Politecnico di Torino

Politecnico di Torino

Master Degree course in Computer Engineering
A.a. 2022/2023
Sessione di Laurea 07/2023

Study and functional improvements development for an Options Strategy Builder Platform

Relatori:
Alessandro Fiori

Candidati:
Ruben Rinaldi

Contents

List of Figures

Chapter 1

Introduction

Nowadays, the accessibility of financial instruments is widespread. Banks and online brokers provide a wide diversity of financial instruments. Some of these instruments are particularly sophisticated not only in terms of their use, but also in a conceptual aspect that requires the end user to be familiar with their functioning and to have a general understanding of the global market. The original project aimed to implement a web app for evaluating investment strategies on both American and European financial options. Trading financial options is a popular method of investing that allows individuals to buy or sell the right to purchase or sell an underlying asset at a specific price and time. However, the process of evaluating financial options is complex and unintuitive, and sometimes the appropriate tools are not provided to make an accurate assessment. This platform aims to provide the tools necessary to describe the characteristics of an investment strategy based on one or more options that relate to a specific underlying asset, in order to bring the user to a deeper awareness.

The primary emphasis of this thesis is the design and implementation

of extra features for usage on the front end of a web platform dedicated to the study of financial derivatives. Derivatives in the financial market are complex financial products that call for a great deal of knowledge and experience in the relevant industry. Financial options, in essence, takes use of a kind of leverage that enables the investor to place a stake on the price of a stock without having to purchase or sell it directly. Instead, the investor merely incurs a small fee to the seller in order to engage in the deal. The objective is to further enhance the functioning of the application's front-end by including advanced visualizations and partially restructuring the server architecture by splitting the frontend and backend into two different units in order to improve development independence.

1.1 Relevant Studies and Implementations

The platform evaluates different indicators related to financial options, the study of these instruments and associated strategies has been expanding and evolving since the 1970s. The first significant contribution to option pricing was conceptualized in 1973 by Fischer Black, Myron Scholes and Robert Merton, known as the Black - Scholes model [3], and is currently the model that is used to calculate the profitability of call and put options on the platform. The model was created for the calculation of European options, thus options are exercisable only at the expiry of that contract.

However, the Black - Scholes model has some limitations in its validity assumptions such as the constant interest rate and the geometric

Brownian motion of the share price and does not take into account certain factors such as dividend payments. For these reasons, subsequent studies have extended the validity range of the previous model that takes into account the possibility of early exercise of American options, indeed, where the contract can be closed before expiry.

The topic of American option pricing is gaining wide acceptance in the literature today. The Bjerksund-Stensland model [2], developed by Per Bjerksund and Gard Stensland in 2002, is the model implemented in the platform for calculating the price of American options. The latter takes into account the early exercise feature and provides a closed-form formula, a relatively simple and fast way to price options with a discrete level of accuracy. Other studies suggest different approaches for calculating options such as the Barone-Adesi and Whaley model [1], developed by Giorgio Barone-Adesi and Robert Whaley in 1987. It is an approximation method that uses the Black-Scholes model to price American options. This model is relatively simple to implement, but it is less accurate than other methods, especially for options that are close to the expiration date.

Other models based on Monte Carlo simulations such as the ones discussed in this book [12] are more accurate, however they are not are not massively usable in a web platform as the calculation time would be too long under some circumstances, making the platform unusable.

1.2 Competitor

Another significant factor for which the original project was launched was the lack of a tool on the market that is easily accessible to all players in the options market. The motivation behind the choice of platform functionality is the possibility to provide a tool that includes various functionalities to support traders in the choice and evaluation of options strategy, which are not jointly present in other platforms present in the market today.

Platforms that allow strategy analysis, especially through the use of visual tools such as charts and indicators, are not widely available, very often the platforms are sold at a high subscription price, which is not affordable for new market entrants and often lacks useful elements for evaluating strategy and assessing trends of the market with a global vision but only offering a partial view.

Option Vue

OptionVue is a software developed specifically for the purpose of trading options. It includes a number of useful features, including a highly customizable charting interface, an option matrix, and tools for determining which investment strategy offers the greatest potential return.

The program computes profits and losses, which are shown via some sophisticated graphics, and it enables comparison with many alternative options strategies. The monthly membership fee is rather expensive, and there are certain limits on the degree of type of Greeks that may be used.

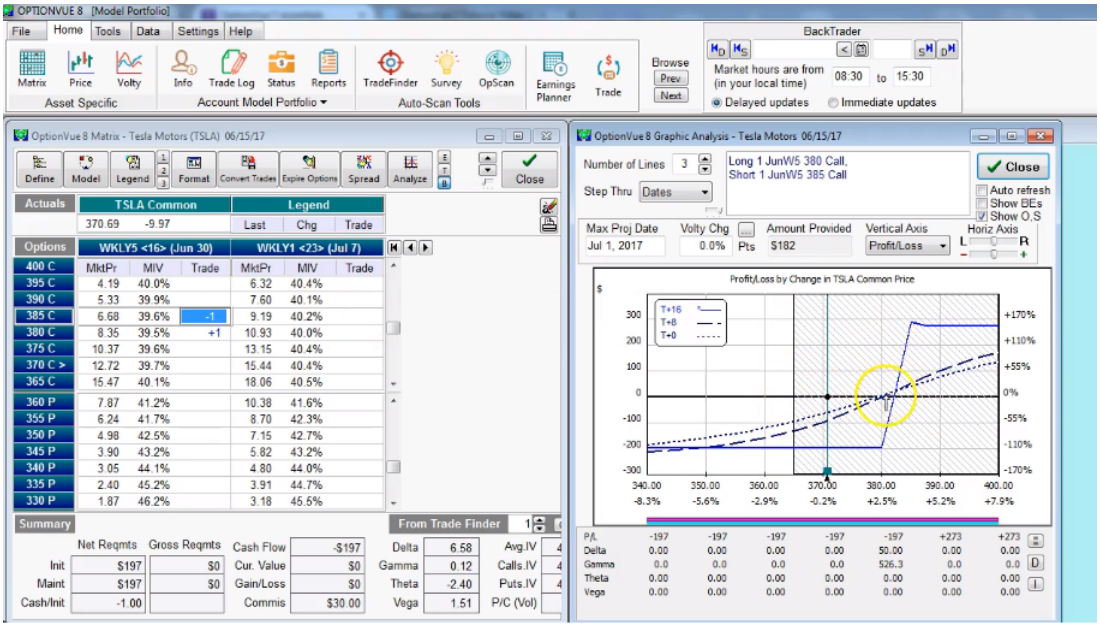


Figure 1.1: OptionVue Dashboard

OptionStrat

OptionStrat is a web-based platform that provides a range of tools and resources for analysing and trading options. This intuitive platform guarantees a good level of comprehensiveness in terms of the instruments provided. Furthermore, it offers a range of educational resources, including tutorials, webinars, and articles to help traders learn more about options trading and improve their trading skills. It also includes interesting features such as position flow and a profit chance indicator. However, it has limitations in terms of the availability of European options.

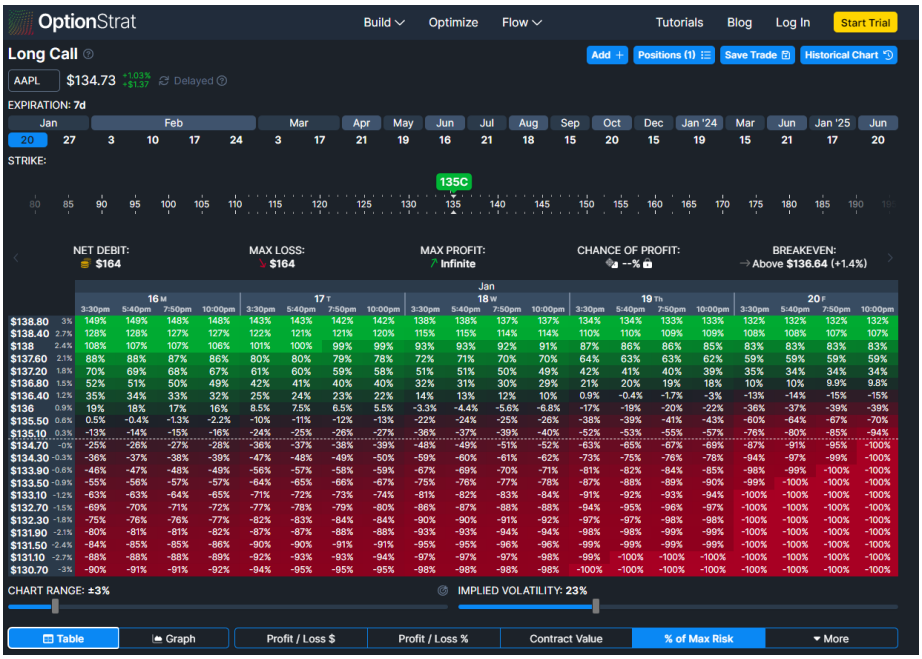


Figure 1.2: Optionstrat Strategy Builder

BeeTrader

Beetrader is a very complete installable software for analysing strategies, offering charts and indicators for technical analysis, strategy comparators and scenario analysis. It provides a wide range of features and tools to help traders analyse and trade financial markets. Moreover, It also offers the possibility to connect the software to your broker and rapidly interact directly with it. It is a versatile and powerful trading platform that can meet the needs of both novice and professional traders, however, the product becomes very expensive for an individual investor who wants to approach the world of options trading.

