knowledge

MDPI

Article

Development of a Backtesting Web Application for the Definition of Investment Strategies

Antonio Sarasa-Cabezuelo

Department of Computer Science, Complutense University of Madrid (UCM), 28040 Madrid, Spain; asarasa@ucm.es

Abstract: Backtesting represents a set of techniques that aim to evaluate trading strategies on historical data in order to verify their effectiveness before applying them to a market in real time. This requires processing large amounts of data from different periods and applying different simulation techniques to them. In general, these types of tools are not very popular for reasons such as the amount of data that must be evaluated and maintained, the computational resources that are required, and the need to have a deep conceptual understanding of these techniques in order to use them. This article presents a web application that implements a set of backtesting functionalities that allow evaluating different trading strategies, managing portfolios, representing the results of simulations, and optimizing a stock portfolio, all from an intuitive and visual interface that makes these techniques accessible to new investors in this field.

Keywords: backtesting; strategy; trading; trader; portfolios; quantitative finance

check for updates

Citation: Sarasa-Cabezuelo, A. Development of a Backtesting Web Application for the Definition of Investment Strategies. *Knowledge* **2023**, *3*, 414–431. https://doi.org/10.3390/knowledge3030028

Academic Editor: Gabriele Santoro

Received: 30 May 2023
Revised: 26 July 2023
Accepted: 9 August 2023
Published: 14 August 2023

1. Introduction

The behavior of the stock market and the variations in the value of financial assets constitute a phenomenon that is difficult to model [1] due to the enormous number of variables on which it depends and the non-linear nature of the problem. That is why it is difficult to know how it works and what its future behavior will be. However, different tools [2] have been developed in order to better understand this phenomenon. All of them are based on analyzing data from the past [3] with the aim of finding patterns or markers that can be used to obtain information on how future behavior will be [4]. In this way, based on the information obtained, strategies can be defined, called trading strategies, about how to operate on a financial market to obtain the maximum benefit [5]. A trading strategy is a system made up of a set of previously defined rules that indicate which financial assets to buy or sell [6], when to buy or sell said financial assets, what percentage of the capital to allocate to the corresponding purchase or sale, and how to manage the risk of both the operation and the account as a whole.

The main problem with a trading strategy is that its validity can only be verified when it is applied in a real market, which implies an economic risk. That is why, before applying a trading strategy, its feasibility must be assessed in a controlled environment in which market behavior and the result of applying the strategy in said market can be simulated. For this, the so-called backtesting techniques [7] have been defined. They are simulation methods that aim to evaluate how a trading strategy would have performed on a set of historical data [8].

The efficiency of these methods is based on their ability [9] to manipulate and process large amounts of historical data. The more data that are considered, the more valid and true the results of the backtesting operation [10]. However, this imposes the limitation of not being able to apply these techniques manually as it is a very computationally expensive operation, and therefore the need to use computer processing applications that implement these techniques [11].

Backtesting software applications can be a valuable tool for traders, but they also have some limitations. These limitations may include:

- Data: Backtesting can only be as good as the data it is based on. If the data are inaccurate or incomplete, the backtesting will also be inaccurate. This is because backtesting uses historical data to simulate the performance of a strategy. If the data are inaccurate or incomplete, the backtesting will not be able to simulate the performance of the strategy accurately.
- Model: Backtesting can only be as good as the model it uses. If the model is inaccurate, the backtesting will be inaccurate. This is because backtesting uses a model to predict the performance of a strategy. If the model is inaccurate, the backtesting will not be able to predict the performance of the strategy accurately.
- Execution: Backtesting does not consider execution. This means that the results of backtesting may not be representative of what would happen in the real world. This is because execution is an important part of trading, and it can have a significant impact on the results. For example, if a strategy includes stop orders, the results of backtesting may not be accurate if execution costs are not taken into account.
- Bias: Traders can bias the results of backtesting by choosing the data they want to use and the models they want to test. This can lead to unrealistic results. For example, if a trader chooses only data that show good performance of their strategy, the results of backtesting will be more optimistic than what the actual performance of the strategy would be.

This article describes a web application that implements the necessary functionalities for the execution of backtesting in a comfortable and intuitive way and implements a trading strategy based on buying or selling a financial asset at its maximum or minimum for a certain time frame. In this sense, the application covers the following objectives:

- Allow the user to configure a portfolio, choose the financial assets, and parameterize the trading strategy on which to run the backtesting.
- Allow the user to manage financial portfolios.
- Allow the user to upload historical data on the prices of financial assets to the system.
- Allow the management, collaboration, and sharing of information (backtestings and financial portfolios) between subscribed users. To do this, it will provide functions for managing user accounts, saving and sharing the "backtesting" executed, and sharing a financial portfolio.
- Offer useful features to a trader, such as a trading journal, or perform "backtesting" manually.
- Design an API that is intuitive, minimalist, and contains sufficiently comprehensive information.

The structure of this paper is as follows: A brief review of other similar applications is described in Section 6. Section 3 defines the architecture and model of the developed application. Section 4 describes the backtesting strategy that has been implemented. Next, in Section 5, the functionality of the application is described. Section 6 proposes a set of conclusions and lines of future research.

2. Review Section

There are numerous applications that aim to help the investor who wants to operate on the financial market. Most of them are aimed at facilitating functions to act in real time on the markets or manage the securities portfolio, such as consulting a market, buying or selling securities, obtaining recommendations in real time, scheduling operations, or configuring change notifications of value [12]. However, there are not too many applications that implement backtesting techniques due to different reasons [13], such as the need to have and maintain a large historical database, the availability of computational resources capable of performing the necessary processing, and the low popularity of these techniques because they work on data from the past and use techniques that require deeper theoretical

and conceptual knowledge. Some examples of these tools are as follows: MetaTrader 5 [14] is a tool to invest in forex, stocks, and futures. It implements a backtesting functionality called MetaTrader's Trading Strategy Tester that allows evaluating a trading strategy before its implementation in live trading. MetaStock [15] is a tool that allows you to backtest up to 58 different trading systems. Trader Workstation is a platform that offers among its functionalities the portfolio manager tool, which allows you to configure a portfolio on which to run backtesting based on fundamental analysis. TrendSpider [16] is a trading platform that has a strategy tester that allows users to write the strategy code and its subsequent execution. TradingView [17] is one of the most popular trading platforms worldwide that allows its users to backtest a given trading strategy. Additionally, NinjaTrader [18] is a trading platform that offers backtesting functionality that allows you to simulate, optimize, and analyze the performance of strategies.

These tools have some limitations [19], such as the complexity of how they work for non-expert users, the cost of having to pay a license for their use, the types of backtesting techniques they implement, the limitations in terms of the types of financial assets, markets, and trading strategies that they allow to be evaluated, or the limitations regarding the direct application of the results of the backtesting to optimize the securities portfolio [20].

3. Architecture and Data Model

The system architecture follows a client-server model where one or more clients make requests and receive service from a central server. To do this, the client has a visual interface through which to make requests and receive responses from the server. When the requests arrive, the server processes them and sends the corresponding response to the client that made the request. On the other hand, the server communicates with a MySQL database that implements the persistence of system information.

Regarding the information that is necessary to store. The data required are financial asset data, user data, and user-generated data. The data on financial assets are also of two types: data referring to the specifications of the financial asset contracts (the name of the asset, the market in which the financial asset in question is listed or the currency in which it is denominated said contract) and the historical data of the daily prices of financial assets (date, the opening price, the closing price, the maximum of the day, the minimum of the day, the adjusted closing price and the volume). Regarding user data, they refer to user accounts (a username, a full name, a password, and a role, such as administrator or user). Finally, the data generated by the user can be of four different types (depending on the functionality with which they are associated): backtesting, portfolio management, manual backtesting, and trading journal.

For the execution of a backtesting, it is necessary to store the user who executes said backtesting, the name that this user gives it, the information of the portfolio that has been configured for the execution of this, and the start and end dates. Finally, if the user has saved it or not, if the user has made it public or not, the parameterization of the strategy to be tested and the financial assets on which to run the backtesting. The user, the parameterization of the strategy, and the financial assets on which the backtesting is executed are independent entities, and the rest of the attributes belong to the backtesting entity. Portfolio management also generates a series of data: portfolio configuration data (start and end dates, capital, the currency in which the portfolio is denominated, and whether the user has shared it or not), transactions, and positions. For each manual backtesting that a user creates and executes, it is necessary to know the user who created it, the name they gave it, the configuration of the user's wallet, whether or not the user has saved it for later recovery, and operations added through the application by the user. Finally, the system also saves the data generated by the user in the trading journal functionality: the user, the date the text was created, the date the text refers to, and the text itself.

In order to maintain the persistence of all the information, a MySQL database with several groups of tables has been used:

(a) Tables on user accounts. Formed by the table:

- Table users includes the necessary information that every user account created in the web application must have.

(b) Tables on backtesting. Formed by the tables:

- Table backtesting includes all the data needed to execute a backtesting.
- Table strategy_type relates the backtesting strategy (previous table) with the type of strategy it is.
- Table strategy_buy_short_max_min includes the parameterization of the strategy implemented in this web application (remember that the strategy implemented here is a global strategy that can be divided into four types of more specific sub-strategies: buy at maximums, sell at maximums, buy at lows, and sell at lows).
- Table backtesting_assets stores the list of financial assets on which a backtesting is executed.

(c) Tables on portfolio management. Formed by the tables:

- Table portfolios includes configuration information for each financial portfolio created by a user.
- Table positions contains the information of each existing position.
- Table position_portfolio relates each position in the positions table with its corresponding financial portfolio in the portfolio table.
- Table transactions includes all the information on all purchase and sale operations carried out during portfolio management.

