

# POLITECNICO DI TORINO

Master's Degree in Data Science and Engineering



**Politecnico  
di Torino**

Master's Degree Thesis

**Development of a mobile application to  
explore financial options strategies**

Supervisor

Prof. Alessandro FIORI

Candidate

Riccardo BALDASSA

October 2022



## **Abstract**

The aim of this project is to design and develop a mobile application that meets the requirements of a professional financial trader in the derivatives market, especially with the goal of simulating options and futures strategies. A lot of data and charts have to be displayed on a mobile device screen, so the design choices are fundamental. The mobile application was developed using Flutter, a cross-platform framework that enables the creation of a fluid high-performance application like a native one. The application allows the user to visualize a variety of data and interactive charts thanks to a modern user interface. These data and charts are essentially of two types: one type relates to the simulation and the performance of the strategies implemented by the user while the other type relates to derivative instruments, specifically options and futures, and their underlying. This latest type of data is updated daily and covers the American and European markets.



# Acknowledgements

Concluding this course of study is a great pride for me, I want to share it thanking people who have been part of my journey in these years and have contributed to make it special.

First of all, I thank my supervisor, Prof. Alessandro Fiori, for giving me the opportunity to test myself by developing a project of my interest, correcting and advising me in times of difficulty. Together with him I thank the professional trader Marco Rossi for his tips and guidance.

I would like to thank my family for supporting and encouraging me in these university years, especially in the most difficult moments.

Thank you Sara, your constant support was indispensable. Thank you for giving me joy all these years and for what you will give me in the years to come, choosing to walk this wonderful path together.

I am grateful to my friends, especially Daniele, Lorenzo, Luca and Riccardo for being there and for sharing important moments with me, making all these years full of happiness. Thanks also to all the other people I met and who have, each in their own way, shared with me a piece of their journey.



# Table of Contents

<b>List of Tables</b>	VI
<b>List of Figures</b>	VII
<b>1 Introduction</b>	1
1.1 Structure of the thesis . . . . .	2
<b>2 Finance</b>	3
2.1 Trading . . . . .	3
2.2 Financial markets . . . . .	4
2.3 Exchanges . . . . .	5
2.4 Derivatives . . . . .	6
2.5 Options . . . . .	6
2.5.1 Option contract specifications . . . . .	7
2.5.2 Call options . . . . .	9
2.5.3 Put options . . . . .	11
2.5.4 Other options strategies . . . . .	16
2.5.5 Greeks . . . . .	20
2.6 Futures . . . . .	21
<b>3 Data sources and technologies</b>	22
3.1 Data sources . . . . .	22
3.2 Technologies . . . . .	24
3.2.1 Django . . . . .	24
3.2.2 MongoDB . . . . .	26
3.2.3 Celery . . . . .	27
3.2.4 Docker . . . . .	28
3.2.5 Nginx . . . . .	28
3.2.6 Flutter . . . . .	29

<b>4</b>	<b>Architecture</b>	<b>34</b>
4.1	Client-server model . . . . .	34
4.2	Database . . . . .	35
4.3	Asynchronous tasks . . . . .	43
4.4	REST APIs . . . . .	44
4.4.1	User . . . . .	44
4.4.2	Market . . . . .	46
4.4.3	Chain . . . . .	46
<b>5</b>	<b>Mobile application</b>	<b>48</b>
5.1	Data management . . . . .	48
5.2	User interface . . . . .	50
5.2.1	Markets . . . . .	51
5.2.2	Strategies . . . . .	55
5.2.3	Portfolio . . . . .	56
<b>6</b>	<b>Use cases</b>	<b>57</b>
6.1	Analysis of a position . . . . .	57
6.2	Exploration of a strategy . . . . .	59
6.3	Exploration of the composition of portfolio . . . . .	60
<b>7</b>	<b>Conclusions and future works</b>	<b>63</b>
	<b>Bibliography</b>	<b>64</b>

# List of Tables

4.1	User document description . . . . .	36
4.2	Market document description . . . . .	37
4.3	Market expiration document description . . . . .	37
4.4	Chain document description . . . . .	38
4.5	Strike document description . . . . .	38
4.6	Option document description . . . . .	39
4.7	Future document description . . . . .	39
4.8	Group document description . . . . .	40
4.9	Strategy document description . . . . .	40
4.10	Position document description . . . . .	41
4.11	Portfolio document description . . . . .	42
4.12	Market history document description . . . . .	42
4.13	Chain history document description . . . . .	43

# List of Figures

2.1	Long Call Payoff Diagram [1]	10
2.2	Short Call Payoff Diagram [2]	12
2.3	Long Put Payoff Diagram [3]	14
2.4	Short Put Payoff Diagram [4]	15
2.5	Covered Call Payoff Diagram [5]	17
2.6	Covered Put Payoff Diagram [6]	18
2.7	Protective Put Payoff Diagram [7]	20
3.1	Model-View-Template architecture [10]	25
3.2	Docker architecture [16]	29
3.3	Flutter architecture [20]	30
4.1	Client-server model [21]	35
4.2	Web application - Markets page	36
5.1	Login page	49
5.2	Navigation drawer	49
5.3	Example of how the app works	50
5.4	Markets page	51
5.5	Stocks page	51
5.6	Price history chart	52
5.7	Positions page	52
5.8	Futures page	53
5.9	Chains page	53
5.10	Strategies page	54
5.11	Strategy information	54
5.12	Strategy chart	55
5.13	Portfolio information	55
5.14	Portfolio performance	56
5.15	Portfolio strategies	56
6.1	AMZN Oct Long Call strategy	59

