fondeo

Lista de Símbolos

Nombre	Bid	Ask	Spread
EURUSD	1.12474	1.12488	14
GBPUSD	1.32962	1.32992	30
USDCHF	0.83129	0.83177	48
USDJPY	145.362	145.392	30
USDCAD	1.39378	1.39402	24
AUDUSD	0.64084	0.64101	17
AUDNZD	1.08472	1.08538	66
AUDCAD	0.89315	0.89356	41
AUDCHF	0.5327	0.53315	45
AUDJPY	93.15	93.195	45

Figura 60: Apartado símbolos, información de los símbolos disponibles

En las figuras 59 y 60 se muestra un ejemplo de funcionalidad de formulario, se ha puesto como ejemplo mi propio correo electrónico.

- **Probador de estrategias:** módulo modular que realiza simulaciones de trading mediante estrategias personalizables; cabe señalar que esta funcionalidad está asegurada por autenticación, por lo que necesita que el usuario esté registrado e inicie una sesión para acceder.

En las figuras x muestro la información de la empresa de fondeo y en la figura x muestro la información de símbolos disponibles, con sus precios de compra y venta actuales.

- **Inicio de sesión:** el apartado que facilita a los usuarios registrados poder acceder a su sesión mediante su correo electrónico o mediante una cuenta de terceros (Google o GitHub).

La página principal cuenta con un diseño para favorecer la claridad y la usabilidad de la página, lo que permite a los usuarios comprender rápidamente la propuesta de valor en el sistema y permite el acceso a cada uno de los segmentos del flujo funcional, por ultimo mencionar que también se presenta al final de la página principal un footer con los datos de contacto, todas estas secciones queda por recalcar que se han respetado según el diseño principal del mockup.

9.4.2. Inicio sesión y registro

La aplicación presenta un sistema de registro y login moderno, seguro, limpio y cómodo para el usuario gestionado a través del proveedor de autenticación Clerk.

En lo que respecta al proceso de logueo, el usuario tiene tres opciones disponibles, siendo éstas:

- Loguearse con su email y contraseña.
- Utilizar su cuenta de Google.
- Utilizar su cuenta de GitHub.

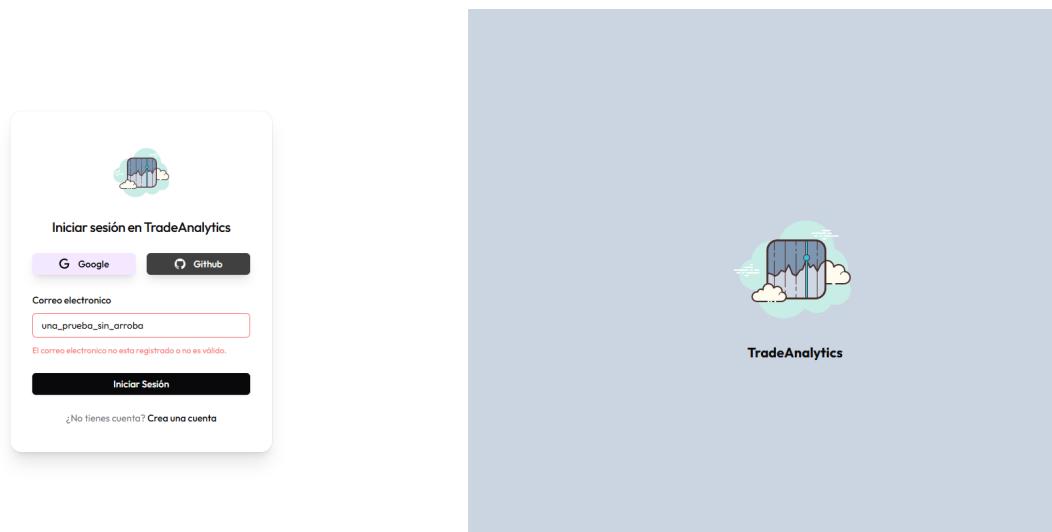


Figura 61: Inicio sesión

En la figura 61 se muestra el diseño de el inicio de sesión, con un ejemplo en el caso al poner un valor incorrecto. Este sistema de autenticación permite al usuario loguearse de una forma rápida como si tuviese una cuenta en la plataforma, y en este sentido, se ahorra la necesidad de crear una nueva cuenta si el usuario ya tuviese de estas plataformas.

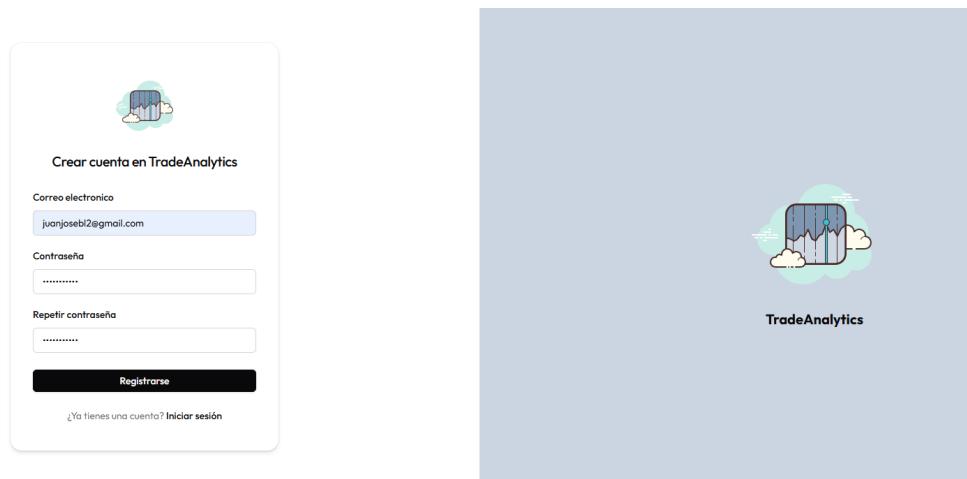


Figura 62: Registro

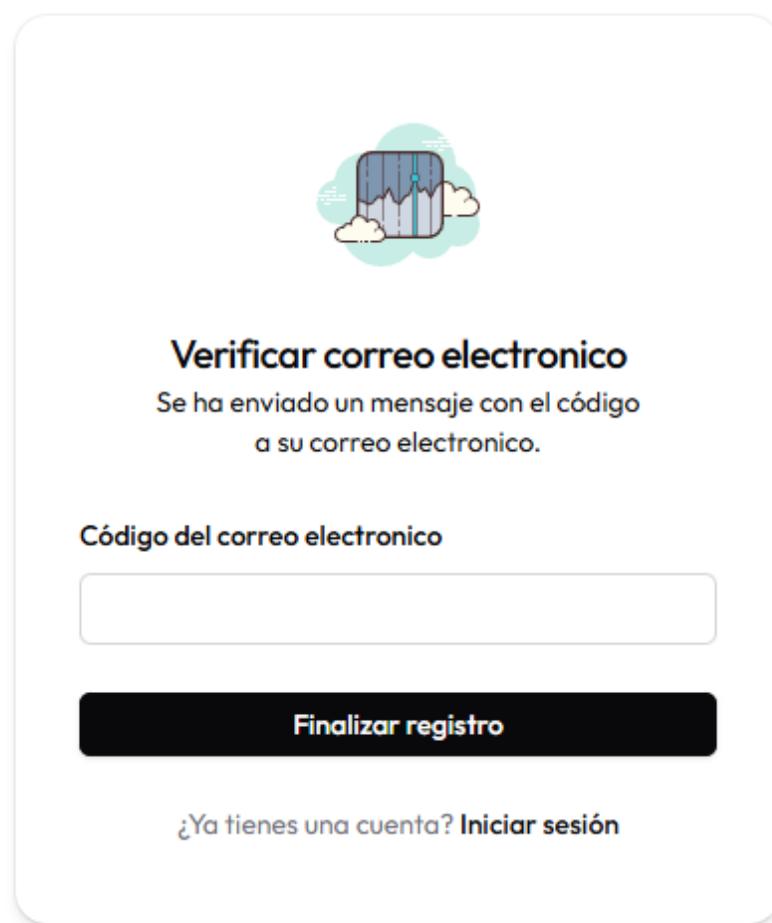


Figura 63: Registro verificación

En el caso de registro he puesto la figura 62 que es la interfaz del registro y la figura 63 que es la interfaz de la verificación de registro con vía email, el usuario debe introducir un email válido y una contraseña. En este punto, para garantizar la seguridad y verificar la identidad del usuario,

se envía a su dirección de email una confirmación, por lo que el usuario sólo quedará totalmente logueado una vez se haya confirmado el correo electrónico que ha introducido.