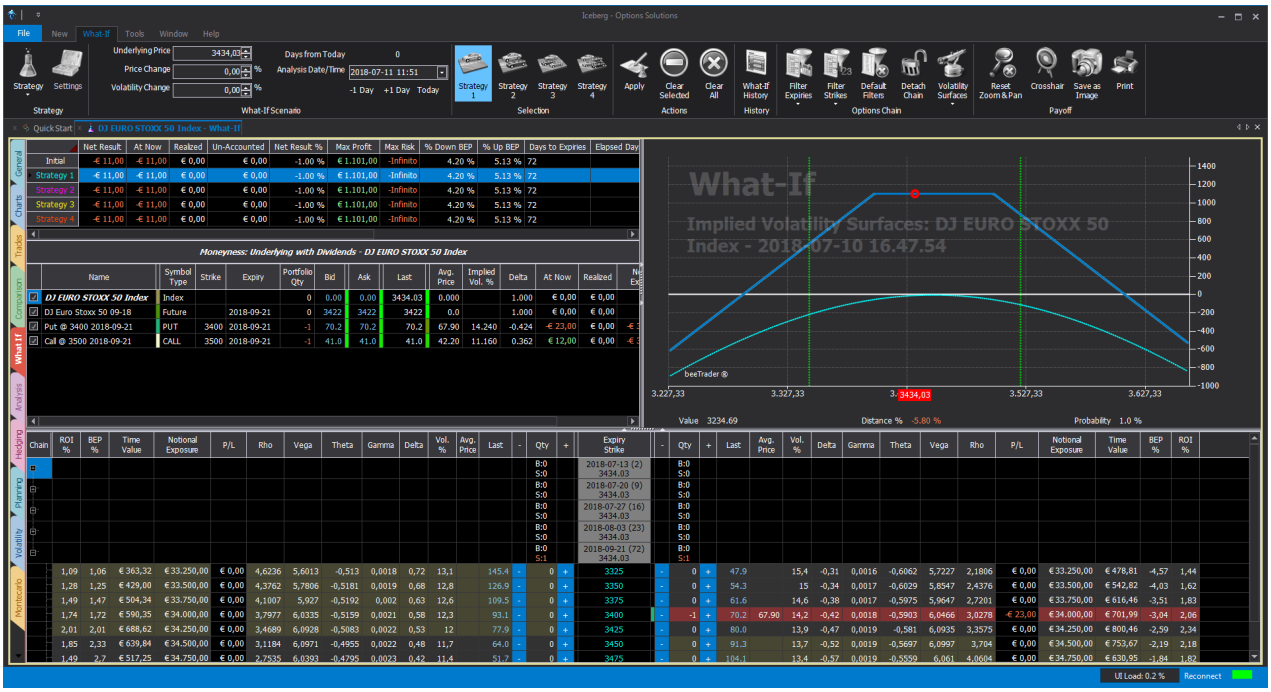


Figure 1.3: Beetrader Dashboard

Chapter 2

Option Trading

The purpose of this chapter is devoted to a general description of the financial instruments used by the platform with a focus on financial options.

2.1 Markets

There is a wide variety of financial instruments, and each kind has its own specific market that facilitates the buying and selling of that particular asset. The most important markets trade a wide variety of financial assets, including stocks, bonds, foreign exchange, and derivatives, among several others. This section gives a quick introduction to the most important financial assets and types of markets.

Stocks: Shares of equity in the property of a publicly traded corporation are referred to as stocks. Investors may purchase and sell stocks on the stock market, as well as earn dividends from the companies whose stocks they own.

Indexes: An index is a collection of financial products that represents the market for investable assets in a certain country or area. They are gathered in order to monitor the price performance of the market they each represent. The index's assets are typically weighted based on their market capitalization. Some of the most noted are the SP500, MSCI World, Dow Jones Industrial Average, DAX and others.

Bonds: Bonds are a portion of a business's (or state's) debt that is represented by a security, and the buyer of the bond merely becomes a creditor of the firm or state. The latter receives the amounts that were subscribed to in addition to the interest that is defined in the contract at a certain maturity date.

Derivatives: Because they draw their value from the fluctuation in the value of an asset or the happening of a future event, derivatives are so named. The asset or event, which may be of any sort or kind, serves as the derivative product's "underlying". There are several derivatives in the financial sector, including: Options, futures, forwards, swaps

Commodities: The term "commodity" refers to basic resources, more specifically, the category of products that are bought and sold on the market despite the lack of any significant quality distinctions. To be more specific, we are discussing so-called fungible goods, which are representative of a very diverse group of goods, each of which has a unique application, a unique set of characteristics and qualities, a distinctive capacity for storage, and a different level of intensity in terms of its ability to regenerate.

Forex: Forex is the acronym for the foreign exchange market,

which is a network within which buyers and sellers exchange a currency at a fixed price. Forex is the way through which individual traders, organizations, and central banks convert one currency into another.

2.2 Exchanges

An exchange is an open and organised market where different varieties of assets are dealt using various financial instruments such as stocks, commodities, and derivatives, among several others.
By centralising commercial activity, exchanges facilitate and guarantee efficient and equitable trade. In this project, we will evaluate the activity of the following exchanges: CBOE, CME, and EUREX.

CBOE

The Chicago Board Options Exchange is the most important market for the sale and purchase of options throughout the entire world[9]. They provide trading solutions and products in a wide variety of asset classes, such as stocks, derivatives, digital assets, and foreign exchange, and they operate in North America, Europe, and Asia Pacific.

CME

It is a centralized marketplace for buying and selling futures and options. The Chicago Mercantile Exchange (CME) [4] was established in 1848, primarily for the purpose of exchanging manufactured commodities like corn and flour. Nowadays, it primarily operates in the futures industry, with a significant emphasis on commodities goods such as

agribusiness and metals, in addition to stock indices, real estate, interest rates, and energy.

EUREX

Eurex[11], stands for European Exchange, is one of the main derivatives markets, controlled by the German stock exchange and SWX. Eurex mainly trades derivatives, i.e. options and futures in the form of contracts. It offers mainly interest rate derivatives, equity derivatives, volatility index derivatives and energy derivatives.

2.3 Options

Options are categorized as derivative contracts due to the fact that their value is determined by the price of an underlying asset. This is exactly why options are considered complex financial instruments. The buyer of an option, in contrast to the buyer of other derivative contracts, is granted the right, but not the liability, at the conclusion of a certain period of time to purchase or sell a predetermined quantity of the underlying asset in exchange for a payment of some amount.

Options are mainly divided into two primary geographic groups, namely American options and European options, depending on the sort of exercise that is being performed. The most significant distinction is that American options may be exercised at any moment up to the expiry date, but European options cannot. Because European options are riskier than American ones, their prices are lower. The only time they may be exercised is on the option's expiry date. The fundamental

difference between options, on the contrary side, is whether they are Call or Put options, and this is determined by the kind of option.

Option Call

The buyer of a call option is granted the right to purchase the underlying asset at a certain price, referred to as the strike price, providing a payment f a premium to the seller of the option. The long call position is held by the option buyer, while the short call position is held by the option seller.

The *long call* (2.1) strategy for trading options is one of the fundamental trading methods. Being optimistic about potential price increases for the underlying asset while holding a positive market stance. This approach is buying a call option in a single position. our exposure to risk is limited to the amount of the option's premium that you paid, regardless of what the price of the underlying asset is on the option's expiry date. In addition, the "trade" is considered profitable only when the profit made by closing the option at a higher price of the underlying asset is equal to or greater than the amount of the premium that was paid.

In contrast to the long call (2.1), the *short call* is executed from the seller's current perspective; in essence, this approach entails selling call options. Since a pessimistic outlook on the market, we anticipate that the price of the underlying will decrease. In this scenario, the benefits are restricted to the amount of premium that is collected, but the threats are limitless and dependent on the market volatility of the underlying asset.

As a result, in the case of the highest possible price, the value of the underlying asset declines, and the option is not exercised. The amount of the received premium is equivalent to the highest possible profit. The most severe situation would occur in the event that the underlying instrument increased in value while the option was being exercised. This suggests that there is no upper limit to the amount of loss.

Option Put

The buyer of a put option receives the right, in exchange for payment of a premium to the seller of the option, to sell the underlying security to the seller of the option at a price that has been specified in advance. The person who purchased the option has a long position, also known as a *long put*, while the person who sold it has a short position, also known as a *short put*, as in the case of call options.(2.1)

The buyer has the ability to exercise the option by purchasing the underlying asset on the stock market at the market price, which is lower than the strike price, and then reselling it at the strike price. This occurs when the price of the underlying asset is lower than the strike price on the expiration date of the option, which indicates that the option is "in the money." If, at the time of expiry, the price of the underlying asset is higher than or equal to the strike price, the option is said to be either out of the money or at the money, and the buyer of the option will not exercise it.

