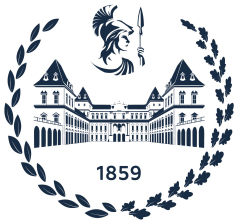
POLITECNICO DI TORINO



Master's Degree course in Computer Engineering

Master's Degree Thesis

Development of a social trading application

Supervisor
Prof. Alessandro Fiori

Candidate
Alexander Carlo Gustavo Spira

October 2022

Summary

The thesis is about the development of social features for a web platform used for analyzing financial derivatives. Financial derivatives are a complex concept that requires a lot of knowledge and experience in the financial market. They are complex since are financial instruments that allow people to either speculate or hedge a position by giving the buyer the possibility to purchase or sell a certain quantity of a specific underlying asset at a fixed price at some point in the future. They are used essentially as a form of leverage allowing an investor to make a bet on a stock without having to purchase or sell the shares directly, but by paying a small sum to the seller. In the application different types of derivatives can be combined in a "strategy" depending on the intention of the investor.

Since strategies can become very complicated and dynamic, the objective of the thesis was to allow people to have a look at other peoples' strategies in order to learn, take inspiration and grow their portfolio. Users should be able to make their strategies open to the public, share them inside the application to other users or inside communities called "Clubs", and outside the application on social media. Clubs are small groups of people in which administrators can add strategies that all the members can view.

The sharing features of the application allow users to receive or give financial knowledge and tactics from or to other people.

Acknowledgements

First of all I want to thank the Professor Alessandro Fiori that welcomed me in this project and guided me by giving advice on how to advance and improve. I would also thank the support of Dr. Marco Rossi Bassignana which helped understanding what a professional trader would have looked for in a financial application.

I would also like to thank my family for the encouragement and support through all my studies.

Last but not least, my friends. We have always supported and helped each other and i would like to thank them for a cherished time spent together.

Contents

List of Figures

List of Tables

Chapter 1

Introduction of the application

This thesis is about implementing sharing features in a web platform that allows to simulate financial options strategies. Strategies are very powerful financial instruments that in recent years have gained lot of attention. A **strategy** is the simultaneous, and often mixed, buying or selling of one or more derivatives of the same type that differ in one or more variables. Since now banks and online broker firms have developed reliable trading systems that enable fast market orders. The problem is that these complex systems are used mostly for task execution, meanwhile other specific external platforms are being used for the analysis and forecasting portions.

These tools, that have different instruments to analyse strategies with visual elements, are difficult to find, particularly in the futures and options markets. Often they have high prices and if they are cheap, they are old and out of date, with user interfaces that don't make new users comfortable and instead, make the whole learning process more challenging and intimidating.

Therefore the concept for this application was proposed. The objective was to deliver a web-based platform, user friendly and most importantly with every tool necessary to analyze the market. The application enables both experienced and inexperienced traders to evaluate the markets and display statistical findings on their trend through an easy and intuitive interface.

For this thesis, the idea was to expand the capabilities of the application by giving the user the possibility to make their strategies open to the public and share them. In addition to that, the user should be able to create communities called Clubs in which share strategies that will be accessible by the members.
This idea originated by the trader need of observing what other traders are doing in order to keep up with trends and have different point of view on the evaluation of the market.

1.1 Overview

All of the key instruments and design decisions needed to get to the established goal will be discussed in this document.

The next chapter, Chapter 2, will give readers a deeper understanding of the financial sector by focusing on the major providers of futures and options contracts, such as CME and CBOE for the American markets and EUREX for the European one. It will explain what futures and options contracts are, highlighting their features and context of use along with the relevant benefits and achievable operational tactics. This stage gives the appropriate background information for better comprehend what strategies are made of.

Afterward, we continue with the Chapter 3 that will describe what are the basic features of a social application and the meaning of social trading.

Chapter 4 will describe how to make the usage of the application user friendly and more engaging through the use of micro-interactions. In general this chapter is about design principles and how to apply them in a web application.

We next go on to the Chapter 5 in which we will cover the architecture and design of the system on which runs the application. Frameworks, libraries and programs will be described with their advantages and use in this project.

The Chapter 6 is about the implementation of the back-end with the methods used for accessing it. In particular it concentrates on the non-relational database with the definition of its models and collections.

In Chapter 7 we'll have a look at the user interface, how the data is managed and at the many application functionalities. It will show how user actions are managed and how the response is delivered.

Chapter 8 shows some use cases that are possible with the new functionalities added.

Finally the last chapter is about the conclusions and the future works.

1.2 Description of the starting point

The platform first features will be summarized here in order to understand the starting point. It was divided in three sections: Markets, Strategies and Portfolio.

1. The first one, `Markets`, allows the user to look for a market in order to analyze it through different types of graphs. One market tab contains general information of the market selected, different charts that can be explored, a part dedicated to the strategies created by the user in that market with their positions (open and closed), a part for futures and one for chains with options put and calls, both with relative prices, volumes and open interests.

2. `Strategies` shows all the strategies that the user has created and lets the user modify the positions of an open strategy. The strategy shows a graph with the payoff, its group of markets and the currently open positions.

3. The `Portfolio` part shows the balance of the user's account, a graph with the trend of the balance and a table with all the strategies with their costs and profits.

The primary functions are offered to the user through graphs, with the option to store them on their own computer through the download of the image. The decision to utilize graphs was made to give the user a clear and understandable perspective of the present state of the market and of the strategy created.

For what concerns the data collection, for our purpose using a Web Service [1] was convenient for its type of data and simplicity of usage. In particular the convenience is because of guidelines called REST API [2] make using online resources, that the service exposes, extremely simpler. This is why the focus of the research has been on locating reliable APIs that can track all financial summary, stock, quotes and options.

The data was collected from these exchanges: Chicago Board Options Exchange (CBOE), Chicago Mercantile Exchange (CME) and EUREX Exchange (EUREX). Some of the data of the EUREX had to be scraped [3] from dedicated HTML pages but other then that the rest was from free public APIs that have a 15 minutes of delay from real time data.

The underlying of the option belongs to different categories: stocks, equity indices, commodity futures, index futures and bond futures.

From here, as said in the introduction, the thesis will develop new features and components. The strategies will be modified and extended with social features and the clubs will be build from scratch.

[1] A service running on a computer device, listening for requests at a particular port over a network, serving web documents (HTML, JSON, XML, images), and creating web applications services, which serve in solving specific domain problems over the Web.

[2] An application programming interface (API) that queries data, parses responses, and sends instructions between one software platform and another.

[3] Automated process implemented using a bot or web crawler used for extracting data from websites.

Chapter 2

About Option Trading

This chapter will first attempt to give an overview of the actors and components involved in trading and then provide more in-depth understanding of the financial instruments that are available on the platform.

First of all trading is referred as buying and selling financial assets such as shares, options, currencies, and futures, whose values are listed during the opening hours of international exchanges. A trade transaction requires a seller of goods and services as well as a buyer. Various intermediaries such as banks and financial institutions can facilitate these transactions by financing the trade.

A trader is defined as a person who buys and sells financial instruments with the aim of making a profit. Millions of businesses, people, institutions, and even governments trade simultaneously and continuously on the financial markets. While some traders maintain a more specialized portfolio, others concentrate to just a single instrument or asset class. Also before entering a trade, some traders conduct extensive research, while others scan charts and keep an eye out for trends. However, every transaction has one thing in common: risk. The concept of risk is fundamental to all forms of financial trading. Balancing possible profit against risk is essential regardless of the instrument being traded, the person doing the trading, or the location of the trade.

But what is the difference between a trader and an investor? The key distinction is in how long each party keeps the asset; investors typically have a long-term time horizon, whereas traders typically hold assets for shorter amounts of time in order to profit from short-term trends.

Now let's see some more specific definition of the words cited before and some that will be encountered in the following chapters:

Share: *"A certificate giving the person or company listed a portion of ownership in a stock, mutual fund, or some other investment vehicle. A share is the smallest unit of ownership. They may be bought or sold on or off an exchange."* [1]

Financial Instrument: *"Any document with monetary value. Examples include cash and cash equivalents, but also securities such as bonds and stocks which have value and*

may be traded in exchange for money." [2]

Stock: *"A portion of ownership in a corporation. The holder of a stock is entitled to the company's earnings and is responsible for its risk for the portion of the company that each stock represents. Stock may be bought or sold, usually, though not always, in the context of a securities exchange. It is important to note that a single share of a stock usually represents only a tiny amount of ownership, and, therefore, most stocks are traded in batches of 100."* [3]

Bond: it is debt that is put out for a period of more than one year. Governments, companies and institutions sell bonds. The act of buying bonds means that the investor is lending money, but the bond is repaid with the principal amount at the specified time agreed in the buying process. Also the bond holder usually receive an interest periodically. [4]

Commodity: is a physical good like metal, food or mineral that is yet to be processed. They are usually sold via futures contracts. [5]

2.1 The Markets

A financial market is any location or system that gives buyers and sellers the ability to trade financial assets, such as shares, bonds, derivatives and various international currencies. They make the connection easier between those who have capital to invest with those who need capital. [6]

There are different types of markets with different characteristics and different purposes. Now we'll have a look at the main ones.

Capital markets which consists of: `Stock markets` that allows to buy and sell shares or common stocks which are the ownership claim on businesses, and `Bond markets` where parties can issue new debt or buy and sell debt securities. With bonds investors can buy those securities in order to loan money for a defined period of time for a established interest rate.

The **Commodity markets** are where products of primary economic sector are traded.

Money markets provide debt financing and investment under the form of short-term funds that generally are for a period of a year or less.

The **Derivatives market** offers instruments that draw their value from other, such as bonds, indices, equities, and other form of assets. These financial instruments can be futures contracts and options.

The **Foreign exchange markets** is a global decentralized market for trading currencies. It is the one responsible for the foreign exchange rates for every currency. It offers buying and selling but also exchanging currencies at current or determined prices.

Finally the **Cryptocurrency markets** are decentralized and let you trade digital currencies, assets and financial technologies.

Now let's see what our platform has to offer in order as they are presented in the *Markets* page.

For the Stock market it has the top American Companies with high capitalization like: *Apple Inc., Microsoft Corp., Tesla Inc., Google Inc., Amazon, Facebook Inc. e Intel Corp.*

For what concerns Indexes there are:

- *S&P 500 or Standard & Poor's 500 Index*: which is a stock market index tracking the stock performance of 500 large companies listed on exchanges in the United States.
- *S&P 500 Volatility Index*: it is a measurement of the volatility of the stock market based on the options of the S&P 500 index. In contrast to other markets indices this is traded through derivative contracts rather than being purchased or sold directly. The quoted price of each option represents the market's estimate of volatility for the subsequent 30 days.
- *DAX or Deutscher Aktien-Index*: is similar to the S&P 500 but tracks the performance of the 40 largest companies trading on the Frankfurt Stock Exchange.
- *EuroStoxx50*: is a stock index of Eurozone stocks and it is composed of 50 stocks from 11 countries in the Eurozone. The index futures and options are among the most liquid products in Europe and the world.

After there are the futures contracts which involve stock indices, bond and commodities. Other than the ones cited before there are also:

- *Russel 2000*: which is a small-cap stock market index that makes up the smallest 2,000 stocks in the Russell 3000 Index. It is by far the most common benchmark for mutual funds that identify themselves as "small-cap".
- *Nasdaq-100*: is a stock market index made up of 102 equity securities issued by 101 of the largest non-financial companies listed on the Nasdaq stock exchange. The stocks' weights in the index are based on their market capitalizations, with certain rules capping the influence of the largest components.
- *Vstoxx*: is based on real-time prices of options on the Euro Stoxx 50 index and reflects the market expectations on volatility.

Finally referring to Commodities there are only two which are gold and crude oil.

2.2 Exchanges

"An exchange is a marketplace where securities, commodities, derivatives and other financial instruments are traded. The core function of an exchange is to ensure fair and orderly trading and the efficient dissemination of price information for any securities trading on that exchange. Exchanges give companies, governments, and other groups a platform from which to sell securities to the investing public." [7]
It might be an actual place where traders congregate to conduct business or an online platform. For each business or entity wishing to list securities for trading on an exchange, there are particular listing requirements that must be met. The fundamental standards for stock exchanges include regular financial reporting, audited earnings reports, and minimum capital requirements. Some exchanges are more strict than others.

The ones present in the project are: CBOE, CME and EUREX.

CBOE (Chicago Board Options Exchange): provides trading in a variety of asset classes and regions; including options, futures, European and American equities, exchange-traded products, foreign exchange, and products that take into account several asset classes volatility. By value traded, it is both the largest stock exchange in Europe and the largest options exchange in the United States. The VIX index is the leading indicator of volatility in the equities market. This Index, which is based on current pricing for near-the-money options on the S&P 500 Index (SPX), aims to capture investors' perceptions of predicted stock market volatility over the next 30 days. [8]

CME (Chicago Mercantile Excange): is an organized exchange for the trading of futures and options. The CME trades futures, and in most cases options, in the sectors of agriculture, metals, stock indices, real estate, foreign exchange, energy and interest rates. [9]

EUREX (Eurex Excange): is one of the world's biggest marketplaces for futures and options.It primarily deals with European futures, but it also gives electronic access to traders connecting from over 700 locations worldwide. [10]

2.3 Derivatives

Derivatives are one of the three main categories of financial instruments besides equity (stocks or shares) and dept (bonds). A derivative is a contract which value depends on the performance of an underlying entity (*underlying*) that can be an asset, bond, index or interest rate. Forwards, futures, options, swaps, and variations of these, such as synthetic collateralized debt obligations and credit default swaps, are some of the more popular derivatives.

The contract consists on an agreement made between two or more parties who can trade over-the-counter (OTC)[1] or on an exchange. It has its own risk and can be used to trade a wide range of assets. It can be used to hedge a position to mitigate risk or assume risk by speculating.

The investor is not required to possess the underlying asset when using derivatives to speculate on the price movement of the asset. In fact, due to the fact that many derivative instruments are leveraged, only a little amount of capital is needed to own a big portion of the value of the underlying. The fact that they are leveraged is a good thing like increasing the amount of return with little money, but can have drawbacks like increasing the accumulation of losses.

Also, since the derivative has no intrinsic value because it comes from the underlying asset, its price is susceptible to the market sentiment regardless of what is happening with the price of the underlying asset. Supply and demand factors may cause a derivative's price and its liquidity to rise and fall.

Derivative items fall into two categories: "*lock*" and "*option.*"

- **Lock products** (such as futures, forwards, or swaps) obligate the parties to the terms of the contract from the beginning.
- **Option products** (such as stock options) give the holder the right but not the obligation to buy or sell the underlying asset or security at a particular price on or before the option's expiration date.

Now we'll dive down and have a look at what the platform offers: futures and options.

2.3.1 Futures

A futures contract, often known as a futures, is a agreement between two entities for trading of an item at a certain price at a later time. The key here is that the parties are required to follow through the promise to acquire or sell the underlying asset at the price stipulated. Futures contracts are used to manage their risk or make predictions about the price of an underlying asset.

A few words on how it works: the buyer opens a long position[2] and agrees to pay a price on a certain date for the underlying asset; in order to supply the underlying asset in

[1] Trading securities over-the-counter means doing it through a network of broker-dealers rather than on a centralized exchange.

[2] The purchase of an asset with the expectation it will increase in value.

return for a price and by an expiration date, the seller opens a short position[3]. So if the price of the underlying increases, the person who purchased the future gets money while the person who sold the future suffers a loss and vice versa if the price falls.

2.3.2 Options

An options contract is a financial instrument that resembles a futures contract with the important distinction that the buyer has the right, but not the obligation, to purchase or sell a certain quantity of a specific underlying asset at a fixed price at some point in the future. A contract's terms outline the underlying securities, the strike price[4] at which it can be traded, and the expiration date.

Like the future contracts they are used for hedging purposes or for speculation, and also options cost a fraction of the underlying. This means that, like the futures, also options are used as a form of leverage allowing an investor to make a bet on a stock without having to purchase or sell the shares directly. The option buyer gives the party selling the option a premium[5] in return for this benefit.

Options may be divided into two primary groups, American options and European options, depending on the type of exercise. Up to the expiration date, American options can be exercised whenever you choose, meanwhile European options are riskier than American options because they can only be exercised on the expiration date. For this reason European options are priced less than American options.
At this point we can have a look at the two types of options: call and put (Figure 2.1).

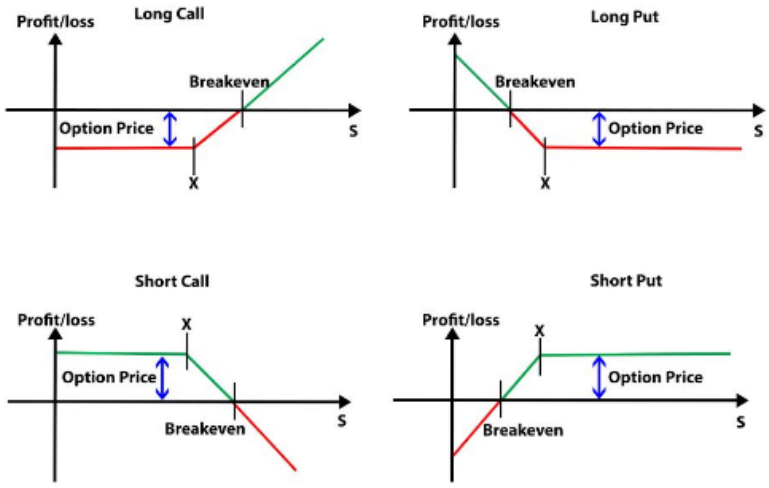


Figure 2.1: Types of options. Image taken from here.

[3]Strategy that speculates on the decline of the price.

[4]A strike price is a price at which an option can be exercised

[5]Option premium: current market price of an option contract.

Option Call

Financial contracts known as call options provide the buyer the right, but not the obligation, to purchase a stock, bond, commodity, or other asset or instrument at a particular price within a predetermined window of time. The buyer also pays a fee called *premium* which is the total cost to buy the contract, so if the price of the underlying at expiration is under the strike price, the buyer loses the premium paid which represents the maximum loss. Otherwise, the profit is the price difference between the underlying asset's current market price and the strike price at expiry less the premium. The amount is then multiplied by the number of shares that the option buyer owns. There are 2 types of call options:

- **Long Call Option**: gives the right to buy shares for a preset price at a later date. It is executed when there is hope that the price of the underlying will rise so that exercising the right to buy it lower at the predetermined price will make a profit when selling it after.
- **Short Call Option**: it's the same as the previous one but from the seller's perspective, so it gives the right to sell shares for a preset price at a later date. In this case, the seller bets that the price of the stock will go down from its strike price and the call option will be worthless to the buyer by the expiration date. The profit of the seller is the premium paid by the buyer at the purchase, but not only, if the buyer doesn't exercise the ability to buy, the seller keeps the stocks.