Una vez logueado el usuario, automáticamente este será redirigido a su panel de usuario (dashboard) donde podrá empezar a utilizar las funcionalidades más relevantes, como el probador de estrategias o la gestión de su historial de simulaciones.

9.4.3. Dashboard o panel

A continuación se redacta todas las secciones disponibles en la zona de dashboard o panel, tanto si el usuario es administrador o si no es administrador.

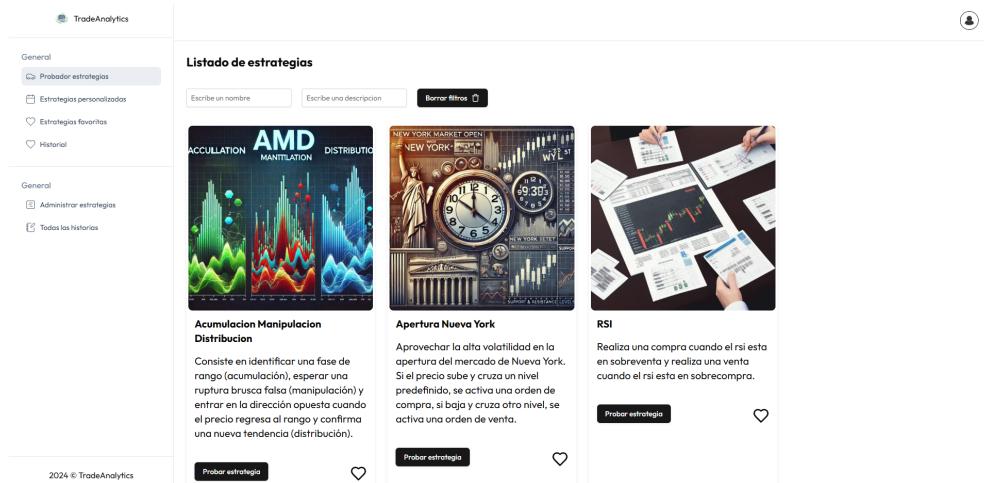


Figura 64: Gestión de estrategias

En la figura 64 se muestra a primera vista el dashboard, en la parte derecha superior tiene un botón donde puede cerrar sesión, en la parte izquierda se muestra un menú desplegable donde se puede acceder a todas las funcionalidades del sistema.

Zona administrador: Administrar estrategias

En el panel de control para el administrador donde solo accede los usuarios administradores. Las funcionalidades fundamentales para manejar las estrategias de trading algorítmico que puede encontrar la plataforma están centralizadas en esta zona del panel de control. El panel de control está construido mediante una interfaz visual intuitiva y accesible que permite llevar un control completo acerca de la gestión de cada estrategia.

Figura 65: Gestión de estrategias

En la figura 65 se muestra la sección donde se gestionan las estrategias predefinidas. Las estrategias son representadas en una tarjeta que contiene la imagen representativa, su título, una breve descripción y su estado de publicación (publicado o no publicado). Desde esta zona del panel de control, el administrador puede:

- Crear nuevas estrategias a través de un formulario o modal en el cual se pueden introducir el nombre, una breve descripción y la imagen representativa de la estrategia.

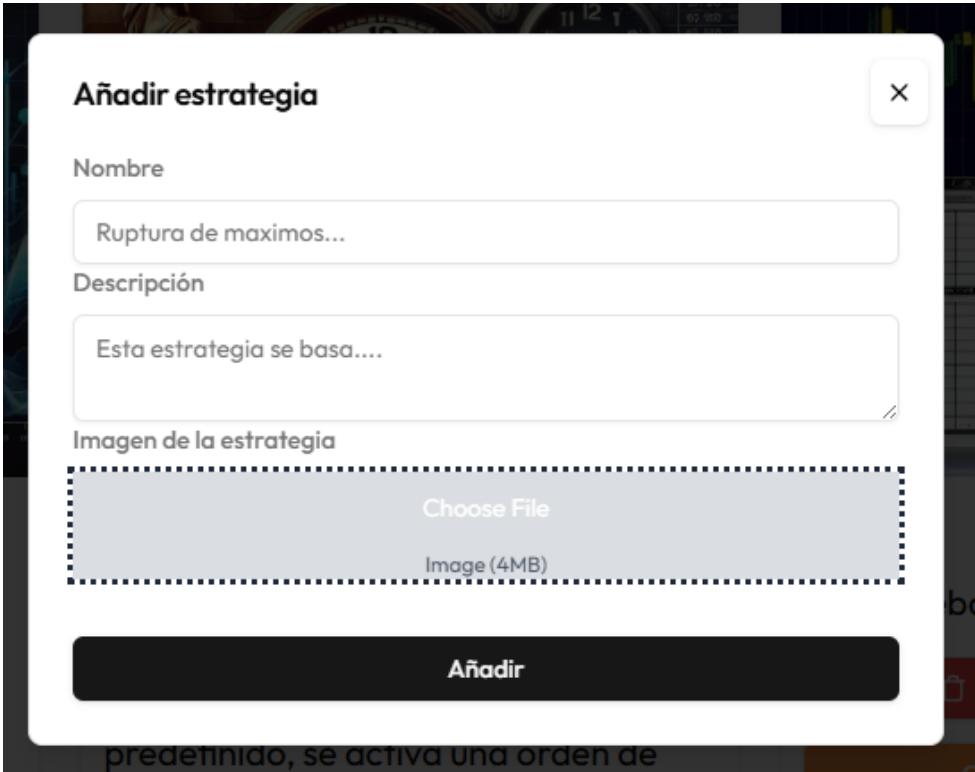


Figura 66: Gestión de estrategias agregar

En la figura 66 se muestra la interfaz para crear una estrategia, para acceder a esta funcionalidad hay que pulsar el botón añadir estrategia.

- Actualizar estrategias existentes a través de un formulario.



Figura 67: Gestión de estrategias modificar

En la figura 67 se muestra la interfaz para modificar una estrategia, similar a crear una estrategia.

- Eliminar estrategia que ya no se requieran en la plataforma, pulsando el botón de eliminar.
- Publicar o despublicar estrategias para poder tener control sobre su visibilidad, para agilizar el trabajo y en un futuro ya solo se tenga que publicar la estrategia visible a los usuarios.

Por otro lado, cada estrategia cuenta con el botón “Gestionar Parámetros” que lleva al gestor a una pantalla específica de parámetros, que permite gestionar los parámetros de esa estrategia.

Gestionar parámetros						Añadir Parámetro +
Nombre	Valor	Valor filtro mínimo	Valor filtro máximo	Descripción	Acciones	
RSI_SOBRECOMPRA	70	50	80	Valor del rsi en sobrecompra donde se realiza la venta. Entre 50 y 80	Editor	Eliminar
RSI_SOBREVENDIDO	30	20	50	Valor del rsi cuando está en sobreventa, entre 20 y 50	Editor	Eliminar
RSI_PERIODO	14	2	50	Período de velas que escoge el rsi para realizar el análisis de sobrecompra y sobreventa	Editor	Eliminar
RATIO	2	0	20	Ratio de beneficio según el stop loss	Editor	Eliminar
LOTAJE_STATIC	0.1			Lotaje de la operación	Editor	Eliminar
SL_POINTS	1000	0	20000	Stop loss es el rango de pips que estás dispuesto a perder en la operación	Editor	Eliminar

Listado de parámetros

Volver

Figura 68: Gestión de parámetros

En la figura 68 se muestra la interfaz de estas funcionalidades. En esta pantalla gestión de parámetros, el gestor puede:

- Añadir nuevos parámetros, definiendo su nombre, el valor inicial, la descripción que lo acompaña y, de forma opcional, los valores límite mínimo y máximo que pueden proporcionar un filtro.