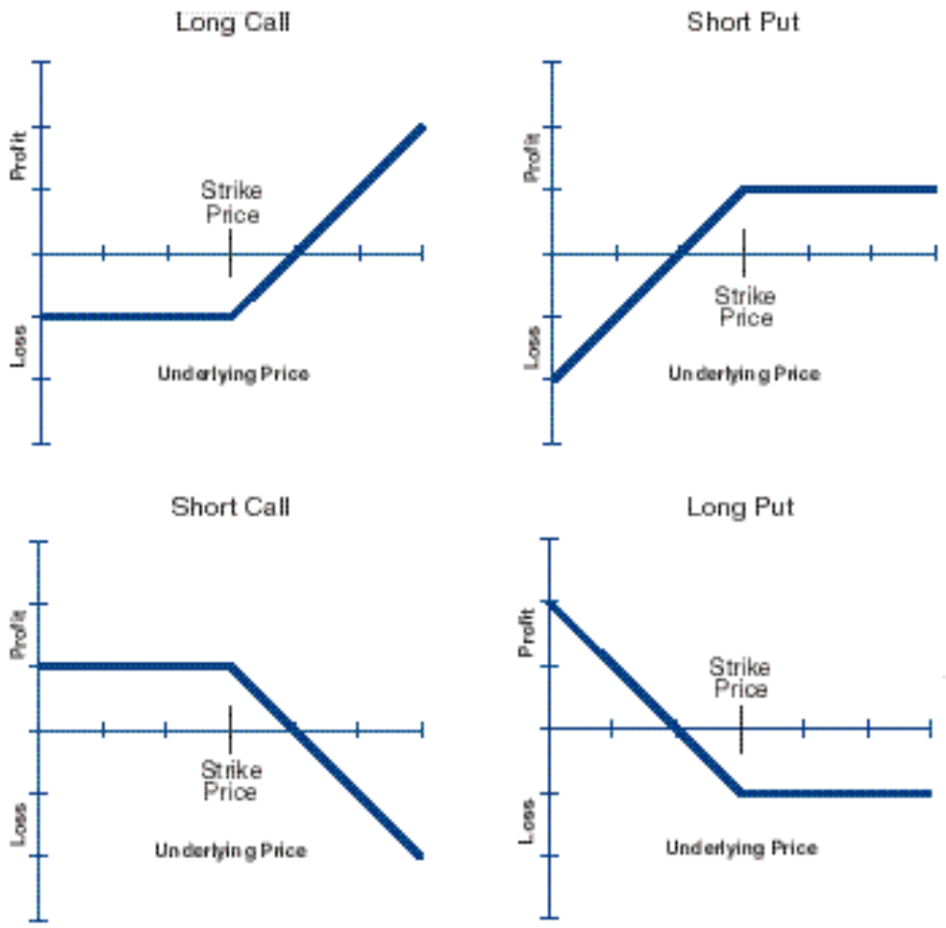


Figure 2.1: Option basics strategy graphic representation [16]

2.4 Greeks

Greeks are indicators that make it feasible to quantify the risk that is associated with holding a position in derivative instruments such as options. They evaluate and forecast the movement of option prices in response to the change in the principal risk variables that determine the value of the derivative.

Greeks are exactly proven to be extremely meaningful by employing them to minimize the risk in the position that is currently taken, obtaining the appropriate degree of exposure to risk factors in the approach. First-order Greeks such as delta, theta, rho, and vega are among the most frequently employed, while second-order Greeks include vanna, vomma, and gamma. The order of the Greeks denotes the degree of the partial derivatives that are being considered.

Delta

The delta of an option denotes the option's price sensitivity to its underlying asset. Depending on the type of Option position, various delta value ranges occur. The delta for a call option ranges from zero to 1, whereas the delta for a put option spans from 0 to -1.

If V is the option's value and S is the price of the underlying asset, then delta is defined as follows :

$$\Delta = \frac{\partial V}{\partial S} \tag{2.1}$$

Creating a delta-neutral position is a typical approach, and as predicted, the delta in this position indicates the hedging ratio.

Theta

Theta is often referred as "time decay", which refers to the process through which the value of an option decreases as more time passes. Time sensitivity is another name for time decay. Theta is calculated as follows, where V is the value of the option and τ is the time to maturity for the option contract:

$$\Theta = \frac{\partial V}{\partial \tau} \tag{2.2}$$

Long positions on call and put options commonly have negative theta, while short positions on call and put options typically have positive theta.

Vega

A measure of the rates at which the implied volatility of the underlying asset and the value of the option fluctuate over time is denoted by the symbol "Vega." Given V the value of the option and σ the volatility of its underlying:

$$\nu = \frac{\partial V}{\partial \sigma} \tag{2.3}$$

In other terms, it refers to the degree to which the value of the option is sensitive to changes in the volatility of the asset that it is based on,

and therefore, it is considered a risk factor.

In this case, the value of call options and put options both rise in tandem with the increase in volatility. If the Vega is close to zero or approaching that value, then the influence of volatility on the option's value is minimal. A portfolio's Vega may be interpreted to be the degree to which the value of the portfolio is sensitive to variations in the portfolio's level of volatility.

Rho

The rate of change in the price of an option relative to a one percent change in the interest rate is measured by the Greek "rho". It determines the degree to which the option is sensitive to changes in interest rates r.

$$\rho = \frac{\partial V}{\partial r} \tag{2.4}$$

The value of an option is often less sensitive to shifts in the risk-free interest rate than it is to shifts in the other parameters that decide it. This is because the risk-free interest rate is one of the factors that determine it. Because of this, the letter rho is not used as often as the other Greek.

When interest rates go down, the price of put options goes up, and this rise is greater for options that are now at the money and have a lengthy period before maturity.

Gamma

Gamma is the pace at which an option's delta changes in relation to the value of the underlying asset. It is a measure of second-order price sensitivity that quantifies the projected change in delta for every dollar change in the price of the underlying asset. Gamma is regarded to be a measure of second-order price sensitivity, computed as follows:

$$\Gamma = \frac{\partial^2 V}{\partial S^2} \tag{2.5}$$

If the value of gamma is low, then the delta is insensitive to changes in the price of the underlying asset. On the other hand, if gamma is significant, then the delta is likely to move drastically in response to even relatively minor shifts in the price of the underlying asset.

It is possible to structure a portfolio such that it is delta gamma neutral, which will protect it against risk regardless of the range of volatility that the underlying asset may suffer.

Chapter 3

Technologies

In this chapter, we will discuss the technologies used in the realization of the web-app, with reference to the components and frameworks used. The Django framework takes care of server-side platform management; the React JavaScript library is used for front-end development; and MongoDB for database management. Moreover, Docker containers are used to handle the entire system.

3.1 React

React is a free open-source library written in JavaScript for the development of interfaces or UI components. The concept behind React is the creation of reusable components that are rendered on demand. The rendering of the component is then translated into calls to the React API that act quickly and efficiently on the DOM of the page, generating the requested elements. In essence, React is just in charge of the application's view layer. Other frameworks may be used to create single-page, mobile, or server-rendered apps utilizing React as its core.

As data descends the component tree in React, interactions between the components are made possible via the usage of props and callback functions. In big React projects, this composition becomes deep, intricately interwoven, less maintainable, and vulnerable to props-drilling. This is one of the factors for the need for implementations like the Redux pattern in sophisticated React projects. React has multiple advantages, such as:

- **Simplicity:** It was founded on the principle of implementing just the View layer instead of the complete MVC stack, which simplifies its implementation;
- **Reusability:** The primary architectural element of dividing the view into components promotes the code's reusability and simplifies the administration of massive projects;
- **Efficiency:** It has a remarkable execution speed in addition to exceptional performance;
- **Documentation:** Since it is maintained by Facebook, there is a considerable measure of documentation on the web, which is constantly growing due to the very high number of developers in the community;
- **Flexibility:** It integrates easily with other frameworks into an existing project, can be used both client-side and server-side and can be used to construct both web and native applications using React. React Native enables the development of native applications.
- **Compatibility:** In comparison to other frameworks, migrating from one version to another is simpler;

React Redux

React-Redux [14] is a library that allows for the integration of the Redux library with React for writing React-based web applications. Its purpose is to make the state change predictable by imposing certain restrictions on how and when this can occur. Redux is a state container, represented by a mutable JSON object only after the invocation of a specific pure function called reducer. These reside within a single data structure called Store and are accessible by each individual component of the application.

The store is in a read-only single source of truth for the entire state, thus the library offers the Reducer function to alter it. It provides a number of methods, including $getState()$, which retrieves the application's State object, and $dispatch(action)$, which launches an Action. However, only Reducer functions are permitted to modify the information stored in the Store, the object meant to retain the state of the whole program. As pure functions, reducer accept the current State object as input and return a new, updated State object.

Actions are straightforward JavaScript objects used to deliver data to the Store. This is the sole way to request an update to the information stored in the Store, the object responsible for maintaining the state of the whole application.

As is illustrated in 3.1 as soon as a user performs a specific interaction that falls within the functions defined in the developed actions, Redux executes the implemented code and triggers the side effects middleware. In the end, it transfers the result to the reducer, which takes care of receiving the current state of the application as input, updates

it with the new data and returns it to React, which will display the new information. Note that the Effect middleware of the React Redux library was used outside a Reducer in every component and enables asynchronous operations such as HTTP requests to external APIs.

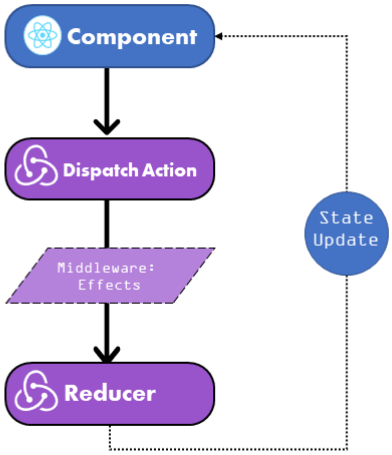


Figure 3.1: React-Redux Flow

3.1.1 Alternatives

The use of React in our field of use allows us to develop and deploy a web platform with good characteristics in terms of performance and robustness. Clearly, there are other viable alternatives to web application development that have been discarded in favour of React's better fit for our needs.

Angular

Angular is a platform primarily used for the development of online, mobile, and desktop applications maintained by Google LLC. By convention, with Angular refers to framework versions beginning with version 2, whereas AngularJS refers to version 1 had the characteristics of being slower and less responsive for some capabilities, and was built on JQuery. The name Angular 2+ denotes the most recent versions of the new framework.