Option Put

A put option is a contract that gives the buyer the right to sell a predetermined quantity of shares of an underlying at a set price (strike price) before the put option contract expires. It more profitable as the price of the underlying decreases, so it is used for hedging or to speculate on the downfall of the price. If the holder decides to sell, the contract obligates the writer to acquire shares of the underlying asset from the buyer (or holder) at a certain price (strike price) and within a predetermined window of time. The contract writer maintains the option premium money even if the buyer doesn't exercise their right to sell by the contract's expiration date.

So the buyer is not required to keep the option until it expires. As the underlying stock price moves, the premium of the option will adjust to reflect the underlying price movements. Due to the effect of time decay, a put option's value typically declines as the time to expiry draws near. With less time to earn a profit from the deal, time decay quickens as an option's time to expiry approaches.

The option writer has a similar set of options. They might not take any action if the underlying price is higher than the strike price. This is so they may pocket the entire premium as the option could expire with no value. However, the option writer may choose to simply purchase the option back if the underlying price is nearing or declining below the strike price (which gets them out of the position). The difference between the premium collected and the premium paid to exit the trade represents the profit or loss.

There are 2 types of put options:

- **Long Put Option**: it is the most popular put option strategy, in which the investor assumes the position of the holder of the option contract (buyer). A long put is betting that the price of the underlying stock or asset will drop.
- **Short Put Option**: in a short put, also known as a naked put, the investor serves as the person who creates the option contract (the seller). The investor in a short put believes that the price of the underlying stock or asset will rise. With this tactic, investors hope to make money from the option premium charge that the buyer gives them at the beginning of the contract. Since the investor can be forced to purchase worthless shares of the underlying asset if the market price of the shares falls sharply, short puts can be risky.

2.3.3 Greeks

A Greek is a symbol that indicate different risks in the option market. Each Greek variable comes from a false assumption or connection between the option and another underlying variable, but they are used to evaluate the risk of options. Each of these Greeks has a corresponding number that provides traders with information about the movement or risk of the corresponding option.

The ability to use Greeks is useful to decrease risk in a position taken and achieve the desired exposure value to risk variables. The main Greek letters that we will briefly analyze are delta, gamma, theta, rho, vega are the ones that are most frequently used.

Delta (Δ) : is the ratio of the option's price movement to a 1$ change in the value of the underlying asset. To put it another way, the option's price sensitivity relates to the underlying asset. A call option's delta can be any value between 0 and 1, while a put option's delta can be any value between 0 and -1. Assume, for instance, that a trader owns a call option with a 0.50 delta. As a result, the price of the option would theoretically rise by 50 cents if the price of the underlying stock rose by 1$. Option traders use it for creating delta-neutral [6] positions since the Delta represents the hedge ratio[7]. Not only, it can also be used to get the probability of the option expiring in-the-money[8].

Theta (Θ) : it indicates the rate of variation in an option's price with respect to time, also referred to as time sensitivity or time decay. It is the amount by which the price of an option would decline as the time before expiration grew shorter. Assume, for instance, that a trader is long an option with a theta of -0.50; with all other things being equal, the price of the option would fall by 50 cents every day. Options closer

[6] Delta neutral is a portfolio strategy utilizing multiple positions with balancing positive and negative deltas so that the overall delta of the assets in question totals zero.

[7] The hedge ratio compares the value of a position protected through the use of a hedge with the size of the entire position itself.

[8] Option that presents a profit opportunity due to the relationship between the strike price and the prevailing market price of the underlying asset.

to expiration also have accelerating time decay. Short calls and short puts will have positive theta, meanwhile long calls and long puts will usually have negative theta.

Gamma (Γ) : it reflects the rate at which the price of the underlying asset changes in relation to the delta of an option. It shows how much the delta would change if the underlying security moved by 1$. Options traders can choose to hedge not only delta, but also gamma in order to be delta-gamma neutral, which guarantees that no matter how the underlying price changes, the delta will stay close to zero.

Vega (v) : it is the option's sensitivity to volatility, or the rate of change between the value of the option and the implied volatility of the underlying asset. An high volatility will correspondingly raise the value of an option because it suggests that the underlying is more likely to have extreme values and vice versa for low volatility. So it is used to determine how stable an option's delta is. Gamma values often increase as expiration gets closer because price changes have a greater effect on gamma.

Rho (ρ) : it determines the interest rate's sensitivity. It is utilized far less frequently than the other Greeks due to the fact that an option's value is typically less sensitive to changes in the interest rate than changes to the other parameters that determine it.

2.4 Competitors

This platform takes place because of the flaws of other platforms already available. Now we'll analyse some with their features, pro and cons.

2.4.1 Fiuto Beta

PlayOptions srl's Fiuto Beta is a free program that allows underlying analysis and provides option chains on the chosen underlying. The software provides a picture of the underlying's price trend, historical volatility, open interest, and the quantities of contracts traded on the underlying with relation to underlying analysis. The software enables the display of the profit graph and reports the value of the Greek linked with each option when it comes to the examination of the options strategy (Figure 2.2).

Even though the software has many functionalities, there are some gaps compared to our platform. In particular, Fiuto Beta does not provide graphs relating to the cumulative distribution of open interest, Greek values are only shown as a numerical value, making it impossible to analyze Greek performance as a function of the price of the underlying.

So overall is perfect to start learning about options and their characteristics since is free, but it is missing some key features like the one cited before and also chain data isn't updated unless the application is restarted.

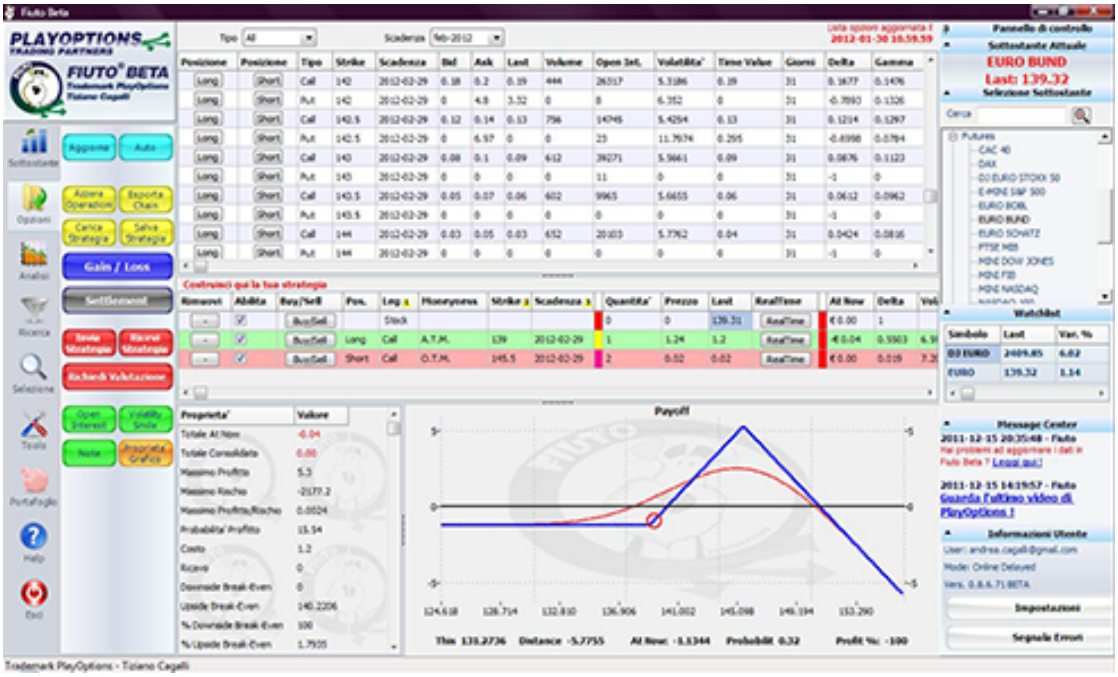


Figure 2.2: Fiuto Beta main page

2.4.2 OptionNET Explorer

OptionNET Explorer is a complete options trading and analysis software platform that enables the user to backtest complex option strategies, analyze their results and monitor them in real-time, all from within a single, user friendly environment. It allows to design and backtest multiple options trading strategies for the same or different underlying symbols and switch between them. Also it automatically keeps track of all adjustments and commissions throughout the life of each position giving you the cummulative profit and loss figure (Figure 2.3). This application is expensive and has only a 10 days trial.

2.4.3 Option Ruler

Option Ruler is a Directa's platform for option trading that is built into dLite (online trading web platform). It allows users to perform simulations and set up option buying and selling strategies that are more in line with their own price expectations in addition to the standard operations. Grid is the default screen (Figure 2.4); it is vertically separated into three sections: call, strike price, and put options data. It is possible to buy or sell a position or add it to a strategy.

There are also other pages that are: Calendar that enables users to examine the average pricing of call and put options with various maturities; Strategy that allows to evaluate the performance of a strategy and Portfolio that let users manage open or temporary orders. Account maintenance costs are not charged by the site, although commissions for European markets are approximately $5 per executed order and were $9 for non-European

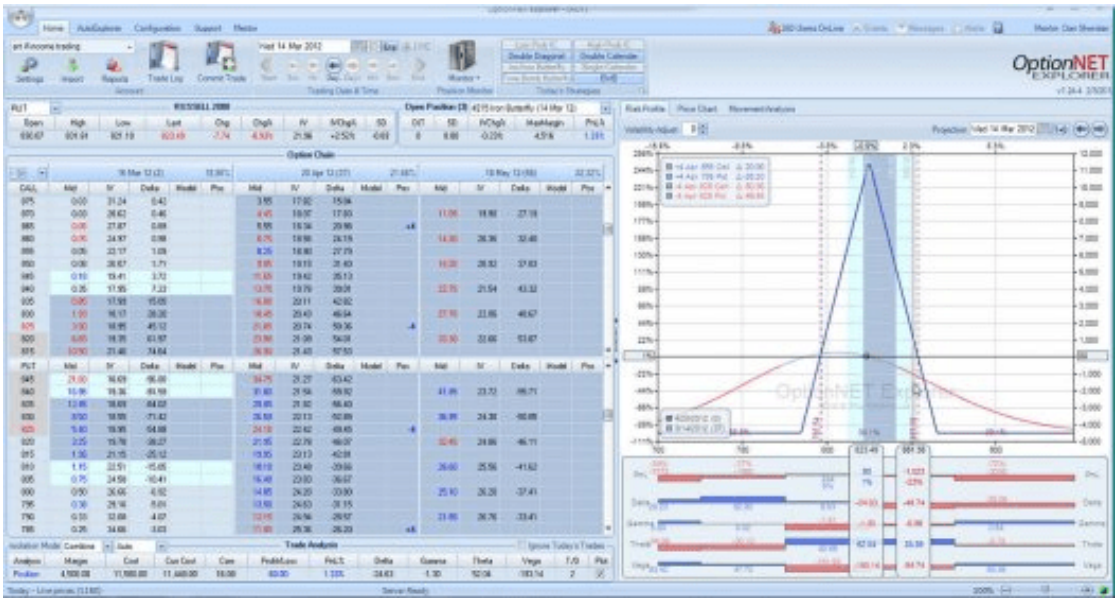


Figure 2.3: OptionNet main page

markets. It is a service that appeals to a very diverse group of investors, from novices to those with more expertise in the field.

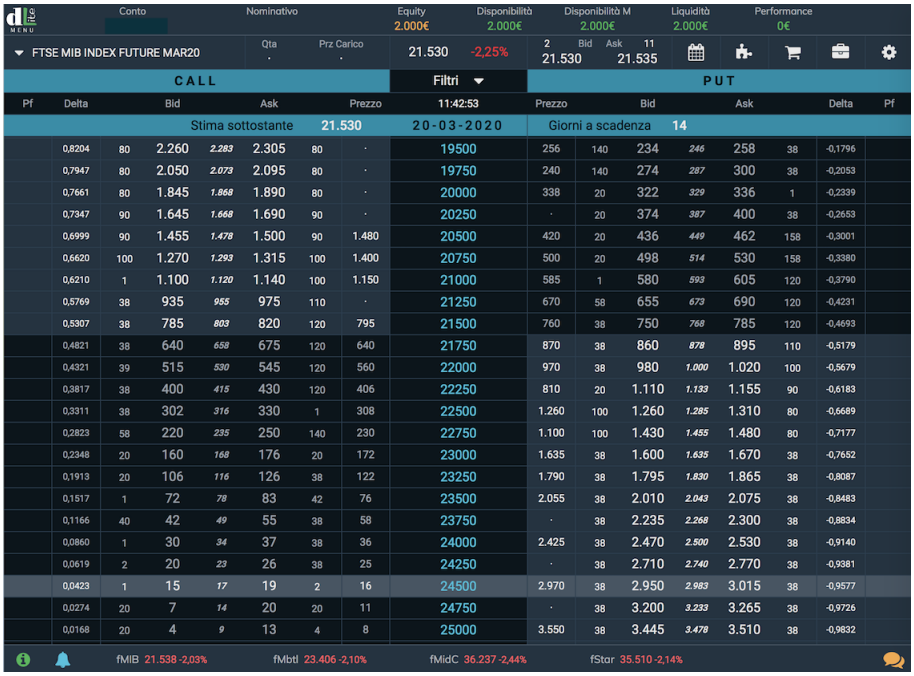


Figure 2.4: Option ruler Grid view

2.4.4 OptionVue

OptionVue is a sophisticated options trading platform with tools for determining the optimum trade strategy as well as facilities for crucial aspects including option matrices (Figure 2.5), highly customisable charting, reporting, and modeling. You can get all the pricing details for the different options contracts for a specific asset here, which is one of the program's remarkable features. The implied volatility, options premium, and the present delta value for each contract are some of the information shown. It will be simple for you to assess the changes in option pricing and implied volatility because you may look across numerous dates and strike prices at once. The ability to enter trades directly into the options chains display is another feature of this matrix. Another tool is the TradeFinder that is also available for comparing various options strategies and locating the best one that adheres to certain basic criteria and filters. The problem is that, other than being advanced, is very expensive for someone just starting to learn.

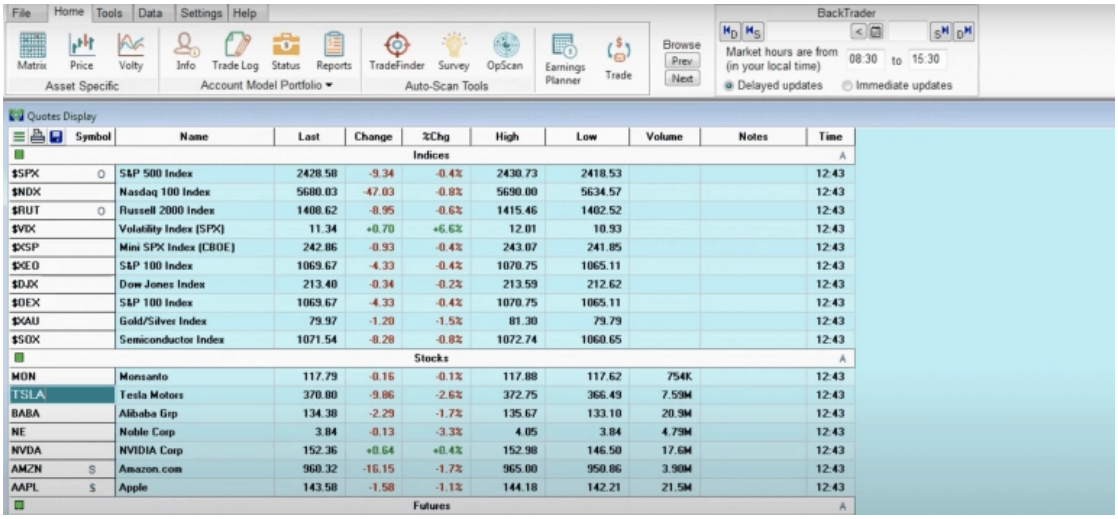


Figure 2.5: OptionVue matrix

Chapter 3

Social trading

Social trading is a form of investing that enables investors to view the trading activities of both novice and experienced traders. The main goal is to imitate or mimic other's trading tactics in order to follow their investment strategies. It can also be a different approach to studying financial data by observing what other traders are doing, comparing it to your own methods, and eventually mimicking them. Furthermore, it can serve as confirmation for other types of analysis by observing market sentiment and other traders' activities. Social trading has been referred by the World Economic Forum [11] as a low-cost, smart alternative to traditional wealth managers and is said to require little to no expertise of financial markets. It serves a wider range of clients and gives users greater control over their financial management.

There are a couple of advantages of having the social trading feature:

- **Information flow**: in finance the exchange of information is a very important feature. In this case people are constantly looking at what other people doing or sharing knowledge to others less experts.
- **Cooperative trading**: it gives investors the chance to create trading groups that can trade the markets cooperatively, whether by pooling funds, dividing research or through sharing information.
- **Transparency**: platforms give members extensive information to evaluate the reliability of the contributors they follow on the platform by disclosing performance statistics, open and historical positions, and market sentiment.

This last point was the subject of a 2017 St. John's University study which found that traders with lots of followers were more susceptible to the *disposition effect* [1] than the ones with less or no followers. In the paper the authors are suggesting that this phenomenon can be explained by: "*leaders feeling responsible towards their followers and an urge to not let them down, by fear of losing followers when admitting a bad investment decision and*

[1] Investors' propensity to hold on to assets with declining value while selling ones with rising value.

signaling confidence in their initial investment choice, or by an attempt of newly appointed leaders to manage their self-image." [12].

Also, social trading networks typically contain a leader-board based on popularity and success rate, experienced traders have an incentive to share their trading tactics because they are frequently rewarded with both money and status.

When talking about social trading, usually there are three words for describing a trade: single (or non-social), copy and mirror trade. The first one is well known and is the normal trade that is placed by a trader.

Copy trade allows to automatically copy opened positions managed by another person. It connects a portion of the copying trader's funds to the copied investor's account. Any future trading decision that the copied investor makes, such as opening a position, placing stop loss and take profit orders, or closing a position, are also carried out in the copying trader's account in proportion to the copied investor's account and the copying trader's allotted copy trading funds. The trader who is copying has the option to detach the cloned trades and manage them independently. Additionally, they have the option to end the copy connection permanently, closing all cloned positions at the current price on the market. Although followers do not deposit funds into the signal[2] providers' accounts, since the latter have indirect control over a portion of the capital of the signal followers, they in fact act as portfolio managers. Social trading platforms therefore offer a cutting-edge framework for delegated portfolio management.

At last, **mirror trading** enables the trader's brokerage account to automatically "mirror" the trades made using a chosen strategies. Individual trading preferences, such as risk tolerance and past trading success, can be taken into account when choosing a trading strategy. All signals generated by a chosen strategy are automatically applied to the client's brokerage account after selection. Since the platform manages all account activity, the client is not obliged to get involved.