(d) Tables on other trading services. Formed by the tables:

- Table backtesting_manual includes the information of the manual backtesting created and executed by the users.
- Table backtesting_manual_operations includes the operations that each user has added during the execution of the manual backtesting.
- Table journal includes all the information from the trading journal of each user.

(e) Tables on managing data. Formed by the tables:

- Table asset_class contains all the financial assets in the system, whatever their class.
- Table stock_prices includes daily data on stock prices.
- Table futures includes the data on the specifications of the futures contracts.
- Table futures_prices includes daily futures price data.

4. Trading Strategy

The trading strategy implemented is based on buying or selling a financial asset (stock or futures) at its maximum or minimum for a given time frame [21]. In this sense, these are four particular cases of a general strategy based on [22] buying or selling at extreme price moments within a specific time frame. Each of these four cases is given by four different input types that constitute a sub-strategy in itself [23]:

- Going long (buying) at the maximum price of a financial asset in the reference time frame.
- Go short (sell) at the maximum price of a financial asset in the reference time frame.
- Going long (buying) at the minimum price of a financial asset in the reference time frame.
- Go short (sell) at the minimum price of a financial asset in the reference time frame.

For each of these entries, the corresponding backtesting is allowed [24], with the maximums and minimums of the following reference time frames: 5 days, 10 days, 15 days, 1 month, 3 months, 6 months, and 1 year.

On the other hand, the user can choose between two different types of output [25]:

(a) Exits with a temporary stop (exit once a certain period of time has elapsed since the entry was made). This output is allowed to be implemented in three different ways:

- First Temporary Stop: 50% of the days of the reference time frame.
- Second Temporary Stop: 100% of the days of the reference time frame

- Third Temporary Stop: 150% of the days of the reference time frame.

(b) Exit with a trailing stop (update the stop as the price moves). Backtesting of this strategy is allowed with an exit based on a trailing stop defined by the ATR indicator (Average True Range: volatility indicator) [26] using up to three different ATR multipliers: 1, 2 and 3.

For the implementation of the backtesting, an event-driven design [27] has been followed. This type of design consists of the implementation of a loop algorithm that receives events and responds to each of these events accordingly. In this way, it is possible to simulate that responses are being handled in real time, even though the backtesting is based on historical data. This form of design has a number of advantages associated with it [28]:

- Avoid anticipation bias [29]. This consists of the use of data during the execution of the backtesting that would not have been obtained during the simulated period. Therefore, an event-driven design guarantees that the simulation executed when making the buying and selling decisions only has the data that would actually be available.
- Code reusability [30]. Specifically, the same code can be used both for the execution of backtesting on historical data and for the implementation of a strategy in real time. Only a few changes are necessary.

5. Functionality

Next, the five main functionalities of the application will be explained.

5.1. Dashboard

It is the main screen of the registered user, which shows four different sets of objects (Figure 1). First of all, the last three backtests ("Lista de Backtesting") created and saved. Second, the three most recently created portfolios("Lista de carteras") Finally, the user has direct access to his last three previously saved manual backtests and the last three comments added to the trading journal.

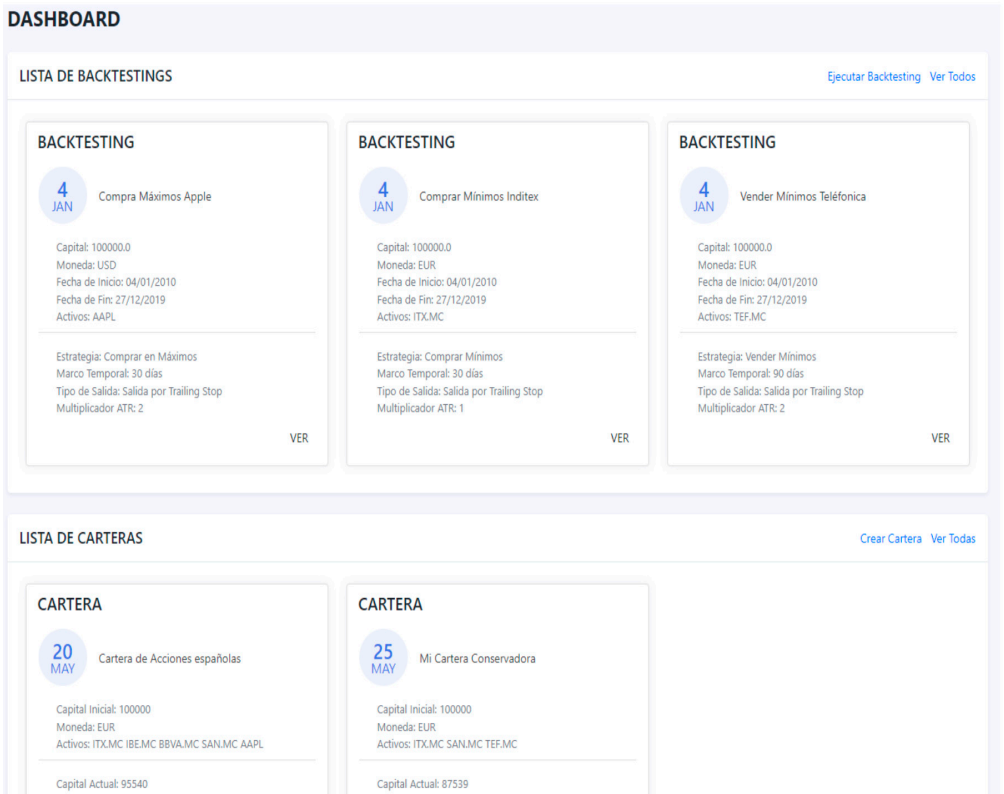


Figure 1. Dashboard view.

In addition, in the upper navigation bar there is a search engine through which you can search for backtesting and financial portfolios whose owners have previously published. The search engine has two fields. In the first, the user selects the option they want to search for: backtesting, portfolio, user, or active. In the second field, the user enters the text that he wants to find. In the case of having selected backtesting, portfolio or asset, the user must enter the ticker of a valid financial asset (the ticker must be the same one with which said financial asset is identified on the page from which data extraction is performed). If the user enters a valid ticker, then the backtesting published by the rest of the users that have been executed on the financial asset whose ticker was entered is displayed on the screen ("Lista de Backtesting"). If it does not find any backtesting or the financial asset does not exist, it shows an empty search. Figure 2 shows an example of a backtesting search through the AAPL ticker (the Apple ticker), as well as its result.

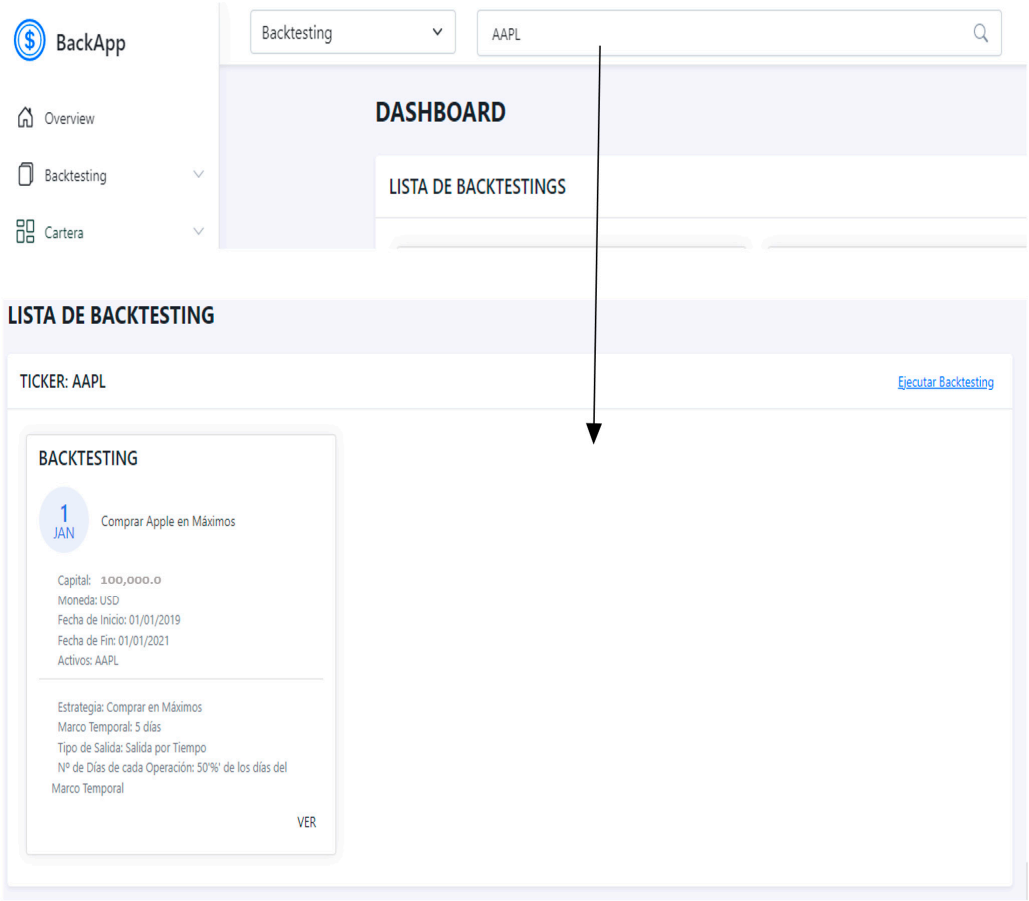


Figure 2. View search backtesting.