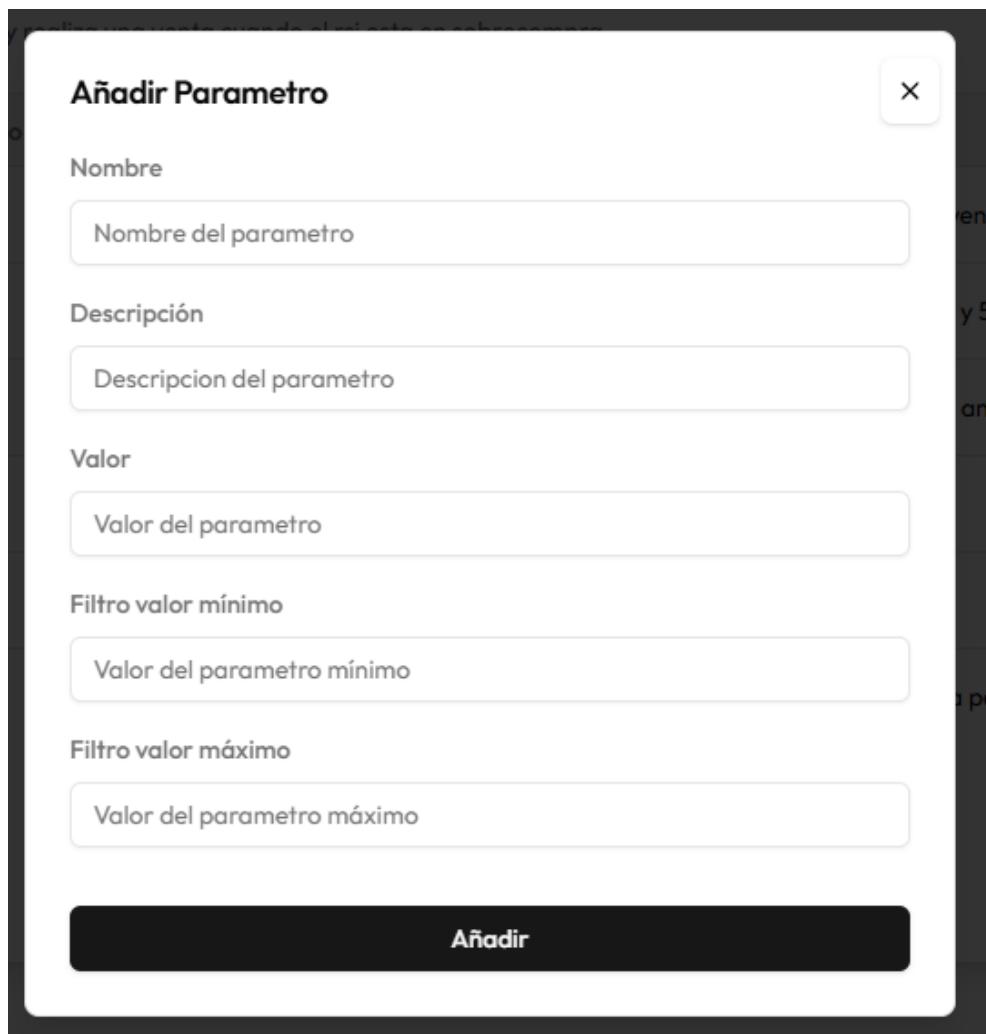


Figura 69: Gestión de parametros agregar

En la figura 69 se muestra la interfaz para crear un parametro, para acceder a esta funcionalidad hay que pulsar el botón añadir parametro.

- Editar parámetros existentes hasta adaptarlos a nuevos requerimientos o configuraciones.

Formulario editar

Nombre
RSI_SOBRECOMPRA

Valor
Valor del rsi en sobrecompra donde se realiza la venta. Entre 50 y 80

Valor
70

Filtro valor mínimo
50

Filtro valor máximo
80

Editar

Figura 70: Gestión de parametros modificar

En la figura 70 se muestra la interfaz para modificar un parametro, similar a añadir un parametro.

- Eliminar algún parámetro si ya no resulta necesario, pulsando el botón de eliminar.

Todas estas funcionalidades permiten configurar cada estrategia con un alto grado de personalización, más aún si se tiene en cuenta el tipo de usuario y las diferentes condiciones del propio mercado. La organización visual que existe en los formularios permite una gestión rápida y eficiente de todas aquellas estrategias ya implementadas.

Zona administrador: Todas las historias

La opción “Todas las Historia” dentro del panel de administración del Dashboard nos permite visualizar la totalidad de los backtesting que han realizado los usuarios dados de alta en la aplicación, ofreciendo una vista ordenada de todos ellos. Se nos presenta en forma de una tabla detallada que recoge la configuración utilizada para cada tipo de backtesting, permitiendo que los administradores tengan un gran control del uso del sistema y de las estrategias evaluadas.

Estrategia	Configuración Básica							Configuración de la estrategia				Realizado	Acciones
	Divisa	Depósito	Apalancamiento	Periodo	Fecha inicio	Fecha fin	SL_POINTS	RATIO_TP	LOTAJE_STATIC	PUNTOS_PRECIO	HORAS_EXPIRACION		
Apertura Nueva York	USD	100000	33	M15	1 may 2025	13 may 2025	2000	2	0.1	2000	2	15 may 2025, 9:11	<button>Eliminar</button>
	USD	100000	33	M15	1 dic 2024	16 dic 2024	200	2	0.1	200	2	11 may 2025, 17:49	<button>Eliminar</button>
	USD	100000	33	M15	1 dic 2024	16 dic 2024	200	2	0.1	200	2	12 feb 2025, 11:38	<button>Eliminar</button>
	USD	100000	33	M15	1 may 2024	1 dic 2024	2000	2	0.1	2000	2	12 feb 2025, 11:37	<button>Eliminar</button>
	USD	100000	33	M15	1 dic 2024	16 dic 2024	200	2	0.1	200	2	12 feb 2025, 11:25	<button>Eliminar</button>
	USD	100000	33	M15	1 dic 2024	16 dic 2024	200	2	0.1	200	2	21 ene 2025, 17:49	<button>Eliminar</button>
	USD	100000	33	M15	4 nov 2024	16 dic 2024	200	2	0.1	200	2	24 dic 2024, 12:18	<button>Eliminar</button>
	USD	100000	33	M15	4 nov 2024	16 dic 2024	200	2	0.1	200	2	24 dic 2024, 12:17	<button>Eliminar</button>
	USD	100000	33	M15	1 dic 2024	16 dic 2024	200	2	0.1	200	2	24 dic 2024, 11:49	<button>Eliminar</button>
	USD	100000	33	M15	1 sept 2024	16 dic 2024	200	2	0.1	200	2	23 dic 2024, 12:10	<button>Eliminar</button>
	USD	100000	33	M15	1 sept 2024	16 dic 2024	200	2	0.1	200	2	23 dic 2024, 12:09	<button>Eliminar</button>
	USD	100000	33	M15	1 dic 2024	16 dic 2024	200	2	0.1	200	2	23 dic 2024, 12:08	<button>Eliminar</button>
	USD	100000	33	M15	1 dic 2024	16 dic 2024	200	2	0.1	200	2	23 dic 2024, 12:07	<button>Eliminar</button>
	USD	100000	33	M15	1 dic 2024	16 dic 2024	200	2	0.1	200	2	22 dic 2024, 20:57	<button>Eliminar</button>

Figura 71: Todo el historial de simulación realizado por todos los usuarios

En la figura 71 se muestra la interfaz representada en una tabla con los datos respectivos de cada estrategia que ha simulado cada usuario. En cada fila de la tabla se especifica el nombre de la estrategia utilizada, la divisa base, el capital depositado, el apalancamiento, el periodo de tiempo de los gráficos (por ejemplo M15) y el rango de fechas en el que se ha desarrollado el backtest.

En la misma tabla se vienen a mostrar los parámetros concretos que el usuario ha introducido para esa prueba, en este caso la estrategia de apertura de new york. Se incluye también otra columna que viene a mostrar la fecha y la hora en la que se ha llevado a cabo dicha simulación y un botón para borrar el historial en concreto de una simulación.

Esta vista resulta eficaz para obtener patrones de uso y de la validación del hecho que las estrategias se están probarando realmente, ayudando con ello tanto al análisis como a la mejora del sistema.

Historial

La sección Historial resulta ser una sección que permite tener una visualización completa de todas las pruebas de backtesting que se han realizado en la aplicación. Cada fila que se muestra en esta sección corresponde a una prueba con todos los detalles, pues no sólo muestra la configuración básica, sino también los parámetros específicos de la estrategia.

En este caso los datos mostrados son: símbolo del activo (EURUSD o BTCUSD), divisa de base, el depósito inicial, el apalancamiento, el periodo temporal (siendo M15) y el rango de fechas donde se realiza la simulación. En esta tabla también se pueden visualizar todos los datos que afectan el comportamiento de las estrategias: SL_POINTS, RATIO_TP, LOTAJE_STATIC, PUNTOS_PRECIO, HORAS_EXPIRACION, etc.