The difference between copy and mirror trading is that while in mirror trading investment decisions are based on algorithms created from the trading patterns of several successful traders, in copy trading the trader actually duplicates the actions of a single successful trader.

Not everyone should engage in social trading. While it has received acclaim for removing some obstacles to financial inclusion, it has also drawn criticism for undervaluing a significant portion of the knowledge required to successfully navigate financial markets. One of the worst mistakes a social trader may make is to believe that a strategy entirely eliminates risk. Risk is a component of trading, and losses are almost always a possibility. The concept of relying on a third party's judgment while yet bearing the full risk of loss is therefore seen to be a significant disadvantage of social trading. Social trading can lower the amount of preparation required, but it also increases the likelihood that the trader will rapidly find himself out of his depth. Furthermore, there is no assurance that the third party he has selected to copy has performed the necessary amount of research.

[2] A trade signal is a trigger for action, either to buy or sell an asset, generated by analysis.

Moreover, according to a recent experiment, just telling someone about the success of others might significantly improve their risk taking. When participants are given the chance to directly emulate others, this rise in risk-taking may even be greater. In light of this, copy trading might encourage taking unwarranted risks.

Although social trading may allow to skip a few steps in the financial markets, it does so at the sacrifice of expertise and experience. It is crucial to ensure that the trader is doing everything exactly as he intended and that he has a suitable risk management plan in place. When beginning social trading, the trader is adopting the trading strategy of an another person, but a strategy should be particular to him and his objectives. Although he can utilize other people's methods as inspiration for his own trades, he has to keep in mind that other's plans will be tailored to their own objectives, drives, etc. Since everyone's risk tolerance and financial availability vary, it is not always a smart idea to trade in the same way as another person.

Before the introduction of social trading, traders and investors relied on fundamental or technical analysis to help them make investment decisions. Using social trading, traders and investors can incorporate social indicators from other traders' trading data feeds into their investment decision-making processes. So in a few words, social trading platforms can be considered a subcategory of social networks, so much that some of them also define themselves as "trading social network". In fact they have lot of similar features that now we will analyze.

3.1 Social networks features

Let us introduce a definition and features of Social Network Services. A Social Network Service is a website or online platform that enables users to create social networks or interpersonal connections with other users that have similar personal or professional interests, hobbies, backgrounds, or connections in real life.

Social networking services come in a variety of formats and feature sets. A variety of modern information and communication tools can be incorporated depending on the purposes and users. Usually, it provides an individual-centered service in contrast to online community services that are groups centered, but at the same time, they provide a space for interaction beyond in-person interactions.
There are different types:

- **Socialization** social networks are mostly used for interacting with current friends. These are the ones with the most users and include the biggest ones like Facebook and Instagram.

- **Communication** social networks allows users to communicate over the internet and they include WhatsApp and Telegram.

- **Networking** ones are used for non-social interpersonal communication. For example there is LinkedIn that is a career and employment-oriented site.

- **Social Navigation** services instead are utilized mostly to facilitate users' access to certain information or resources. One example is Reddit which is a social news aggregation, content rating, and discussion website.

One thing that all have in common is the User-Generated Content (UGC) which is the one thing that keeps these services alive. UGC is any type of online content that users have submitted, including text, audio, video, and photographs on wikis, social networking platforms, and other online communities. In this way developers need to focus on creating tools for create / modify content and places where to publish and interact with it. In this way, the company only needs to deal with the technical part of the platform, while all the content inside is created by the users themselves.

Since profits are fueled by the user time spent on the platform, these companies try to keep the user engaged and interested as much as they can. Social media engagement usually refers to the number of public shares, likes and comments. This data is also used by external companies for marketing measurements.

Different platforms offer different ways to interact with the content but are always the same concepts such as likes, comments and the possibility to share with other people either inside the platform itself or outside on other ones. The user has possibility to either be active and post personal content or to interact with other people's content through comments, likes and shares.

While shares and likes let you know how popular a post is, followers show a higher level of interest, indicating that users want to regularly see more of your work. Based on this, platforms usually promote to users accounts that are active and post frequently. In this way, users are kept busy for longer periods of time by fresh content.

Another way to keep users on a platform is to have a user-friendly, intuitive and pleasant interface design, so that the user doesn't end up frustrated that he doesn't understand or find what he's looking for and leaves the platform.

In the next chapter we will have a look at some of the principles in designing a system used by non-expert users and then see some web design trends.

3.2 Common ground

Unlike copy trading, which is the mimicking of the trades of a chosen trader, social trading places a greater emphasis on the social aspects. Because of this, some platforms have established it as a sort of social network where traders may communicate with one another, observe one another's trades and methods, and get more knowledge about developing trading strategies and making decisions. So lets have a look at what social trading apps have in common with social networks.

Just like social networks, every user has his own profile, but in this case, instead of being the personal information and posts of the user, there are statistical data on the performances and risks, and a summary of the composition of the portfolio. A profile usually has a nickname, description and picture. Sometimes there is also the flag that indicates where the user is from.

A trader can be searched in a page where all profiles are displayed in different categories, like "Top traders" or "Trending" and they can be filtered by different parameters. Each profile displayed is summarized with the most important information in order to let the user decide if it is worth checking them out. The summary usually includes some information on the user, like username, photo and then there are some statistical and performance data. Some examples of explore pages can be seen in Figures 3.1 and 3.2.

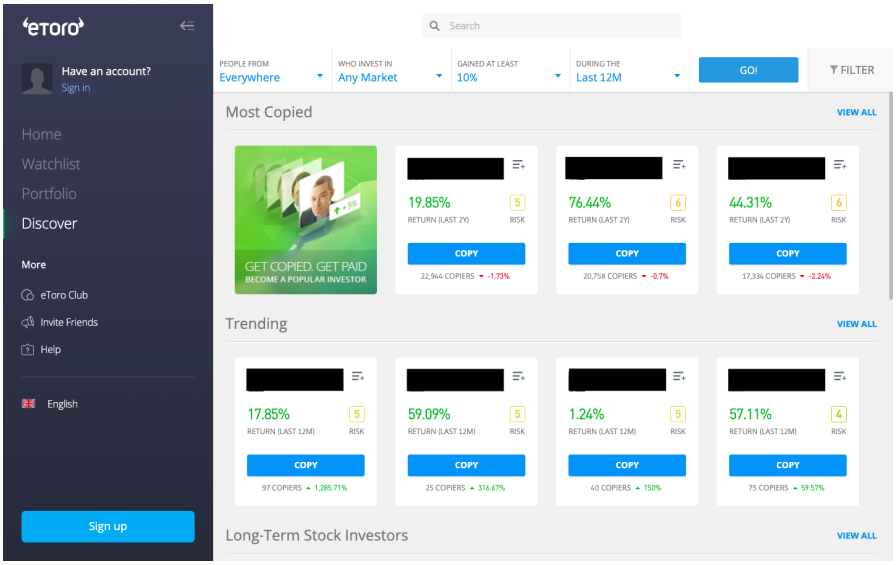


Figure 3.1: eToro copy investor explore page.

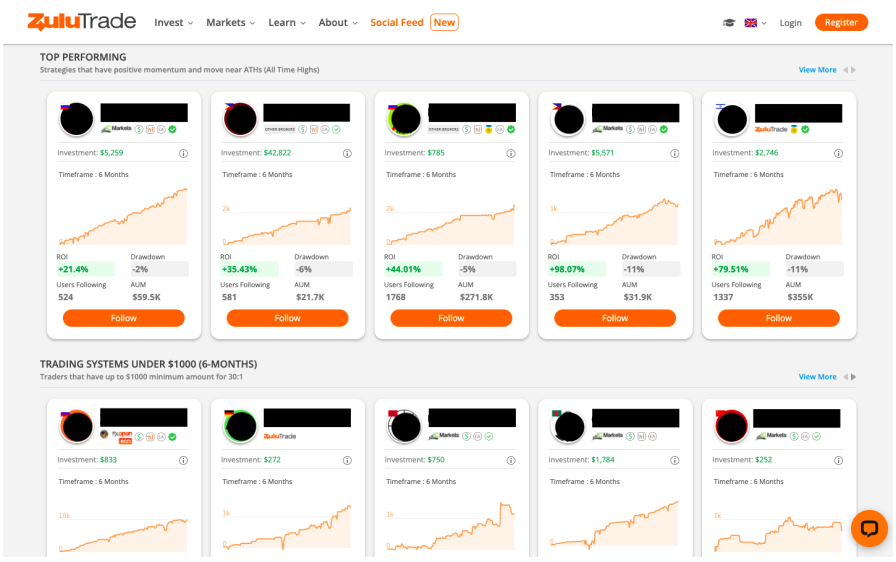


Figure 3.2: ZuluTrade copy investor explore page.

Once opened a profile, all the information and statistical data of the trader is presented in order to understand his skills, achievements and if he is worth to copy. Among the information usually there is a feed with the operations performed, statistical data on the performance, the portfolio composition and different charts. An example can be seen in Figure 3.3.

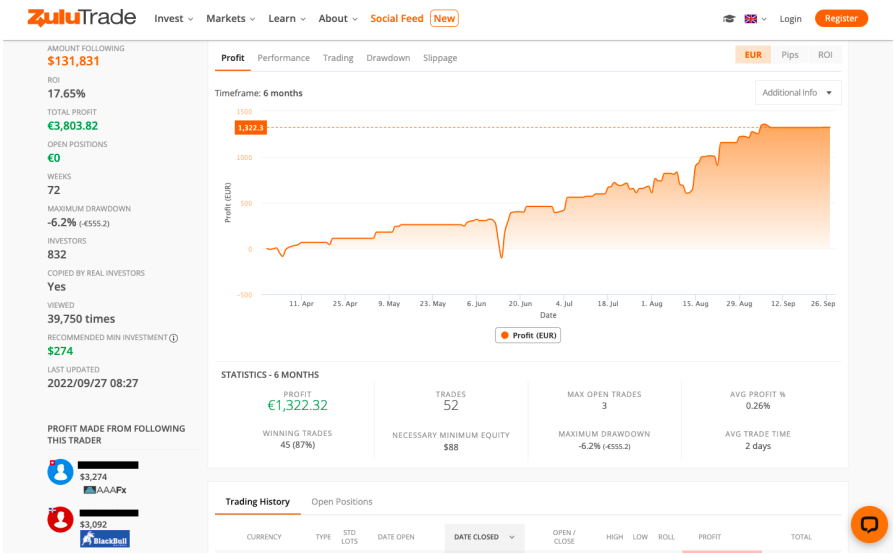


Figure 3.3: ZuluTrade user page.

Another thing that some platform use for distinguish users, that are also present in some social networks, are badges that are obtained according to milestones achieved, and they are displayed in the profile. These badges are made in an effort by the broker to promote a positive social environment and are not only for showing off one's accomplishments, but they can also aid potential copiers in forming a proper opinion of a certain trader. You can learn about a trader's involvement in the community, whether or not the public likes or dislikes them, whether they copy other traders or prefer to rely on their own judgment by looking at the kind of badges they have accumulated. Badges represents some achievement in different areas like capital gain, time spent trading and many others.

3.3 Social trading platforms

The first platform to have introduced the feature was eToro in 2010. eToro is an international social trading company based in Israel that specializes in offering financial and copy trading services. Along with being used for trading stocks, making investments online, trading cryptocurrencies, and much more, it has also developed many novel social trading features. The main feature is the *CopyTrader*, that lets you select the traders you want to replicate, choose your investment level, and with just one click, automatically and in real-time, copy everything they do (modal for copying in Figure 3.4). This is just the basic concept, but other than that there are quite a few more settings and elements to

the system. For example you can copy all the trades or only the new ones and copy the stop losses.

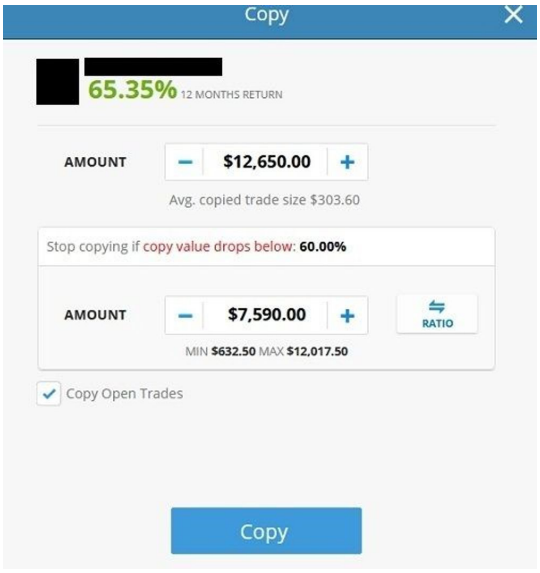


Figure 3.4: eToro CopyTrade interface

The decision on whether or not copy someone is based on statistical and historical data that is displayed on the user profile (examples in Figure 3.5). In particular there is the gain and fail/success ratio, the *Risk Score* (shows the risk the investor is taking on a scale of 1 to 10) and the portfolio composition.

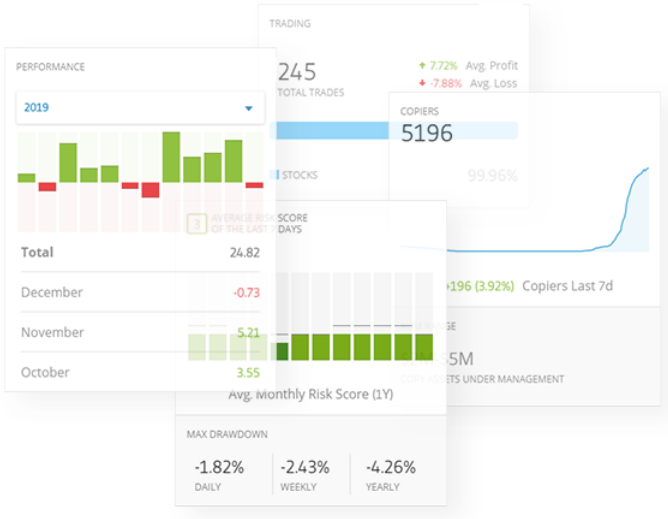


Figure 3.5: eToro investor data. Image from here.

In a very similar way other platforms jumped in implementing these features. The first one was Wikifolio in 2012, then others like NAGA which is Europe-based and Zulu Trade. All have similar characteristics and tools to look at profiles performance and risk in order to decide whether to follow their strategies or not.

The problem with the platforms cited above and others similar, is that they don't have the possibility to trade derivatives, but they allow to trade Stocks, Commodities, Indices, Forex, ETFs and Cryptocurrencies.

The only popular one that has the ability to trade also derivatives such options and futures with features of social trading is Tastyworks. Tastyworks was launched in 2017 as a subsidiary of Tastytrade and is an online brokerage platform specialized in options, futures, and stock trading. It is aimed for experienced and active traders, so it can be intimidating for beginners, but it has a lot of educational content and research tools for learning available on Tastytrade website. The platform has a lot of focus on derivatives since they form the 98% of the trades placed by the customers.

Coming back to the social trading aspect, just like eToro, it lets the user view and copy real-time trades from other traders. There is a tab called "FOLLOW" in which there are all the trades placed by the people that the user follows (Figure 3.6). By clicking on one trade in which the user is interested there are a set of informations and a button named "trade" or "duplicate this trade" according to the device used (Figure 3.7). Clicking on the button automatically fills the trade ticket with the same trade with the possibility to add an underlying security.

The limitation of Tastywork is that the people that one can follow are only the teachers of the show Tastytrade which are just 21 (showed in Figure 3.6). This has been done so that if someone is following the course wants to recreate also the trades that they are showing, they can in real time on their account.

Here is where our application comes in and fills the gap in this business, a social trading application that trades with derivatives.

3.4 Regulations

Since copy traders are effectively making judgments for other traders, the MiFID[3] recently classified copy trader platforms as portfolio management services due to their automatic nature. The MiFID does not expressly address the issue of copy trading, but it does provide general guidelines that would serve as the foundation for any explicit rules to be developed. The tendency among regulators is to view copy trading as combining elements of portfolio management, which is a MiFID-regulated financial service. As a result, the

[3]The MiFID stands for The Markets in Financial Instruments Directive and is a European regulation that increases the transparency in the EU financial markets and normalizes the regulatory disclosures required for firms that operate in Europe.

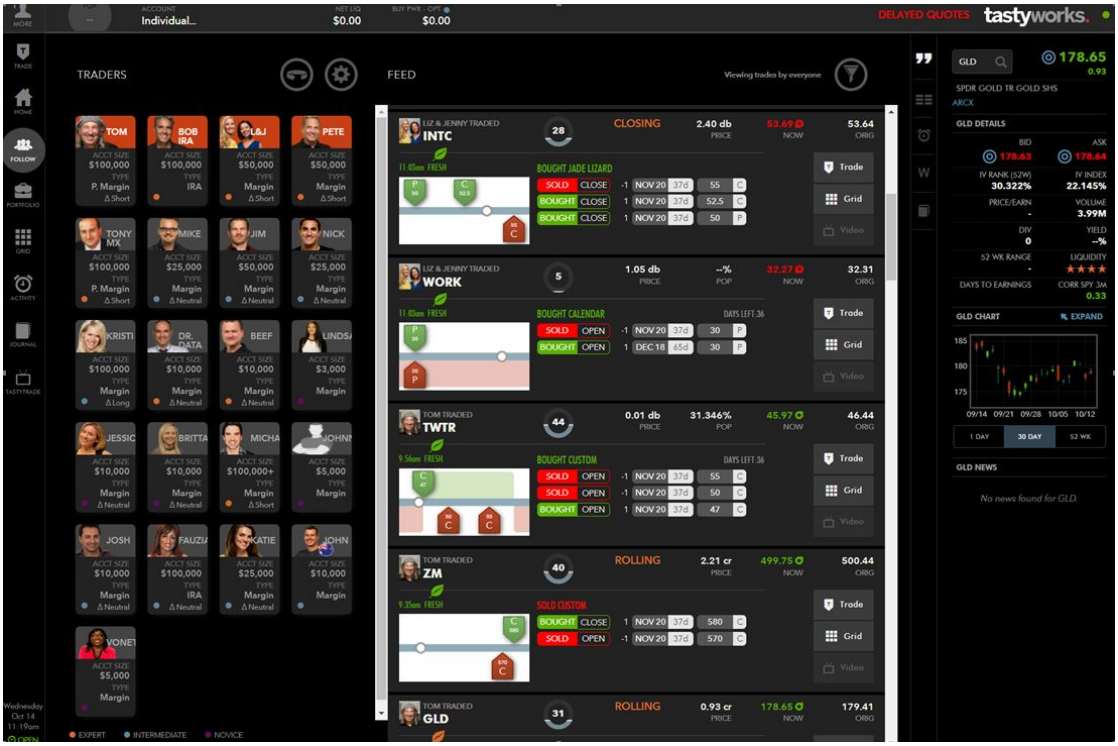


Figure 3.6: TastyWork FOLLOW page. Image from here.