6.2	AMZN Oct Long Call payoff diagram . . . . .	61
6.3	AMZN Oct Long Call share function . . . . .	61
6.4	Portfolio performance . . . . .	62
6.5	Portfolio performance . . . . .	62



# Chapter 1

## Introduction

Nowadays, trade financial instruments has become easier and more accessible to everyone, allowing people to execute market orders in a very simple way and at a very low cost. This is possible thanks to the new upgrades of the banks and the online brokers. Despite the increasing of the usage of this tools, especially for the derivative instruments like options and futures, it is difficult for the persons to analyze and study the data and statistics about them. The reason of this problem is that banks and online brokers do not provide a lot of data and charts, so if people want to understand better the world of derivative instruments, they have to use external platforms but these platforms are often expensive and hard to use for a beginner trader. In response to this necessity, the idea of design and development a platform which respect the requirements suggested by a professional trader was born and a web platform was created.

As a following stage, we desired to extend this platform also for mobile devices. The work of this thesis consists in fact in the design and development of a mobile application. The reason is that even more functionalities and activities that before was performed only on desktop device, now are performed on the mobile devices. In this way the user can access to the application in every moment and in a very simple way. However, the mobile application does not substitute completely the web application because these two versions are complementary. The main purpose of the app is to provide to the user a consultative instrument. All the data and charts that are available in the web applications are also displayed in the mobile app, obviously they have been adapted to be usable on a much smaller screen. Instead, the functionalities of creation and deletion of strategies are maintained only for the web application because these operations need to display a lot of data and the mobile screen is clearly not suitable for this situation. But, in addition to the consultation function, the app implements social activities. The user has the possibility to share charts and data about markets and strategies through external social apps.

## **1.1 Structure of the thesis**

This thesis is divided in chapters where each of them describe in detail one specific argument which is part of the whole project. The Chapter 2 explains in a exhaustive way all the financial knowledge that is necessary as background to clearly understand the functions of the application. Specifically, there is a complete description of the options, the most important derivative instrument. In addition to the theory part, different options strategies that the user can implement are also presented.

After seeing the financial part, we move to the part of design and development of the application, in the Chapter 3 you will find the information about the data sources and technologies adopted. The data sources chosen are listed and for each of them it is indicated how the data necessary for the application are retrieve while for the technologies there is an overview of them where all the technologies used in this project are presented, justifying its choice.

The Chapter 4 contains the explanation of the whole architecture of the platform, in particular it is presented in a precisely way how the data are stored in the database, which are the asynchronous tasks used to retrieve the financial information and which are the APIs made available for the user with a proper description.

The complete mobile application is introduced in the Chapter 5. The data management, the user interface and the various implemented functionalities are here explained from a technical point of view.

Finally, in the Chapter 6, three different use cases are presented, they include the most common operations that the user can perform with a complete functional description.

# Chapter 2

## Finance

In this chapter, we are going to see an overview about the financial instruments and main actors regarding options trading which build the core argument of the mobile application.

### 2.1 Trading

Trading consist of buying and selling different types of financial instruments, such as stocks, bonds, currencies, commodities and derivatives. Trading should not be confused with investing, which instead suggests a buy-and-hold strategy. The aim of trading is to be profitable over time and this depends on the trader's ability. In particular, in this work, the focus is on options trading. With this instrument, we are going to discover and understand how to trade in the options markets using a wide range of options strategies.

Some terms and metrics about trading which are necessary to know before go ahead are presented below.

**Payoff diagram:** is a graphical representation of the potential outcomes of a strategy. The vertical axis of the diagram represents the profit or the loss of the strategy while the horizontal axis represents the underlying asset price.

**Volume:** is the amount of security that is traded over certain period of time (usually during the course of a day). Securities with more volume are more liquid which represents an important indicator in technical analysis.

**Open interest:** is the total number of derivative contracts that are still open and change with the new opening or closing positions. This metric is uses very often in options markets, and it provides an accurate picture of the derivatives trading activity.

**Open price - close price:** are the respective price at which a security first trades when an exchange opens on a trading day and the price of the last transacted security before the market closes on a trading day. **High price - Low price:** are the respective highest price at which a security is traded during the course of a trading day and the lower price at which a security is traded during the course of a trading day.

**Last price:** theoretically it corresponds to the closing price but it is not always like this. The true last trade may be posted anywhere from 30 seconds to 30 minutes after the market close. This because a few minutes are required to process the orders and determine which among the trades actually was the last trade.

**Bid-ask spread:** is the difference between the ask price and the bid price for an asset in the market. The bid price corresponds to the highest price that a buyer is willing to pay for an asset while the ask price is the lowest price that a seller is willing to accept.

**Profit:** is calculated as the revenue less the cost of a certain strategy.

## 2.2 Financial markets

As we said before, there are different types of financial instruments and, for each of them, there is a specific market that allows their trading. The principal markets are the following: stock market, index market, bond market, money market, derivatives market, forex market and commodities market.