Historial

Estrategia	Configuración Básica								Configuración de la estrategia					Realizado	Acciones
	Símbolo	Divisa	Depósito	Apalancamiento	Período	Fecha inicio	Fecha fin	SL_POINTS	RATIO_TP	LOTAJE_STATIC	PUNTOS_PRECIO	HORAS_EXPIRACION			
Apertura Nueva York	BTCSUD	USD	100000	33	M15	1 may 2025	13 may 2025	2000	2	0.1	2000	2	13 may 2025, 9:11	<button>Guardar parámetros ↕</button>	
	EURUSD	USD	100000	33	M15	1 dic 2024	16 dic 2024	200	2	0.1	200	2	11 may 2025, 17:49	<button>Guardar parámetros ↕</button>	
	EURUSD	USD	100000	33	M15	1 dic 2024	16 dic 2024	200	2	0.1	200	2	12 feb 2025, 11:38	<button>Guardar parámetros ↕</button>	
	EURUSD	USD	100000	33	M15	1 may 2024	1 dic 2024	2000	2	0.1	2000	2	12 feb 2025, 11:37	<button>Guardar parámetros ↕</button>	
	EURUSD	USD	100000	33	M15	1 dic 2024	16 dic 2024	200	2	0.1	200	2	12 feb 2025, 11:25	<button>Guardar parámetros ↕</button>	
	EURUSD	USD	100000	33	M15	1 dic 2024	16 dic 2024	200	2	0.1	200	2	21 ene 2025, 17:49	<button>Guardar parámetros ↕</button>	
	EURUSD	USD	100000	33	M15	4 nov 2024	16 dic 2024	200	2	0.1	200	2	24 dic 2024, 12:18	<button>Guardar parámetros ↕</button>	
	EURUSD	USD	100000	33	M15	4 nov 2024	16 dic 2024	200	2	0.1	200	2	24 dic 2024, 12:17	<button>Guardar parámetros ↕</button>	
	EURUSD	USD	100000	33	M15	1 dic 2024	16 dic 2024	200	2	0.1	200	2	24 dic 2024, 11:49	<button>Guardar parámetros ↕</button>	
	EURUSD	USD	100000	33	M15	1 sept 2024	16 dic 2024	200	2	0.1	200	2	23 dic 2024, 12:10	<button>Guardar parámetros ↕</button>	
	EURUSD	USD	100000	33	M15	1 sept 2024	16 dic 2024	200	2	0.1	200	2	23 dic 2024, 12:09	<button>Guardar parámetros ↕</button>	

Figura 72: Todo el historial de simulación realizado por el usuario registrado

En la figura 72 se muestra la interfaz de esta sección mostrando todos los datos en una tabla. Una característica muy interesante de esta sección es el botón Guardar Parámetros que permite al usuario guardar de forma permanente la configuración exacta que le ha servido para realizar la la simulación actual. Esto sirve para facilitar su reutilización o su análisis futuro. En este sentido, es muy interesante para guardar las configuraciones de las estrategias y poder llevar un control de los experimentos a lo largo del tiempo.

Estrategias personalizadas

La sección Estrategias personalizadas da acceso a una serie de configuraciones de simulaciones que el usuario ha decidido guardar anteriormente. Estas estrategias contienen los datos básicos (símbolo, divisa, fechas, apalancamiento, etc.) y los parámetros asociados a cada tipo de estrategia. Desde allí, el usuario puede volver a lanzar el backtesting dando al botón de "Probar" utilizando una configuración que ya ha demostrado ser útil o interesante.

Estrategia	Configuración Básica								Configuración de la estrategia					Realizado	Acciones
	Símbolo	Divisa	Depósito	Apalancamiento	Período	Fecha inicio	Fecha fin	HORA_INICIO_RECOGIDA	MINUTO_INICIO_RECOGIDA	HORA_FINAL_RECOGIDA	MINUTO_FINAL_RECOGIDA	HORA_FINAL_OPERACION			
Acumulación Manipulación Distribución	EURUSD	USD	100000	33	M15	1 dic 2024	16 dic 2024	8	30	14	0	22			
	EURUSD	USD	100000	33	M15	1 dic 2024	16 dic 2024	8	30	14	0	22			
Estrategia: Acumulación Manipulación Distribución															
Estrategia	Configuración Básica								Configuración de la estrategia					Realizado	Acciones
	Símbolo	Divisa	Depósito	Apalancamiento	Período	Fecha inicio	Fecha fin	RSI_SOBRECOMPRA	RSI_SOBREVENDIDO	RSI_PERIODO	RATIO	LOTAJE_STATIC	SL_POINTS		
RSI	EURUSD	USD	100000	33	M15	1 oct 2024	3 nov 2024	70	30	14	2	0.1	1000	23 nov 2024, 18:36	<button>Probar 🚀</button>

Figura 73: Historial de simulación guardados por el usuario registrado

En la figura 73 se muestra el contenido de la sección similar a las secciones anteriores.

Estrategias favoritas

La sección Estrategias favoritas ha sido creada para que el usuario acceda rápidamente a todas aquellas estrategias que marcó como "me gusta". Cuando el usuario hiciera clic en el botón de favorito (ícono del corazón) en el listado de estrategias del dashboard, esa estrategia queda guardada como favorita para posteriormente mostrarse en esta sección. De esta forma, el usuario puede gestionar su lista de estrategias preferidas, ya que éstas son accesibles o revisables para modificarlas, sin tener que buscar entre el conjunto general, lo que optimiza la experiencia del usuario con la funcionalidad de la aplicación, ayudando a la personalización y al rápido acceso.

Probador de estrategias

La sección Probador de estrategias hace posible que el usuario pueda probar y elegir las diferentes estrategias del trading algorítmico disponibles en la aplicación, siendo cada estrategia mostrada en forma de tarjeta con su correspondiente imagen, título y una corta explicación detallada, además de un botón para probar la estrategia que inicia el asistente de configuración y backtesting.

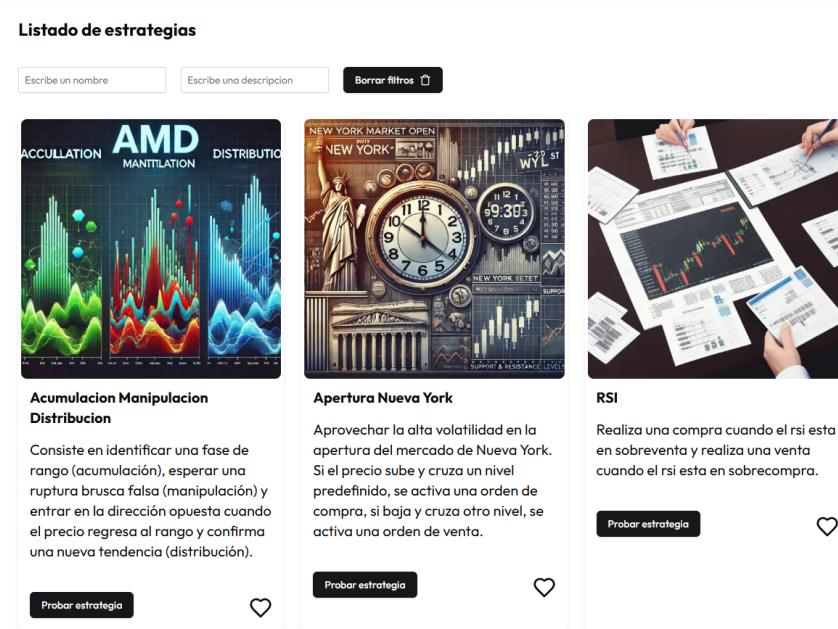


Figura 74: Probador de estrategias

En la figura 74 se muestra la interfaz donde contiene la tarjetas o cards y en la parte superior los filtros. En la parte superior de esta sección hay unos campos de búsqueda que permiten filtrar las estrategias por nombre o por descripción, facilitando la búsqueda de estrategias concretas por parte del usuario. También existe el botón de "Borrar filtros" que vuelve a mostrar todas las estrategias.

Cada tarjeta incluye un ícono en forma de corazón, que permite marcar la estrategia como favorita. Esta funcionalidad enlaza directamente con la sección de estrategias favoritas, donde se guardan las que se han marcado como favoritas para un acceso rápido y más personalizado.