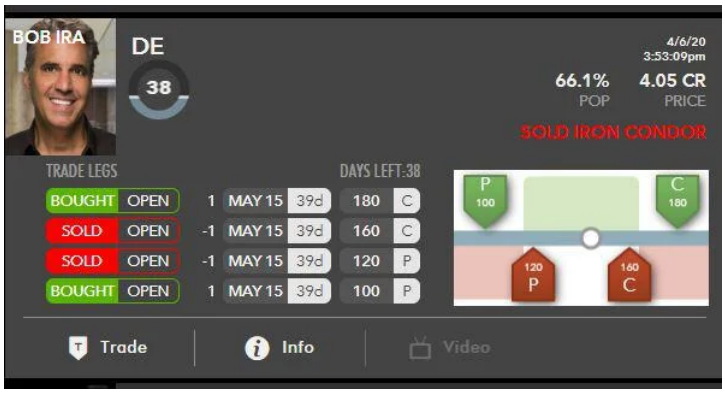


Figure 3.7: TastyWork copy trade.

platforms had to incorporate more features to meet regulatory requirements, particularly those that would reduce risk and increase public awareness of it.

For this reason many trading systems provide each trader a risk profile so those thinking about mimicking them can have more information. Additionally, platforms have included additional tools to assist clients diversify their portfolios and pair traders with one another based on risk profiles. These come at an additional cost to comply with the law, but many

professionals view this as a positive development and essential to creating a sustainable product.

So, platforms, in order to reduce the copy trading potential backlashes, have created "programs" in which investors can join once they meet some requirements. This is done in order to help inspire confidence in copying someone since they follow some criteria. Some of the requirements could be that the investor is verified (with his real credentials and personal information), is active, has invested a certain amount of money, has been investing for a certain period of time, respects some set rules about leverage and has a risk factor below a threshold. This types of users usually are the finest and the ones being copied the most.

Once the trader is associated with a program, he gains perks and benefits according to the number of copiers to maintain the position and the active work.

Chapter 4

Design

The design of systems that interact with humans is an entire research field in computer technology called **Human computer interaction** (HCI). Researchers in this field study how people use computers and develop solutions that let people use them in new and different ways. As said many times, the interaction with the interface must be fluid and easy, the user has to find what he's looking for in order to avoid frustration and generate disinterest in the application. Basically the objective is to reduce errors committed by the user or try to avoid them happening in the first place. If there is one the user must have a way of correcting it or go back and redo the stage where something went wrong.

Also the user has to have everything needed visible, accessible, and the navigation inside the pages has to be intuitive and not too complicated.

Researchers in the field may have different ideas of what they want to accomplish, for example for a cognitive perspective they try to make computer interfaces more consistent with the mental picture that people have of their activities. They aim to match computer interfaces with current sociocultural values or social practices.

In addition to studying interaction paradigms they are also interested in building design approaches, experimenting with hardware and software, prototyping software and hardware systems, and developing models and theories of interaction.

4.1 Interface design principles

Now lets have a look at the "golden rules" of interface design present in the *Designing the User Interface: Strategies for Effective Human-Computer Interaction* book [13], in particular in chapter 8:

1. **Strive for consistency**: similar scenarios should call for the same set of actions, the same terminology should be used in prompts, menus, and help screens, and the same color scheme, layout, capitalization, typefaces, and other design elements should be utilized throughout.

2. **Seek universal usability**: users have different needs, so the interface should adapt

and let the content be transformed. For example adding explanations for novices and shortcuts for experts enriches the interface design and improves perceived quality.

3. **Offer informative feedback**: every action the user performs should be associated with an interface feedback. The response can vary according to the action.

4. **Design dialogues to yield closure**: action sequences need to be grouped into units with a beginning, middle, and end. Feedback after the completion gives the user the satisfaction of accomplishment. For instance, e-commerce websites guide consumers through the checkout process ending with a clear confirmation page.

5. **Prevent errors**: the interface should be design so that the user can't make mistakes. If they still make one, the interface should offer easy instruction for recovery.

6. **Permit easy reversal of actions**: actions should be reversible when is possible. Because users are aware that mistakes may be corrected, this feature reduces fear and stimulates investigation of new actions.

7. **Keep users in control**: experienced users have a great desire for the feeling that they control the interface and that it reacts to their activities. They dislike unexpected changes in behavior, and they find laborious data entry processes, challenges locating relevant data, and failure to deliver intended results annoying.

8. **Reduce short-term memory load**: interfaces that demand users to memorize information from one display and subsequently use that information on another display should be avoided due to humans' limited ability to process information in short-term memory.

In the application these principles were followed as much as possible. For example, as we will see in the next chapters, every element that can be created by the user can also be updated or deleted; errors are prevented by not showing what the user shouldn't see and disabling buttons he can't access depending to different roles and permissions; and every information needed for actions are showed and well displayed.

4.2 Web design inspired by Social Media

The main characteristics of a typical social media can influence the design of web applications usually are: visuals, images, interaction and use of the user generated content (UGC).

Because there is more stuff available online every day we have developed the practice of speed-reading and quickly scanning websites to find the information we need. To combat information overload we should take advantage of **attention-grabbing visual elements**. For example in the page where admins manage members, the button to add new ones is colored and well visible in order to "call to action" (Figure 4.1).

Promoting images is very important because "a picture paints a thousand words", and in addition to icons, they immediately communicates the purpose and usage of buttons

Figure 4.1: Attention grabbing button

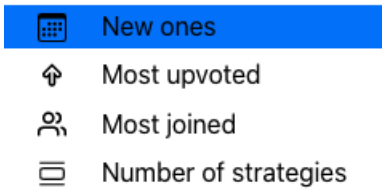


Figure 4.2: Icons in support of text

and features. In the application icons are widely used as support of text since they immediately highlight the action or task (Figure 4.2).

Another important aspect is to **invite to interact** through the use of micro-interactions. They are essential to give the user the impression of showing interest for something and give him some entertainment value. A micro-interaction [14] is a short animated event that has a single purpose, further, they are used to entertain the user and generate a small emotional experience. They have the purpose to get the user to do something, they need to grab the attention and satisfy the user once they have been used. They should not only be captivating animations, but also highly enhance user experience by giving immediate feedback. The thing to avoid is to add too many animation that instead disturb the user experience. For example in the application there are only two main micro-interaction that are the upvote and the bookmark button, which are a way to interact with elements (Figure 4.3).

Finally, the content created by the user (**UGC**) should be customizable in some way so that the user can distinguish himself from the others.

4.3 Reference designs

When designing an interface, is always suggested to have a look at what other companies have done according to the different purposes of the application. For example in our case the objectives were to have pages in which present both content from the logged-in user and content from other users, and also, for the clubs, have a common space within a group of people where to interact. These characteristics are present already in many popular social networks. According to what have been said in the previous section about social networks, the application falls in the *social navigation* category since it is not used to create networks or to chat, but to view content of other users.

In particular the application has nested sections that works at levels, for example in order

(a) Upvote before click

(b) During animation

(c) Bookmark before click

(d) During animation

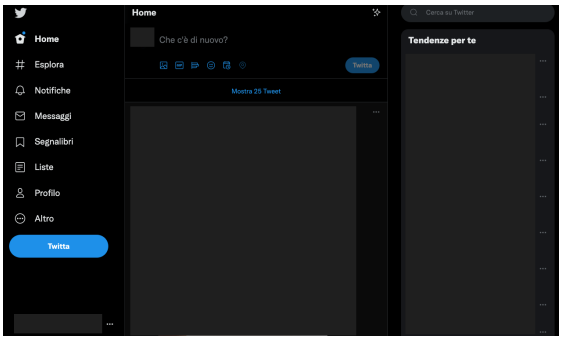
Figure 4.3: Button animation

to look at one strategy from the *Strategies* page, the user has to select a category and then a strategy to open. Each section doesn't have enough information and content in order to be in a separated page from the others, so they are put together in columns or rows. This can also be seen in other social media applications like Telegram, Discord, Facebook, Twitter and many others. The ones cited are the ones that have been used as inspiration for different parts of design.

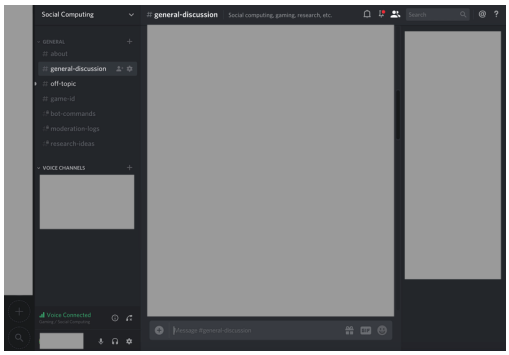
As we can see in Figure 4.4, they all have similar structure: one navigation column on the left, content in the center and suggested or minor content or column on the right. The order of the column is dictated by navigation purposes, and the order goes from left to right since what is selected on the left decides what to show on the right. On the top usually there are some commands and search bars for filtering content.

- **Navigation column**: on the far left there is the navigation section that is needed to decide what type of content display in the main column.
- **Content section**: at the center of the page there is the main component that displays the content.
- **Secondary content**: on the right is usually displayed extra content that can be something like secondary content, advertisement or suggested content.

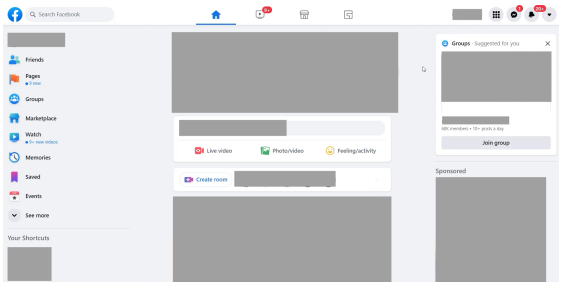
This same type of structure is being used in our application when displaying lists of content. There is the navigation column and the content columns. The content ones can vary according to what is shown on screen. We will have a more in depth look of the interface in the client chapter.



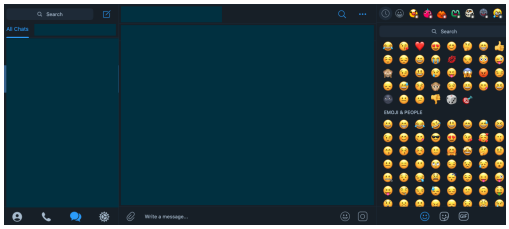
(a) Twitter



(b) Discord



(c) Facebook



(d) Telegram

Figure 4.4: Social media main pages

Chapter 5

Architecture

As previously mentioned, the application is web-based with a client-server architecture used for managing it. Between the client and the server the data is exchanged with the use of the APIs. Since the application is made up of microservices that handle data, the server side has been containerized in order to make the development faster and easier. Microservices are an architectural and organizational approach to software development where software is composed of small independent services that communicate with each other. Each microservice should have a clearly defined role and domain knowledge when performing its tasks, so this approach promotes better results when implementing new features in a system since each one specializes in a particular task. Furthermore, separating concerns into individual microservices promotes scalability since each service can scale independently. These microservices are packed inside containers that are then managed all by a program in order to make the development and use easier and faster.

So in this chapter we will explain everything that have been used for developing including the list of development tools and frameworks with their characteristics and advantages.

5.1 Web-Based Application

Any program that is accessed via a network connection using HTTP[1] rather than being stored in a device's memory is referred to as a web-based application. Web browsers are frequently used to run web-based applications and is the instrument used by the user as an interface for accessing the data. By using the browser nothing needs to be installed or updated on the client machine, but everything is requested to the server. Nowadays browsers are largely widespread and supported on most devices so web-applications can be considered platform-independent.

The usage of terminology like web-based, internet-based, and cloud-based when discussing applications causes a lot of confusion; those that interact with users using HTTP

[1]HTTP stands for Hypertext Transfer Protocol and it is an application-layer protocol for sending hypermedia documents between web browsers and web servers, but it can also be used for other purposes.

are all considered web-based applications.

The architecture that was chosen is the client-server because it is the most common and highly recommended design for enabling users to interact with online applications over internet using a web browser.

5.1.1 Client-Server Architecture

In the client-server computing model, the server hosts, provides, and controls the majority of the resources and services that the client will use. In this form of architecture, a central server is connected to one or more client computers via a network or the internet.

So basically, there is the server which is listening and waiting for request of services and the client that has an interface with which the user interact and makes requests to the server that responds with the results. So the exchange of messages between client and server follows the *request-response* pattern: the client sends a request, and the server returns a response. Many clients can make requests to the same server simultaneously (Figure 5.1).

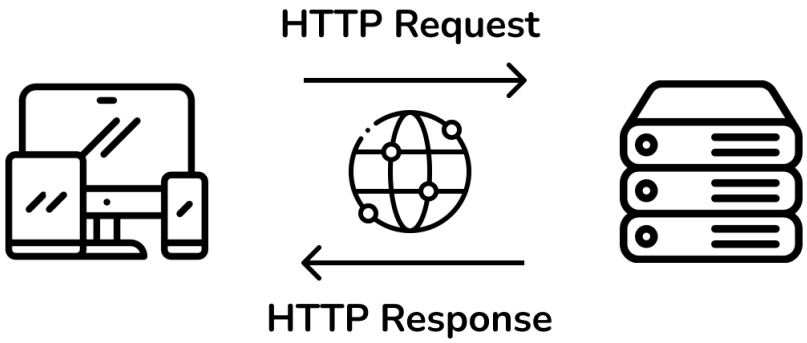


Figure 5.1: Client-server architecture

The requests and responses are made through a specific network protocol, which in our case is HTTP. Other than that, there is the *application programming interface* or API that is an abstraction layer[2] for accessing a service on the server. This is needed for facilitate cross-platform data exchange through a communication with specific content format. They strive for rapid performance, reliability, and scalability by reusing parts that can be controlled and modified without having an impact on the system as a whole, even while it is in use. In particular for the server interface it was implemented a REST API[3] approach.

[2]Technique for concealing how a subsystem works.

[3]A software architecture approach known as "representational state transfer" describes a consistent interface between physically distinct components.

REST API

REST in web development enables what is frequently referred to as "Dynamic Content" or content that is rendered as it is requested. In order to create a website and transmit it to the requesting web browser, it is required to have server-side rendering. The requesting web browser then understands the server's code and renders the page in the user's web browser.

Using URL-encoded parameters, HTTP requests are used to retrieve data or resources in web applications. The data is often formatted as either JSON or XML[4] for transmission. REST guidelines recommend using a specific HTTP method on a given type of call of a server call. HTTP provides operations or methods that are described in Table 5.1.

HTTP method	Description
GET	Retrieve resources representation or information
POST	Create new resource in a collection of resources
PUT	Update an existing resource
PATCH	Make a partial update on a resource
DELETE	Delete resources

Table 5.1: HTTP methods

5.1.2 Containerization of the server

Using Docker containers (that will be explained later), the server-side portion of the web application may be built and deployed. With this architecture, an application can be packed along with all of its dependencies without having to worry about the real server's hardware or software features. This is due to the fact that a container always carries all of its dependencies with it. Additionally, by running many web applications separately on the same physical server, containers enable IT businesses to drastically lower the cost of managing data centers and administrative overhead.

Due to this, we choose to use Docker containers to build the trading platform. Looking at Figure 5.2, it is able to observe how the Docker compose tool structured and connected all necessary applications.

Now we will go through all these applications and frameworks and see how they work.

[4]JSON and XML are two human-readable formats for storing or transmitting data.

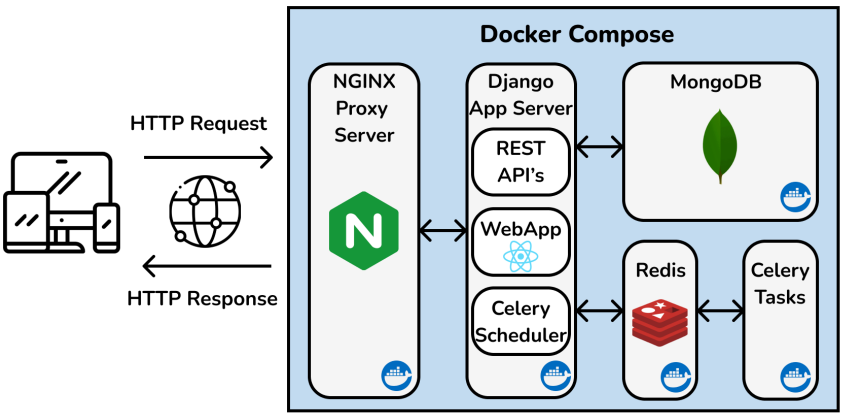


Figure 5.2: Architecture of Docker-Compose

5.2 Technologies

As seen from the previous image we used Docker Compose that is a configuration tool used for running multiple containers at once keeping all the components in the same place letting them interact with each other. All of the configurations are defined in one file so that the containers can be started with one command. Containers run the applications in isolated environments so that they don't enter in conflict with each other.

Now we will have a look at the components one by one.

5.2.1 Docker

Docker is an open platform for developing, shipping, and running applications [15]. You may divide your apps from your infrastructure with the help of Docker, allowing a fast software delivery. Users can package programs into a Docker-Image, which is a compact, independent, executable bundle of software that includes all the operating system libraries and dependencies needed to run the code in any environment. When launched on Docker Engine[5], a Docker image becomes a container, a loosely separated environment. An advantage is that the container can be easily shared because the host doesn't have to worry about what he has installed on his device.

Architecture: docker uses a client-server architecture (Figure 5.3) where the client communicate with the Docker *daemon* (server) through a REST API over a UNIX sockets or network interface if the server is remote or through the CLI[6] if both client and daemon run on the same system. So the Docker daemon manages Docker objects like images, containers, networks, and volumes while listening for requests made over the Docker API. To manage Docker services, a daemon can also talk to other daemons. Docker registry is

[5]Open source containerization technology for building and containerizing your applications

[6]Command line interface

used to publish or download public or private images.