The portfolio search works in a similar way. The user selects the portfolio option in the upper navigation bar and then enters the ticker of the financial asset on which he wants to perform the search. All published portfolios ("Cartera") containing the asset whose ticker was entered are then displayed ("Lista de carteras"). Figure 3 shows an example of a portfolio search through the ticker NFLX (Netflix ticker).

On the other hand, the user can also search for financial assets in order to consult the daily price data available in the system. To do this, the user selects the asset option in the top navigation bar and then enters the ticker of the financial asset he wants to search for. If the user enters a valid ticker, then a graph is displayed with the evolution of the asset's price and, below, a table with all the daily price data (Figure 4).

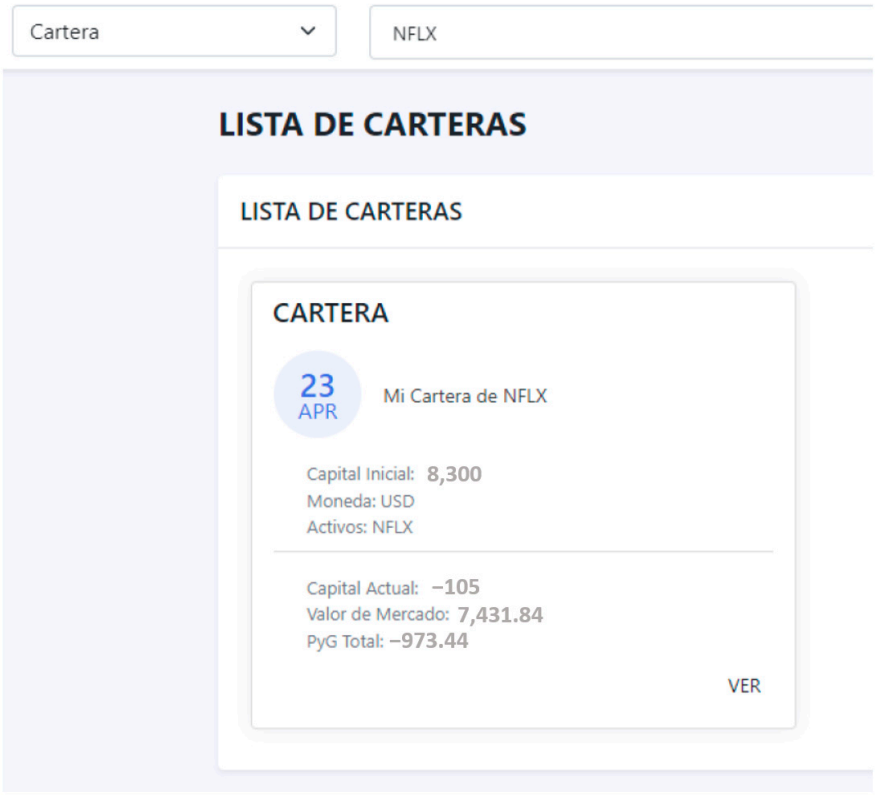
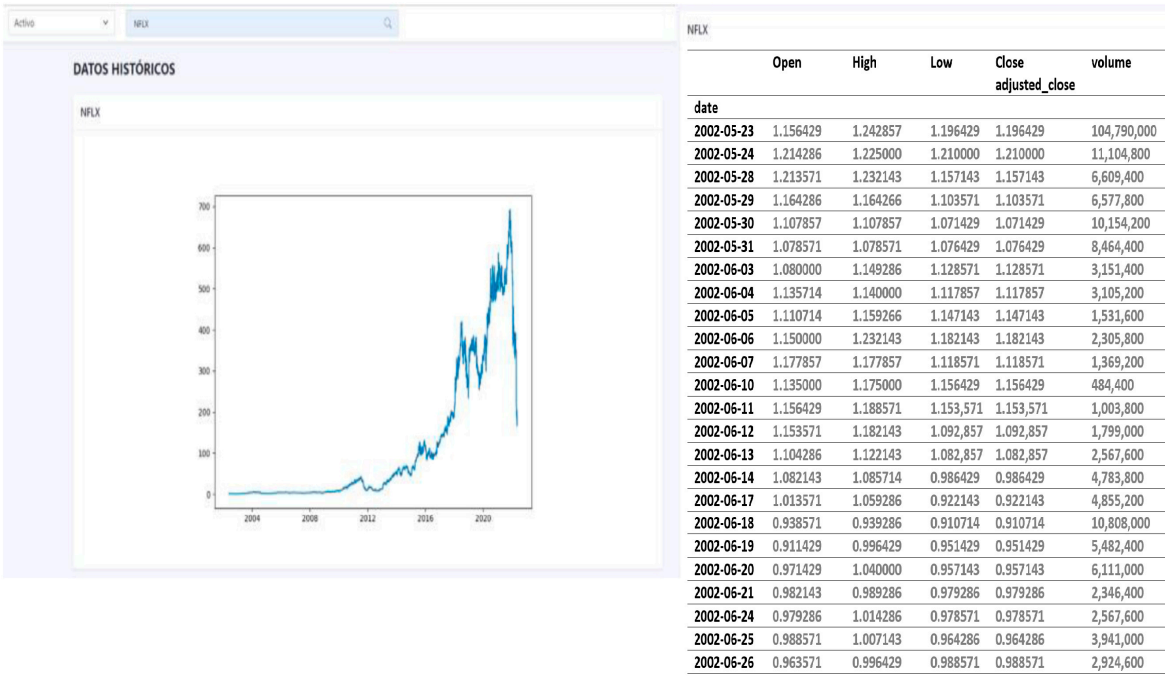


Figure 3. Portfolio search view.



	Open	High	Low	Close adjusted_close	volume
date					
2002-05-23	1.156429	1.242857	1.196429	1.196429	104,790,000
2002-05-24	1.214286	1.225000	1.210000	1.210000	11,104,800
2002-05-28	1.213571	1.232143	1.157143	1.157143	6,609,400
2002-05-29	1.164286	1.164266	1.103571	1.103571	6,577,800
2002-05-30	1.107857	1.107857	1.071429	1.071429	10,154,200
2002-05-31	1.078571	1.078571	1.076429	1.076429	8,464,400
2002-06-03	1.080000	1.149286	1.128571	1.128571	3,151,400
2002-06-04	1.135714	1.140000	1.117857	1.117857	3,105,200
2002-06-05	1.110714	1.159266	1.147143	1.147143	1,531,600
2002-06-06	1.150000	1.232143	1.182143	1.182143	2,305,800
2002-06-07	1.177857	1.177857	1.118571	1.118571	1,369,200
2002-06-10	1.135000	1.175000	1.156429	1.156429	484,400
2002-06-11	1.156429	1.188571	1.153,571	1.153,571	1,003,800
2002-06-12	1.153571	1.182143	1.092,857	1.092,857	1,799,000
2002-06-13	1.104286	1.122143	1.082,857	1.082,857	2,567,600
2002-06-14	1.082143	1.085714	0.986429	0.986429	4,783,800
2002-06-17	1.013571	1.059286	0.922143	0.922143	4,855,200
2002-06-18	0.938571	0.939286	0.910714	0.910714	10,808,000
2002-06-19	0.911429	0.996429	0.951429	0.951429	5,482,400
2002-06-20	0.971429	1.040000	0.957143	0.957143	6,111,000
2002-06-21	0.982143	0.989286	0.979286	0.979286	2,346,400
2002-06-24	0.979286	1.014286	0.978571	0.978571	2,567,600
2002-06-25	0.988571	1.007143	0.964286	0.964286	3,941,000
2002-06-26	0.963571	0.996429	0.988571	0.988571	2,924,600

Figure 4. Asset search view.

Finally, the user also has the option to search for other users. To do this, you must select the user option in the upper navigation bar and then enter the username of the user you want to find.

5.2. Backtesting

The main functionality of the application is to execute the backtesting of a trading strategy based on buying or selling at the maximum or minimum of the price in a given time frame. To do this, the user can access the functionality both through the dashboard and the navigation bar. Through the dashboard, the user has to click on the run backtesting button in the backtesting list section. Additionally, if the navigation bar is used, then it must be clicked on the backtesting button on the side navigation bar, which will open a drop-down menu, and the user must click on the run backtesting option (Figure 5).