Probador de estrategias: Rellenar formulario de una estrategia seleccionada

Una vez que el usuario ha escogido una estrategia desde el probador, tiene a su disposición un formulario donde puede introducir la información necesaria para poder llevar a cabo el backtesting. Esta vista se divide en dos bloques principales: la configuración de parámetros base que requieren todas las estrategias y los parámetros específicos de la estrategia seleccionada.

Probar Estrategia: Acumulacion Manipulacion Distribucion

Configuración base

Símbolos	EURUSD	Representan los activos financieros que puedes negociar, como pares de divisas (EUR/USD), acciones (sp500=US500.cash), materias primas (oro=XAU/USD, petróleo=OIL)
Periodo	M15	
Depósito	100000	Divisa
Apalancamiento	1.33	USD

Fechas de la prueba

Fecha inicio de la prueba	2024.12.01	Fecha fin de la prueba	2024.12.16
---------------------------	------------	------------------------	------------

Parametros de la estrategia

HORA_INICIO_RECOGIDA	-4	MINUTO_INICIO_RECOGIDA	30
HORA_INICIO_RECOGIDA debe estar entre 0 y 24			
HORA_FINAL_RECOGIDA	14	MINUTO_FINAL_RECOGIDA	0
HORA_FINAL_OPERACIONES	22	MINUTO_FINAL_OPERACIONES	0

Figura 75: Probador de estrategias: Formulario 1

Parametros de la estrategia

HORA_INICIO_RECOGIDA	8	MINUTO_INICIO_RECOGIDA	30
HORA_FINAL_RECOGIDA	14	MINUTO_FINAL_RECOGIDA	0
HORA_FINAL_OPERACIONES	22	MINUTO_FINAL_OPERACIONES	0
SL_POINTS	200	RATIO_TP	2
LOTAJE_STATIC	1		

Ver resultado

Volver

Figura 76: Probador de estrategias: Formulario 2

En la figura 75 y 76 se muestra la interfaz limpia e intuitiva de esta sección. En cuanto a la primera parte, el usuario va a poder introducir el símbolo (por ejemplo EUR/USD), el capital inicial (por ejemplo 100.000), el apalancamiento, el período temporal (por ejemplo, M15, para velas de 15 minutos), la divisa y las fechas de inicio y fin que se quiera realizar la simulación del back-testing. Dicha información resulta importante ya que servirá para poder definir el escenario de la estrategia que se evaluará.

Finalmente, en el bloque de parámetros de la estrategia, habrá que introducir los valores específicos que condicionan el comportamiento de dicha estrategia por ejemplo en este caso con la estrategia AMD los horarios de recogida de información, niveles de stop-loss, ratio de beneficios, etc. Esta sección está definida con una validación automática, que indicará al usuario ha introducido un valor fuera del rango permitido. Por ejemplo, al introducir una hora negativa se mostrará un aviso de que el valor tiene que estar entre 0 y 24.

Además de esta validación junto a cada campo hay un ícono de información (i) que, al pasar con el cursor por este ícono muestra una descripción explicativa sobre la utilidad de este parámetro. Esta particularidad hace mucho más fácil la comprensión sin necesidad de incrementar el interrup-

tor de salida.

Si ha finalizado el formulario correctamente, el usuario podrá pulsar el botón de "Ver resultado" para iniciar el proceso de backtesting. Al hacer esto los datos son enviados al sistema de la api local, el cual lanza la simulación correspondiente y que finalmente mostrará los resultados obtenidos.

Probador de estrategias: Backtesting

Los resultados de la estrategia seleccionada y configurada con anterioridad se encuentran en la sección de backtesting. En esta pantalla se ve el comportamiento histórico simulado de la estrategia de trading, lo que permite analizar el rendimiento.



Figura 77: Probador de estrategias: Backtesting gráficos

En la figura 77 se muestra la interfaz superior de la sección backtesting. Estamos ante una pantalla que muestra en forma de gráfico de líneas, la evolución del balance de la cuenta durante el tiempo que hemos predefinido en el formulario. Este gráfico permite de un vistazo detectar la ganancia o pérdida, así como la estabilidad o la variabilidad del rendimiento de la estrategia.

Y justo debajo de este gráfico se presentan otros gráficos con el desglose del comportamiento de la estrategia en distintas gráficas, concretamente:

- **Entradas por horas:** es un gráfico que señala en qué franjas horarias se han registrado las transacciones.
- **Entradas por días de la semana:** en este gráfico veremos qué días han registrado más entradas de media en la semana.

- **Entradas por meses:** a partir del comportamiento histórico de la estrategia se obtiene un gráfico que marca los meses con más entradas.
- **Ganancias y pérdidas por horas:** se muestran las horas del día con mejores o peores resultados.
- **Ganancias y pérdidas por días de la semana:** en este gráfico podemos saber qué días han sido los mejores o los peores en términos de rentabilidad.
- **Ganancias y pérdidas por meses:** se refleja el rendimiento de cada uno de los meses y su comportamiento, obteniendo la rentabilidad por mes.

Bajo los gráficos se pueden encontrar tres botones con función:

- **Guardar parámetros:** guarda la configuración utilizada en la prueba de estrategia que después se visualizar en la sección de estrategia personalizada.
- **Información:** abre una nueva página y proporciona un análisis exhaustivo y en profundidad del informe generado por MetaTrader 5, aportando todas las estadísticas.

Explicación de los Datos

Explicación de los Términos

- **Calidad del histórico:** Indica la precisión y completitud de los datos históricos utilizados para análisis. Si es menos del 100% no quiere decir que los datos históricos estén mal, sino que falta una pequeña información en tiempo real. Segun la estrategia esta información puede ser importante.
- **Barra:** Representan períodos de tiempo en los datos históricos, como velas en un gráfico financiero.
- **Ticks:** Cada cambio en el precio del mercado. Un tick es el nivel más bajo de detalle en el movimiento de un precio.
- **Símbolos:** Los activos financieros utilizados en el análisis, como pares de divisas (ej., EUR/USD).
- **Beneficio Neto:** La ganancia o pérdida final después de restar todas las pérdidas del beneficio bruto.
- **Reducción absoluta del balance:** La mayor caída en el saldo desde el punto más alto alcanzado.
- **Reducción absoluta de la equidad:** La mayor caída en el valor de la cuenta (incluyendo posiciones abiertas) desde su punto más alto.
- **Beneficio Bruto:** La suma total de todas las ganancias sin considerar las pérdidas.
- **Reducción máxima del balance:** La mayor pérdida observada en el balance durante el análisis.
- **Reducción máxima de la equidad:** La mayor caída observada en la equidad durante el análisis.
- **Pérdidas Brutas:** La suma total de todas las pérdidas sin considerar las ganancias.
- **Reducción relativa del balance:** La reducción máxima del balance expresada como un porcentaje del balance inicial.
- **Reducción relativa de la equidad:** La reducción máxima de la equidad expresada como un porcentaje del balance inicial.
- **Factor de Beneficio:** La relación entre el beneficio bruto y las pérdidas brutas. Un valor mayor a 1 indica rentabilidad.
- **Beneficio Esperado:** La ganancia o pérdida promedio esperada por operación.
- **Nivel de margen:** Muestra la proporción entre la equidad de la cuenta y el margen requerido para abrir posiciones.
- **Factor de Recuperación:** Mide la relación entre el beneficio neto y la reducción máxima. Un valor positivo indica que las pérdidas han sido recuperadas.
- **Ratio de Sharpe:** Indica la rentabilidad ajustada por el riesgo. Un valor mayor a 1 es considerado bueno.
- **Z-Score:** Mide la aleatoriedad de las operaciones. Un valor bajo puede indicar que el resultado de las operaciones es predecible.
- **AHPR:** Indica el rendimiento promedio de las operaciones en términos percentuales.
- **GHPR:** Muestra el rendimiento compuesto promedio de las operaciones.
- **LR Correlation:** Indica qué tan bien se relacionan los datos de las operaciones.
- **LR Standard Error:** Mide la precisión de la correlación lineal.
- **Total de operaciones ejecutadas:** Número total de operaciones realizadas durante el análisis.