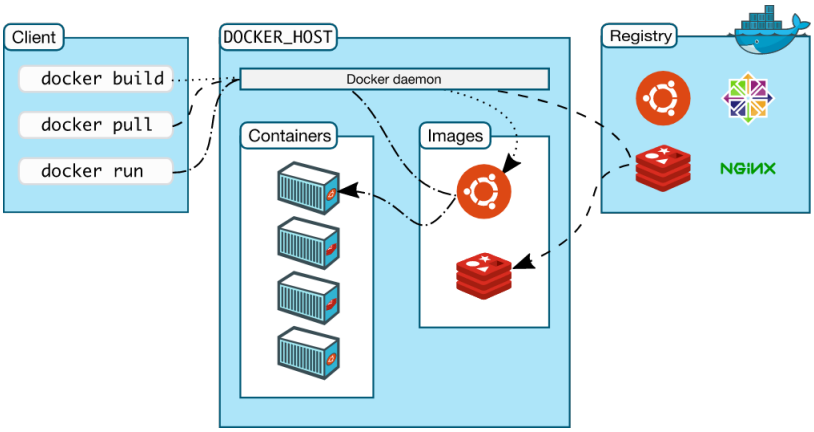


Figure 5.3: Docker client-server architecture. Image took from [15]

Docker Compose

A tool called Docker Compose is used to create and manage services, which are multi-container Docker applications. These may all be created and started with a single command after being specified using a specific file for each environment. Some features are that one host can have multiple isolated environments and that it recreates containers only when they change.

5.2.2 Celery

Celery is an open source distributed message forwarding based asynchronous task queue or job queue. Although it enables scheduling, its emphasis is on real-time operations. Using multiprocessing, the execution units, known as tasks, are carried out concurrently on one or more worker nodes. Tasks can be executed asynchronously (in the background) or synchronously (wait until ready). Despite being built in Python, the Celery protocol can be implemented in any language and is simple to integrate with a wide range of web frameworks, like Django in our case.

Redis

Celery typically requires a different service called a message broker to transmit and receive messages between the application and workers because it interacts via messages. Redis is one of the most popular message brokers for this use due to its compatibility with Celery. In particular, it is a open source, in-memory data store used by millions of developers as a database, cache, streaming engine, and message broker. It popularized the concept of a system that functions as both a store and a cache. In addition to being saved on disk in a format that is not suitable for random data access, it was created so that data is always

modified and read from the main computer memory. Once the system has restarted, the formatted data is solely restored into memory.

5.2.3 MongoDB

MongoDB [16] is a cross-platform document-oriented database application that is open source. MongoDB, a NoSQL database application, employs documents that resemble JSON and may or may not include schemas[7]. A document, or record, in MongoDB is a data structure made up of field and value pairs. Values can be native data or also other documents, arrays, and arrays of documents. Some key features are:

- High performance data persistence since it supports embedded data models to reduce I/O activity on the system and has indexes for query faster.
- The Query API supports read and write operations (CRUD[8]) and also Data Aggregation (pipeline that process documents with one or more stages), Text Search and Geospatial Queries.
- High Availability through its replication facility called replica set. This is a group of servers that maintains the same data set, providing redundancy and increasing data availability.
- Horizontal Scalability by using a process called *sharding*, which consists in distribute data across a cluster of machines.
- Supports multiple Storage Engines, which are the responsible for managing data storage, both in memory and on disk. The performance of the applications can be dramatically impacted by selecting the best storage engine for the use case.

Before we cited NoSQL, now we will have a look at what it is and where it comes from.

SQL and NoSQL

The programming language known as SQL, or "Structured Query Language," has been used extensively since the 1970s to manage data in relational database management systems (RDBMS)[9]. When storage was still expensive in the beginning, SQL databases concentrated on minimizing data duplication. Today they are still very used using tables with columns for the field type and rows for the records. One table record may be connected to another record in the same table, to many other records, or to many records in another table.

[7] An organization of data as a blueprint of how the database is constructed

[8] It is the acronym for CREATE, READ, UPDATE and DELETE that are the operations for managing persistent data in relational and NoSQL databases.

[9] Is a type of database that allows to identify and access data in relation to another piece of data

NoSQL is a non-relational database, which means it offers more flexibility to use the format that best suits the data and different architecture than a SQL database.

When working with connected data, SQL is good. Relational databases are effective, adaptable, and simple for any application to use. A benefit of a relational database is that every instance of the database automatically refreshes when one user modifies a specific record, allowing for real-time information to be delivered.

While SQL is preferred for assuring data validity, NoSQL is effective when it's more crucial that massive data can be available quickly. Additionally, it's a wise decision when a business needs to expand in response to shifting customer demands. It's a wise decision as well when a business must expand to meet shifting demands. [17]

So, in conclusion, in our case was chosen NoSQL with MongoDB because it is the best DBMS when looking for high flexibility and versatility to real world scenarios.

5.2.4 Django

Django is a Python-based server-side web framework[10] that is free and open-source. It follows the *model–template–views* (MTV) architectural pattern and the primary goal is to facilitate a quick creation of complex, database-driven websites.

Before diving into Django, lets have a few words about Python. Python is an interpreted, object-oriented, high-level, dynamically semantic programming language. The straightforward syntax that prioritizes readability makes it simple to learn, which lowers the cost of program maintenance. Python's support for modules and packages promotes the modularity and reuse of code in programs. For all popular platforms, the Python interpreter and the comprehensive standard library are freely distributed and available in source or binary form.

Coming back to Django, Django is versatile since it can work with all client-side frameworks, and can deliver content in almost any format, including HTML, RSS feeds, JSON, and XML. It can scale for more traffic by adding hardware at any level because the various components are completely separated. Other than that Django is very secure, because by default it has protection against several vulnerabilities, such as SQL injection, cross-site scripting, cross-site request forgery, and clickjacking. It also has an integrated authentication mechanism that manages user counts, groups, and cookie-based user sessions, support for multiple cache mechanisms, end-to-end application testing and translating text into various languages, local formatting dates, times, numbers and timezones.
It also has a lot of external libraries; for example in our case was used the REST-Framework one that provides support for building APIs with validators and authentication protocols with few lines of code.

At the beginning was mentioned that Django follows a model–template–views architecture, so now lets see how it works. There are separate files that deals with different steps of the communication between client and server which follow this schema (Figure 5.4):

[10]Software for designing web applications

URLs : it is a mapper that is used to route HTTP requests based on the request URL to the appropriate view. It can also look for specific strings or number patterns in a URL and transmit those patterns as data to a view function.

View : is a function that handles requests and sends HTTP responses in response to HTTP queries. Models provide views with access to the data they require to fulfill requests, and templates are given control over the formatting of the answer.

Models : are Python objects that provide the data structure for an application and offer tools for managing (add, change and remove) and querying database records.

Templates : is a text file that specifies a file's structure or layout with placeholders are used to indicate actual content.

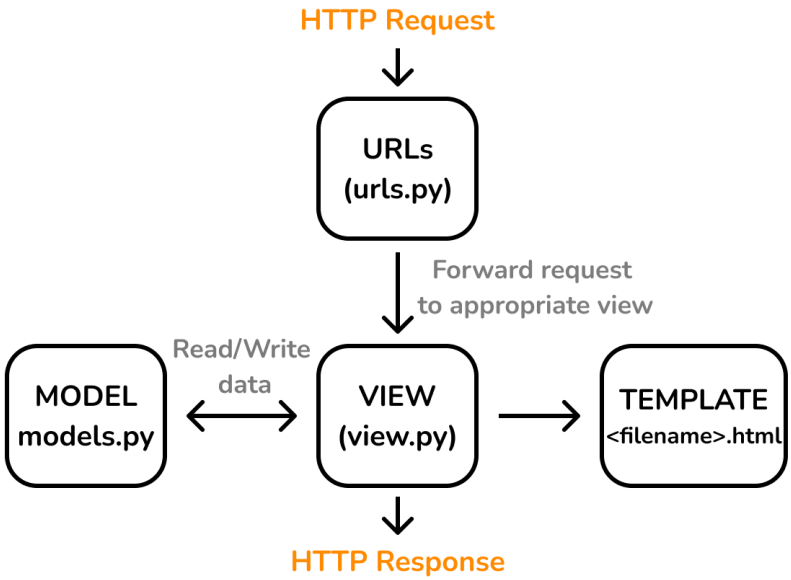


Figure 5.4: Django MTV architecture.

5.2.5 React

React is a front-end JavaScript library that is free and open-source used for creating user interfaces based on UI components. The basic idea is to create user interfaces by assembling little pieces of code (called *components*) into complete websites without imposing any type of architecture or pattern to developers.

These components are build using JSX, that is a combination of JavaScript and XML, and they are the smallest building pieces with an inner state and properties that make them flexible and reusable. JSX enables developers to construct elements containing both HTML and JavaScript at the same time.

Props and callback functions are used to interact among the components in the component tree as data travels downward through it. This composition gets deep, tightly connected, less maintainable, and prone to *props-drilling* in large React apps. This is one of the reasons why complex React apps require architectural patterns like the Redux pattern.

React-Redux

Is an open-source JavaScript package that is used to control application state while creating the user interface. It enables Redux Store data to be accessed by React components, who can then send actions to the store to update the data (Figure 5.5). By offering a practical method of managing state through a unidirectional data flow architecture, it helps in the scalability of projects.
So lets define all the elements:

STORE : is the place where the entire state of the application is situated. It has a dispatch (action) function used for handling the application's status change.

ACTION : is dispatched or sent from the view and contains payloads that Reducers can read. It is a pure object designed to keep track of user event data.

REDUCER : it reads the payload of the action and updates the store.

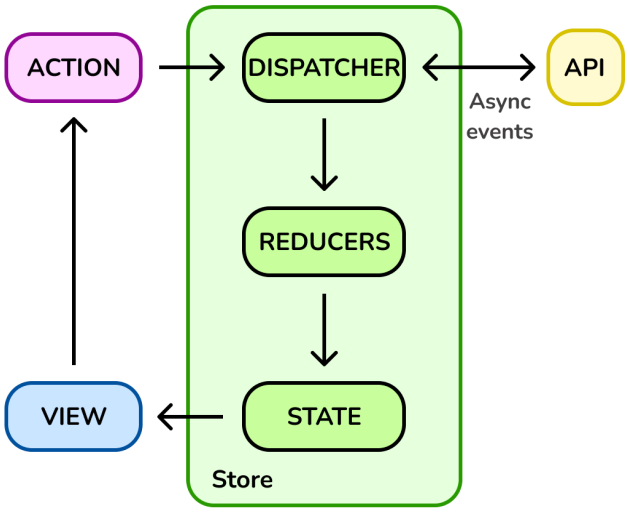


Figure 5.5: React Redux architecture.

JavaScript

JavaScript is a programming language that enables the implementation of sophisticated functionality on web pages like create dynamically updating content, control multimedia, animate images etc. It is the third tier of the *layer cake* made up of standard web technologies in which the first two being HTML and CSS. It is a type-safe, dynamic scripting language, and the user's browser's interpreter executes the code rather than transforming it by compiling it like in other languages like C and C++.

HTML

The most fundamental component of the Web is HTML (HyperText Markup Language). It describes the meaning and structure of web content meanwhile the appearance / presentation and functionality / behavior of a web page are handled by CSS and JavaScript respectively.

To annotate text, images, and other content for display in a Web browser, HTML uses *markups* that include *Tags*. Tags consist of the element name enclosed by "<" and ">". They are used to distinguish HTML elements from other content in a page. Case is not relevant when naming an element inside a tag.

CSS

Cascading Style Sheets is a language for style sheets that is used to describe how a document written in a markup language, like HTML, is presented. Layout, color, and font may all be separated from the content by specifying the CSS in a separate file, which reduces complexity and repetition in the structural content. This separation can improve content accessibility, flexibility and control of the presentation characteristics, enabling multiple web pages to share formatting, and enabling the style file to be cached to improve page load speed between the pages that share the same file.

5.2.6 Nginx

Nginx is an open-source web server that is currently utilized as a reverse proxy, HTTP cache, and load balancer in addition to its original function as a web server due to its success. It was designed to have high concurrency and little memory utilization. Nginx employs an asynchronous, event-driven approach where requests are handled in a single thread rather than starting new processes for each web request.

One master process can manage numerous worker processes, so while the workers carry out the actual processing, the master manages the worker processes. Each request can be processed by the worker simultaneously without affecting other requests because Nginx is asynchronous.

Nginx is utilized by a number of well-known businesses, including Microsoft, IBM, Google, Adobe, LinkedIn, Cisco, Facebook, Twitter, Apple, Intel, and many others.

Chapter 6

Implementation

This chapter describes the back-end part of the application, which refers to the part of a software system that is not visible to the user. This part of the system performs tasks such as data storage and processing and is used to power many different types of online applications. A server, a database, and a back-end application make up a website's back-end. The information is stored in the database, which is operated by the server. By using server-side scripts to transmit requests and queries, the back-end application creates a barrier between your server and the database. A good database should have the features necessary to support an application's needs, which means that, in our case, the database should have a flexible schema and the ability to scale efficiently with the different types of data. A well-implemented database will have adequate performance and security features, and this will ultimately improve the overall functionality and efficiency of an application's system architecture.

Now the implementation of the back-end will be presented with the database and the services exposed.

6.1 Database

As was already mentioned in the technologies section, for data persistence, a non-relational database, specifically MongoDB, is being used. The collections are: User, Market, Chain, Future, Group Strategy, Portfolio, Market History, Chain History and Club. Here we will focus on the new ones and those that were modified or used.

6.1.1 MongoDB Collections

As said before, in MongoDB data records are kept as documents, specifically BSON documents[1], which are organized into collections. Also for the collections definition it was decided to not define Python classes but to use JSON dictionaries directly so that one document can have a different structure from one another inside the same collection.

[1] A binary representation of JSON documents that has more data types than JSON

User

Its a collection of user records that includes details for identifying each individual user, such as email, name, and surname, as well as details about how they used the platform (Table 6.1).

Field	Description	Data Type
id	Unique identifier of the user inside of the database	Number
email	Unique identifier for the authentication	String
username	Unique identifier for the user inside the application	String
password	Unique string used for logging in	String
first_name	Name of the user	String
last_name	Surname of the user	String
is_superuser	If the user is authorized to administrative tasks	Boolean
is_active	If the user is using the application	Boolean
last_login	Last time the user logged in	Datetime
date_joined	Date the user registered	Datetime

Table 6.1: User's table

Strategy

Collection of all the strategies created by the users and contains all the operations made inside the application within a specific group of markets (Table 6.2). Every strategy is identified by an unique *id*, but other than that it can be distinguished by its name, belonging group of markets, and creator username. The strategy itself is composed by a number of market *positions* that the user decides to open or close. Each position can refer to either an option or a future. A strategy also has a "what-if" logic that enables the user to alter parameters of one or more positions, such as the initial or final price, volatility, the number of days before expiration, and many others, in order to run simulations on the strategy's performance.

Since strategies can be shared, each has a field called *published* that implies whether or not other people in the platform can see the details of the strategy. When a strategy is published, other people can interact with it by giving votes (*upvotes*, which can be seen as likes) or saving it for another moment (*bookmarks*).

A strategy not only can be shared in a traditional way, but it can also be added to a **club**, and this is tracked in the *clubs* field.

Club

A club is a place where a group of people gather and share strategies that the other members can look (Table 6.3). In particular a club is composed by *admins*, *members*

Field	Description	Data Type
_id	Unique identifier auto-generated by the database	ObjectId
userId	Username of the creator	String
groupId	Symbol of the group of markets which the strategy belongs	String
name	Name of the strategy given by the user	String
positions	Array of *position* objects	Array
created	The strategy's creation time and date	Datetime
disabled	If the strategy in the portfolio is disabled	Boolean
closed	If there are no open positions left in the strategy	Boolean
whatif	Object with all the simulation information	Object
clubs	List of clubs ObjectId in which the strategy is present	Array
published	Whether the strategy is public or private	Boolean
bookmarks	List of usernames that saved the strategy	Array
upvotes	List of usernames that voted the strategy	Array

Table 6.2: Strategy's table

and strategies. One club, other than being defined by its *id*, is also unique in the pair *creator_userId* and *name*. This means that one user can't create two clubs that have the same name. One club also can have an *image*, a *description* and some contact or social *links*.

Like the strategies, also clubs can be shared, and for this reason there is the *published* field that defines if the club can be seen by the public. They also have the *upvotes* like in the strategies.

Portfolio

This collection contains the user's portfolios that are a virtual account that keeps the overall balance of each user (Table 6.4). When a user creates a profile, a portfolio is created with an initial value of €100,000. This value changes through time based upon the performance of the strategies created by the user. Since the user and his portfolio currently have a one-to-one relationship, it is identifiable by his username.

A portfolio has a list of strategies and clubs that were created by the user, and these are inside the fields *strategies* and in *clubs*. The *other_strategies* field contains strategies that are not created by the user, but created by somebody else and connected in some way to the portfolio:

- The *saved* ones are those that the user have bookmarked while exploring other peoples' strategies.

Field	Description	Data Type
_id	Unique identifier auto-generated by the database	ObjectId
creator_userId	Username of the creator	String
name	Name of the club given by the user	String
img_path	Position of the image in the disk	String
created	The club's creation time and date	Datetime
description	Text that describes the club	String
links	List of objects containing contact information	Array
admins	List of usernames that manage the club	Array
nmembers	Number of members	Number
published	Whether the club is public or private	Boolean
upvotes	List of usernames that voted the club	Array
nstrategies	Number of strategies in the club	Number
strategies	List of ObjectIds of the strategies	Array

Table 6.3: Club's table

- *Opened* are the last 10 strategies that the user opened, and they can be both from other people or the user personal strategies.

- The *shared* field contains the strategies that some other user sent to the user owner of the portfolio. They are saved in objects with the id of the strategy and the username of the sender.

6.2 REST APIs

This section is about the most important APIs created to allow the user to interact with the data through the client application. These web services follow the REST guidelines and serve as an interface for the following elements: user, market, chain, future, strategy, portfolio and club. Same as before, we will describe only the most important ones that were involved in this work and the new ones.

Different HTTP methods have been used to define special URLs for each of them. To distinguish them from other web services that do not require authentication and are not a part of the REST API architecture, it is vital to highlight that all declared paths begin with the prefix */api*.

It is essential to note that all of the APIs are secure and connected through HTTPS before beginning the discussion of each one. In fact, an HTTP 401 Unauthorized error is returned for every request made by an unauthenticated or unauthorized client. A user,

Field	Description	Data Type
_id	Unique identifier auto-generated by the database	ObjectId
userId	Username of the holder	String
name	Name of the portfolio	String
value	Total balance	Number
currency	Currency used for the portfolio	String
created	The date of creation	Datetime
strategies	List of ObjectIds of the strategies associated to the portfolio	Array
clubs	List of ObjectIds of the clubs associated to the portfolio	Array
other_strategies	Object with different types of strategies arrays	Object
other_strategies .saved	List of ObjectIds of the strategies saved	Array
other_strategies .opened	History of the last strategies' ObjectIds opened	Array
other_strategies .shared	List of Objects of the strategies shared to this portfolio with the sender username	Array

Table 6.4: Portfolio's table

to be authenticated, goes through the login process in order to obtain his session *cookie*[2]. In the login process the server follows the default implementation provided by Django Authentication System [18] which consists in four steps:

1. With the user's username and password, the client application posts an HTTP request to the server.

2. The server queries the database for the username, hashes the provided login password, and compares it with the previously stored hashed password. If it is invalid, the client will receive an HTTP 401 Unauthorized error and access will be prohibited.