Angular is a component-based framework for building scalable web applications developed in TypeScript. It is a full-fledged model-view-controller (M.V.C.) framework and enables the implementation of responsive and fast web pages by exploiting dynamic HTML and data binding. This framework is widely used in large data-heavy applications with large amounts of information to be rendered.

One of the common and widely known problems of Angular is its difficult learning curve, and given its nature in providing multiple out-of-the-box functionalities it usually produces heavy packages, resulting in applications that are sometimes slower than those developed with other frameworks.

VueJS

VueJS is an open-source JavaScript front-end Model-View-Model package, particularly famous for its ease of use and implementation. VueJS can be included as a library within your project and is developed to facilitate the creation of responsive UIs. Moreover, Vue.js is a progressive

framework that provides a core that is able to create apps by using its logic and core engine.

However, in order to build extra functionality, it is required to make use of external components. In terms of performance, VueJs, being particularly light, is obviously faster than the other frameworks mentioned, but it must be taken into account that the integration of additional components is not always linear and may cause slowdowns and bugs.

Having taken these three frameworks into consideration, it was decided to continue development with React, particularly because the application already featured existing code and components developed in React, and so it was decided to continue the project with the existing framework.

3.2 Django

Django is an open-source framework relying on the Python programming language. [7] It is a great option because of its primary attributes, which include speed, scalability, exceptional flexibility, and a wide range of libraries that enable the avoidance of frequent security mistakes. Versatility and scalability are two of the fundamental guiding concepts. Modularity, which makes it responsive to the demands of the developer, is another fundamental notion. The many integrated tools that direct the programmer throughout the development process are intended to make the design of applications, no matter how simple or complicated they are, more agile.

The framework is also created and built to be incredibly secure,

giving the developer the ability to combat critical vulnerabilities like cross-site scripting, clickjacking, SQL injection, and cross-site request forgery. When using modular programming, the developer may simply and transparently include code created by other parties in the project. As a result, it is easy to do code maintenance or upgrading on only one specific module without affecting how the system is configured as a whole.Built-in authentication uses cookies and includes protection measures against cross-site request forgery by including a CSRF token for authenticating API calls.

Django exploits a MVT (model-view-template 3.2) approach that served as the foundation for the framework. It divides the representation of the data model (model), the user interface (view) and the application logic (controller).[5]

- As soon as an HTTP request is processed, it is sent to the appropriate View. The URL mapper may also match certain patterns of strings or numbers inside a URL and transfer the matched data to a view function;

- Views handle all incoming requests by accessing requested components and resources through models;

- Models are Python objects that define the data structure of the application, providing an interface for accessing and modifying information residing in the database. They return the required data to the views;

- Templates are used to define the structure or layout of a file, they format the data and return the output to the views, which then

send the response under HTTP;

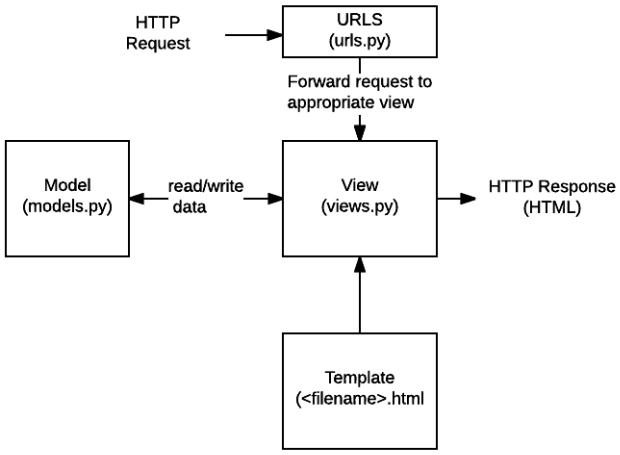


Figure 3.2: Django Data Flows[5]

3.2.1 Alternatives

Alternatively to Django, other framework solutions exist in order to implement the backend component, here are some of the main features:

Flask

Flask is an open-source backend framework written in Python for web applications. Flask is based on the WSGI Werkzeug(server framework) toolkit and the Jinja2 template engine. Flask is a fairly small and rather lightweight microframework, ideal for developing and releasing non-extensive applications from relatively small companies.

Although Flask has advantages in terms of speed of development, simplicity and flexibility, it was discarded in favour of Django, since

the latter, although heavier and less flexible, has numerous advantages in terms of stability and security.

Firstly, Flask does not have a default and robust ORM like Django, although in this context it does not interface with an RDBMS and an additional framework called Djongo has been introduced to work with NoSQL databases.

In terms of security, Django and Flask are both considered secure solutions for different reasons. Django boasts multiple security features such as CSRF, SQL and XSS. In addition, many other security-relevant features are added and executed automatically.

However, using Flask the security risks are lower due to the compactness of the code, but security problems are more likely to arise when adding third-party extensions.

As far as performance is concerned, Flask is considerably faster than Django due to the compactness of the code. However, performance deteriorates significantly when adding additional plugins and handling big amount, which makes Django a more complete and integrated solution considering the project requirements and the possible future developments.

NestJS

The NestJS framework is a comprehensive framework for building server-side (back-end) applications in a scalable and efficient manner, based on Node.js and developed in JavaScript. NestJs includes a large number of

libraries supporting Typescript and OOP (Object-Oriented Programming). They include modules and middleware to organize the application structure and manage request-response loops.

On the other hand, NestJs has performance disadvantages when it has to handle heavy workloads. Furthermore, due to the intrinsic nature of the JavaScript language, it can be complicated to manage the task queue and keep the code up-to-date.

Django was selected over NestJs in developing the application, not only because of the previously mentioned disadvantages but also in order to exploit the potential of the Python language in terms of machine learning and data analysis libraries for future implementations.

3.2.2 Celery

Celery is an open source distributed message forwarding-based asynchronous task queue or work queue. Celery is written in Python and can be efficiently integrated with Django. Although scheduling is possible, the focus is on real-time operations. Tasks may be performed synchronously or asynchronously (in the background) (wait until ready) Furthermore, tasks are processed concurrently on one or more work nodes through multiprocessing execution units.

3.3 MongoDB

MongoDB is a C++-based, open-source, document-oriented, scalable, and very efficient database. It is designed to have excellent read-and-write performance. In addition, searches are simpler and quicker, bigger reads may be split among numerous replicated servers, and the document method allows for the representation of intricate hierarchical interconnections through the use of nested documents and arrays.

This database architecture adopts the doc-oriented paradigm's storage approach, which involves storing each record as a document with certain properties. Contrary to the relational model, doc-oriented implements a different methodology: merging as many objects as possible to produce macro entities with the highest information richness. These objects include all the information required for a certain semantics.

Therefore, MongoDB lacks a schema and each document is not organized; instead, it only has a necessary key that is used to identify each document specifically; this key is similar in meaning to the main key of relational databases.

3.3.1 Relational Databases vs NoSQL Databases

Interaction with the DBMS takes place via structured languages. Databases are developed according to two main implementation approaches. The first are called relational databases and are created and managed in SQL (Structured Query Language). These tables are interconnected by one or more relationships.

Keys define the relationship between tables. The essential idea is built on the key concept provided to a column that holds unique values for each record. Then, each table will be joined to one another via systems relying on the uniqueness of the main key.

In contrast, non-relational databases, referred as NoSQL, do not store data in tables and records. However, in these types of databases, the data storage structure is developed and optimized for technical specifications, and a variety of data models are used to access and manage the data.NoSQL databases employ object-relational mapping (ORM) to allow communication with its data instead of SQL, which is used by relational databases.

NoSQL databases nowadays are popular since data is constantly evolving and developers must adapt in order to manage the massive amount and diversity of data produced by mobile devices, cloud, social media, and Big Data.

Time-Series Database

The adoption of time series has increased at an exponential rate over the last decade, with a special emphasis on applications in the financial and management sectors as well as, more generally, in the industrial and information technology industries.

TSDB(Time-Series Databases) are optimized for time series, which implies that they are optimal for applications that place a main focus on scanning a large number of data points over an extended period of time and then processing those data points.[10] The timestamp at which the

data was collected serves as the primary key in these databases.

The primary objective of these databases is to optimize performance for specific operations and well-defined data structures and is related to its ability to aggregate, filter and process Time Series Data efficiently and reliably. In fact, TSDB databases use compression algorithms to ensure efficiency, even in scenarios where the frequency of data collection is very high.

3.3.2 Alternatives

Given the mentioned characteristics, it would be pertinent to discuss some alternative Data Base Management System (DBMS) frameworks that could potentially enhance the process of database administration:

Apache Cassandra

Cassandra is a distributed, fault-tolerant and elastic database, originally developed for Facebook and first released in 2008. Cassandra is a DBMS suitable, therefore, for handling very large amounts of data that common database systems do not offer sufficient capacity. In such cases, it is necessary to resort to big data applications that are also scalable, as the actual data volume often cannot be estimated from the outset.

The most significant feature of this system is that the server is usually installed in a clustered configuration, in which several Cassandra nodes cooperate to optimize and distribute the data. Cassandra is generally used by large organizations and social networking such as Netflix,

Uber, Reddit, Red Hat and many others, which require extreme speed in data management and prevent any malfunctions and data loss from any given datacenter's hardware failure.

Considering the context of application development and used data, it was discarded as disproportionate for our needs.

InfluxDB

InfluxDB is a time series database well-suited for storing and retrieving time-stamped data such as financial data, sensor measurements, application metrics, and real-time analytics data. The database management system is programmed in the Google Go programming language. It offers a query language similar to SQL and allows batch inserts and real-time querying.