*Stock market:* in this market, buyers and sellers meet to exchange equity shares of public corporations. Investors will own company shares in the expectation that share value will increase or that will get dividend payments or both.

*Index market:* indexes measure the performance of a group of securities with the aim to replicate a certain area of the market. So, the calculation of the index value comes from the prices of the underlying holdings. The securities belonging to an index, can be weighted in a different ways, the most common is the market-cap-weighting.

*Bond market:* in this market, investors buy debt securities that are brought to the market by either corporations or governmental entities. These last use the earnings from bonds to finance infrastructural improvements and pay down debts. Bonds are usually fixed-income instruments because paid a fixed interest rate to debt holders at the maturity date.

*Money market:* this market is characterized by a high degree of safety and relatively low rates of return. An investor can purchase a money market mutual fund, short-term certificates of deposit (CDs) or opening a money market account at a bank.

*Derivatives market:* derivatives are financial contracts, set between two or more parties where the value derives from an underlying asset. Prices of derivatives come from fluctuations in the underlying asset and can be trade on exchanges. The most common derivatives are options and future contracts.

*Forex market:* is the market the establishes the exchange rate for currencies around the world. It is possible to buy, sell, exchange, and speculate on the relative exchange rates of various currency pairs.

*Commodities market:* is the market where is possible to buy, sell, and trade raw materials or primary products like oil, gold, or sugar. Natural resources are called hard commodities while livestock or agricultural goods are call soft commodities.

## 2.3 Exchanges

An exchange is a marketplace where stocks, bonds, currencies, commodities, derivatives and other financial instruments are traded. Exchanges guarantee fair and orderly trading giving a efficient diffusion about price information for the instruments traded. Each exchange trades only certain types of financial instruments. Since we are interested on options trading, we will focus on exchanges which trade these instruments, in particular: CBOE, CME and EUREX.

*CBOE Exchange:* originally known as the Chicago Board Options Exchange, is the world's largest options exchange, it offers options on over 2,200 companies, 22 stock indices, and 140 exchange-traded funds (ETFs). CBOE is the creator of the CBOE Volatility Index (VIX) that is the most widely used instrument to measure market volatility.

*CME Exchange:* the Chicago Mercantile Exchange is a global derivatives marketplaces based in Chicago. CME especially trades options concern the following sectors: agriculture, energy, stock indices, foreign exchange, interest rates, metals and real estate.

*EUREX Exchange:* one of the largest futures and options markets in the world. Mainly operates with European-based derivatives but provide also the possibility through electronic access to traders to connect from more than 700 locations around the world.

## 2.4 Derivatives

The focus of this work concerns derivative instruments, so we are going to understand them in detail. A derivative is a financial contract whose value is derived from another entity which is also known as the underlying, which include for example stocks, bonds, commodities, currencies and financial indicators such as interest rates, market indexes.

There are two main activities that can be done by using derivative products: hedging and speculation. Hedging consists of use derivatives to remove the risk of losses to their holdings that are caused by fluctuations in the value of underlying. Instead speculation, taking advantage of the high leverage of derivatives, use them to increase the profit arising if the value of the underlying mover in the direction they expect.

Different types of derivatives exist and the most common are: options, futures, forward contracts, and contracts for the difference (CFD). Specifically we will see in details the first two types. It is also possible to combine different basic derivatives to create more complex derivatives.

## 2.5 Options

An option is a contract between two parties, the option buyer purchases the right (but not the obligation) to buy/sell (depending by the option) a certain quantity of an underlying at a predetermined price from/to the option seller within a fixed period of time. The option buyer is also called holder while the option seller is also called writer. Options holders are said to have long positions while writers are said to have short positions. We can distinguish two different types of options trading strategies: bullish and bearish: bullish strategies (e.g. Long Call) indicates that the options trader expects the increase of the price of the underlying while bearish strategies (e.g. Long Put) indicates that the options trader expects the decrease of the price of the underlying.

**Moneyness:** is a term which describes the relationship between the strike price of an option and the current trading price of its underlying. The moneyness is shown in the following terms: in-the-money (ITM), at-the-money (ATM) and out-of-the-money (OTM).

ITM options cost more than ATP and OTM because their premiums consist of significant intrinsic value. Call options are defined as ITM when their strike price are below the current trading price of the underlying while put options when their strike price is above the current trading price of the underlying.

ATM options have a strike price that is equal to the market price of the underlying. They have no intrinsic value but only time value.

OTM options are less expensive than ITM and ATP because, like ATP, they do not have intrinsic value and therefore the premium is only made up of the time value. Call options are defined as OTM when their strike price are above the market price of the underlying while put options when their strike price are below the market price of the underlying.

**Exercise and assignment:** the holder of an option can execute the right to buy or sell (depending if is a call option or a put option) the underlying at the strike price. Two different types of exercise style exist: American and European. With the American style, options may be exercised at any time before the expiry date whereas with the European style, options may only be exercised on the expiry date.

The options writer has a duty to provide the terms of the options when the written option is exercised by the options holder. If the call option is assigned, the writer must sell the required amount of the underlying at the strike price. Instead, in the event of an assignment of a put option, the writer will have to buy the obligated quantity of the underlying at the strike price.