3. If the request is legitimate, the database will generate and store a session ID that specifically identifies the user's current session. The session ID may also have a time or date limit to prevent indefinite use of the user's session. It will then be attached to a cookie that will be sent back to the client as a response.

4. Cookies will be connected to any future client requests that requires user authentication, and the server will just need to verify that the session ID stored inside the cookies is still valid. If so, access is given; if not, a fresh login request is necessary.

[2]Small text file that the website stores on the user's device with different type of information.

In addition to this Django comes in handy also for other security features like the protection against Cross Site Request Forgeries (CSRF). In this case it offers a middleware[3] and a template tag [19] that help avoid the "login CSRF" attack, where when a malicious site manipulates a user's browser to log in to a site using another person's credentials.

In particular, a CSRF attack is a sort of attack in which a malicious web page, email, blog, instant message, or application directs a user's web browser to carry out an undesirable activity on a reliable site after the user has authenticated. Browser requests automatically contain all cookies, including session cookies, which is how a CSRF attack operates. As a result, if the user is authenticated on the website, the website is unable to tell the difference between genuine approved requests and fake authenticated ones. It follows that a challenge-response system that authenticates the identity and authorization of the requester is necessary in order to prevent this type of attack.

6.2.1 User APIs

These APIs are in charge of providing data about the user, as well as his portfolio, strategies and clubs. For the strategies and clubs, these services not only provide information, but they can also add, modify and delete elements already existing.

– User general purpose –

api/users/

Only the HTTP GET method is permitted, and it only returns information about the user who is now signed in. This is done without requiring any sensitive data to be sent in, like a username or email address because after authentication, Django actually saves its identifier (such as a username) in the session that is accessible for any subsequent HTTP requests made by the same user until the session expires.

api/users/search/

Only the HTTP GET method is available because it provides the list of all users. This is needed for searching for users to add to a club. It searches all users that match the first letters of the *username* that has been written in a form. This has been done by using a regular expression[4] that determines whether a string (username in the database) begins with the characters of a specified string that corresponds to the ones typed in the search form in the client interface.

[3]Type of software that permits various forms of connectivity or communication between various applications or application components.

[4]Also referred as regex or regexp, is a a string of characters that designates a text search pattern.

api/users/search/share/strategy/

Similar to the previous one, it searches for users that don't already have a specified strategy shared to them. It is used when a user want to share a strategy to another user inside the application. It works just like the previous API, with a regular expression on the username, but here users are filtered if the strategy has already being shared to them.

– Portfolio –

api/users/portfolio/

Only the HTTP GET method is permitted, and it returns information on the portfolio of the logged-in user. Since a user is limited to having a single wallet, as was already said, only the username already present in the session is needed.

– Strategies general purpose –

api/users/strategies/

For the strategies both HTTP GET and POST methods are available:

1. **GET method**: returns the list of strategies of the logged-in user. Each element has details that summarize the single strategy. The user can also filter the results by name.

2. **POST method**: it lets the user create a strategy to be added to his portfolio. Before the insertion in the database the server checks if the fields in the body are valid:

   - **name** field: is the name given by the user.
   - **groupId** field: identifier of the market group to which the strategy belongs.
   - **published** field: boolean field that indicates if the strategy is public or private. By default is set to false, that means not published.

If the validation procedure is completed successfully and without any problems, the new strategy is inserted into the database after being associated to an auto-generated id, and at the end the server responds with the produced id.

api/users/strategies/:id/

For the single strategy are available HTTP GET, POST and DELETE:

1. **GET method**: returns the data of a single strategy. After receiving the request, the server checks if the *id* exists in the database. If it exists, the server returns all the information of the strategy requested, otherwise it responds with an HTTP 404 Not Found error.

2. **POST method**: it lets the user modify one of his own strategies. The server validates the body of the request after having checked if the strategy exists. The user can modify these strategy fields:
   - **name**: change the name.
   - **positions**: add, modify or remove one or more positions.
   - **whatif**: update the "whatif" values.
   - **published**: change the visibility of the strategy.
3. **DELETE method**: the user can remove a owned strategy from the database. After this operation the strategy is removed permanently from the application without the possibility to recover it.

api/users/strategies/:id/:chart-id

By using the HTTP GET method, it is possible to get related chart information for each individual strategy. For the strategy Payoff and Greeks information, the chart-id path parameter may be enhanced with profit or greeks, respectively. The API always returns an array of JSON objects with data for each possible market price.

– Strategies organization –

api/users/strategies/:page-category

Returns strategies belonging to a specific category though a HTTP GET method. A query string can be used for filtering results according to strategy name, group of belonging and creator username. Some of the categories also have a POST method.

1. **GET method**: returns a different list of strategies according to the category requested. Just like in the GET of the strategies of the user, each element returned in the list has details that summarize the single strategy.
   - **explore** category: retrieves all the strategies that are public (*published* field equal to true).
   - **saved** category: retrieves the strategies that the user saved by giving them a bookmark.
   - **history** category: gets the last 10 strategies that the user opened. The strategies can be both of the user or of other users.
   - **share** category: recovers the strategies that had been shared with the logged-in user.
2. **POST method**: adds a strategy id to the correct list according to the category.
   - **history** category: once a strategy is opened, it is saved in the list of "opened" strategies. There is a max of 10 strategies that will be tracked, so the database first checks if there aren't already 10 strategies in the list and: if there aren't,

it adds the id at the top of the list, otherwise it "pulls" or removes the last one and adds the new one as first.

- **shared** category: when a user shares a strategy to another user, the recipient receive in his "shared" list another entry which is an object with the sender username and the strategy-id.

api/users/strategies/mypublic/

Returns, through a GET method, all the public strategies that the user has created. This is needed when the user wants to add a strategy to a club.

– Strategy votes –

api/users/strategies/:id/upvote/

It manages, through a POST method, the upvote of a strategy by adding or removing the username to the array of users that upvoted the strategy in question.

api/users/strategies/:id/bookmark/

It manages, through a POST method, the bookmark of a strategy by adding or removing the username to the array of users that bookmarked the strategy.

– Clubs general purpose –

api/users/clubs/

For the clubs both HTTP GET and POST are available:

1. **GET method**: returns the list of clubs of the logged-in user. Each club of the list has summary of the single club. The user can also filter the results by name.

2. **POST method**: creates a club to be added to the user's portfolio. When a club is created, the user creator is automatically also promoted to admin. Before adding it in the database, the server checks if the fields in the body are valid:

   - **name** field: name given by the user.
   - **published** field: boolean value that indicates if the club is public or private and by default is private.
   - **img_path** field: is a string with the directory path where is situated the display image. If no image is selected a default one is placed.
   - **description** field: is a brief description of the club with a limit of 256 characters. The description can also be empty.
   - **links** field: is an array of objects with the link information. The object is made by the link title and by the URL itself.

api/users/clubs/:page-category

Returns the clubs that belong to the category selected through a HTTP GET method. The results can be filtered by name and username of the creator with a query string inserted by the user. Both the categories have only the GET method:

- **joined** category: retrieves all clubs in which the user is member or admin but doesn't own.
- **explore** category: retrieves all the clubs that are public (published field equal to true).

api/users/clubs/names/

Retrieves all the names of the clubs created by the user through a GET. There is to note that the club names for the same user are unique. When creating or modifying a club a control is being done and in case of a duplicate the user is alerted and informed.

api/users/clubs/admin/administrate/

Returns all the clubs in which the user logged-in is admin with a GET method. This is needed when the user wants to share a strategy to a club because only admins can add or remove strategies.

api/users/clubs/:id/

Same as for the single club, there are HTTP GET, POST and DELETE methods.

1. **GET method**: returns the data of a single club. Upon receiving the request, the server checks if the club exists in the database. If it exists, the server returns all the information of the club, otherwise it responds with an HTTP 404 Not Found error.
2. **POST method**: it lets the user modify the club. After confirming that the club is present, the server verifies the request's body. The user can modify these fields:
   - **name**: change the name.
   - **published**: change the visibility.
   - **img_path**: replace the display image.
   - **description**: modify the text of the description.
   - **links**: add, remove or update a link.
3. **DELETE method**: the user can remove a owned club. This operation is allowed only if there aren't any members still in the club.

api/users/clubs/:creator_userId/:name

This HTTP GET is very similar to the GET of the one above, but is made not with the id, but with the pair creator_userId and name of the club. As mentioned before, a club name can't be duplicated by the same user, so this pair should be unique. The requests returns all the information about the club. The purpose is that one club should be possible to be retrieved from the club page url (which includes the username and the name), so that if someone shares the link, the page can load the club from only the information of the url. The decision was made so that the url is more readable since it contains who made the club and the club's name, instead of only the id like in the strategies.

– Club votes –

api/users/clubs/:id/upvote/

Just like in the strategies, it manages the upvotes through a POST method.

Clubs members and admins

api/users/clubs/:id/member/:username/

It manages members with a POST and DELETE method.

1. **POST method**: adds the user to the club by adding one entry to the user's *clubs* in the portfolio collection. It also increment the counter of members in the club collection. A user can join a club if some admin invites him or, if the club is public, he can join autonomously through the club's page.

2. **DELETE method**: this method is used when the user is being expelled or just wants to leave the club. This works both for users and admins. Similarly as the POST, it decrements the member counter in the club collection and removes the club id from the *clubs* field in the portfolio collection. If the user was an admin, it also removes the username from the *admins* field in the club collection.

api/users/clubs/:id/admin/:username/

Just like for the members, admins have the POST and DELETE method:

1. **POST method**: is used to add an admin to the club. When a user creates a club, he is automatically promoted to admin. At this point he has the power to promote other users that are members of the club. In particular this method just adds to the list of *admins* in the club collection the username of the user.

2. **DELETE method**: in the contrary to the previous one, this demotes an admin to a member by removing the username from the array of *admins* in the club.

Both methods don't have an impact to the portfolio collection, where there are all the clubs in which is present the user, because the user, if is not admin, still remains a member of the club. These methods just concern the array of admins in the club collection.

– Club strategies –

api/users/clubs/:id/strategies/:strategy-id

This API manages the club's strategies with a HTTP POST and DELETE methods:

1. **POST method**: simply adds the strategy id to the list of strategies of the club.
2. **DELETE method**: in the contrary this removes the strategy id form the list.

Both operates not only on the club collection, but also on the strategy one. In the strategy record, in the *clubs* field, is added or deleted the club id. This is done in order to know in which club the strategy is found.

Chapter 7

Client application

We had a look at the back-end of the application, in particular at the server, so now we will have a look at the front-end part, by discussing the client side. A client is a program that uses the services provided by the server. These services are accessed by the client by doing requests that we already saw in the previous chapter.

Front-end web development is the process of creating a website's graphical user interface using HTML, CSS, and JavaScript so that visitors can view and interact with it. One of the greatest challenges for a developer is to create a user interface that each of the users can easily understand and use. There is also the need to ensure that they have access to the necessary data for their application to function properly, so that they will find much easier to accomplish tasks. Consistent navigation, formatting, and design elements help the user quickly and easily navigate through the web application. Some users may have limited access to certain functionalities according to their role and permissions.

The client application's user interface will be covered in this chapter. Along with it there will be explained the data management that supports and interacts with the interface providing the necessary information and data.

7.1 User interface

As already told at the beginning of the document, the application front-end or interface is being developed using the React framework with the support of HTML and CSS.
The application is made up of many parts: Markets, Strategies, Portfolio and Clubs. The focus will be on the Strategies and Clubs pages since they have been the main components being developed for this thesis.
After the login the user will be presented with a navigation bar with all the parts and some user management buttons like the profile and the log-out button and under the bar the Market page is open by default.
From here the user can navigate through the pages with the names in the navigation bar.

7.1.1 Strategies

This page is divided in three columns. The first one indicates what types of strategies are displayed, the second one shows the strategies available for that category and the third is a blank space where the open strategies will be displayed (Figure 7.1).

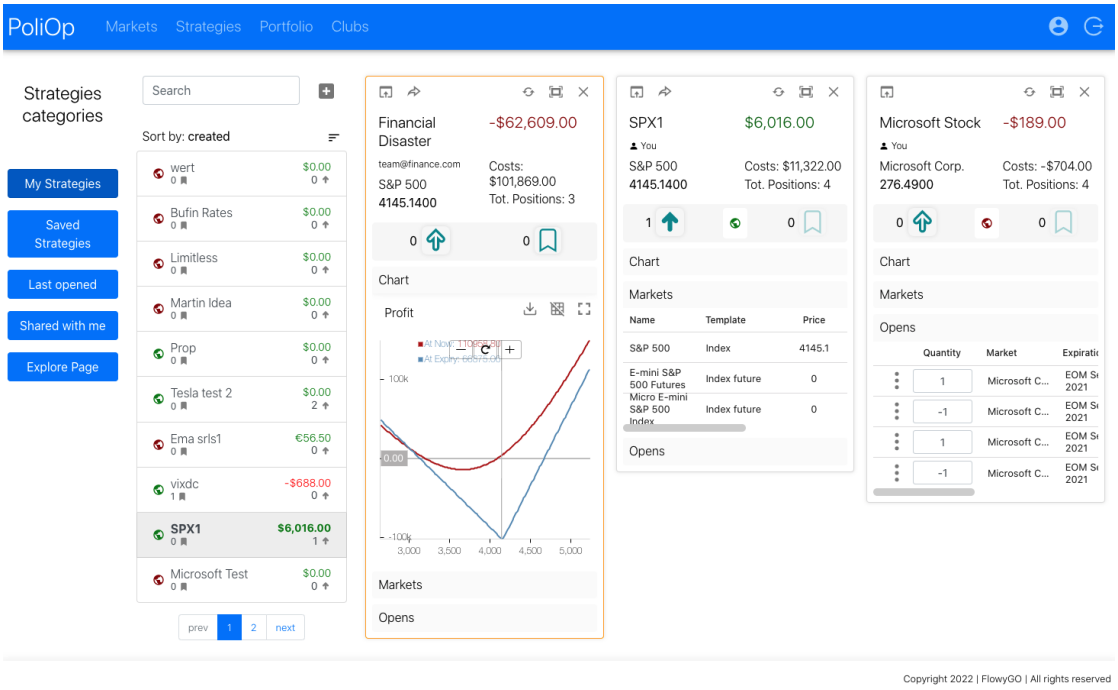


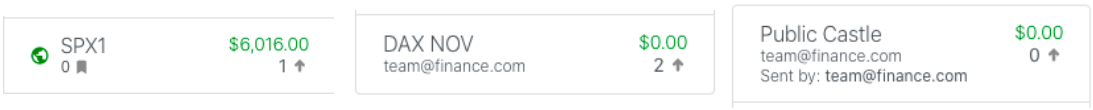
Figure 7.1: Strategy page with three open strategies

The first column, with the buttons aligned vertically, is where there are all the categories of strategies, these are:

- **My strategies**: category in which there are all the strategies that have been created by the user.
- **Saved strategies**: contains all the strategies which have a bookmark given by the user.
- **Last opened strategies**: when a strategy is opened as a card, it is saved here in order to track the latest opened strategies. It is a history of the strategies that the user has looked at.
- **Shared with me strategies**: place where the logged-in user receives strategies shared by other people.
- **Explore page**: here users can go and look at what strategies other users have created and made public.

According to the category selected, in the second column, are presented the strategies in a summarized format. Strategies can be searched by name or creator username and ordered by date, popularity, number of positions and number of clubs they are in. At the top there is also a button that opens a modal for creating a new strategy by inserting the name, choosing the visibility and the group it will belong to. Depending on what category is selected, the summary is different (Figure 7.2):

- **My strategies**: here the strategies are presented with: the visibility, the name, the profit, the number of bookmarks and upvotes.
- **Saved strategies**, **Last opened strategies** and **Explore page**: same as for the *My strategies* but instead of the upvotes, there is the username of the creator of the strategy.
- **Shared with me strategies**: also same as for the *Saved strategies* but it has also the username of the user that shared the strategy.



(a) My strategy (b) Saved, history and explore (c) Shared

Figure 7.2: Different types of representation for the strategies

Also, for the categories saved, last opened and shared, if another user's strategy becomes private from public, the table entry will disable the possibility to open the strategy, view the summary and will display: "[name of the strategy] has become private" as shown in (Figure: 7.3). In this way if a user decides to transform one of his strategies from public to private, the strategy information will only be accessible to him even if it is situated in other peoples feeds. All of this is not valid if the strategy in question is owned by the user, so if the strategy has become private but the user owns it, the information is visible and the strategy can be opened.

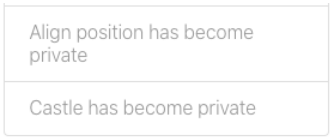


Figure 7.3: Private strategy seen by other users that are not the owner

The third column is like a whiteboard where strategies selected from the second column are opened as *Cards* in a three column grid. Here the user can open more than one strategy at the time with an overview of the trend and profits. Strategies can be opened from different categories and will stay open even if the category changes.

Card : the card represents a single strategy with at the top a brief summary on the trend, costs and popularity and at the bottom three expandable windows to view some details:

- the payoff graph, both at now and when it expires
- the group of markets in which it belongs
- the currently open positions

The card is different if is owned by the user logged-in or not:

- Owned strategies show also if the strategy is public or not and have the *bookmark* button disabled because the user can't save his own strategies.
- Not owned strategies have a colored border to have a distinct difference from the owned ones. In the *Markets* window, the link for accessing the market page is disabled just like the position numbers in the *Opens* window because the user does not own the strategy and have not authorization to modify it.

In the very top left of the card there are some control buttons and in the top right there is one to open the strategy in a page and the other one for sharing it.