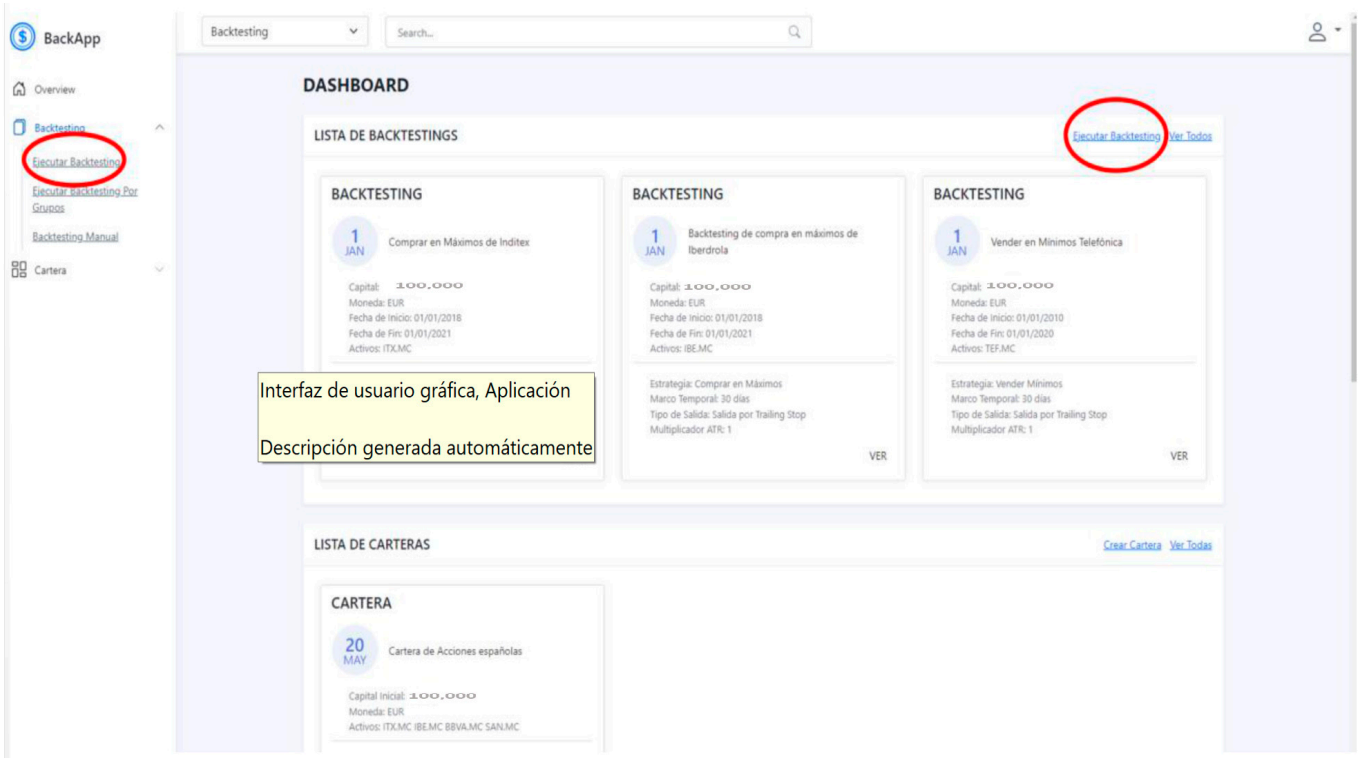


Figure 5. Select run backtesting.

Next, the user must configure the backtesting by completing a form divided into five steps:

- In the first step (Figure 6a), the user gives a name to the backtesting and configures the portfolio on which the backtesting is going to be executed. In the currency field("Moneda"), the user must enter the currency symbol (EUR, USD. . .).
- In the second step (Figure 6b), the user enters the list of tickers of the financial assets on which the backtesting is to be carried out.
- In the third step (Figure 6c), the user selects the dates ("Fecha de inicio", "Fecha de fin") between which he wants to run the backtesting. If the user chooses dates for which there are no data for the chosen financial assets, the application will inform the user after submitting the form.
- In step 4 (Figure 6d), the user chooses the sub-strategy ("Estrategia") to be evaluated. It is possible to choose between the following: buy highs, sell highs, buy lows, and sell lows.
- Finally, in the last step (Figure 6e), the parameters of the chosen sub-strategy are configured: the time frame ("Marco temporal") and the type of output desired ("Salida").

Once all the fields of the form have been completed, the user clicks (Figure 6f) on the run backtesting button, and if the data are correct, the backtesting is created.

Once executed, the results of the backtesting are displayed. First, the system displays an alert about whether or not the use of this strategy is recommended (Figure 7a). Next, it

shows the statistical results of the backtesting executed: the cumulative return ("Retorno Total"), the Sharpe ratio (which measures the relationship between return and risk), the maximum fall or highest drawdown (Max Drawdown), and the duration of this drawdown ("Drawdown duration"). In addition, a graph is generated with the evolution of the capital curve of the strategy. Likewise, there is the option of viewing a graph (Figure 7b) for each of the financial assets on which the backtesting has been carried out, which shows the evolution of the price of the financial asset in question together with the buy and sell signals set by the strategy. Figure 7c shows an illustration of this type of graph ("Backtesting de compra en máximos de Iberdrola") with the evolution of the price for Iberdrola and the buy (green) and sell (red) signals set by the strategy. Finally, to save the backtesting for later recovery, the user must click on the save button, and once saved, the buttons to share ("Compartir") and delete the backtesting("Borrar") will appear (Figure 7d).

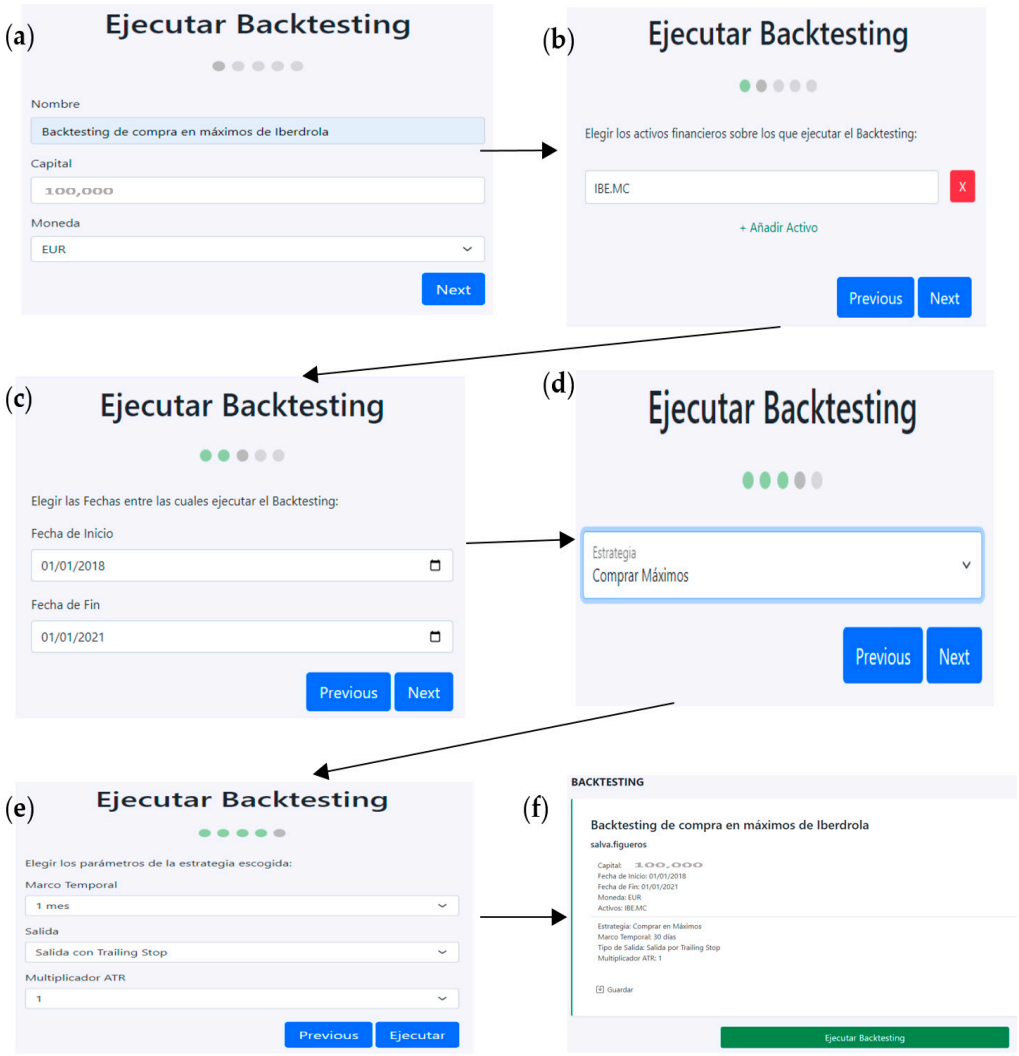


Figure 6. (**a**) Backtesting form step 1. (**b**) Backtesting form step 2. (**c**) Backtesting form step 3. (**d**) Backtesting form step 4. (**e**) Backtesting form step 5. (**f**) Backtesting configuration view.

It is also possible to carry out the execution of the backtesting by groups. To do this, the user must click on the backtesting element in the navigation sidebar. Once the drop-down menu is open, the user has to click on run ("Ejecutar Backtesting por Grupos") backtesting by groups (Figure 8a). Next, the user must complete a form similar to the previous one about run backtesting. The only difference resides in completing a previous step in which the user must choose the metric to use to divide the financial assets into two groups. To do this, the user has two options: market capitalization ("Capitalización del Mercado")

and ("Volumen") volume (Figure 8b). Once the form is completed, the user can review the backtesting created (Figure 8c) for each of the two sets of financial assets formed. Then, the management is done in the same way as in the previous case (view, execute, save, publish or delete).