InfluxDB is also able to handle heavy write and query loads. Its supports for SQL-like queries, sophisticated data modelling, and horizontal scalability make it an excellent option for a broad variety of use cases.

However, in view of our context, InfluxDB and, generally, TSDBs were discarded, as although the data to be processed within the application is predominantly financial, other data cannot be represented as a time series, but requires greater flexibility in terms of modelling the structure.

MySQL

MySQL is a popular open-source relational database management system (RDBMS) that is widely used for managing and organizing data in web and application development. It is known for its reliability, ease of use, and high performance.

MySQL use of SQL (Structured Query Language) for accessing, managing and manipulating data in a database. SQL is a standard language for working with relational databases, and it allows users to easily add, update, and retrieve data from a MySQL database. MySQL also supports a wide range of programming languages and frameworks, including popular languages such as PHP, Python, and Java. This makes it easy for developers to integrate MySQL into their web applications and projects.

In addition to its core features, MySQL also provides a number of tools and utilities for managing and administering databases. These include the MySQL Command Line Client, the MySQL Workbench, and the MySQL Administrator. These tools provide a wide range of functionality, including data modelling, database management, and performance tuning.

However, it does have some limitations and disadvantages as well. In particular, SQL databases might have performance problems, particularly when performing a significant number of read and write operations. This can be challenging for real-time applications that need low latency. Due to SQL's rigid schema foundation, it might be problematic to adapt to new data formats or shifting needs. SQL's ability to

maintain data integrity may also result in decreased speed when working with massive amounts of data. Additionally, firms that depend on SQL databases may incur significant maintenance expenses since SQL databases need frequent maintenance and optimization.

3.4 Docker

Docker is an open-source containerization platform by which developers are able to bundle and deliver apps efficiently and portably. It is based on a number of essential elements (Figure: 3.3), each of which serves a particular function and plays a crucial role in the larger Docker ecosystem.

The Docker platform's main building block is the Docker daemon. The lifespan of containers, involving their beginning, halting, and deletion, must be managed and coordinated by it. Additionally, the daemon interacts through REST API, with Docker users and other remote Docker daemons and responds to requests for containers and image creation, management, and inspection.

Docker CLI: The command-line interface used to communicate with the Docker daemon is known as the "Docker Client." It enables programmers to provide the daemon commands, such as those for building, launching, and halting containers. The client also gives access to the daemon and the containers it is controlling to analyse their states.

Docker Images: Docker Images: A Docker image is a small, self-contained package that includes the code, runtime, system tools, and

libraries necessary to execute a piece of software. Images may be created by following a set of instructions in a Dockerfile and distributed and saved via a registry like Docker Hub.

Docker Register: A Docker registry is an online service that hosts and shares Docker images. Developers would download and utilize images made by others, as well as save and distribute their own images using the registry. The platform's default registry is named Docker Hub, however, The Google Container Registry and the Amazon Elastic Container Registry are two other registries that developers may utilize.

Docker Container: A running instance of a Docker image is referred to as a "Docker container". While running in a distinct namespace and using the host's kernel and system libraries, containers are isolated from the host system and other containers. This enables developers to install several programs on a single host without worrying about conflicts.

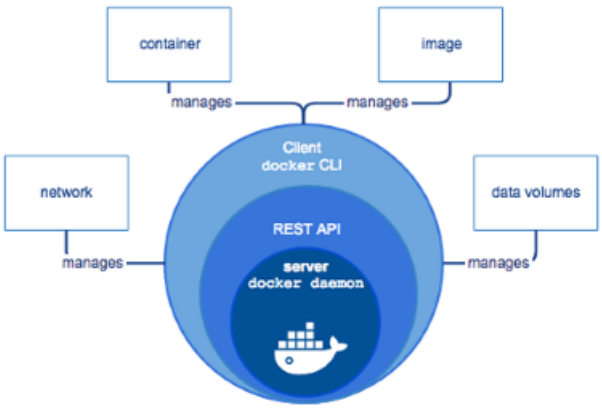


Figure 3.3: Docker Main Structure [15]

Chapter 4

Architecture

As previously stated, the program is web-based and is managed using a client-server architecture (Figure: 4.1). APIs allow for the transmission of data between the client and the server. Since the program is composed of data-handling microservices, the server side has been containerized to facilitate the development process. Microservices are an architectural approach to creating software applications. As an architectural framework, microservices are distributed and loosely connected in order to preserve the independence of one service without compromising the development of the others. Due to the fact that each microservice specializes in a certain duty and has a well-defined function, this strategy encourages better outcomes when integrating new features into a system. In addition, splitting concerns into separate microservices improves scalability since each service may expand separately. These microservices are packaged into containers that are maintained by software in order to make development and usage simpler and more efficient. In this chapter, we will discuss everything that has been utilized for development, including a list of development tools and frameworks along with their benefits.

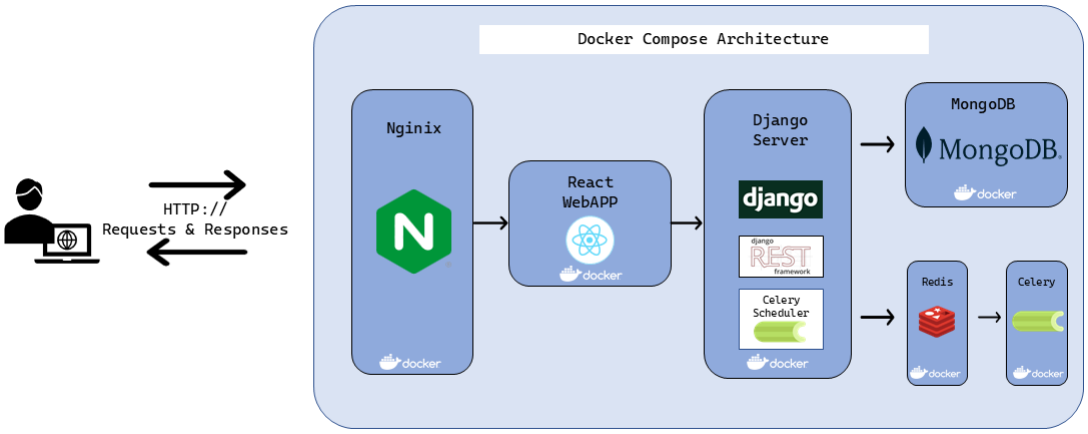


Figure 4.1: Architecture Structure

4.1 Dockerization

The entirety of the project is created and executed inside the Docker environment. Docker is a system that enables applications to be containerized and exploits the virtualization capabilities offered by the operating system to run containers, making it incredibly efficient since containers share the operating system kernel and resources while preserving isolation across containers. Specifically, a container is an environment that is separated from the execution environment and includes all the essential libraries, dependencies, and configuration files for a containerized programme to work.

Containers are produced from a file known as an image that describes how a container should be built. Docker-compose is used in this scenario since the design demands distinct containers to handle services. Docker-compose enables the setup and execution of containers to be

orchestrated and managed. However, in contrast to the initial setup, the react-developed application no longer resides in the original container with the backend, but rather in a separate one. As previously described, this enables more independent development of new features and debugging.

4.2 Nginx

Nginx is a free and open-source strong and adaptable web server, generally used as a load balancer, reverse proxy and HTTP cache. It is a popular option for web hosting and content delivery due to its fine performance, reliability, and minimal resource usage. It has an event-driven design, which enables it to process several requests concurrently without the need of numerous threads or processes. NGINX's capacity to manage a high number of concurrent connections is one of its primary characteristics, becoming ideal for websites and online apps with significant traffic that need quick and trustworthy performance.

Furthermore, NGINX has a load-balancing function that enables the distribution of incoming traffic across other servers. Additionally, It offers content caching, which lessens the amount of data that must be transmitted over the network and speeds up the rendering of web pages and other data. The support of a variety of protocols is another crucial aspect of NGINX such as HTTP, HTTPS, TCP, and UDP.

4.3 Client-server model

The client-server paradigm is a distributed application architecture that divides responsibilities between servers and clients that sit on the same system or interact across the Internet or a computer network. The client-server architecture is the most prevalent and recommended architecture for interacting with a web application. In order to use a service made accessible by a server, the client must make a request to another software. The server executes one or more programs which pool resources with clients and distribute their tasks.

The client-server interaction uses a request-response message pattern and must conform to a common communications protocol that specifies the rules, terminology, and dialog patterns to be utilized. Typically, client-server communication conforms to the TCP/IP protocol suite. The communication between the client and server is governed by the HTTP protocol, which handles all requests.

4.3.1 HTTP Protocol

The HTTP stands for Hypertext Transfer Protocol. This protocol is the basis for data transmission on the World Wide Web and is built on a request-response concept, in which a client submits a request to a server, which therefore provides a response. This protocol has different components that enhance communication between the involved machines.

- These are the verbs that specify the action to be taken on the requested resource. In particular, methods as GET, POST, PUT,

and DELETE are the most popular;

- URI (Uniform Resource Identifier) is a string that identifies the resource on the server that was requested. Generally, URIs have the form of URL (Uniform Resource Locator), but they could also be developed following the URN (Uniform Resource Name) pattern;

- Request and Response Headers are used to provide the server with extra information about the client or the request/response. In addition to the basic ones, in our case it is interesting to mention the 'X-CSRF-Token' used to authenticate API calls by the client and prevent man-in-the-middle attacks, and the 'Set-Cookie' header used by the server to set the CSRF token within the client current session;

- Request and Response Body are employed to transfer information from the client to the server and vice versa. Usually, the information within the body is encoded and represented in JSON.