Page : the strategy page simply has the card information displayed at full screen. This is useful if someone wants to have every data information of the card open and with more space (Figure 7.4). In this page the user can also share the strategy (if its visibility is public) and modify its attribute like name and visibility through the settings if the strategy is owned by him. If the strategy becomes private after being

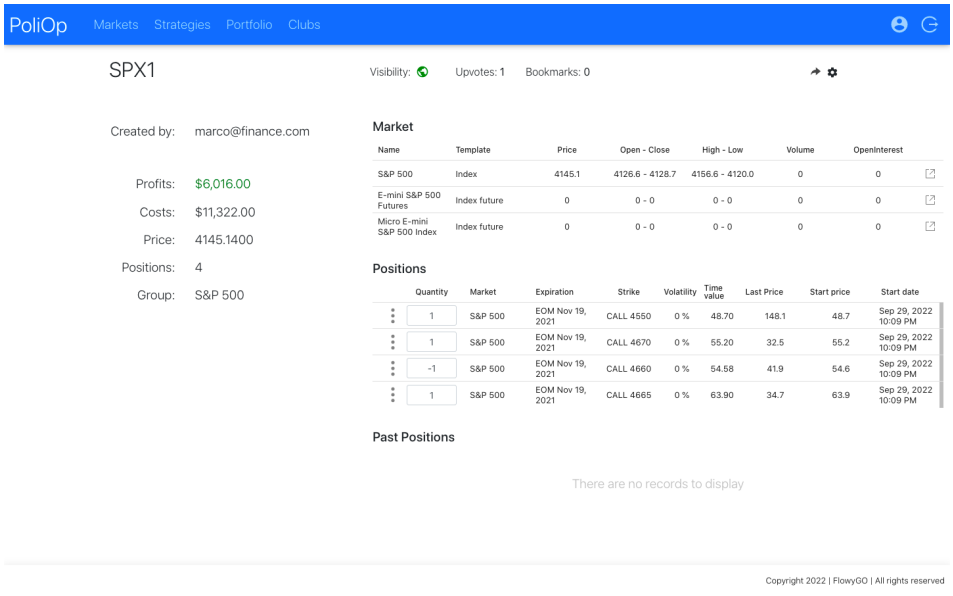


Figure 7.4: Page of a strategy

shared, the page will show a blank page with written that the strategy is private like shown in Figure 7.5.

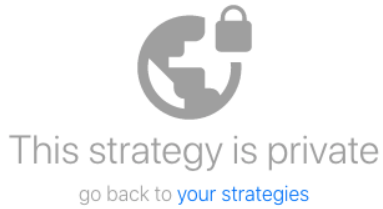
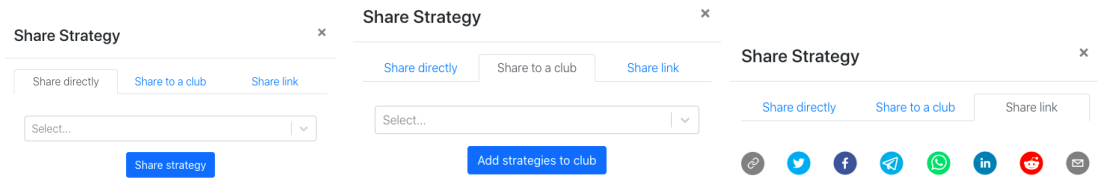


Figure 7.5: Accessing to a not owned private strategy page

Share Modal : this modal (Figure 7.6) can be accessed through the Card or the Page by clicking on the share button (arrow). The button is visible only if the strategy is public. Here the strategy can be shared:

- inside the application with another user, and the receiver will find it in the *Shared with me strategies* category.
- to a club only if the club is owned by the user or he is one of the admins.
- outside the application through the link of the strategy's page.



(a) Share directly (b) Share inside a club (c) Share on social media

Figure 7.6: Different tabs for sharing a strategy in different ways.

7.1.2 Clubs

The club page is composed by a column with the club strategies and the column with the list of clubs with a top part with some controls (Figure 7.9).

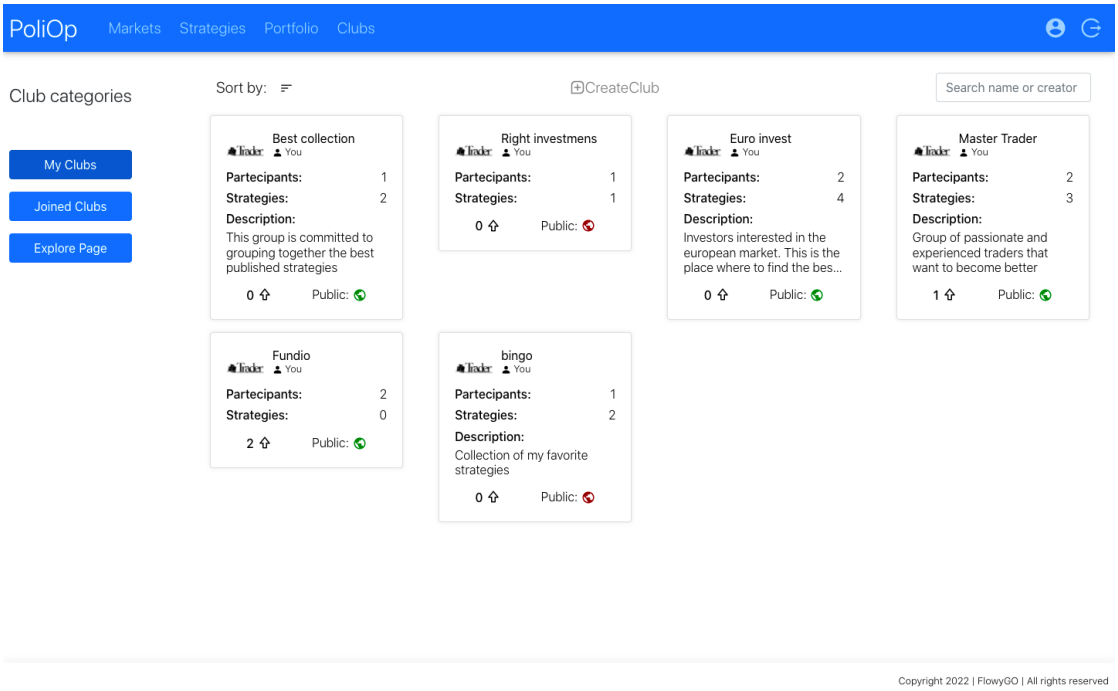


Figure 7.7: Club page

On the left, just like in the strategies, there are the clubs categories:

- **My Clubs**: category in which there are all the clubs created by the user.
- **Joined Clubs**: where all the clubs that the user joined are situated.
- **Explore Page**: place where users can go and join public clubs created by other users.

The rest of the page is occupied by the list of the clubs that are represented by cards with some information which include: name, creator username, number of members and strategies, description and number of upvotes that it received. In the *My Clubs* category the club also shows the visibility of the club.

At the top there are some controls to filter by name or creator and sort the clubs according to: date, upvotes, number of members and number of strategies. If the category selected is *My Clubs*, the user has at the top one more button that opens the modal for creating a club.

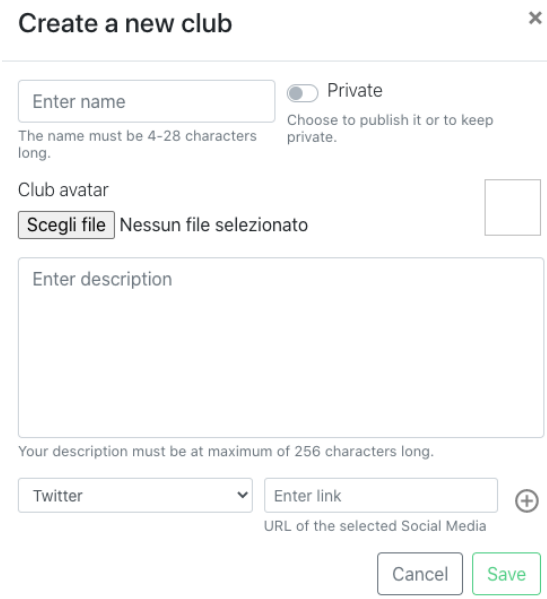


Figure 7.8: Modal for creating a club

Upon clicking one of the cards, the selected club page opens up (Figure 7.9).

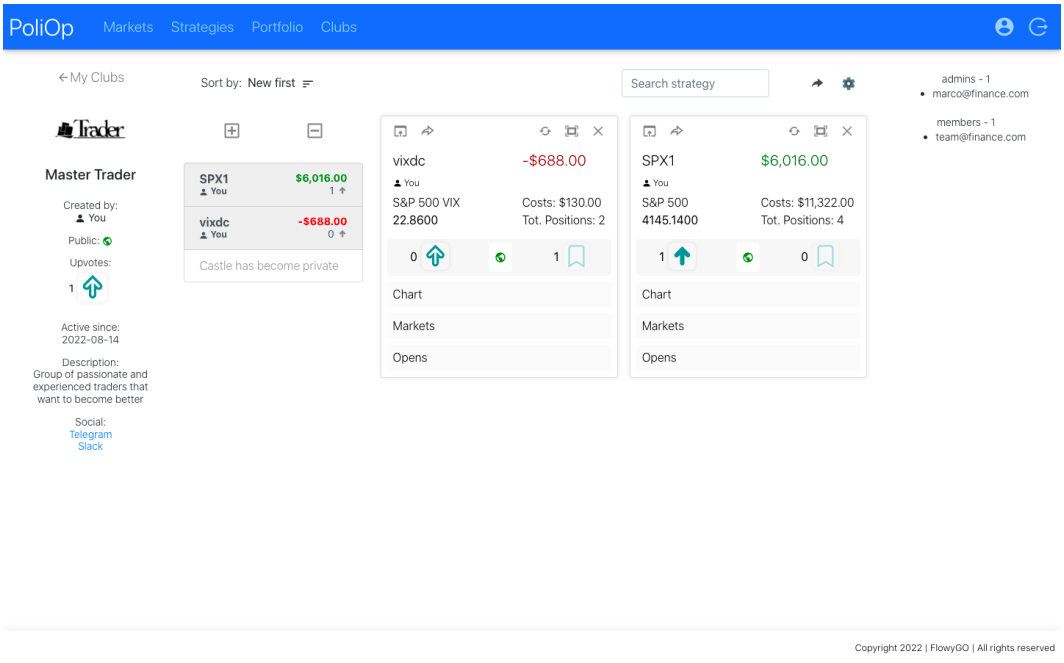


Figure 7.9: Page of an open club

This page is also divided in different parts:

1. The first column has a description of the club with all his attributes and at the top a button to go back to the page with the list of the clubs.

2. The second one has the list of all the strategies that had been added to club. They can be opened, filtered by name and creator and sorted by different parameters. If the user is an admin he will also have two buttons for adding and deleting strategies from the club. If the one of the strategies changes visibility from public to private, it will show the same message that was displayed in the strategies page. If this strategy is owned by the user, the table entry will be disabled, but the information of the summary will still be visible as show in Figure 7.10.

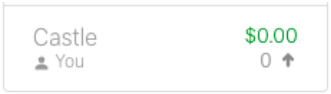


Figure 7.10: Owned private strategy inside a club

3. The third part, just like the strategies, is a empty space where strategies cards can be opened. The cards are the same as in the strategy page.

4. The last part on the left is composed by the list of the admins and members.

At the top left of the page there will be the button that let a new user join (Figure 7.11) and if he's already in, there will be a button with commands (Figure 7.9).

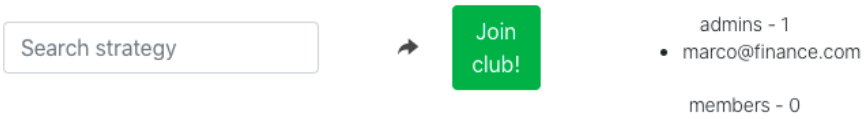


Figure 7.11: Button for joining a club

If the user is a member, the user can just leave the club through the button in the modal that appears when clicking the gear button in the top right corner. The leave club is the only available option for members when opening the dropdown.
If instead he is an admin (Figure 7.12), he can change the club settings, leave the club, close the club or, by accessing the manage members page (Figure 7.13), promote members to admin, demote admins to member or expel members.

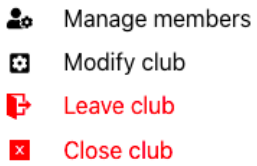


Figure 7.12: Club dropdown

An admin can't leave the club if he is the only admin and there are still members. He can close the club if he is the only admin and there are no members (Figure 7.14).

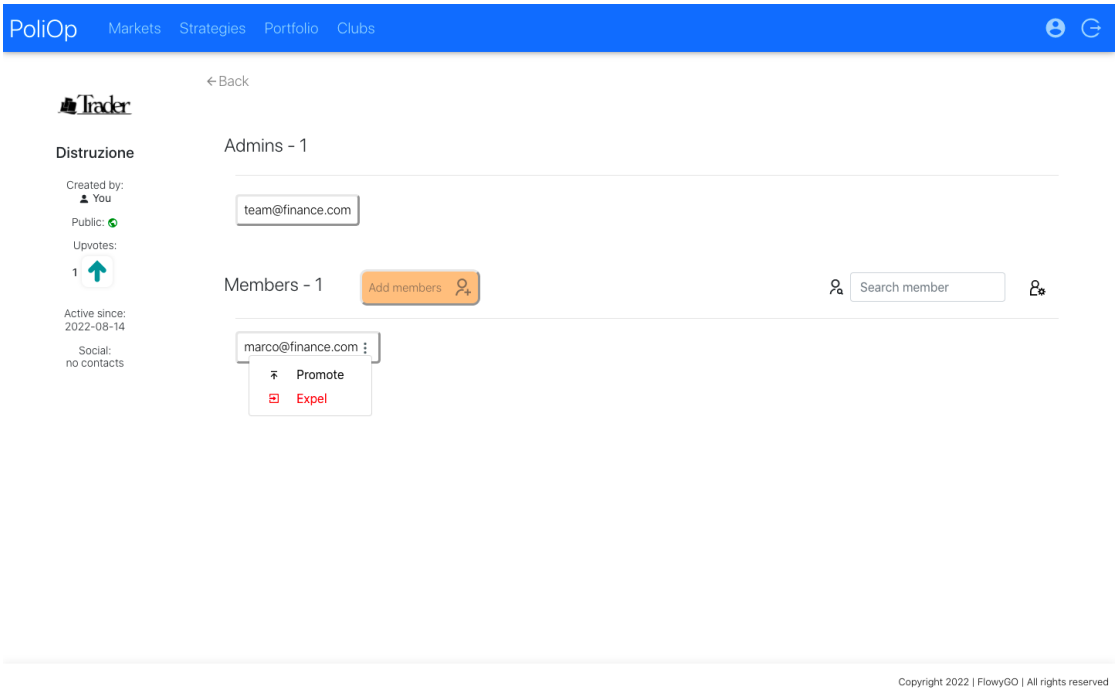


Figure 7.13: Manage members page

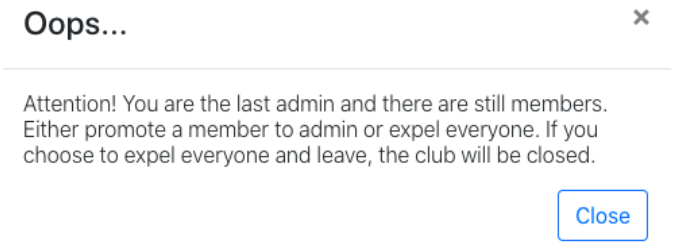


Figure 7.14: Modal of warning when the last admin is trying to leave.

If a user try to access to a private club via url, but the user is not either owner or member, the page will not be accessible and it will display a suggestion as in Figure 7.15.

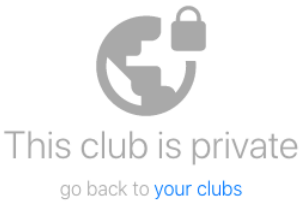


Figure 7.15: Private club page

7.2 Data management

Now we will have a look at how the data is handled in the front-end after being sent from the server. The managements has been done using the React Redux library of the React framework. It works by having one single data source called *Store* which is accessible by every component (Figure 7.16).

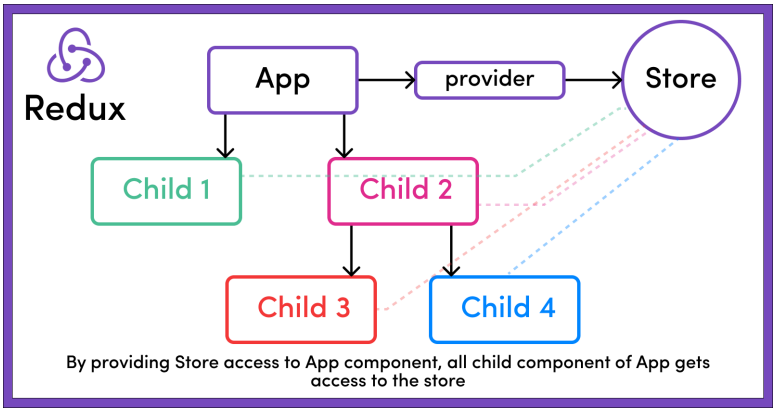


Figure 7.16: React-Redux Store, image from freeCodeCamp

It comes handy in order to avoid duplication of inconsistent data between different components and is also very useful for simplifying the development and the maintenance since it is strict about the structure of the code, which makes it simpler for someone who is familiar with Redux to comprehend the organization of any application.

Another key feature is that the state is always predictable, this is because reducers are pure functions, they always deliver the same result when the same state and action are supplied to them.

7.2.1 React Redux Store

A **store** is an immutable object tree, or a state container, which is in JSON format and is stored inside the browser and whose access is restricted to the application that instantiated it. Only one store can exist for one application and it's lifecycle is related to the program one.

In our case the object consists of other nested objects, one for each section of the application. In this way each section has its own data which corresponds to its object with its elements inside.

```
{
    "app": {
        "mode": "Authenticated",
        "wait": 0, "open": false,
        "loading": false, "failed": false,
        "groups": [{"symbol": "^GSPC", "name": "S&P 500","type": "index"}, ...],
        "exchanges": [{"symbol": "CBOE", "days": [1,2,3,4,5],"name": ...}, ...],
    },
```

```
    "user": {"email": "team@finance.com", "name": ...},
    "modal": {"show": false, "title": ""},
    "markets": { "open": true, "loading": false, "failed": false, "search": "", ... },
    "strategies": {
        "search": "",
        "results": [...],
        "charts": [...],
        "page": "My Strategies",
        "pagination": {},
        "strategy": {"markets":[], "upvotes": [], "published": false, ...}
        ...
    },
    "portfolio": {
        "currency": "EUR",
        "value": 0, "stats": {},
        "chart": {...},
        "strategies": [...],
    }
    "clubs": {
        "results": [...]
        "page": "My Clubs",
        "club": {"_id": "", "name": "", "admins": [], ...}
        "pagination": {},
        ...
    }
}
```

The store is a read/only state, so in order to change it the library provides the *Reducer* function. This function accept as input the current state of the store and one *action* which is an object that describes what happened. The Reducer is called like that because combines the two inputs to produce the new instance of the updated state (Figure 7.17).

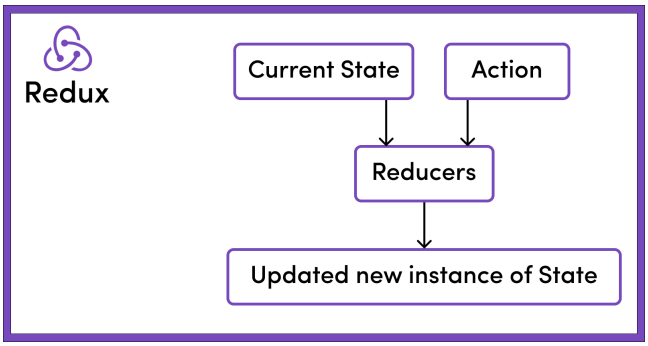


Figure 7.17: React-Redux Reducer, image from freeCodeCamp

Actions are often created following a user interaction with the view that results in an event, or within an asynchronous process.