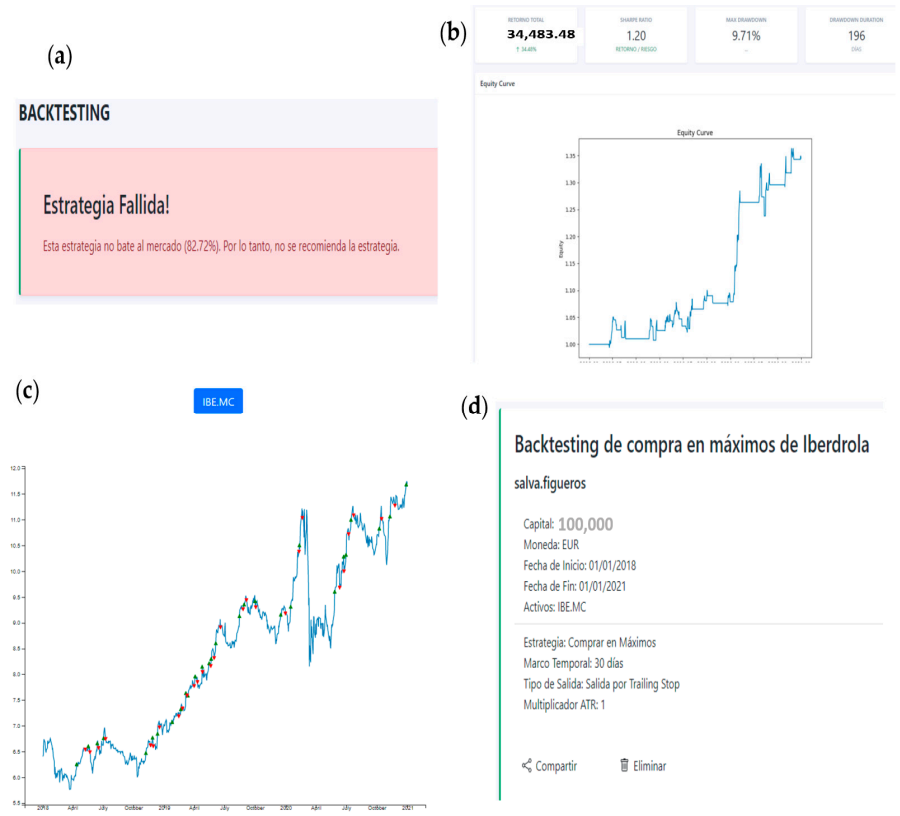


Figure 7. (**a**) Alert result backtesting. (**b**) View backtesting results. (**c**) Chart view of trading signals. (**d**) Share and delete buttons backtesting.

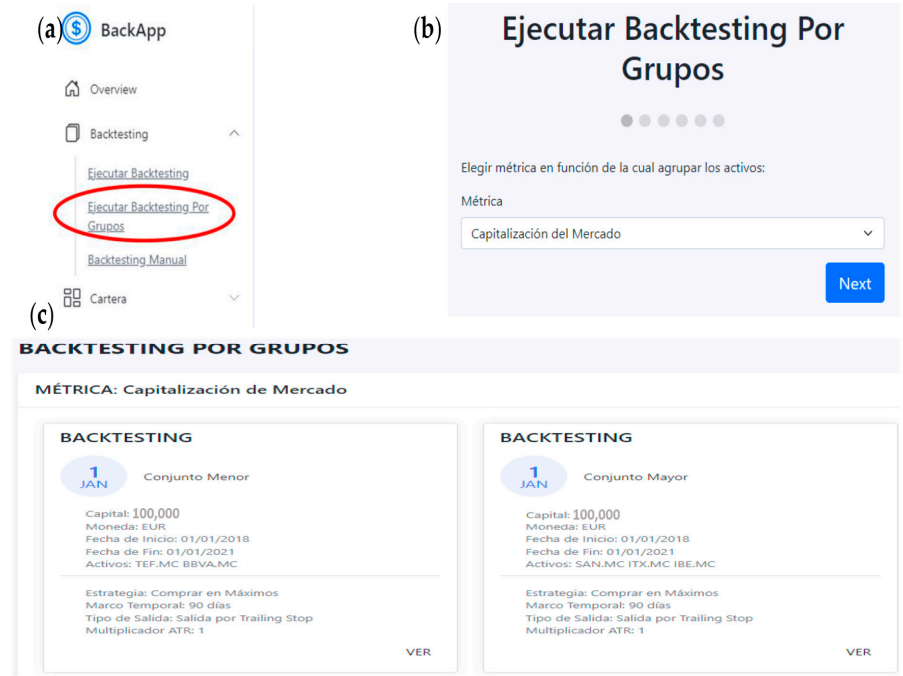


Figure 8. (**a**) Form run backtesting by groups". (**b**) Form run backtesting by groups". (**c**) View backtesting by groups.

5.3. Portfolio Management

This functionality allows the creation and maintenance of financial portfolios. The user accesses by clicking on the create portfolio button ("Crear cartera") within the portfolio list section of the Dashboard or using the navigation bar by clicking on the portfolio button in the side navigation bar (this opens a drop-down menu and the user clicks on the option create portfolio (Figure 9a). Next, a form is shown where the portfolio data are filled in (Figure 9b): name ("Nombre Cartera"), capital("Capital"), currency ("Moneda"), and financial assets ("Elegir los activos financieros que van a componer la cartera") of the portfolio. Once the portfolio is created, the user can access its most relevant statistics and visually consult the losses or gains, the market value ("Valor de Mercado") of the portfolio, the capital ("Capital") that it has in its account, and the equity (market value plus capital). In addition, it also has a graph that represents the distribution of the assets ("Distribución activos") in the portfolio (Figure 9c).

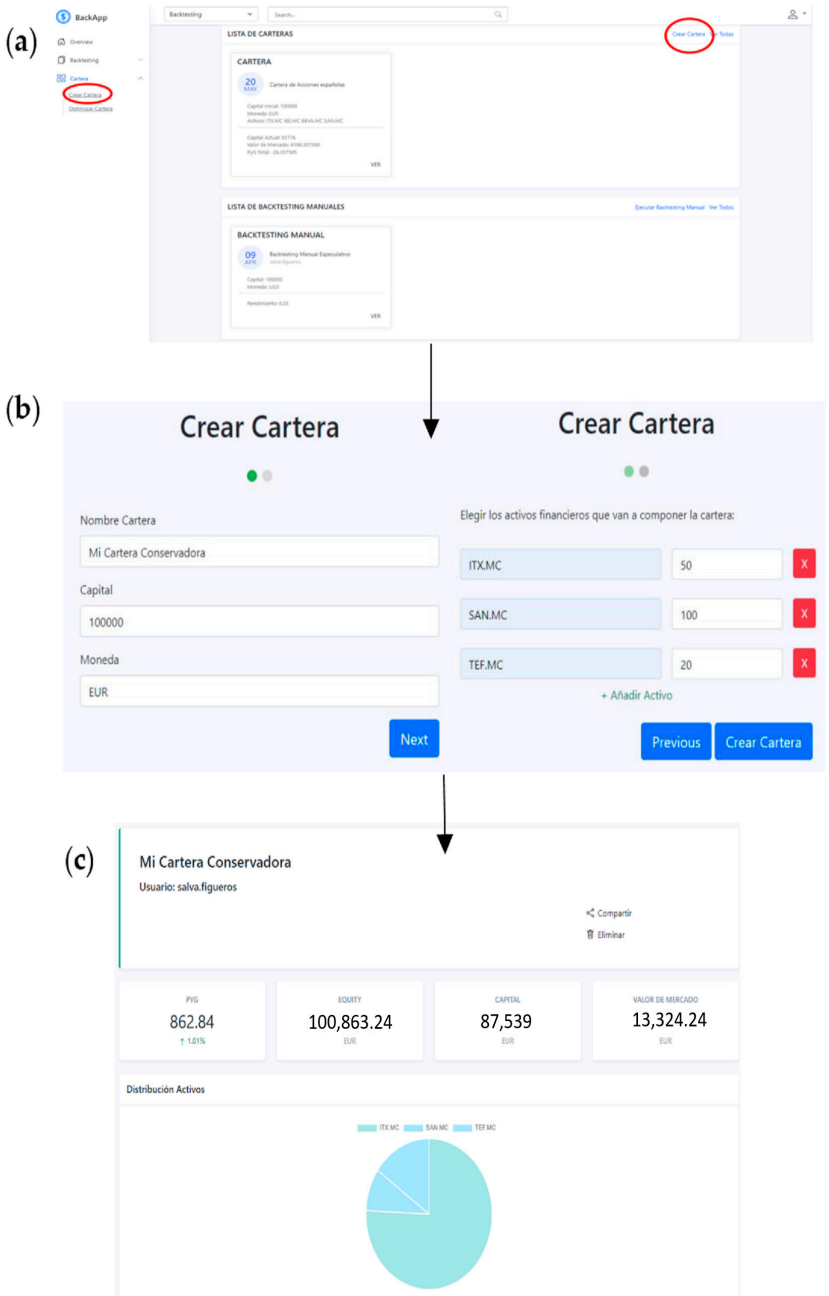


Figure 9. (**a**) Select create portfolio. (**b**) View create portfolio form. (**c**) Portfolio view.

To share or delete the portfolio, the user clicks on the buttons that appear in the upper header on the right of the interface. In addition, the user also has access to a set of statistics related to the positions and transactions in the portfolio.

The modification of the portfolio consists of the purchase or sale of contracts for any of the financial assets that make up said portfolio. To do this, there is a form at the top where the user indicates the ticker (Ticker), the type of operation (buy or sell), and the number of contracts to buy or sell (N°). For example (Figure 10), if the user buys 100 more Inditex shares, then the user must select the buy option, enter 100 in the next field, and click the send button (Enviar).

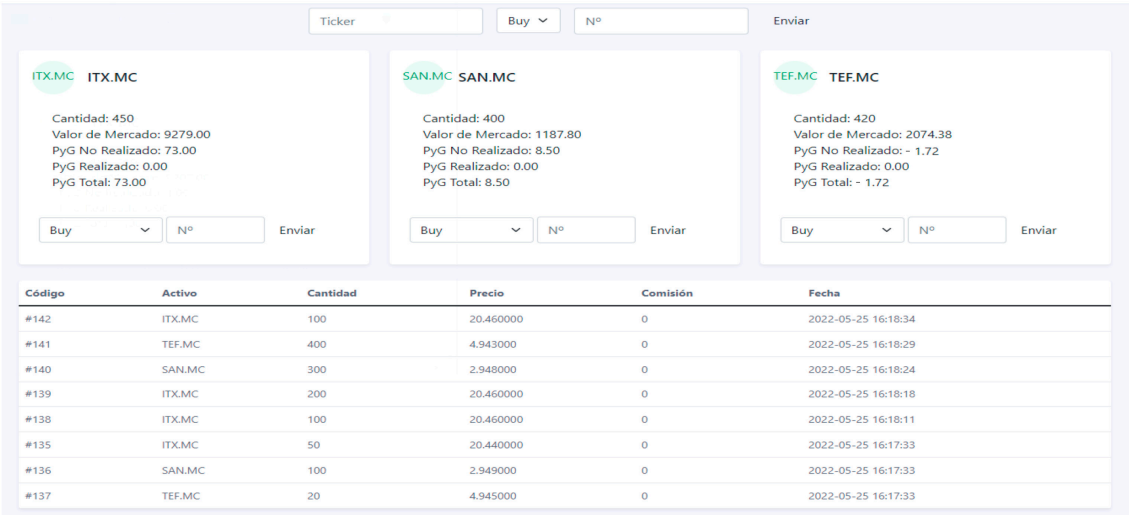


Figure 10. Modify portfolio view.