Figura 78: Probador de estrategias: Backtesting información resultado 1

- **Posiciones cortas:** Transacciones realizadas esperando que el precio baje.
- **Posiciones largas:** Transacciones realizadas esperando que el precio suba.
- **Total de transacciones:** La cantidad total de operaciones abiertas y cerradas.
- **Posiciones rentables:** Operaciones que resultaron en ganancias.
- **Posiciones no rentables:** Operaciones que resultaron en pérdidas.
- **Número máximo de pérdidas consecutivas:** La mayor cantidad de operaciones consecutivas con pérdidas.
- **Máximo de pérdidas consecutivas:** La suma total de las pérdidas de las operaciones consecutivas con pérdida más alta.
- **Correlación (Beneficios y MFE):** Relación entre los beneficios y la máxima excursión favorable (MFE).
- **Correlación (Beneficios y MAE):** Relación entre los beneficios y la máxima excursión adversa (MAE).
- **Correlación (MFE y MAE):** Relación entre la máxima excursión favorable y la adversa.
- **Tiempo mínimo para retener la posición:** El tiempo más corto que una posición se mantuvo abierta.
- **Tiempo máximo para retener la posición:** El tiempo más largo que una posición se mantuvo abierta.
- **Tiempo medio para retener la posición:** El promedio de tiempo que las posiciones permanecen abiertas.

Figura 79: Probador de estrategias: Backtesting información resultado 2

En la figura 78 y 79 muestran la información que presenta los siguientes apartados sobre las estadísticas que muestra la simulación en detalle. En la primera sección se puede apreciar toda la información explicada sobre el apartado de resultados que se presentara en la siguiente sección del backtesting. Esta información relevante nos proporciona si dicha simulación de la estrategia se puede aplicar en producción según su criterio.

Explicación de las Tablas

Tabla 1: Órdenes

- **Hora Apertura:** Fecha y hora en que se realizó la orden.
- **Orden:** Número único que identifica cada orden.
- **Símbolo:** Activo financiero asociado a la orden.
- **Tipo:** Compra buy o venta sell.
- **Volumen:** Tamaño de la transacción.
- **Precio:** Precio al que se ejecutó la operación.
- **S/L y T/P:** Niveles de Stop Loss y Take Profit.
- **Estado:** Estado de la orden (ej., ejecutada o pendiente).
- **Comentario:** Información adicional sobre la orden.

Tabla 2: Transacciones

- **Transacción:** Número único que identifica cada transacción.
- **Símbolo:** Activo financiero involucrado.
- **Tipo:** Tipo de operación (compra o venta).
- **Dirección:** Entrada in o salida out.
- **Volumen:** Tamaño de la operación.
- **Precio:** Precio de ejecución de la operación.
- **Orden:** Orden asociada a la transacción.
- **Comisión:** Costo de la operación.
- **Beneficio:** Ganancia o pérdida generada.
- **Balance:** Balance actualizado después de la operación.

Figura 80: Probador de estrategias: Backtesting información tablas

En la figura 80 se visualiza la información que presenta los siguientes apartados sobre las tablas con todas las transacciones y órdenes ejecutadas durante el periodo de la simulación. Esta información beneficia al usuario para saber como se comporta la estrategia con la pre configuración establecida.

- **Volver:** permite volver a la página del probador de estrategias para elegir o establecer una nueva prueba.

Esta sección es fundamental para probar visualmente las estrategias y de forma interactiva para hacerlo en un entorno real o simulado más avanzado.

Resultados

Calidad del historial: 100%	Barras: 960
Ticks: 1169465	Símbolos: 1
Beneficio Neto: 545.00	Reducción absoluta del balance: 63.50
Reducción absoluta de la equidad: 163.50	Beneficio Bruto: 1230.50
Reducción máxima del balance: 210.50 (0.21%)	Reducción máxima de la equidad: 364.00 (0.36%)
Pérdidas Brutas: -685.50	Reducción relativa del balance: 0.21% (210.50)
Reducción relativa de la equidad: 0.36% (364.00)	Factor de Beneficio: 1.80
Beneficio Esperado: 45.42	Nivel de margen: 3130.59%
Factor de Recuperación: 1.50	Ratio de Sharpe: 4.28
Z-Score: 0.42 (32.55%)	AHPR: 1.0005 (0.05%)
LR Correlation: 0.75	Resultado de OnTester: 0
GHPR: 1.0005 (0.05%)	LR Standard Error: 186.47
Total de operaciones ejecutadas: 12	Posiciones cortas (% rentables): 6 (83.33%)
Posiciones largas (% rentables): 6 (33.33%)	Total de transacciones: 24
Posiciones rentables (% del total): 7 (58.33%)	Posiciones no rentables (% del total): 5 (41.67%)
La transacción rentable: 199.50	La transacción no rentable: -202.50
Promedio de transacción rentable: 175.79	Promedio de transacción no rentable: -133.50
El número máximo de ganancias consecutivas (\$): 4 (794.00)	El número máximo de pérdidas consecutivas (\$): 2 (-203.00)
El máximo de beneficio consecutivo (número de ganancias): 794.00 (4)	El máximo de pérdidas consecutivas (número de pérdidas): -203.00 (2)
Promedio de ganancias consecutivas: 2	Promedio de pérdidas consecutivas: 1
Correlation (Profits,MFE): 0.90	Correlation (Profits,MAE): 0.88
Correlation (MFE,MAE): 0.7647	Tiempo mínimo para retener la posición: 0:00:59
Tiempo máximo para retener la posición: 5:26:23	Tiempo medio para retener la posición: 1:47:34

Figura 81: Probador de estrategias: Backtesting sección resultados

En la figura 81 se muestra la interfaz de como se visualiza las estadísticas. En la sección de resultados del backtesting se ofrece un resumen detallado de las estadísticas más significativas obtenidas de MetaTrader 5 después de simular una estrategia de trading. Esta sección permite al usuario analizar la rentabilidad de la estrategia testeada con el propósito de poder decidir sobre su calidad.

Las medidas que se extraen se organizan a su vez en métricas de rendimiento y métricas de riesgo. En las métricas de rendimiento se destacan:

- **Beneficio Neto:** es capaz de mostrar la ganancia total resultante de la estrategia, después de restar las pérdidas, con el objeto de permitir valorar la rentabilidad de la simulación.
- **Factor de beneficio:** representa la relación entre los beneficios brutos y las pérdidas brutas. Un valor superior a uno indicaría que la estrategia es rentable.
- **Reducción máxima del balance y de la equidad:** representan la mayor caída desde el máximo en el capital o en el valor de la cuenta durante el test, uno de los parámetros más importantes para los riesgos de la estrategia.
- **Posiciones rentables:** el porcentaje de operaciones que terminaron en beneficio, es un dato que podemos usar para medir la efectividad de la estrategia.
- **El número máximo de ganancias y pérdidas consecutivas:** nos habla de la manera en que se ha comportado la estrategia a lo largo de las operaciones.

Las estadísticas permiten identificar la fuerza, las debilidades e incluso la estabilidad de una estrategia de trading.

Transacciones

Fecha/Hora	Transacción	Símbolo	Tipo	Dirección	Volumen	Precio	Orden	Comisión	Beneficio	Balance		
Fecha/Hora	Transacción	Símbolo	Tipo	Dirección	Volumen	Precio	Orden	Comisión	Swap	Beneficio	Balance	Comentario
2024.12.01 00:00:00	1		balance					0.00	0.00	100 000.00	100 000.00	
2024.12.02 18:23:31	2	EURUSD	buy	in	1	1.04962	2	-1.50	0.00	0.00	99 998.50	
2024.12.02 22:00:00	3	EURUSD	sell	out	1	1.04903	3	-1.50	0.00	-59.00	99 938.00	
2024.12.03 20:40:35	4	EURUSD	sell	in	1	1.05306	4	-1.50	0.00	0.00	99 936.50	
2024.12.03 21:51:32	5	EURUSD	buy	out	1	1.05106	5	-1.50	0.00	200.00	100 135.00	tp 1.05106
2024.12.04 16:21:33	6	EURUSD	buy	in	1	1.04828	6	-1.50	0.00	0.00	100 133.50	
2024.12.04 17:00:27	7	EURUSD	sell	out	1	1.05028	7	-1.50	0.00	200.00	100 332.00	tp 1.05028
2024.12.04 17:31:43	8	EURUSD	sell	in	1	1.05285	8	-1.50	0.00	0.00	100 330.50	
2024.12.04 21:02:05	9	EURUSD	buy	out	1	1.05085	9	-1.50	0.00	200.00	100 529.00	tp 1.05085
2024.12.06 16:03:17	10	EURUSD	sell	in	1	1.05935	10	-1.50	0.00	0.00	100 527.50	
2024.12.06 16:11:30	11	EURUSD	buy	out	1	1.05735	11	-1.50	0.00	200.00	100 726.00	tp 1.05735

Figura 82: Probador de estrategias: Backtesting sección tabla transacciones

La figura 82 muestra en una tabla esta sección de transacciones. La última parte de la sección de backtesting es la tabla de las transacciones, se muestra todas y cada una de las operaciones que ha realizado el sistema a lo largo de toda la simulación, siendo clave en el estudio detallado del comportamiento de la estrategia porque nos permite ver, paso por paso, la lista de operaciones de compras y ventas que ha realizado, en función de todos los parámetros que antes hemos definido.