Additionally, it was used a middleware of the React Redux library called *Effect* which manages actions and allows them to not necessary being handled in a Reducer. The Effect lets perform asynchronous tasks like HTTP calls to external APIs.

7.2.2 Data flow

Now lets examine what happens when the user interacts with the User Interface. A unidirectional flow of data between the views and the store is obtained and ensured by using the set of elements describes before. Lets make a real example (shown in Figure 7.18) in which the user wants to change a "club category".

As a first action, the user clicks on a category from the list that launches an event (1).

The event is then intercepted by its handler that creates a JSON object with: the requested action (LOAD_CLUBS), and the information of the event (that contains some information about the category clicked). This object then becomes an input of the *dispatch* function provided by Redux, which is the only way of updating the state (2).

The middleware *Effect* receives the action as input and decides how to proceed with the following tasks based on the type of action. In this case it will call a HTTP GET to the API /users/clubs/:category (3).

The asynchronous nature of the call requires the input of two callback functions, one to handle the response and the other to handle any errors. The dispatch function will be applied in both scenarios to communicate the server's response via an *action* (such as LOAD_CLUBS_SUCCESS or LOAD_CLUBS_FAILED) (4).

At this point the Reducer will take the action and the latest state in the store and based on the action type, the store will be updated with the new data of the API (5).

Finally the view will be updated to show the new information by React (6).

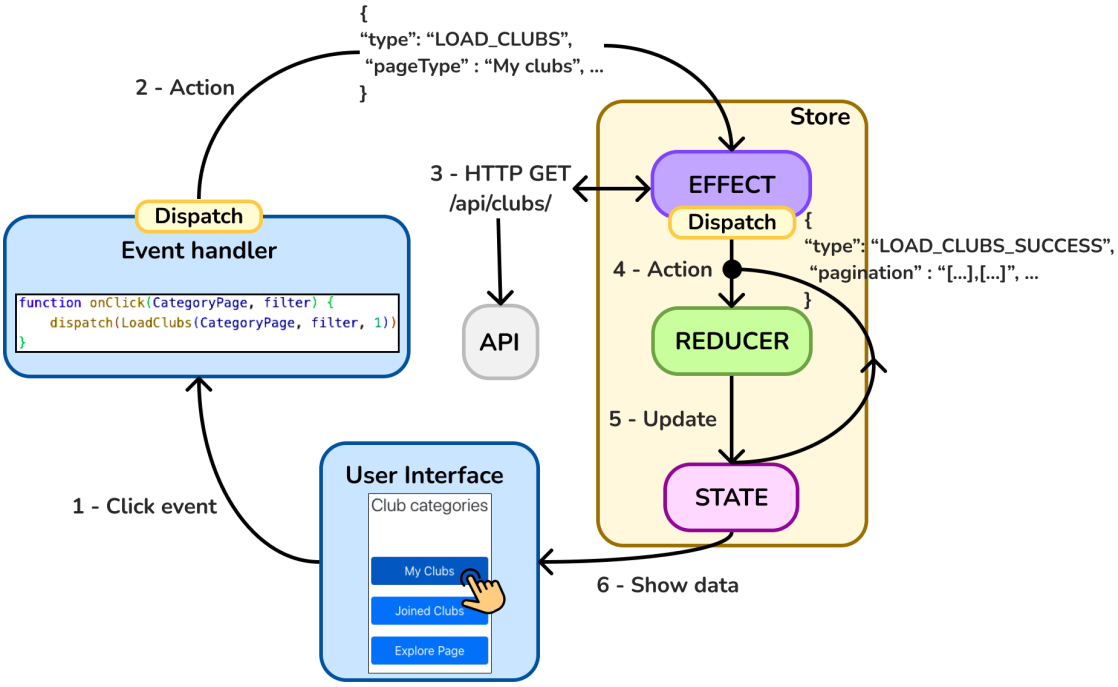


Figure 7.18: React-Redux data flow example

Chapter 8

Use cases

A use case is a detailed explanation of how users will utilize your website to complete tasks. It describes how a system behaves in response to a request from the user's perspective. A use case starts with a goal, describes the actions that users take, and specifies what the system response is to each action. So each use case is described as a series of actions that start with the user's objective and finish when that objective is achieved. Use cases are valuable because they help to describe how the system should function and ensure that there is a match between expectations and the delivered response.

It provides clear instructions on how a user should utilize a system or software and identifies common tasks that should be performed through it. It is an essential tool for software development as it helps with both conceptualization and design patterns. It can also help with user training and testing of new systems or software modules.

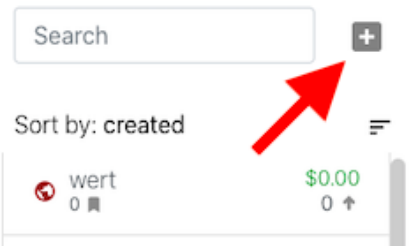
A use case can also be defined as a set of scenarios that collectively describes, from the perspective of the system, what a set of actors can do with it and what outcomes can be expected from a particular sequence of interactions. Use cases can also be defined as a set of functional requirements specifying what tasks and services should be supported by the system from the perspective of the system, what a set of actors can do with it, and what outcomes can be expected from a particular sequence of interactions.

Every use case should at least include: who is using the system, called an "actor," the user's goal, and the steps that he takes to accomplish it with the responses from the system. A use case is described as scenarios, also called flows, that describes the system's behavior in different conditions. The main one is called *Main success scenarios* that is when everything goes well. Then there are *Alternative paths* that are variation of the main one. A scenario also can't occur if the *preconditions* are not satisfied.

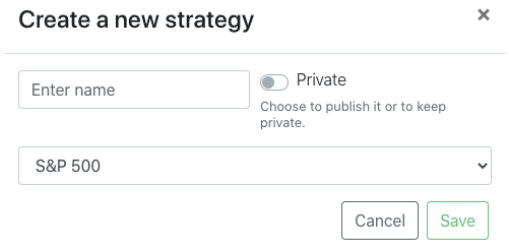
Create a strategy : in this use case the user creates a new strategy (Table 8.1). In the strategies page, if the *My strategies* category is selected, the user can click on the plus button (Figure 8.1a) and, through the modal, decide the name, the market group and the visibility (Figure 8.1b). Once confirmed the chooses with the save button of the modal, a HTTP POST will be called with the information provided by the user. The information is then saved in the database and the user will be able to observe the new strategy in the list of his strategies after an HTTP GET called automatically updates the list. In case of some error, the user will be informed by a error modal.

Use Case 1	Create strategy
Actor	User
Pre-condition	The user is authenticated and is in the *Strategies* page in the *My strategies* section.
Post-condition	The user has created a new strategy that can see in the list of owned strategies.
Basic flow	The user clicks on the plus button, inserts a valid name, selects a market group and the visibility in the modal opened. Once finished confirms his choices by creating the strategy.
Alternative flow	The user inserts a name that is not valid and the system doesn't let him create the strategy.

Table 8.1: Use case: create strategy



(a) Create strategy button.



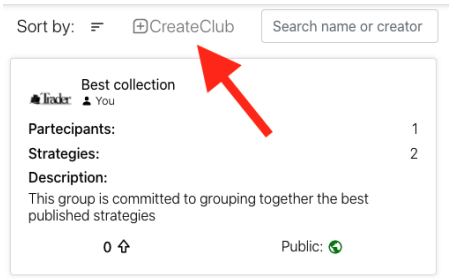
(b) Create strategy modal.

Figure 8.1: Create strategy stages

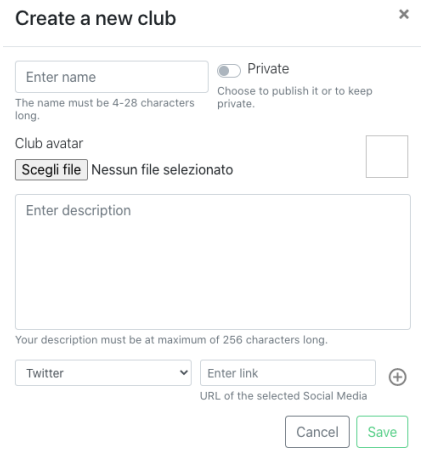
Create a club similar to the previous use case, the user creates a new club in the clubs page only if the *My Clubs* category is selected (Table 8.2). The user can click on the *Create Club* button (Figure 8.2a) and, through the modal, decide the name, the visibility, display image, description and links (Figure 8.2b). He's obliged only to have a name in order to continue, the other fields are optional. An HTTP POST will be called with the information provided by the user when the user clicks on the save button of the modal. In this way the information is saved in the database. At this point a HTTP GET will retrieve the list of clubs updated with the new club just created. In case of some error in the process, the user will be informed by a error modal.

Use Case 2	Create a club
Actor	User
Pre-condition	The user is authenticated and is in the *Clubs* page in the *My Clubs* category.
Post-condition	The user finds the new club in the list of his own clubs.
Basic flow	The user clicks on the *Create Club* button, inserts a valid name in the modal and saves. Once finished confirms his choices by creating the strategy.
Alternative flow	The user inserts a name that is not valid and the system doesn't let him create the club.

Table 8.2: Use case: create club



(a) Create club button.



(b) Create club modal.

Figure 8.2: Create club stages

Share a strategy to a club in this use case the user wants to add a strategy to a club in which he's one of the admins (Table 8.3). First the user needs to select a public strategy. Once chosen and opened as a card, he has to click on the arrow at the top of the card (Figure 8.3). A modal will appear with three tab. He will choose the second that says *Share to a club*, click the search bar and either choose a club from the list or write for the name and click on it and finally click on the *Add strategies to club* button (Figure 8.3). At this point a HTTP POST will be performed to save the id of the strategy in the club collection and the strategy will be visible in the club page.

Use Case 3	Share a strategy to a club
Actor	User
Pre-condition	The user is authenticated, has the *Strategy* page open, has at least a public strategy in the list, and is admin in at least one club.
Post-condition	The user finds the strategy in the list of strategies in the club chosen.
Basic flow	The user clicks on a public strategy that he wants to share. This will open as a card and present the share button. Once clicked the share button a modal will appear with three tabs. The user chooses the *Share to a club* one and selects a club from the search bar after clicking on it. After the selected club is present in the search bar he clicks on the button below to add the strategy.
Alternative flow 1	The user selects a strategy that is not public and doesn't have the share button.
Alternative flow 2	In the sharing modal the user doesn't select the right tab and shares the strategy to a user instead of a club.

Table 8.3: Use case: share strategy to a club

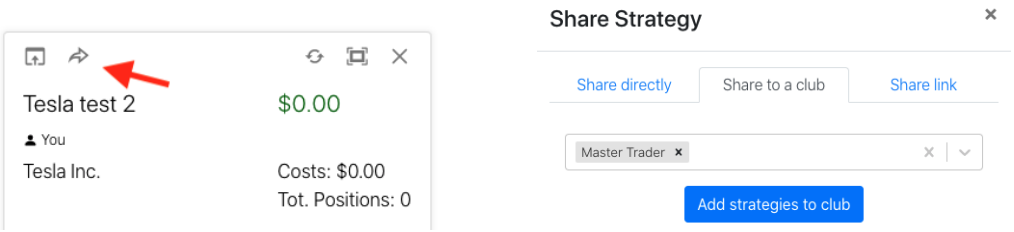


Figure 8.3: Share strategy to a club stages

Close a club in this case the user wants to delete one of his own clubs (Table 8.4). The user has to open the club that he wants to close, click on the settings button in the top corner and select "close club", which is the last option (Figure 8.4). This option appears only under the circumstance of the user being admin and the club being without any members. After the club is closed the user will be redirected to the clubs page with all his other clubs displayed. When the button is clicked a HTTP DELETE will be called, and this will go in the database and remove every information of the club by eliminating the entry in the club collection and the id saved in other collections.
If there are some members or other admins they have to be expelled either one by one or all together.

Use Case 4	Close a club
Actor	User
Pre-condition	The user is authenticated, has the club page open, is the only admin and there are no members.
Post-condition	The club is being deleted and disappears from the list of owned clubs.
Basic flow	The user clicks on the settings button in the top corner and on the "close club" in the drop-down menu.
Alternative flow 1	The club still has members or admins, so the "close club" button in the drop-down doesn't show up.

Table 8.4: Use case: close a club

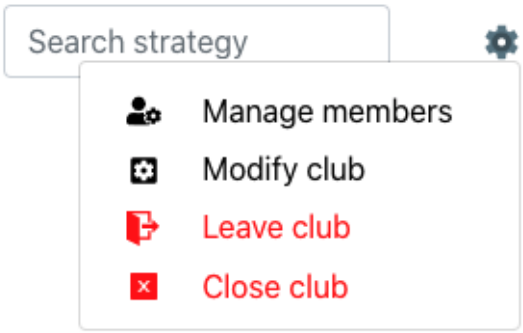


Figure 8.4: Close club button location

Chapter 9

Conclusions

Since strategies on derivatives can become complicated for retail investors, it is therefore important, in order to encourage investments, to observe strategies that other people, maybe more experienced, have made available to the public.

In this context is inserted this work, which aim is to create sharing features in a platform in which users have the ability to conduct analysis of the derivatives market and create operational strategies on future and option contracts.
This has been achieved in this thesis by giving the user the possibility to make his strategies publicly visible and share them not only through a web link, but also by sending them directly to another user or to a group of people inside the application.

Aside from that, users can interact with strategies by saving or upvoting them to let others know that they liked them. This means that the best strategies are quantified by the number of votes received.

It is also possible to form groups of people, called "clubs", within which investment strategies can be shared. The entry allowance for a club can be free or not depending on whether the club is open to the public or private. Just like the strategies, clubs can be evaluated by the number of votes or members.

From a technological perspective, the implemented technologies serve as strong foundations for future functional extensions with advancements. The ones utilized, like React, Django, and MongoDB, are already the most famous and used among the "over the top" companies. These tools are well documented and supported, and are continuously updated with new features and upgrades.

The sharing features of the application allow users to study financial data by observing what other traders are doing, enabling investors to track the trading activities of both novice and experienced traders.

Having considered everything, in conclusion, this application can allow to welcome a broader number of users in the complex derivatives world, as it tries to fill the gap between expert and novice investors.

9.1 Future work

For what concerns future works, the application can be extended on different fronts for the social part. For example, a feature missing is the user's profile that can be looked at by other people to study the different approaches and decisions taken in the past and the performances of these decisions and tactics. This should be different from the portfolio page since it would be open to the public. On this page, for example, people can have a look at the public strategies and clubs created by the user, and maybe have a brief summary of the user's performances, like a dashboard.

Also, the possibility to have *friends*, which are users that accepted the invite to be the user's friend, with which to compare different approaches, ideas and performances. They can also be the only *type* of user with which a user can interact, like sending strategies or inviting to a club.

Notifications both in the app or via mail, can be added in order to notify the user about certain tasks and actions that have been performed. In particular some essential ones would be some responses from the server when something is done. For example, as for now, when sending a strategy to another user, the sender doesn't know if it has arrived or not except from the page loading. Notification via mail can be used for more important and big events, like the announcement that a club, in which the user is present, will be closed.

Bibliography

[1] Campbell R. Harvey. *Shares*. Accessed: 2022-08-19. URL: https://financial-dictionary.thefreedictionary.com/Shares.

[2] Farlex Financial Dictionary. *Financial Instrument definition*. Accessed: 2022-08-19. URL: https://financial-dictionary.thefreedictionary.com/Financial+instrument.

[3] Campbell R. Harvey. *Stock*. Accessed: 2022-08-19. URL: https://financial-dictionary.thefreedictionary.com/stock.

[4] Campbell R. Harvey. *Bond*. Accessed: 2022-08-19. URL: https://financial-dictionary.thefreedictionary.com/bond.

[5] Campbell R. Harvey. *Commodity*. Accessed: 2022-08-19. URL: https://financial-dictionary.thefreedictionary.com/commodity.

[6] Office of the Comptroller of the Currency. *Financial Markets*. Accessed: 2022-08-16. URL: https://www.occ.treas.gov/topics/supervision-and-examination/capital-markets/financial-markets/index-financial-markets.html#:~:text=Financial%20Markets%20include%20any%20place,who%20have%20capital%20to%20invest..

[7] Will Kenton. *Exchange*. Accessed: 2022-08-16. URL: https://www.investopedia.com/terms/e/exchange.asp.

[8] Gordon Scott. *CBOE Options Exchange*. Accessed: 2022-08-16. URL: https://www.investopedia.com/terms/c/cboe.asp.

[9] James Chen. *Chicago Mercantile Exchange (CME)*. Accessed: 2022-08-16. URL: https://www.investopedia.com/terms/c/cme.asp.

[10] Rajeev Dhir. *EUREX*. Accessed: 2022-08-16. URL: https://www.investopedia.com/terms/e/eurex.asp.

[11] World Economic Forum. *Newly Empowered Investors*. Accessed: 2022-08-16. URL: https://reports.weforum.org/future-of-financial-services-2015/newly-empowered-investors/?doing_wp_cron=1662191016.5359039306640625000000.

[12] Matthias Pelster and Annette Hofmann. *About the Fear of Reputational Loss: Social Trading and the Disposition Effect*. Page 26, accessed: 2022-08-20. URL: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3057533.

[13] Ben Shneiderman et al. “Designing the User Interface: Strategies for Effective Human-Computer Interaction”. In: 6th ed. New York: Pearson, 2016. Chap. 8.

[14] Evernine. *MICRO INTERACTIONS*. Accessed: 2022-09-6. URL: https://evernine-group.de/en/designtrends-2022-micro-interactions-optimized-user-experience/.

[15] Docker Documentation. *Docker Overview*. Accessed: 2022-08-27. URL: https://docs.docker.com/get-started/overview/#the-docker-platform.

[16] MongoDB Documentation. *MongoDB Introduction*. Accessed: 2022-08-27. URL: https://www.mongodb.com/docs/manual/introduction/.

[17] Benjamin Anderson Brad Nicholson. *SQL vs. NoSQL Databases: What's the Difference?* Accessed: 2022-08-27. URL: https://www.ibm.com/cloud/blog/sql-vs-nosql.

[18] Django Documentation. *Using the Django authentication system*. Accessed: 2022-08-30. URL: https://docs.djangoproject.com/en/3.2/topics/auth/default/.

[19] Django Documentation. *Cross Site Request Forgery protection*. Accessed: 2022-08-30. URL: https://docs.djangoproject.com/en/3.2/ref/csrf/.