As for the optimization of portfolios, the process is very similar to that of portfolio creation. It is accessed from the side navigation bar by clicking on optimize portfolio once the menu associated with the portfolio option is displayed (Figure 11a). Next, the user completes a form very similar to the previous one for the creation of portfolios: the capital("Capital"), the currency("Moneda"), and the financial assets ("Elegir los activos financieros que van a componer la cartera optimizada") that are going to compose said portfolio, and the user clicks on the ("Optimizar") optimize button (Figure 11b).

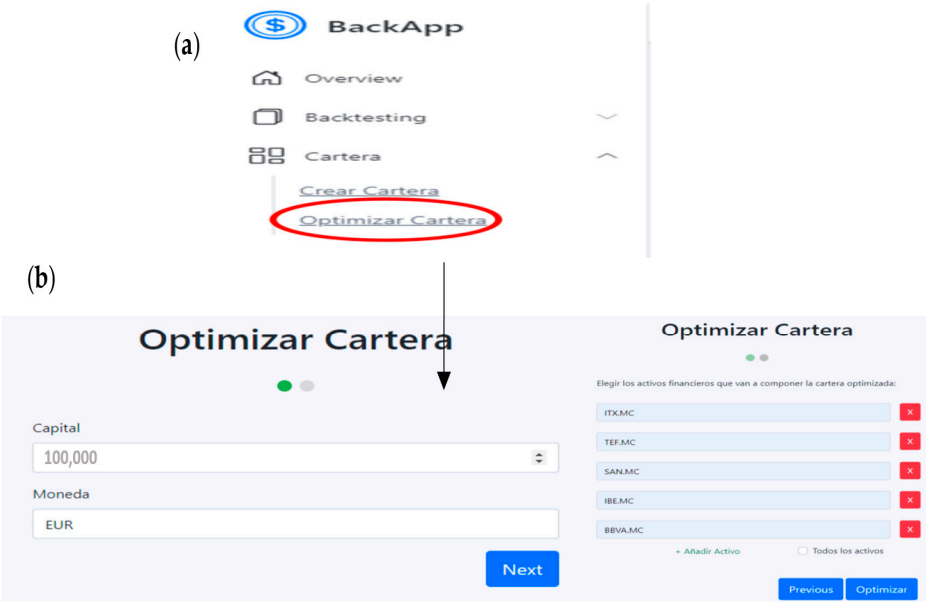


Figure 11. (**a**) Select optimize portfolio. (**b**) Portfolio optimize form view.

Once the form is submitted, the user will observe the results obtained from the optimization (Figure 12): The performance ("Rendimiento"), the volatility ("Frontera eficiente"), the weight distribution of the optimized portfolio ("Distribución de activos"), and the graph of the efficient frontier that represents each one of the portfolios that have been generated during the execution of the algorithm (the abscissa axis represents the standard deviation of the portfolios (volatility) and the ordinate axis the daily performance of these).

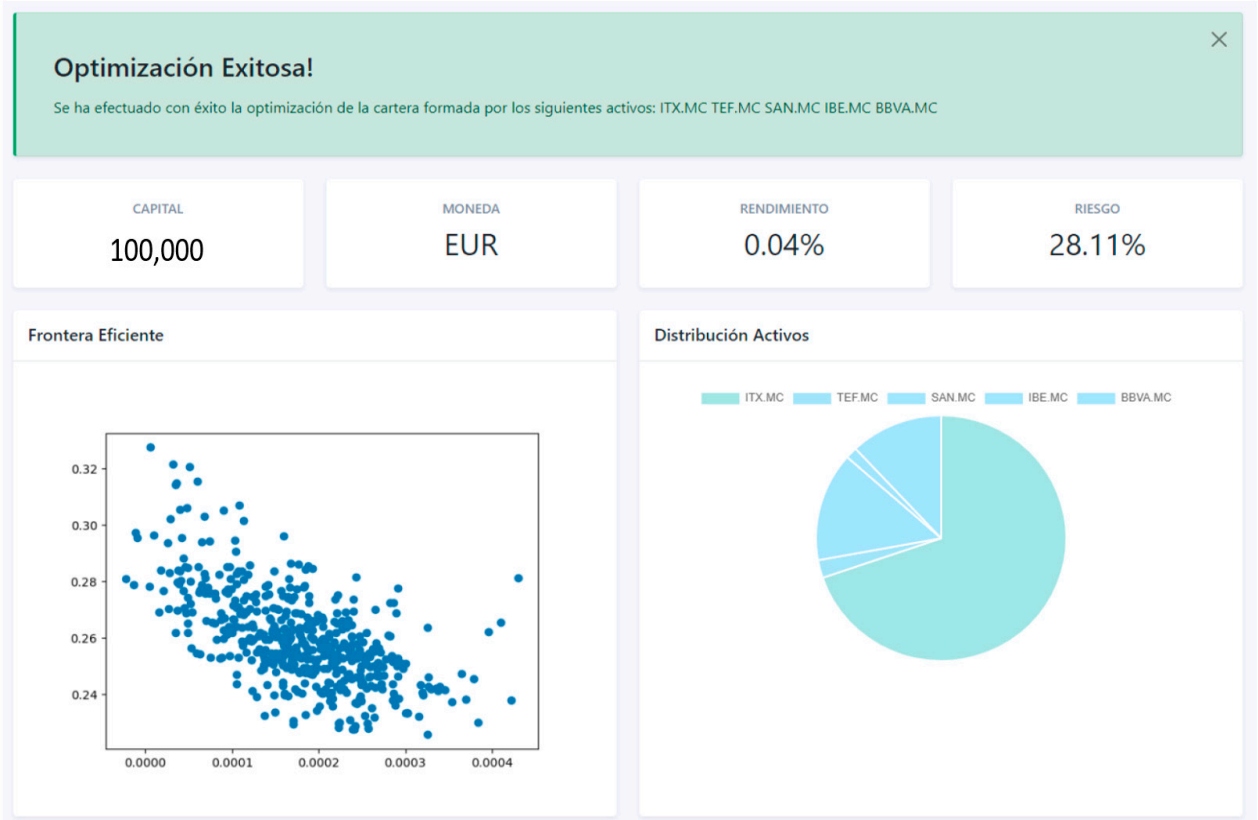


Figure 12. Portfolio optimization view.

5.4. Other Trading Services

Two additional trading services have been implemented. First, the user can run manual backtesting. To access this functionality, the user can click on the run manual backtesting button within the manual backtesting list ("Lista de Backtesting manuals") section of the dashboard, or use the navigation bar and click on the backtesting button on the toolbar lateral navigation (this will open a drop-down menu, and the user must click on the manual backtesting option) (Figure 13a). Next, the user completes a form whose fields refer to the capital "Capital") of the portfolio to be configured and the currency ("Moneda") in which said portfolio is denominated (Figure 13b), and an interface is generated in which the user enters the profits("Añadir beneficio") and losses("Añadir pérdidas") to be recorded and the associated date (Figure 13c). Once the data have been entered, the system updates all the associated statistics and displays them automatically. Finally, if the user wants to save said manual backtesting for later recovery, then the user must click on the green Save button ("Guardar") and enter a name for the manual backtesting (Figure 13d).

The other trading service is the trading journal, where the user also has the opportunity to write comments. To write, the user accesses the dashboard by clicking on the write button in the trading journal section (Figure 14a). Next, a form is shown where the user will write the text and the date, automatically showing a view with the text written by the user (Figure 14b).