HTTP is a stateless protocol, which means it does not maintain a connection between consecutive requests. However, some header values may be used to keep the connection alive. Moreover, the protocol also implements control codes to quickly check the status of the response from the server.

When a user accesses a website using a web browser, the complete connection is made over HTTP. The protocol permits the reception of data, including text, photos, video, style sheets and scripts, among others.

HTTPS

HTTPS (Hypertext Transfer Protocol Secure) is an extension of the HTTP protocol used to encrypt data transferred between a web browser and a web server. When a user accesses a website using HTTPS, its browser creates a secure connection with the server using SSL (Secure Sockets Layer) or TLS (Transport Layer Security). This secure connection encrypts the data exchanged between the browser and the server, preventing third parties from intercepting or reading it.

4.4 Architectural Pattern

The design of the structure of a web application can be performed according to different architectural patterns depending on the type of rendering of the content of the pages, some main of these styles in detail.

Multi-Page Application

The client sends a request to the server whenever a page has to be shown. The server then processes the request and sends back the requested content along with any resources that were required for the page. Therefore, navigation occurs between various HTML pages, and each page has the capability of executing additional JavaScript in order to tailor the content and make it dynamic.

To provide more context, the browser makes the request for the HTML page, while the JavaScript code makes the request for the data and the application logic is managed both on the server and on the client. Since it is the server that is responsible for sending the client

the right page to show, server-side rendering is carried out in accordance with this pattern. As a result, one is required to constantly wait for the new page to load.

Single-Page Application(SPA)

This is a development of the classic paradigm, in which a single application page with identical content for each URL sits on the server. The page, originally an HTML file with no content, modifies the displayed content via the execution of JavaScript code in response to the user's activities while browsing. Therefore, the content of the DOM structure changes dynamically, adding and rendering components only when required by the user or the page structure, reducing page loading time, which happens virtually immediately since the new HTML resource does not need to be retrieved from the server.

In this situation, the frontend is solely responsible for displaying the content, and contact with the server is limited to the exchange of data to be presented. Thus, the server's effort is decreased, while the browser and front-end take on a far greater role in controlling the service. Among the principal benefits are:

- Faster execution of the applications and a reduction in the amount of time spent waiting while exploring;

- The whole of the application is implemented in a centralized point, for this reason, it is also simpler to debug the application by inspecting the Document Object Model (DOM) and monitoring the data that is transmitted to and from the server over the network;

- improvement of the user experience.

This implementation style, however, exposes the Web App to certain disadvantages. Firstly, these SPAs may usually be exposed to certain security problems relating to Cross-Site Scripting and Cross-Site Forgery. Secondly, since the content of the page is dynamically rendered, the effectiveness of the web history is lost, and since there is no HTML page to inspect, some search engines and SEO (Search Engine Optimization) may have difficulty in indexing the application.

However, given the advantages, not only in terms of performance, but also in terms of development and release, a structural modification of the application was to switch from a hybrid architectural pattern to the one described in this subsection. Initially, the rendering of pages was partially handled server-side, to which it was supplied with react resources.

At the current moment, frontend and backend have both been completely separated from each other and are operating as two independent entities from the other. React is used throughout the development of the client in its completeness. Instead, Django is responsible for managing the Backend exclusively, and it exposes itself to the SPA as a REST API server in order to establish a connection between it and the database, providing Database resources and authentication. In this scenario, the two servers each take care of their own unique domain. This structural choice was made with the intention of making the development of the two different parts more independent and agile.

In fact, with this type of architecture, the evolution of the first does not affect in any way the executive cycle of the other, which makes the

release of additional features and the correction of any problems more straightforward.

4.5 REST-API

According to the previous explanation, the Representational State Transfer (REST) Application Programming Interface (API) provides the capability for one application or service to access a resource located inside another application or service. REST is an architectural constraint set that is used in the API. These architectural constraints govern the characteristics of the API's implementation.

As soon as the server receives a request, it processes the instance and sends the response, in which the requested information is encoded, back to the client. The type of HTTP method used depends on the purpose of the request. The most commonly used methods include GET, POST, PUT, and DELETE. Typically, the data in the responses is formatted as either JSON or XML.

4.6 CORS

The initial approach was to render the hybrid web app from the Backend. Subsequently, the Frontend was spun off and made a standalone instance. Currently, the Client exclusively calls the API for Database querying and session and user authentication, however for the current configuration, the frontend and backend belong to two different domains. Django restricts cross-domain connections, for this reason, has

been needed to enable Django Rest Framework to receive calls external to your domain by modifying the server configuration and allowing resource sharing.

A server may specify any origins besides its own from which a browser should authorize database queries using the Cross-Origin Resource Sharing (CORS) HTTP header method. In order to verify that the server hosting the cross-origin resource would allow the actual request, CORS additionally uses a method wherein browsers send a "preflight" request to the server hosting the resource.

Additionally, due to ambient authority, a cross-domain request from an attacker to your website will include the user's authentication. This is a Cross-Site Request Forgery attack.[6] For this reason, Django requires a CSRF Token to be included with every request to the Backend. However, coming from different domain origins, it was required to set the necessary parameters for the correct use of the Token.

4.7 Design Implementation

Implementing an interface and dashboard that is functional and makes data consultation quickly accessible and configurable is crucial to the complete site experience. An effective dashboard needs to be intuitive and consistent and clear with the proposed tools. In particular, some application components were reengineered with the libraries applied below.

4.7.1 React Bootstrap

Bootstrap is a collection of front-end development components that is available under an open-source licence. By combining JavaScript, CSS, and HTML, Bootstrap provides a framework that is well-suited for the development of dynamic and responsive web pages. It guarantees that the user interface of a website retains its ideal functionality regardless of the screen size being used.

React Bootstrap is a framework that includes Bootstrap styles and functionalities by releasing React components that can be quickly incorporated into web-based applications. This makes it much easier to employ Bootstrap while designing the user interface of the website. As a result, the developer is given access to a vast library of components which are both ready to use and quickly customizable.

4.7.2 Graphic Libraries

It is of the utmost importance to have access to both the charts and the various underlying indicators when it comes to examining and planning investment strategies. This allows it to comprehend the interactions that occur between the various assets, identify patterns and trends, and reach at decisions which thus are more informed.

Whenever it comes to comprehending and analysing the complicated data involved in trading techniques, graphical analysis is a crucial tool. Since it can analyse large amounts of information in a way that is brief and clear, it is a useful and important tool for experienced traders. It is essential, through the creation of visual representations, to identify

areas in which the strategy has performed satisfactorily and those areas in which it is necessary to make adjustments.

React-Stockcharts

React-stockcharts [13] is a free and open-source tool for making dynamic financial charts using React, a JavaScript language for constructing user interfaces. It is based on the d3.js package and enables extensive functionality and customization based on the user's requirements.

React-stockcharts also provides a variety of pre-built chart components, including candlestick, bar, and line charts, that may be effectively included into a React application. In addition, it offers a wide range of indicators and tools for technical analysis and other sorts of charts, making it a flexible tool for a wide range of purposes.

However, the library has some disadvantages that are not entirely negligible. Foremost, the original source code is poorly maintained. In particular, the main repository is locked at React v16 and, following the webapp's upgrade to React v17, it has some compatibility problems, turning out to be obsolete. Secondly, the library requires more development than other libraries to achieve satisfactory results. For these reasons, it was decided to migrate to libraries that, although proprietary, offer faster development and solid future code reliability.

Lightweight-Charts

Lightweight-Charts is a free and open-source library created by the renowned software company Tradingview, whose charting platform and

social network is used by over 30 million traders and investors. Tradingview provides three types of libraries: the one described above, which is less exhaustive and very lightweight; a second, more advanced, but closed source and fee-based library that integrates charts and more advanced analysis tools; and a third, designed for complete technical analysis and trading functionality.

Given the project's current stage of development, which is still at an early stage, let's consider the first, thus Lightweight-Charts[17]. The latter gives the best performance in terms of lightness and chart generation, however, its input data architecture is limited and it does not support general-purpose chart formats. Indeed, makes it unsuitable for several of our platform's analytical charts

Hightchart

Highcharts is a JavaScript library for creating interactive charts and web applications released by the company Highsoft AS. [8] The library is proprietary and requires a commercial licence, but offers a free licence to students, non-profit organizations, personal projects and throughout product development.

One of the library's main strengths is its flexibility for professional customization and development. The library also has APIs that make it easier to develop graphics in a lighter and more understandable manner, minimizing coding.

Highcharts offers a wide range of chart types, divided into four different packages below:

- **HighchartsJS** It is the core library, developed in pure JavaScript. The library is SVG-based, includes all standard graphics and facilitates developers to create responsive, interactive and accessible charts.

- **Highcharts Maps** This product fulfils very specific needs, especially data visualization on interactive maps. It offers an extensive catalogue of thousands of maps, including heatmaps, POIs, map areas and map lines. This library also offers built-in projections and natively supports different coordinate system formats such as TopoJSON and the more popular GeoJSON.

- **Highcharts Gantt**This library is required to create Gnatt charts, and describes timelines and project tasks useful for organizing the workflow.

- **Highcharts Stock** This is the Norwegian company's reference solution for financial or representative time series graphs. This standalone library requires no additional dependencies to operate, and is optimized to handle large amounts of data quickly and efficiently. Another fundamental and useful feature for traders includes the pre-built technical indicators such as SMA, WMA, VbP and many others, which can be easily added to the chart to provide a comprehensive analysis of the data.