**Implied volatility (IV):** is the market's prediction of the volatility of the underlying value of the option. It is calculated through an option pricing model where a mathematical relation between the volatility of the underlying security and the price of its options has been established. When buy or sell options, one should take note of the IV of the underlying because, generally, when the IV is high, options are more expensive while they are less expensive when it is low.

**Options chains:** display the call and put options available for various strike prices for a specific underlying and expiration date. In-the-money options are typically highlighted to differentiate between out-of-the-money options.

**Break-Even Point:** in options trading, is the market price that the underlying asset needs to reach in order for an options buyer to avoid a loss by exercising the option.

### 2.5.1 Option contract specifications

In an option contract, several terms are specified and it is important to be aware of their meaning.

**Option type:** option can be of type put or type call. Call is the option that gives the buyer the right to buy the underlying whereas put is the option that gives the buyer the right to sell it.

**Strike price:** is the price at which the underlying asset must be purchased (in case of call options) or sold (in case of a put options) by the holder when the option is exercised. It is a decisive factor of the option's premium: in the case of call options, the higher is the strike price, the cheaper is the option while in the case of put options, the higher is the strike price, the more expensive is the option. The price distance between the strike prices is dependent on the market price and the type of the asset of the underlying. The lowest values are typically 1, 2 and 2.5 points and the highest are typically 5 and 10 points.

**Premium:** is the price paid by the holder to purchase the option. The premium is dependent not only on the strike price but also on the volatility of the underlying and on the remaining time to expiration. It consists of two parts: intrinsic value and time value. Only in-the-money options have intrinsic value, for call options it can be calculated as the difference between the current trading price and the strike price while for the put options it can be calculated as the difference between the strike price and the current trading price. The time value is dependent on three factors: the time left to exercise the option, the moneyness and the volatility of the underlying asset. The time value diminishes when its expiry date arrives and becomes worthless after the date. For in-the-money options, the time value can be calculated as the difference between the option premium and the intrinsic value whereas for out-of-the-money options the time value is equal to the option premium. In general, higher volatility will result in higher time value.

**Expiration date:** refers to the date when an option contract becomes invalid and the right to exercise it ceases to exist. The expiration month is defined for each option contract. The expiration date depends on the type of options, for stock options the date falls on the third Friday of the expiration month.

**Option style:** as seen before, the option contract can be either American style or European style, this style determines the different manner in which the options can be exercised.

**Underlying asset:** is the security that the holder can buy (in case of call options) or sell (in case of put options) from the writer in the moment in which the option is exercised. The most commonly used types of securities are stocks, currencies, indexes and commodities.

**Contract multiplier:** specifies the amount of the underlying which must be sent when the option is exercised. For instance, for stock options, every contract refers to 100 shares.

## 2.5.2 Call options

A call option is an option contract in which the buyer has the right to purchase a specified amount of the underlying at a strike price up to its expiration date. Rather, the seller of a call option is obligated to sell the underlying at the strike price if the option is exercised. The call option writer receives a premium to take the risk of the obligation.

### Long Call

The easiest way of to trade call options is to buy them, in addition to their simplicity there is also the opportunity to generate a great profit from successful trades (Figure 2.1).

- **Leverage:** compared to buying the underlying shares themselves, the buyer of call options is able to leverage as lower-priced call options appreciate in value more quickly as a percentage for each point increase in the price of the underlying share. However, call options have a limited lifespan. If the price of the underlying share does not exceed the strike price before the option expires, the call option will expire worthless.
- **Unlimited profit potential:** since there is no limit to how high the stock price can be at expiration date, there is also no limit to the maximum profit possible when implementing the long call option strategy. There is a profit when the price of underlying moves above the sum of strike price of long call and premium paid, in this case the profit is computed in this way:  $\text{profit} = \text{price of underlying} - (\text{strike price of long call} + \text{premium paid})$ .
- **Limited loss potential:** the risk for the long call option strategy is limited to the price paid for the call option, regardless of how much low the stock price is trading on the expiration date. The maximum loss occurs when the price of the underlying is less or equal than the strike price of long call, in this case the loss is equal to the premium paid.
- **Break-Even Point:** the underlying price at which the Break-Even Point is achieved for the long call position can be calculated using the following formula:  $\text{Break-Even Point} = \text{strike price of long call} + \text{premium paid}$ .

*A simple example:* let's assume that the stock of a certain company is trading for \$100. A call option contract with a strike price of \$100 expiring within a month is priced at \$5. You are confident that this stock will increase significantly over the next few weeks. So, you paid \$500 to purchase a single \$100 call option covering 100 shares of this stock. Let's say you were correct and the price of the stock reaches to \$110 when the option expires. With underlying stock price at \$110, if

you exercise your call option, you invoke your right to buy 100 shares of the stock at \$100 each and can sell them immediately in the open market to \$110 a shares, the total amount you will receive from the exercise is \$1000. Given that you paid \$500 to buy the call option, so your net profit for the whole transaction is \$500. However, the underlying stock must exceed \$105 (Break-Even Point) at expiration to make a profit. If you were wrong in your forecast and the stock price had instead dived to \$90, your call option will expire worthless and your total loss will be the \$500 that you paid to buy the option.