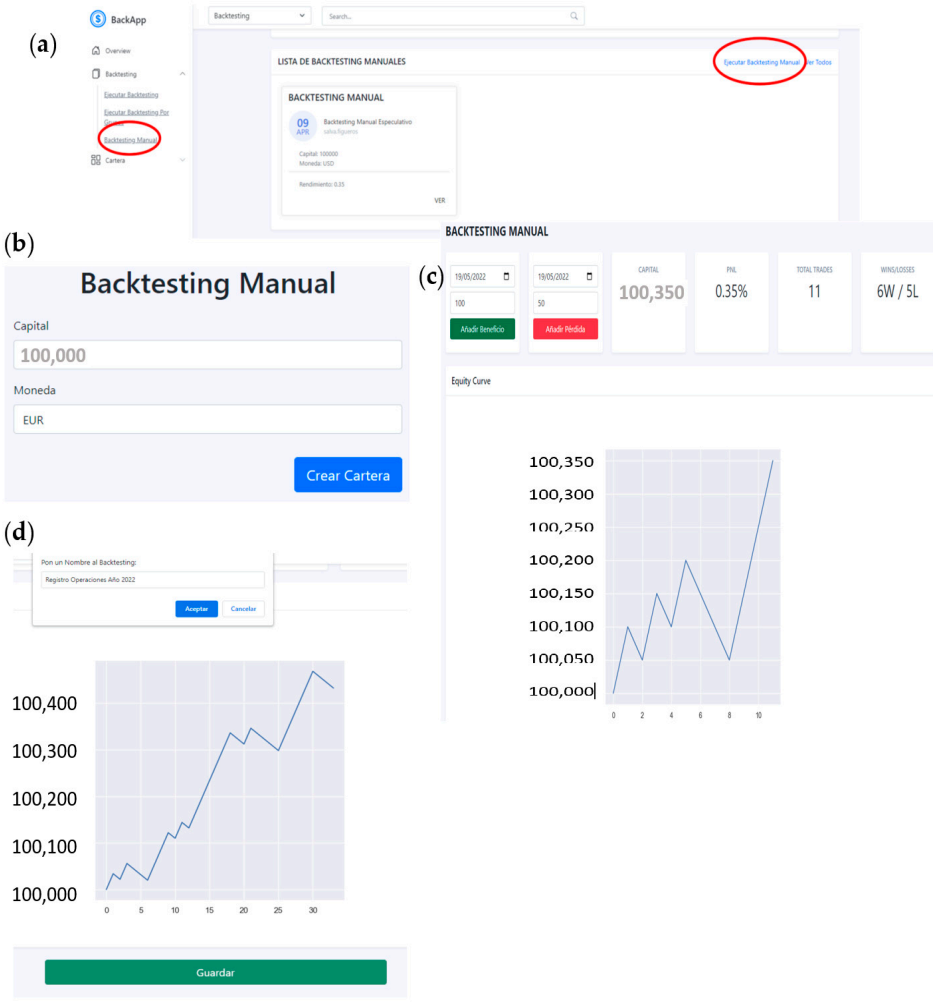


Figure 13. (**a**) Select backtesting manual. (**b**) Form view manual backtesting. (**c**) Manual backtesting view. (**d**) View save backtesting.

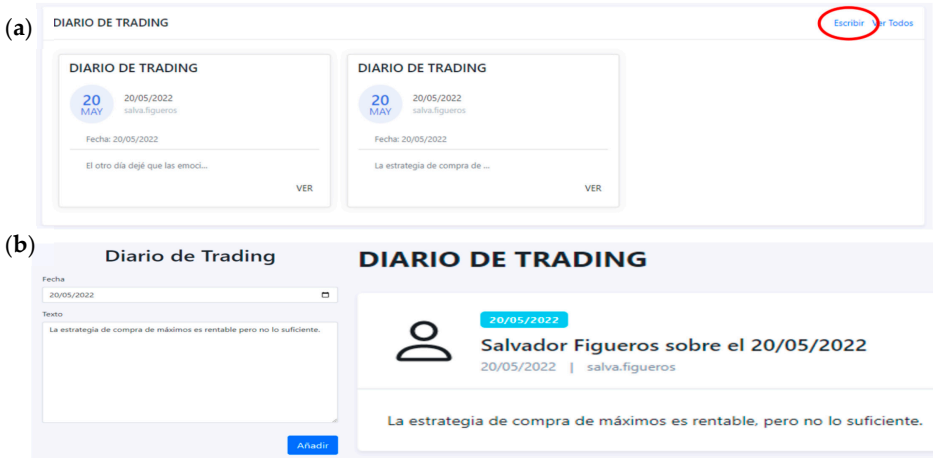


Figure 14. (**a**) Select write in trading journal. (**b**) Trading journal view.

5.5. Historical Data Management

The system administrator can upload historical data on the prices of financial assets to the database. To do this, the administrator clicks on the upload historical data (Figure 15a) button ("Subir Datos Históricos") that appears in the side navigation bar. Next, the user must choose whether to upload data on shares or futures (Figure 15b) and complete a

form indicating the data to upload to the system (Figure 15c): the ticker of the financial asset(“Ticker”), the name of the company(“Nombre completo”), the market (“Mercado”)in which it is listed, stock currency (“Moneda”), and data dates (“Fecha de inicio”, “Fecha de fin”). Observed thatname, market, and currency should be filled in only if no historical data are been previously uploaded for that asset. After the form is submitted, the system fetches the stock price data from Yahoo! Finance, and a price chart view is displayed for confirmation (Figure 15d). The system detects whether the data that the administrator wants to upload already exist or not and only inserts those that do not exist into the database.

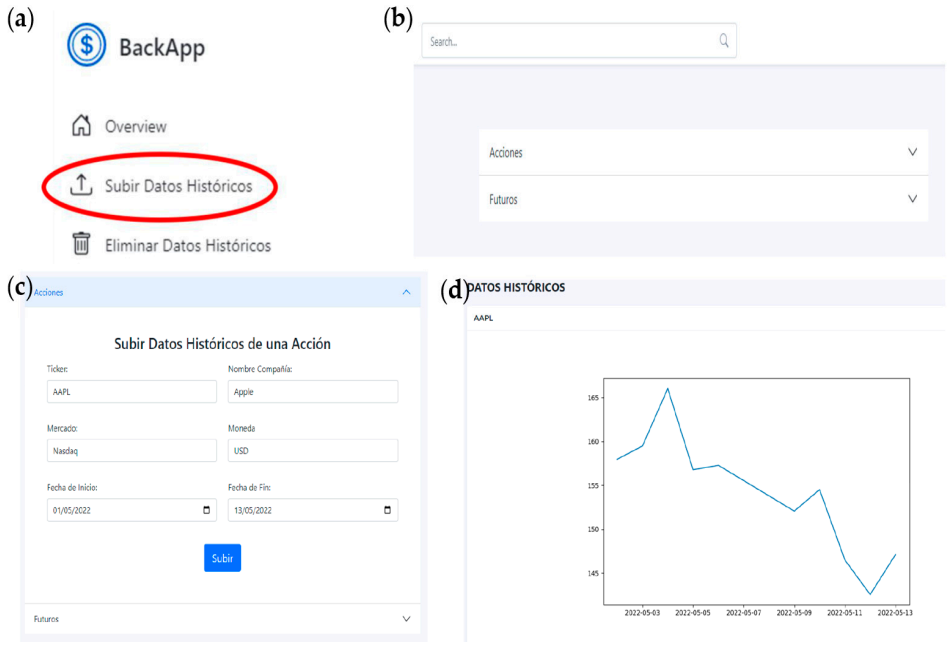


Figure 15. (**a**) Select upload historical data”. (**b**) View stocks or futures. (**c**) View form to upload historical data. (**d**) Confirmation view after uploading historical data.

It is also possible to delete historical data that have been previously uploaded. To do this, the administrator must click on the delete historical data button(“Eliminar Datos Históricos”) (Figure 16a) on the side navigation bar and fill out a form with the following data: the ticker of the financial asset in the Ticker box(Figure 16b) and the dates of the data (“Fecha de inicio” and “Fecha final”) to be deleted(Figure 16c). It is also possible to delete all ticker data by checking the select all box. If the data entered by the user are correct, the system will proceed with the deletion of the indicated data and, later, it will direct the user to the main menu.

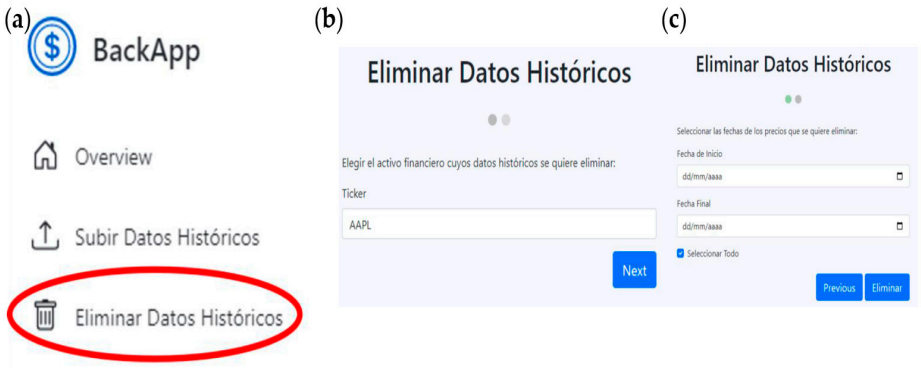


Figure 16. (**a**) Select delete historical data, (**b**) Form view delete historical data (Ticker), (**c**) Form view delete historical data (Dates of the data).

6. Conclusions and Future Work

In this article, a web application has been presented that allows the execution of the backtesting of a strategy based on the purchase or sale of a financial asset at the maximum or minimum price. The implementation of this functionality has been done following an event-driven design, which allows backtesting to be carried out with sufficient guarantees and without falling into anticipation bias. Furthermore, it allows great scalability in terms of the number of trading strategies that can be added to the system. Likewise, other functionalities have been implemented that allow the user to monitor their operations as a trader: portfolio management, manual backtesting, the trading journal, and the administration of historical data.

Existing backtesting applications are often difficult to use, offer a limited variety of simulations, and are not very intuitive. This can make it difficult for traders to test different trading strategies and settings, and find the information they need. The developed application improves these limitations in a number of ways. First, the application is very easy to use, even for people with no experience in backtesting. The application's interface is clear and concise, and the instructions are easy to follow. Second, the application offers a variety of different types of backtesting simulations. This allows traders to test different trading strategies and settings in different market conditions. Third, the application offers a variety of different types of financial operations. This allows traders to test different trading strategies in different markets. Finally, the application is very intuitive. Users can easily find the information they need and perform the tasks they need.

Overall, the developed application is a valuable tool for traders. It is easy to use, offers a variety of different types of backtesting simulations and financial operations, and is very intuitive.