Cada fila de la tabla representa una transacción e incluye la siguiente:

- **Fecha/hora:** momento exacto en que se ejecutó la transacción.
- **Transacción:** número secuencial de la operación.
- **Símbolo:** activo financiero que ha sido objeto de la operación (por ejemplo, EURUSD).
- **Tipo y Dirección:** indica si la transacción es de tipo *buy* (compra) o *sell* (venta), y si se trata de una entrada o salida del mercado (qué tipo de operación se ha realizado).
- **Volumen y Precio:** reflejan el tamaño de la posición abierta y el precio al que se generó la operación.
- **Orden:** número que identifica la orden.
- **Comisión y Beneficio:** reflejan los costes de ejecución de la operación y el resultado de la operación (el beneficio que se ha obtenido).
- **Balance:** saldo acumulado tras la operación, que refleja el resultado inmediato de la operación, en la cuenta.
- **Comentario:** a veces se especifican datos adicionales como los niveles de *take profit* (tp) que se han alcanzado.

Esta tabla nos permite no sólo estudiar la rentabilidad de cada orden, sino también ver si existen patrones, errores de configuración o puntos a mejorar en el proceso de construcción de la estrategia. En combinación con los gráficos y las estadísticas, proporciona una visión del backtest completa.

10 Resultados y casos de uso

En esta sección se ofrecen evidencias de forma práctica de los algoritmos de trading que se han desarrollado de manera práctica en el proyecto, en el ejemplo en concreto se basa en la estrategia AMD. Con el fin de realizar la validación de las estrategias de trading se ha hecho uso de cuentas demo, es decir, dinero ficticio facilitadas por la empresa de fondeo FTMO, lo que permite simular las condiciones del mercado real cumpliendo una serie de requerimientos. La finalidad de este tipo de pruebas ha sido cumplir unos requisitos mínimos impuestos por la compañía de trading para validar la consistencia y la rentabilidad del sistema automático.



Figura 83: Ejemplo de resultados con las estrategias de trading gráfico 1

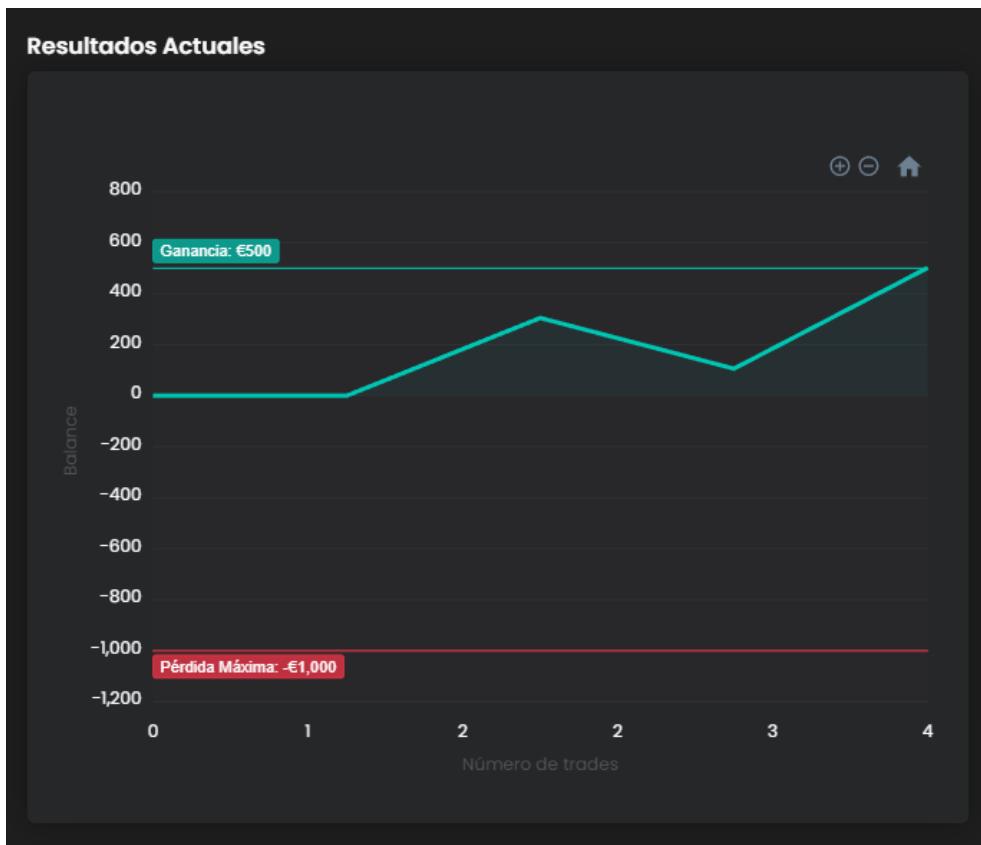


Figura 84: Ejemplo de resultados con las estrategias de trading gráfico 2

A partir de los gráficos de la figura 83 y 84 se puede observar la evolución positiva del balance que se ha obtenido a partir de la primera fase y segunda fase de evaluación. Se puede ver que el control del riesgo ha sido adecuado y que la optimización de ganancias se ha realizado adecuadamente. Las estrategias se han mostrado capaces de llegar a los objetivos mínimos impuestos por FTMO, en este caso se muestra el ejemplo de la primera fase, entre los cuales se encuentran los siguientes:

- Superar la operación un mínimo de días establecidos.
- No superar una pérdida diaria máxima de €500.
- Mantenerse en una pérdida total inferior a €1.000.
- Superar el objetivo de beneficio mínimo de €1.000.

Objetivos		
Objetivos de Trading	Resultados	Resumen
+ Mínimo 4 Días de Trading	13	<input checked="" type="checkbox"/> Aprobado
+ Pérdida máxima del día - €500	-€215.29 (-2.15%)	<input checked="" type="checkbox"/> Aprobado
+ Pérdida Máx. -€1,000	-€153.59 (-1.53%)	<input checked="" type="checkbox"/> Aprobado
+ Objetivo de Beneficio €1.000	€1,023.80 (10.23%)	<input checked="" type="checkbox"/> Aprobado

Figura 85: Ejemplo de resultados con las estrategias de trading estadísticas

En la figura 85 se muestra las estadísticas presentes. La ejecución de estas validaciones pone de manifiesto que el sistema no solo funciona correctamente a nivel técnico, sino que también se

muestra útil en contextos reales de mercado. El hecho de que se hayan cumplido todos los criterios de la fase 1 y 2 en cuentas de fondeo indica que existe la posibilidad de que estos algoritmos sean utilizados para traders en entornos profesionales. Queda por recalcar que son estadísticas a favor, no siempre se tiene resultados efectivos hay variables como el análisis fundamental o circunstancias del mercado que cambien el contexto de nuestra estadística.

11 Conclusiones y trabajos futuros

Este Trabajo de Fin de Grado ha consistido en el diseño, desarrollo e implementación de una aplicación web interactiva destinada al análisis de estrategias de trading algorítmico. Se ha implementado una arquitectura moderna que utiliza Next.js para el frontend, Python con Flask para manejar la lógica del lado del servidor y MetaTrader 5 como plataforma de simulación.

La aplicación permite que el usuario se registre, elija una estrategia y defina sus parámetros, ejecute simulaciones (backtesting) y analice los resultados mediante una interfaz visual sencilla y funcional. Además, se ha conseguido integrar servicios externos como Clerk, UploadThing, Prisma o Ngrok para manejar la autenticación, almacenamiento, base de datos y conexión con el servidor local, respectivamente.

Los objetivos que se establecieron al inicio del TFG se han cumplido por completo, incluida la conexión con MetaTrader 5 en local, el análisis gráfico de resultados, la gestión de estrategias personalizadas y la implementación de una plataforma web funcional para usuarios reales.

El desarrollo de esta aplicación web ha permitido construir una herramienta práctica y funcional para la prueba y análisis de estrategias de trading algorítmico. A través de una interfaz clara y usable, los usuarios pueden diseñar, configurar y lanzar simulaciones de backtesting, interaccionando directamente con la plataforma MetaTrader 5. La integración de tecnologías modernas como Next.js, Prisma, Clerk, UploadThing y Python con Flask ha demostrado ser funcional para la arquitectura y un sistema útil para traders profesionales.