**Figure 2.1:** Long Call Payoff Diagram [1]

### Short Call

Rather than buying call options, one can also sell them for a profit (Figure 2.2). Call options writers, sell call options in the hope that they expire with no value so that they can earn the premiums. Selling calls is more risky, but can also be highly profitable if done in a proper way. The calls sold may be of two different types: covered calls or naked (uncovered) calls.

Covered calls mean that the call option seller holds the required amount of the underlying securities. The covered call is a popular option strategy that allows a stock holder to generate additional income from their securities by selling call options on a periodic basis.

Instead naked call means that the call option seller does not have the mandatory amount of the underlying security, this option strategy is very risky.

The primary purpose of writing naked calls is to collect the premiums when the options expire with no value. For instance, one would write an out-of-the-money naked call every month and if the stock price remains stable or decreases, the premiums are collected and the process is repeated while the perception of the market situation remains the same.

- **Limited profit potential:** maximum earnings are limited and reflect the premium received for the sale of call options. The maximum profit is obtained when the price of the underlying is less than or equal to the strike price of the short call.
- **Unlimited loss potential:** if the stock price rises significantly on expiry, the naked call writer will need to meet the option requirements to sell the obligated stock to the option holder at the lower strike price by buying the stock from the open market at the higher market price. As there is no limit to how high the stock price may be at expiry, the maximum potential loss for writing naked calls is theoretically limitless. A loss occurs when the price of the underlying is higher than the strike price of short call plus the premium received. The formula for determining the loss is:  $\text{loss} = \text{price of underlying} - (\text{strike price of the short call} + \text{premium received})$ .
- **Break-Even Point:** the underlying price at which Break-Even Point is reached for the naked call position can be calculated using the following formula:  $\text{Break-Even Point} = \text{strike price of the short call} + \text{premium received}$ .

*A simple example:* a certain stock is currently negotiated for \$100. An options trader chooses to write a naked call option with a strike price of \$100 expiring within a month at price of \$5. Therefore, he gets \$500 to write the call option. At the expiry date, the stock had increased to \$110. Since the strike price of \$100 for the call option is below than the actual negotiating price, the call is assigned and the writer buys the shares for \$11000 and sell it to the option holder at \$10000, resulting in a loss of \$1000. However, as a result of receiving \$500 earlier on, his net loss is \$500. Instead, if the stock price were decreased to \$90, for example, the call expires without value and the writer of the naked call retains the \$500 in premiums received as profit. While the stock price remains at \$105 (the Break-Even Point value) or less, there will be no loss to the naked call writer.

### 2.5.3 Put options

A put option is an option contract in which the buyer has the right to sell a specified quantity of an asset at a specified strike price until its expiration date. For



**Figure 2.2:** Short Call Payoff Diagram [2]

the seller of the put option, this means an obligation to purchase the underlying asset at the strike price in case of exercise of the option. The writer of the put option receives a premium to assume the risk related to the obligation.

### Long Put

The long put option strategy is a basic strategy in options trading in which the investor purchases put options with the conviction that the price of the underlying will be substantially lower than the strike price before the expiry date (Figure 2.3).

- **Put buying vs. short selling:** in comparison to short selling the stock, it is more convenient to bet against a stock by buying put options, since the investor does not need borrow the stock. Moreover, the risk is capped at the premium paid for the put options, as opposed to unlimited risk during the short of the underlying stock. But put options have a finite lifetime, if the price of the underlying stock does not fall below the strike price before the option expires, the put option will expire without value.
- **Unlimited profit potential:** since stock price in may theoretically reach zero on the expiration date, the maximum possible profit from using the long

put strategy is only limited to the strike price of the purchased put minus the price paid for the option.

- **Limited loss potential:** the implementation risk of the long put strategy is limited to the price paid for the put option, regardless of how high the stock price trades on the expiration date. The maximum loss occurs when the price of the underlying is more than or equal to than the strike price of the long put, this maximum loss is equal to the premium paid.
- **Break-Even Point:** the underlying price at which Break-Even Point is achieved for the long put position can be calculated using the following formula: Break-Even Point = strike price of the long put - premium paid.

*A simple example:* let's assume that the stock of a certain company trades for \$100. A put option contract with a strike price of \$100 that expires in one month is priced at \$5. You think the stock is going to drop sharply in the next few weeks and so you paid \$500 to buy a single \$100 stock put option covering 100 shares. If you were correct and the price of the stock falls to \$90 when the option expires. The put option will now be in-the-money with a \$1000 intrinsic value and you may sell it for that amount. Since you had paid \$500 to buy the put option, your net profit for the whole trade is \$500. However, if you were wrong in your evaluation and the stock price had rather gone up to \$110, your put option will expire with no value and your total loss will be the \$500 that you paid for the option.