All Highcharts libraries are actively maintained and updated, have clear documentation, and are supported by a wide community, allowing

developers to easily find answers to common questions and solve problems. The pick of this library above the previously listed alternatives is based on these considerations.

Chapter 5

Analytics

A section dedicated to the advanced analysis of derivative contracts was introduced in the platform. The aim is to provide the trader with additional advanced tools to identify possible opportunities. The purpose of this section introduces new possibilities for developing and expanding the platform. A first step was to integrate the basic and intuitive frontend interface, focusing on data analysis and visualization. By integrating the analysis on the backend side and exposing the related services via an API, the new functionalities introduced in the application are listed below:

5.0.1 Contract price estimation with Monte-Carlo simulations

Financial option valuation models indisputably play an integral role in the field of modern finance theory. These models are pivotal in determining the fair market price of options, thus enabling investors, traders, and financial institutions to make informed decisions.

In essence, these models make use of the no-arbitrage condition, which postulates that an equivalent expected return cannot lead to differing prices for the same asset. This is founded on the belief that if such a situation were to arise, arbitrageurs would take advantage of the price discrepancy until equilibrium is restored. Moreover, these models are premised on the assumption that operations are conducted within a risk-neutral environment. This risk neutrality assumes that all market participants are indifferent to risk, meaning they neither favour nor shy away from it. Consequently, the value of the option can be derived as the discounted mean of payoffs.

5.0.2 Gamma and Vanna Exposure (GEX & VEX)

The preliminary investigation is predicated upon the computation of Gamma Exposure (GEX), a term that delineates the sensitivity of options to fluctuations in the price of the underlying asset. It is a metric denominated in dollars, representing the coverage provided by market makers for options, and consequently, is defined as the variation in dollar exposure corresponding to a 1% change in the underlying asset. Computed as follows:

The concept of gamma exposure is an intricate one, necessitating an in-depth understanding of various trading strategies. In order to fully appreciate the effectiveness of gamma in gauging market liquidity, it is paramount to familiarize oneself with two fundamental notions.

Limit Order Book

An 'order book' refers to a continually updated compilation of buy and sell orders, pertinent to a specific security or market, systematically

organized based on price tiers. This instrument is leveraged by market operators for ascertaining the fluctuations of demand and supply for a given security across various price thresholds. The 'limit order book' provides an in-depth perspective of the market for the respective security and serves as a valuable asset in informing trading decisions. However, over time, the limit order book has evolved to become increasingly intricate, decentralized, and abstract.

The contemporary trading environment comprises numerous trading platforms, an array of order types and their modifications, and infinite algorithmic smart order routing systems. These were all designed with the primary objective of achieving optimal fills at the lowest aggregate costs. Despite the complexity and variety, there remains a single overarching limit order book outlining supply and demand.

Delta Neutral Positions

A delta-neutral stance implies that the portfolio, consisting of a specific quantity of derivative contracts and an equivalent amount of underlying assets, is arranged in such a way that the overall delta from the derivative holdings perfectly counterbalances the total delta from the positions in the underlying assets.

Essentially, any particular fluctuation in the value of the underlying assets' position is precisely neutralized by a change of equal magnitude, but in the opposite direction, in the value of the derivative position.

From the viewpoint of a market dealer, any option transaction, whether a purchase or a sale, invariably involves a counterparty who assumes the opposite position. Given that the dealer typically aims to avoid

the directional risk tied to the position, they mitigate this risk by dynamically adjusting their position in the underlying asset, thereby influencing the liquidity of the market. This suggests that we are capable of constructing a particularly insightful "implied order book" merely by understanding the hedging requirements of existing options. Consequently, we will be able to identify areas where liquidity stemming from options is plentiful and zones where it is limited. To conclude, the value of gamma exposure is calculated as follows:

$$\text{CallGEX} = \text{callOpenInterest} \times \text{CallGamma} \times \text{ContractSize} \times \text{spotPrice}^2 \times 0.01$$

$$\text{PutGEX} = -\text{putOpenInterest} \times \text{PutGamma} \times \text{ContractSize} \times \text{spotPrice}^2 \times 0.01$$

$$\text{TotalGEX} = \text{PutGEX} + \text{CallGEX}$$

Through GEX analysis, an experienced trader can perceive the position market makers take on the market and their expected actions.

Vanna Exposure (VEX)

Adopting a rationale analogous to that used for Gamma, it is feasible to delineate a concept known as Vanna Exposure (VEX). In this context, however, VEX is characterized as the sensitivity of an option dealer's delta to alterations in the implied volatility of options. In other words, Vanna Exposure is expressed in terms of the dollar value corresponding to a 1% fluctuation in the underlying volatility.

5.0.3 Implementations

The implementation process can be broadly bifurcated into two categories: frontend and backend. From the frontend perspective, users interact with a straightforward dashboard, enabling them to choose their preferred securities for analysis. An introductory interface permits users to establish the parameters for Monte Carlo analysis and transmit the request to the server via an Application Programming Interface (API). The server, in turn, will return the value derived from the simulation.

A secondary interface facilitates the selection of the desired security and initiates the request for the computation of Gamma and Vanna Exposure. In this scenario, the data returned by the function populates three distinct graphical presentations. The initial graph depicts the values of Call, Put, and Total options, segmented by Strike Price, and calculated across all expiration dates. The subsequent graph represents the aggregate values computed for all strike prices, categorized by maturity.

The final visualization showcases the exposure profile corresponding to alterations in the spot price. When simulating the price of the underlying asset, the value of the Greek changes, thereby necessitating the recalculation of each value. By consolidating all the simulation values for each maturity, it is feasible to derive a profile line corresponding to the fluctuating value of the underlying asset. This aids in calculating the zero level, the inflexion point where the sign transitions.

The following endpoints expose the services mentioned above:

- `/api/analytics/mcsimulation/`
- `/api/analytics/gex/`
- `/api/analytics/gex/exp/`
- `/api/analytics/gex/profile/`
- `/api/analytics/vex/`
- `/api/analytics/vex/exp/`
- `/api/analytics/vex/profile/`

Chapter 6

Use Case

Use cases are descriptions of how a system can be used, essentially, describing how a user interacts with a system to achieve a certain goal. The study of a use case involves the fundamental elements described below. The actors are the type of user who interacts with the system in question. The system and software in use-case analysis specify the functional requirements that describe the system's behaviour. Preconditions are the necessary requirements for the development of the use case, and post-conditions are the instances that occur at the end. The steps describe the actions necessary to achieve the final result.

Underlying Price by Volume Chart Visualization

This use case describes the user's opportunity to display the historical chart of the underlying with the volume on the y-axis adjacent. This chart is called the Volume Profile Indicator and draws a histogram on the chart that can be used to reveal significant or predominant price levels based on volume.

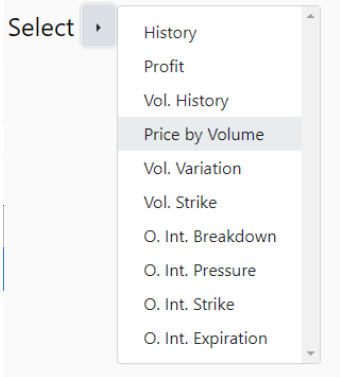
A logged-in user has access to the analysis platform. After the user

has selected the preferred underlying from Stocks, Indices, Bonds, Commodities Future, the tab with the various trading tools is loaded. After selecting the strategy of interest and selecting the Price by Volume chart [6.1a], the historical chart of the underlying is loaded with the volume chart distributed on the vertical profile.

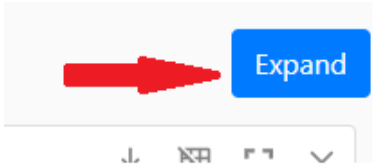
If the user needs a larger view of the charts, he can select the expand button [6.1b]. In this case the other elements are hidden and the chart section is expanded to the whole parent container of the tab and, as a result, the chart is reloaded adapting to the new dimensions [6.1].

Actors Involved	User
Pre-condition	The user is authenticated and is in the Market page, he has selected an underlying and its tab has loaded
Post-condition	The user displays the extended Price by Volume chart
Scenario	The user selects clicks on the select button and selects the graph from the catalogue of those available The user clicks on the "*Exapand*" button The user displays the extended chart in the Market Tab
Alternative	The user selected a graph with unavailable volume data, the graph is not displayed correctly

Table 6.1: Use Case Table



(a) Chart Selection



(b) Expand Button

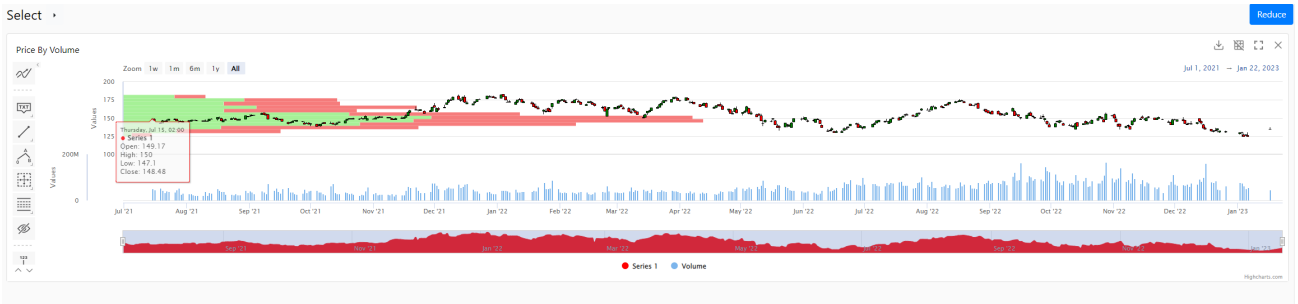


Figure 6.1: Price by Volume Chart

Pricing by Monte-Carlo Simulation

This use case allows the user to perform a Monte Carlo simulation to estimate the price of an option. Through the interface [6.2] in the analytics section, the user selects an underlying asset, and its simulation parameters. After the submit button is pressed, the server returns the value of the option obtained as the discounted average of the payoffs.