Desde un punto de vista técnico, el proyecto ha abordado correctamente la modularización backend, la gestión segura de usuarios o la comunicación con servicios externos como bases de datos o API locales, y se ha conseguido generar una base bien adaptada y escalable que puede servir para desarrollos futuros, siguiendo los principios de la accesibilidad, usabilidad y seguridad.

Existen diversas líneas de mejora y ampliación que se sugieren como trabajos futuros:

- Aumentar la responsividad de la interfaz para optimizar la vista en dispositivos móviles y tablets, ya que en estos momentos se encuentra orientada a la vista de escritorio.
- Mejorar el rendimiento de la simulación de backtesting, permitiendo la ejecución asíncrona o en paralelo en equipos con mejor rendimiento.
- Extender las estrategias de trading personalizadas, completándolas con configuraciones más avanzadas que se adapten mejor a múltiples perfiles, como por ejemplo opción de optimización.
- Incorporar métricas de seguimiento y dashboards analíticos para que los usuarios puedan visualizar la evolución de sus simulaciones en el tiempo.
- Añadir más secciones para adquirir una comunidad donde los traders comparten sus configuraciones de las estrategias predefinidas y valorar información relevante para obtener una

formación robusta como profesionales del trading.

Este proyecto propone las bases para la implementación de una plataforma completa de evaluación y mejora de estrategias algorítmicas, que posee un importante los márgenes de evolución técnica y funcional que se podrían desarrollar en el futuro.

Bibliografía

- [1] Anaconda. Conda: Gestor de entornos y paquetes en python, 2024. Disponible en: <https://anaconda.org/anaconda/conda>.
- [2] apiweb. apiweb, 2024. <https://es.wikieducator.org/images/0/09/AplicacionesWeb.pdf>.
- [3] Gavin Bierman, Martín Abadi, and Mads Torgersen. Ecoop 2014—object-oriented programming: 28th european conference, uppsala, sweden, july 28–august 1, 2014. proceedings 28. pages 257–281, 2014. <https://users.soe.ucsc.edu/abadi/Papers/FTS-submitted.pdf>.
- [4] Clerk. Clerk: Sistema de autenticación para aplicaciones web, 2024. Disponible en: <https://clerk.com/nextjs-authentication>.
- [5] FTMO. Ftmo: Empresa de fondeo para traders profesionales, 2024. Disponible en: <https://ftmo.com/es/>.
- [6] Cory Gackenheimer. Introduction to react. 2015. <https://pepa.holla.cz/wp-content/uploads/2016/12/Introduction-to-React.pdf>.
- [7] Marc Gibert Ginestà and Oscar Pérez Mora. Bases de datos en postgresql. *Slj:[sn]*, 2012. <https://artpatcusco.com/sis/pdf/20140527172742en.pdf>.
- [8] glassdoor. glassdoor: Sueldos, 2024. <https://www.glassdoor.es/Sueldos/index.htm>.
- [9] Investing.com. Horario de los mercados bursátiles y de forex: Aperturas y cierres globales, 2024. Disponible en: <https://es.investing.com/tools/market-hours>.
- [10] Charles M Jones. What do we know about high-frequency trading? *Columbia Business School Research Paper*, (13-11), 2013. <https://ecommons.cornell.edu/server/api/core/bitstreams/5fbf20f8-67ff-4455-bc9a-ae18969878d4/content>.
- [11] Mutiara Auliya Khadija, Astrid Noviana Paradhita, Agus Purbayu, Sahirul Alim Tri Bawono, Abdul Aziz, Sri Haryati, Aubin Sava Rausanfiker, Bagus Brang Wicaksono, and Shafwan Eksa Jayadi. Backend programming techniques in the development of resource manager features in vrms systems based on orm prisma. *Decode: Jurnal Pendidikan Teknologi Informasi*, 4(3):830–843, 2024. <https://bibdigital.epn.edu.ec/bitstream/15000/26137/1/CD>
- [12] Mohammad Fariz Syah Lazuardy and Dyah Anggraini. Modern front end web architectures with react.js and next.js. *Research Journal of Advanced Engineering and Science*, 7(1):132–141, 2022. <https://irjaes.com/wp-content/uploads/2022/02/IRJAES-V7N1P162Y22.pdf>.
- [13] lerma2013aplicaciones. lerma2013aplicaciones, 2024. <https://es.wikieducator.org/images/0/09/AplicacionesWeb.pdf>.
- [14] Fred J Maryanski. Backend database systems. *ACM Computing Surveys (CSUR)*, 12(1):3–25, 1980. <https://dl.acm.org/doi/pdf/10.1145/356802.356804>.
- [15] MetaTrader 5. Metatrader 5: Plataforma de trading multiactivo, 2024. Disponible en: <https://www.metatrader5.com/es/download>.
- [16] Mockitt. Mockitt: Herramienta de prototipado para interfaces de usuario, 2024. Disponible en: <https://mockitt.com/home.html>.
- [17] Jorge A Montoya, Diego A Escobar, and Carlos A Moncada. Test de la variabilidad en las condiciones de accesibilidad urbana ofrecida por el sistema de transporte público en manizales (colombia). *Información tecnológica*, 34(2):99–110, 2023. <https://www.scielo.cl/pdf/infotec/v34n2/0718-0764-infotec-34-02-99.pdf>.

- [18] Ngrok. Ngrok: Túneles seguros para exponer servicios locales, 2024. Disponible en: <https://ngrok.com/>.
- [19] Hui Niu, Siyuan Li, and Jian Li. Metatrader: An reinforcement learning approach integrating diverse policies for portfolio optimization. pages 1573–1583, 2022. <https://dl.acm.org/doi/pdf/10.1145/3511808.3557363>.
- [20] Node.js. Node.js: Entorno de ejecución para javascript en el servidor, 2024. Disponible en: <https://nodejs.org/es/download>.
- [21] npm. npm: Gestor de paquetes para javascript, 2024. Disponible en: <https://www.npmjs.com/package/download>.
- [22] Prisma. Prisma con postgresql: Orm para bases de datos relacionales, 2024. Disponible en: <https://www.prisma.io/postgres>.
- [23] Juan Pérez and Diego Martínez. Desarrollo de una interfaz web para el usuario basada en tecnologías front-end, s.f. Disponible en: <https://sedici.unlp.edu.ar/bitstream/handle/10915/120476/Ponencia.pdf-PDFA.pdf?sequence=1>.
- [24] QuantConnect. Quantconnect: Plataforma de backtesting y desarrollo de estrategias algorítmicas, 2024. Disponible en: <https://www.quantconnect.com/>.
- [25] RangoTrading. Las 3 fases del mercado: Consolidación, manipulación y distribución, 2024. Disponible en: <https://www.rangotrading.com/las-3-fases-del-mercado-consolidacion-manipulacion-y-distribucion/>.
- [26] Resend. Resend: Envío de correos electrónicos con next.js, 2024. Disponible en: <https://resend.com/docs/send-with-nextjs>.
- [27] Pablo Muñoz Romero. Análisis técnico: Diseño de una estrategia automatizada de trading. <https://core.ac.uk/download/pdf/547390505.pdf>.
- [28] shadcn. Shadcn ui: Biblioteca de componentes ui para react, 2024. Disponible en: <https://ui.shadcn.com/>.
- [29] Montserrat Sobrepera Delgado. Análisis de modelos de trading algorítmico en el mercado forex. 2015. <https://repositorio.comillas.edu/xmlui/bitstream/handle/11531/4468/TFG001267.pdf?sequence=1>.
- [30] StrategyQuant. Strategyquant: Plataforma de creación automatizada de estrategias de trading, 2024. Disponible en: <https://strategyquant.com>.
- [31] UploadThing. Uploadthing: Servicio para gestión y subida de archivos, 2024. Disponible en: <https://uploadthing.com/>.
- [32] Vercel. Vercel: Plataforma de despliegue y alojamiento de sitios web, 2024. Disponible en: <https://vercel.com/docs>.
- [33] WAVE. Wave: Herramienta de evaluación de accesibilidad web, 2024. Disponible en: <https://wave.webaim.org/>.
- [34] Wikipedia. Rsi: Índice de fuerza relativa en análisis técnico [wikipedia], 2024. Disponible en: <https://es.wikipedia.org/wiki/>