### Short Put

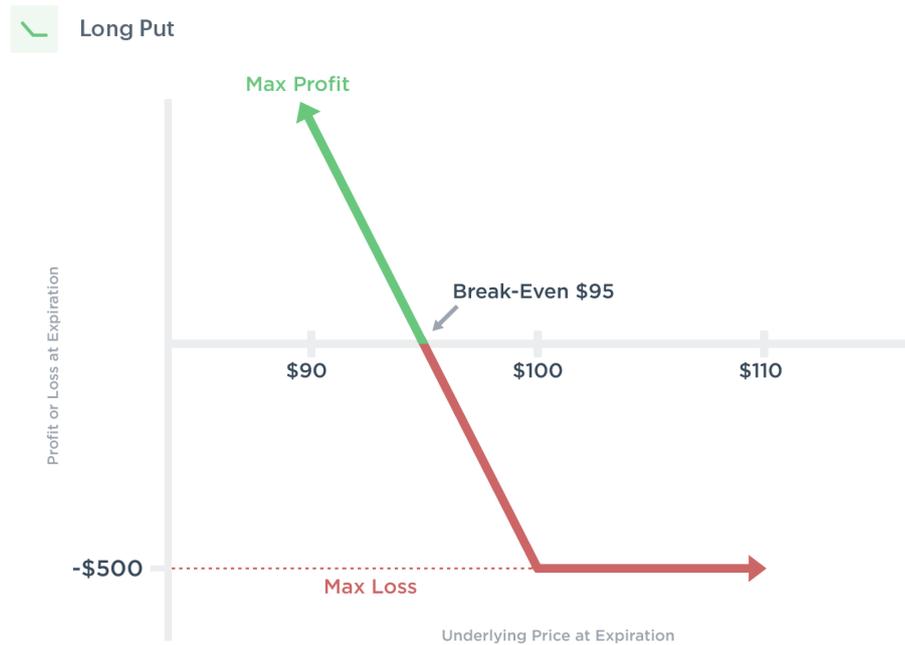
Rather than buying options, it is also possible to sell them for a profit (Figure 2.4). Put options writers, sell the put options hoping that they expire worthless so that they can earn the premiums. Sale puts options is more risky, but may be profitable if done correctly. The puts sold can be of two different types: covered puts or naked (uncovered) puts.

Covered puts mean that the put option writer is also short the required amount of the underlying.

Rather, naked puts mean that the put option writer has not short the required amount of the underlying when the put option is sold.

Naked put write is an option strategy executed to earn a consistent profits through continuous premium collection.

- **Limited profit potential:** the profit from the uncovered put write is limited to the premiums received for the options sold and contrary to the cover put write, since the uncovered put writer is not short on the underlying, he does not have to handle a loss if the price of the security increases at expiration. The maximum profit is reached when the price of the underlying is higher



**Figure 2.3:** Long Put Payoff Diagram [3]

than or equal to the strike price of the short put, and it is equivalent to the premium received.

- **Unlimited loss potential:** whereas the premium received may amortize a slight decline in stock price, the loss resulting from a catastrophic fall in stock price of the underlying can be enormous when implementing the uncovered put write strategy. A loss arises when the price of the underlying is lower than the difference between the strike price of the short put and the premium received. The formula to calculate the loss is the following:  $\text{loss} = \text{strike price of short put} - (\text{price of underlying} + \text{premium received})$ .
- **Break-Even Point:** the underlying price at which Break-Even Point is reached for the uncovered put write position may be computed with the following formula:  $\text{Break-Even Point} = \text{strike price of short put} - \text{premium received}$ .
- **Writing naked puts to purchase stocks:** the greatest risk that must be faced by the writer of uncovered put is that if the price of the underlying falls below the put strike price, he has to purchase the shares at the put strike price. However, for a long-term investor who seeks to go long on the stock at

a discounted price, writing naked puts can be a great way to purchase stock. This can be done by writing uncovered puts with a strike price at or close to its target entry price. If the stock price goes down below the put strike and the puts gets assigned, he gets to make the stock purchase at the desired price. In addition, he receives a further discount in the form of the premium earned through the sale of put options. Even if the put strike price has not been reached and the stock has not been purchased, he still gets to keep the premiums.

A *simple example*: let's say that certain stock is trading at \$100. An options trader chooses to write an put option with a strike price of \$100 expiring within a month at price for \$5, resulting in a \$500 of premium. If the stock remains at \$100 when it expires, the put expires worthless and the trader gets to keep the \$500 in premium as profit. This is also its maximum profit and is reached as long as the stock trades over \$100 at the option expiry date. If instead the stock falls to \$90 on expiration, then the put expires in-the-money with \$1000 in intrinsic value. The put must be bought back for \$1000 and subtracting the initial credit of \$500 taken, resulting in a net loss of \$500.



Figure 2.4: Short Put Payoff Diagram [4]

## 2.5.4 Other options strategies

Some of the other most common and used strategies:

### Covered call

Using the covered call option strategy (Figure 2.5), the investor gains a premium writing calls while appreciating all the advantages of the underlying stock shareholding, such as dividends, unless a notice of exercise is given to him in the course of the written call and is obliged to sell the shares. However, the profit potential of selling covered call options is limited.

- **Limited profit potential:** other than the premium received for writing the call, the profit of the OTM covered call strategy also includes a gain if the underlying stock price increases until the strike price of the call option sold. The maximum profit is reached when the price of the underlying is greater than or equal to the strike price of short call. The calculation of the profit is as follows: maximum profit = premium received + strike price of short call - purchase price of underlying.
- **Unlimited loss potential:** potential loss to this strategy may be very significant and occurs as the price of the underlying security drops. However, this risk is no different than the risk to which the common stock owner is exposed. In fact, the loss of the covered call writer is slightly amortized by the premiums received for writing the calls. The loss arises when the price of the underlying is below the difference between the purchase price of the underlying and the premium received. It can be computed as follows: loss = purchase price of the underlying - premium received.
- **Break-Even Point:** the underlying price at which the Break-Even Point is reached for the covered call (OTM) position may be computed using the following formula: Break-Even Point = purchase price of underlying - premium received.