- Ticket: Underlying Asset
- Expiration: Contract Expiration
- Price: Spot Price (already defined by the last available)
- Strike: Desired Strike Price

- Path: Number of possible underlying price paths
- Option type: Call or Put
- Implied Volatility: Specifying the implied volatility

Actors Involved	User
Pre-condition	The user is authenticated and is in the Analytics page, he has selected an underlying asset and the relevant parameters
Post-condition	The user displays the simulated option Price
Scenario	The user selects an underlying and sets the desired parameters for the simulation The user clicks on the "*Submit*" button The user displays the simulated price
Alternative	The user enters inconsistent parameters for the simulation and receives an error.

Table 6.2: Use Case Table

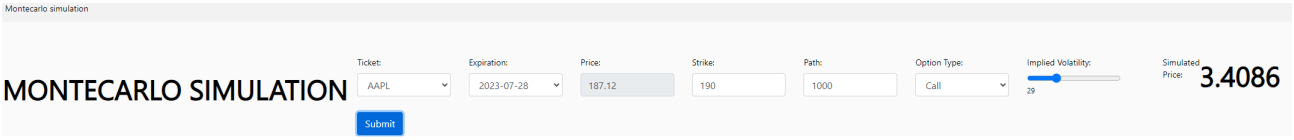


Figure 6.2: Monte-Carlo Simulation Interface

GEX & VEX Chart Visualization

This use case pertains to the user's capability to access and observe six graphs associated with the indicators Gamma Exposure (GEX) and Vanna Exposure (VEX). Initially, the user selects the desired underlying asset from a drop-down menu and proceeds by clicking the submit button. The frontend then invokes the relevant endpoints, and upon receiving the results, populates the corresponding graphs.

The graphs for both indicators encompass three distinct types. The first type [6.3] depicts the corresponding value arranged according to the strike price, calculated for each expiry date, and further categorized into Call, Put, and the cumulative value. The second type[6.4] showcases the total value distributed across different maturities. Lastly, the third graph [6.5] illustrates the profile in response to variations in the underlying asset's price.

Actors Involved	User
Pre-condition	The user is authenticated and is in the Analytics page, he has selected an underlying asset.
Post-condition	The user displays the relevant GEX and VEX charts
Scenario	The user selects the underlying asset from the list of those available The user clicks on the "*Submit*" button The user displays the extended chart in the Analytics Page
Alternative	The user selected the underlying asset with unavailable data, the charts are not displayed correctly

Table 6.3: Use Case Table

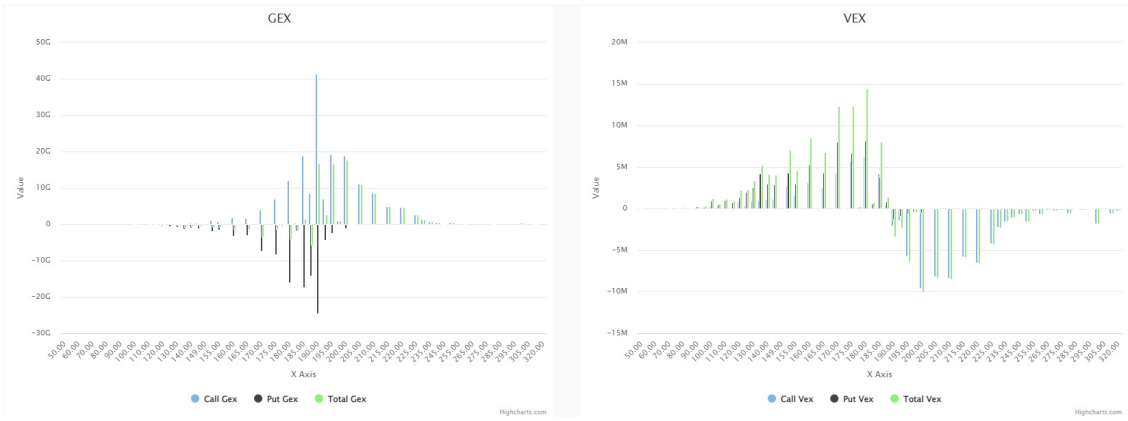


Figure 6.3: GEX & VEX by Strike

[b]

Figure 6.4: GEX & VEX by Expiration

[b]

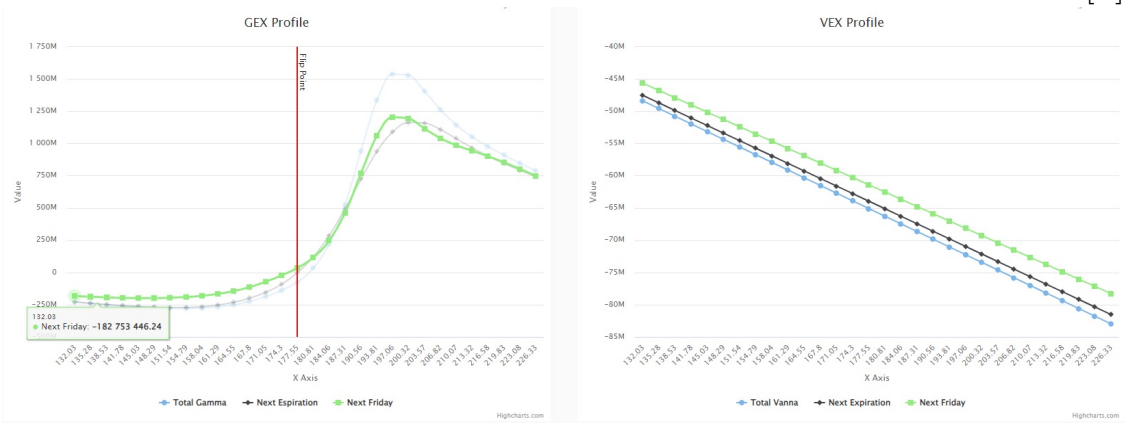


Figure 6.5: GEX & VEX Profile

Chapter 7

Conclusions

In recent years, trading in derivative instruments such as options has become extremely complex. Financial derivatives such as options can be difficult to understand for a user unfamiliar with these products. In this regard, it is important to provide a visualization of the characteristics of compound strategies that are as clear and immediately understandable as possible. The implementation of more structured charts and indicators helps the trader in consulting financial indicators and analysing complex strategies in order to improve their performance in terms of desired characteristics. The charts in question are displayed correctly and data provided and calculated by the models described are consistent. The platform is currently online and after a short registration, it is possible to view the tools supporting the investor, calculated on the individual user's portfolio.

Future Work

The platform offers several possible future developments and improvements in certain aspects. Possible future implementations of the platform lie in both functional and product coverage aspects. In fact, at present, the data used within the platform, which describes the history of stocks, indexes, and bonds, is collected via a free API service that has some limitations on both the number of options available and the price frequency of updates. Future developments could involve a move to more structured services, although for a charge, so as to have greater range and flexibility in the choice of option contracts and their underlying. Furthermore, an automatic option recommendation service and an intelligent portfolio optimization system could be developed and integrated to allow users to have a more efficient and balanced portfolio.

Bibliography

[1] Giovanni Barone-Adesi and Robert E Whaley. *Efficient analytic approximation of American option values*. 1987.

[2] Petter Bjerksund and Gunnar Stensland. *Closed form valuation of American options*. 2002.

[3] Fischer Black and Myron Scholes. *The pricing of options and corporate liabilities*. 1973.

[4] James Chen. *Chicago Mercantile Exchange: Definition, History, and Regulation*. URL: https://www.sofi.com/learn/content/what-is-cboe/.

[5] MDN Web Docs. *Django Web Framework*. URL: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction.

[6] Django Documentation. *Cross Site Request Forgery protection*. URL: https://docs.djangoproject.com/en/4.1/ref/csrf/ (visited on 05/10/2022).

[7] Django Documentation. *Django Overview*. URL: https://docs.djangoproject.com/en/4.1/intro/overview/ (visited on 03/10/2022).

[8] Highcharts. *Highcharts Documentation*. URL: https://www.highcharts.com/docs/index.

[9] Ashley Kilroy. *What Is the CBOE?* URL: https://www.sofi.com/learn/content/what-is-cboe/.

[10] Charles Mahler. *Relational Databases vs Time Series Databases.* URL: https://www.influxdata.com/blog/relational-databases-vs-time-series-databases/.

[11] Brian O'Connell. *What Is the Eurex?* URL: https://www.sofi.com/learn/content/what-is-eurex/.

[12] Andrea Pascucci. *PDE and martingale methods in option pricing.* Springer Science & Business Media, 2011.

[13] Ragu Ramaswamy. *React stockcharts.* URL: http://rrag.github.io/react-stockcharts/documentation.html.

[14] React Redux. *React Redux Documentation.* URL: https://react-redux.js.org/api/hooks/ (visited on 11/11/2022).

[15] Arvind Samantray. *Docker Architecture: Why is it important?* URL: https://www.edureka.co/blog/docker-architecture/ (visited on 01/12/2022).

[16] Sarah Springer. *Options.* URL: https://www.fe.training/free-resources/financial-markets/options/ (visited on 02/12/2022).

[17] Tradingview. *Lightweight Charts API Documentation.* URL: https://tradingview.github.io/lightweight-charts/docs/api.