Some observations are necessary. First of all, regarding the backtesting methodology. This methodology presents some limitations such as the following. In the first place, the availability of a broad set of detailed historical data is necessary so that the conclusions obtained can be used to learn strategies and search for behavior patterns. Another limitation is the inability to model strategies that could affect historical prices. Another problem is confirmation bias, that is, the fact that people are more likely to accept information if it seems to support what they already believe or expect. Thus, one can unconsciously adjust the data or adapt the study parameters to create positive results. In this case, the statistical history is misleading and does not represent the true performance of a strategy or system. The functioning of the stock market depends on some parameters, in particular the bid-ask spreads and slippage. Slippage is the difference between the price of a desired order and the price at which the order was actually executed in the market. In this sense, backtesting cannot take this variable into account, so the market entry and exit values may be inaccurate. In addition, bid and ask spreads vary considerably as market conditions evolve in terms of liquidity and volatility.

Secondly, on the objectives of the work. The objective of the work is to create an application where the user can test different investment strategies with historical data so that, in a controlled environment, they can see if their strategies are good or bad, or where they fail. In this way, you will hone your investment skills and hopefully improve your investments in a real portfolio of securities. It is for this reason that it is claimed that the application will help improve the portfolio.

The set of tools implemented are very useful for investing in financial markets; however, the system can be improved in several aspects:

- Add more strategies: It consists of using event-driven design to code new strategies and add them so that the user can execute them.
- Live trading: It consists of using event-driven design to implement live trading.
- Backtesting efficiency: It improves the implementation of functions that perform backtesting by using a multithreaded or multithreaded implementation.
- Portfolio optimization: The mean variance analysis method has been used in the application for portfolio optimization. In this sense, it is proposed as a line of future

work to use other optimization methods (HRP, mCVAR...) so that the user can choose which one to use.
- Add more types of financial assets: Currently, the application only allows you to manage stocks and futures. It would be interesting to manage new asset classes such as bonds, cryptocurrencies, or currencies (forex).
- Application of machine learning: One line of future work is to implement functions that use machine learning techniques for various tasks such as risk management or efficient configuration of a trading strategy.

Finally, comment that the main ideas for continuing the research focus on the use of machine learning techniques. To achieve more realistic results and discover trends or patterns, techniques are required that allow huge amounts of data to be processed efficiently and so that the algorithms can learn from the data. That is why the use of neural networks is proposed, and more specifically, convolutional neural networks, since these allow automatically extracting characteristics that will later be used for classification and pattern search. Perhaps a third way is a mixed solution between the use of backtesting techniques and machine learning algorithms. So algorithms are used to detect patterns, and backtesting is used to validate the patterns that have been found.

Funding: This research received external funding: Project number PID2021-123048NB-I00 from the Spanish Ministry of Science and Innovation.

Informed Consent Statement: Not applicable.

Acknowledgments: I would like to thank Salvador Figueros Sentana for developing the analyses.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Shah, D.; Isah, H.; Zulkernine, F. Stock market analysis: A review and taxonomy of prediction techniques. *Int. J. Financ. Stud.* **2019**, *7*, 26. [CrossRef]
2. Edwards, R.D.; Magee, J.; Bassetti, W.C. *Technical Analysis of Stock Trends*; CRC Press: Boca Raton, FL, USA, 2018.
3. Zhang, K.; Zhong, G.; Dong, J.; Wang, S.; Wang, Y. Stock market prediction based on generative adversarial network. *Procedia Comput. Sci.* **2019**, *147*, 400–406. [CrossRef]
4. Das, D.; Kannadhasan, M.; Bhattacharyya, M. Do the emerging stock markets react to international economic policy uncertainty, geopolitical risk and financial stress alike? *North Am. J. Econ. Financ.* **2019**, *48*, 1–19. [CrossRef]
5. Yu, D.; Huang, D.; Chen, L. Stock return predictability and cyclical movements in valuation ratios. *J. Empir. Financ.* **2023**, *72*, 36–53. [CrossRef]
6. Huang, D.; Li, Y.; Wang, X.; Zhong, Z.K. Does the Federal Open Market Committee cycle affect credit risk? *Financ. Management.* **2022**, *51*, 143–167. [CrossRef]
7. Chen, C.W.; Lin, T.Y.; Huang, T.Y. Incorporating volatility in tolerance intervals for pair-trading strategy and backtesting. *J. Risk Model Valid.* **2019**, *5*, 10–25. [CrossRef]
8. Chauhan, R.; Nistor, M.S.; Bein, D.; Pickl, S.; Bein, W. Stock Backtesting Engine Using Pairs Trading. In Proceedings of the ITNG 2022 19th International Conference on Information Technology-New Generations, Las Vegas, NA, USA, 10–13 April 2022; pp. 245–253.
9. Rosenberg, E.; Alberg, D. An Open Source Finance System for Stocks Backtesting Trade Strategies. *Int. J. Open Source Softw. Process.* **2021**, *12*, 52–65. [CrossRef]
10. Schumann, E. Backtesting. In *Forthcoming in "Numerical Methods and Optimization in Finance*, 2nd ed.; Gilli, M., Maringer, D., Schumann, E., Eds.; SSRN: New York, NY, USA, 2018.
11. Arnott, R.; Harvey, C.R.; Markowitz, H. A backtesting protocol in the era of machine learning. *J. Financ. Data Sci.* **2019**, *1*, 64–74. [CrossRef]
12. Özyeşil, M. Comparison of Technical and Fundamental Analysis Trading Disciplines on Portfoilo Performance: Short and Long Term Backtest Analysis on Borsa İstanbul National Stock Indices. *J. Contemp. Res. Bus. Econ. Financ.* **2021**, *3*, 128–143. [CrossRef]
13. Fiorucci, J.A.; Silva, G.N.; Barboza, F. RTS: Expert advisor for reaction trend system. *Softw. Impacts* **2022**, *13*, 100331. [CrossRef]
14. Arora, V.; Patel, K.P. Role of Technical Analysis Tools for Trading Decision. *Int. J. Econ. Perspect.* **2022**, *16*, 28–32.
15. Yu, D.; Huang, D. Cross-sectional uncertainty and expected stock returns. *J. Empir. Financ.* **2023**, *72*, 321–340. [CrossRef]
16. Arslan, M.E.; Kirci, P. Makine Öğrenmesi İle Borsa Analizi. *Avrupa Bilim Teknol. Derg.* **2021**, *28*, 1117–1120.
17. Gandhmal, D.P.; Kumar, K. Systematic analysis and review of stock market prediction techniques. *Comput. Sci. Rev.* **2019**, *34*, 100190. [CrossRef]

18. Leung, M.; Li, Y.; Pantelous, A.A.; Vigne, S.A. Bayesian Value-at-Risk backtesting: The case of annuity pricing. *Eur. J. Oper. Res.* **2021**, *293*, 786–801. [CrossRef]
19. Sangweni, X.Z. Empirical Evaluation of Existing Backtesting Techniques for Market Risk Models. Master's Thesis, University of Johannesburg, Johannesburg, South Africa, 2019.
20. Tayalı, S.T. A novel backtesting methodology for clustering in mean–variance portfolio optimization. *Knowl.-Based Syst.* **2020**, *209*, 106454. [CrossRef]
21. Su, Q.; Qin, Z.; Peng, L.; Qin, G. Efficiently Backtesting Conditional Value-at-Risk and Conditional Expected Shortfall. *J. Am. Stat. Assoc.* **2021**, *116*, 2041–2052. [CrossRef]
22. Wu, D.; Wang, X.; Wu, S. Construction of stock portfolios based on k-means clustering of continuous trend features. *Knowl.-Based Syst.* **2022**, *252*, 109358. [CrossRef]
23. Naik, M.J.; Albuquerque, A.L. Hybrid optimization search-based ensemble model for portfolio optimization and return prediction in business investment. *Prog. Artif. Intell.* **2022**, *11*, 315–331. [CrossRef]
24. Chen, W.; Zhang, H.; Jia, L. A novel two-stage method for well-diversified portfolio construction based on stock return prediction using machine learning. *N. Am. J. Econ. Financ.* **2022**, *63*, 101818. [CrossRef]
25. Khan, A.Z.; Mehlawat, M.K. Dynamic portfolio optimization using technical analysis-based clustering. *Int. J. Intell. Syst.* **2022**, *37*, 34–50. [CrossRef]
26. Miftahurrohmah, B.; Wulandari, C.; Dharmawan, Y.S. Investment Modelling Using Value at Risk Bayesian Mixture Modelling Approach and Backtesting to Assess Stock Risk. *J. Inf. Syst. Eng. Bus. Intell.* **2021**, *7*, 11–21. [CrossRef]
27. Zhang, H.; Liang, Y.; Su, H.; Liu, C. Event-driven guaranteed cost control design for nonlinear systems with actuator faults via reinforcement learning algorithm. *IEEE Trans. Syst. Man Cybern. Syst.* **2019**, *50*, 4135–4150. [CrossRef]
28. Chen, Y.; Lu, Z.; Yang, X. The Design and Implementation of a High-performance Portfolio Optimization Platform. In Proceedings of the 2020 IEEE 23rd International Conference on Computational Science and Engineering (CSE), Guangzhou, China, 29 December 2020–1 January 2021; pp. 1–7.
29. Mylnikov, G. Volatility Targeting: It's Complicated! *J. Portf. Manag.* **2021**, *47*, 57–74. [CrossRef]
30. Zhang, Y.; Nadarajah, S. A review of backtesting for value at risk. *Commun. Stat. -Theory Methods* **2018**, *47*, 3616–3639. [CrossRef]