*A simple example:* an options trader buys 100 shares of certain stock trading at \$100 and writes a call with a strike price of \$105 expiring within a month at price for \$5. Thus, he pays \$5000 for the 100 shares of the stock and receives \$500 to write the call option resulting in a total investment of \$4500. At the time of expiry, the stock had risen to \$105. Since the strike price of \$105 for the call option is equal to the current negotiating price, the call is assigned and the writer sells the shares for a \$500 profit. This brings his total profit to \$1000 after gain \$500 in premiums received for writing call. Instead, if the stock price had fallen by 5 points to \$95, the call writer would incur \$500 loss for holding the 100 shares of the stock. However, his loss is amortized by the \$500 in premiums received, which makes the whole trading operation without loss.



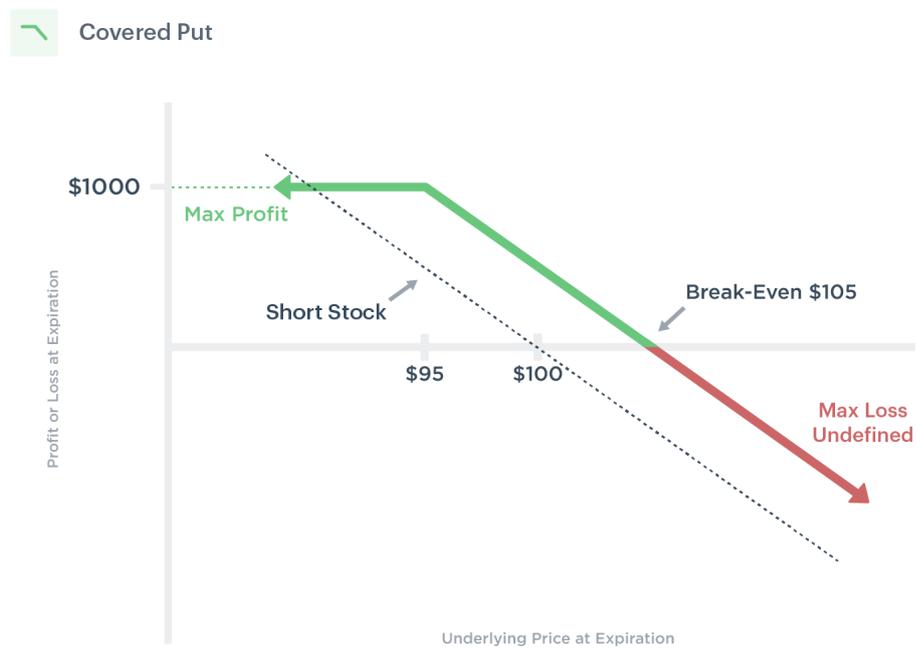
**Figure 2.5:** Covered Call Payoff Diagram [5]

### Covered put

Writing covered put (Figure 2.6) is an options trading strategy that includes the writing of put options while shorting the obligated shares of the underlying.

- **Limited profit potential:** profit for the covered put option strategy is limited and the maximum gain is equal to the premiums received for the options sold. The maximum profit is reached if the price of the underlying is below or equal to the strike price of short put.
- **Unlimited loss potential:** theoretically, the maximum loss for the covered put options strategy is unlimited since there is no limit to how high the stock price may be at expiry. A loss arises when the price of the underlying is greater than or equal to the strike price in addition to the premium received. The formula for determining the loss is as follows:  $\text{loss} = \text{price of the underlying} - (\text{strike price} + \text{premium received})$ .
- **Break-Even Point:** the underlying price at which Break-Even Point is achieved for the covered put position can be calculated using the following formula:  $\text{Break-Even Point} = \text{strike price} + \text{premium received}$ .

A *simple example*: let's assume that a certain stock is trading at \$100. An options trader writes a put with a strike price of \$95 expiring within a month at price for \$10 while shorting 100 shares of the stock. The net credit taken to get into the position is \$1000, which is also its maximum possible profit. On expiration, the stock is trading at \$95. The put expires with no value while the trader covers his short position without loss. In the end, he gets to keep the whole credit taken as profit. If instead the stock falls to less than \$95 at expiry, the short put will expire ITM with a some loss which is compensated with the gain on the short stock position. So the profit is still the initial credit of \$1000 taken on entry into the trade. However, if the stock price goes above the Break-Even Point of \$105 on expiration, there will be significant loss.



**Figure 2.6:** Covered Put Payoff Diagram [6]

### Protective Put

Investors also purchase put options in order to protect an existing long stock position. Put options used in this way are also referred to protective puts.

- **Unlimited profit potential:** there is no limit on the profit which can be made with this strategy. The profit is realized when the price of the underlying is higher than the buying price of the underlying plus the premium paid,

the formula for calculating it is as follows:  $\text{profit} = \text{price of the underlying} - (\text{purchase price of the underlying} + \text{premium paid})$ .

- **Limited loss potential:** the maximum loss for this strategy is limited and represents the premium paid for the purchase of the put option. The maximum loss arises when the price of the underlying is lower than or equal to the strike price of the long put, it can be achieved using the following formula:  $\text{maximum loss} = \text{premium paid} + \text{purchase price of the underlying} - \text{strike price}$ .
- **Break-Even Point:** the underlying price at which the Break-Even Point is achieved for the protective put position can be computed using the following formula:  $\text{Break-Even Point} = \text{purchase price of the underlying} + \text{premium paid}$ .

*A simple example:* an options trader holds 100 shares of a certain stock which trades for \$100. He implements a protective put strategy by buying a put option with a strike price of \$95 expiring within a month for \$1000 to secure his long stock position against a potential crash. The maximum loss occurs when the stock price is \$95 or less on expiry and is equal to \$1000. There is no limit to the profit that can be reached if the stock price rises. Assuming that the stock price rises to \$125, his long stock position will earn \$1500. Excluding the the \$1000 paid for the protective put, his net profit is \$500.

### Options spreads

Options spreads are options trading strategies where an equal number of options of the same class on the same underlying are purchased and sold simultaneously, but with different strike price and/or expiration date. A spread which is composed by using call options is named call spread while a spread which is composed by using put options is named put spread. There are three basic classes of spreads: vertical spread, horizontal spread an diagonal spread.

*Vertical spreads:* made up of options with the same expiration date but different strike price.

*Horizontal spreads:* made up of options with the same strike price but with different expiration date.

*Diagonal spreads:* made up of options with different expiration date and also different strike price.

There are also different possibilities to combine several options strategies: purchasing and/or selling of both call and put options on the same underlying asset. Some of the options combinations are: option straddle, option strangle, option strip and option strap.

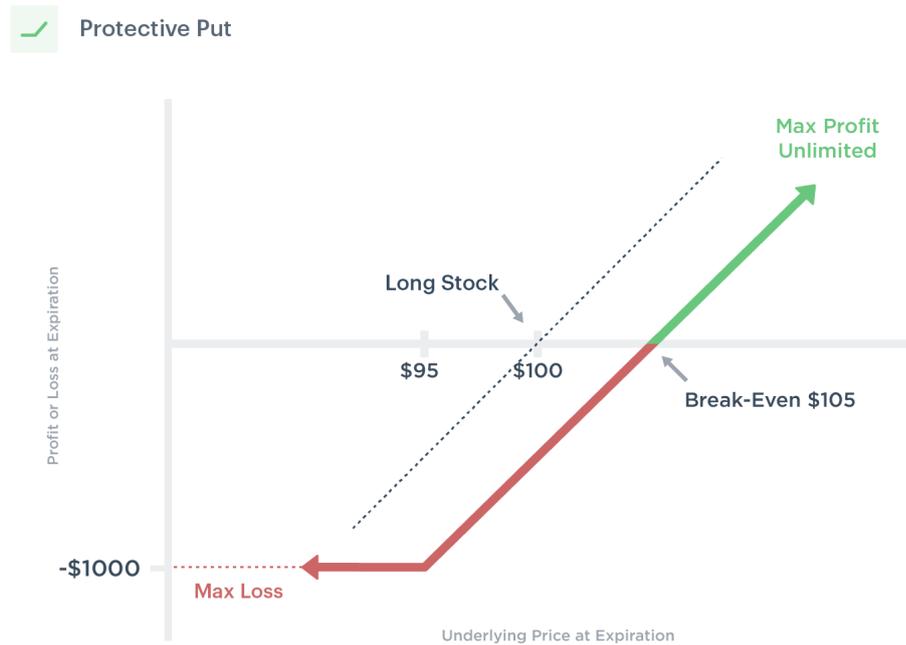


Figure 2.7: Protective Put Payoff Diagram [7]

### 2.5.5 Greeks

In options trading, you may note the use of some Greek alphabets when describing the risks associated with different positions. They are known under the name of "Greeks" and the following is a brief description of the four most widely used.

*Delta*: is the rate at which the option price changes relative to the price of its underlying value. Option delta is a value between 0 and 1 for calls options (0 and -1 for put options) and reflects the rise or fall in the price of the option in response to a 1 point move in the price of the underlying asset. For instance, deep ITM options have delta near to 1 while deep OTM options have delta near to 0.

*Gamma*: is a measure of the velocity of variation of the option delta. It is expressed in percent and reflects the delta variation in response to a movement of one point in the price of the underlying. Like the delta, the gamma is in constant evolution, even with minute movements of the price of the underlying. It is typically at its peak value when the stock price is close to the strike price of the option and diminishes as the option comes deeper ITM or OTM. For example, options which are very deep ITM or OTM have gamma value near to 0.

*Theta*: is a measure of the temporal decline of the option. It measures the rate at which options drop in value, particularly the time value, as the expiration date approaches. Typically expressed as a negative number, the theta of an option reflects how much the value of the option will decline each day.

*Vega*: is a measurement of the impact of variations in underlying volatility on the price of options. In particular, it expresses the variation in the option price for each percentage point variation in the underlying volatility.

## 2.6 Futures

A future contract is a standard contract which requires delivery of a specific amount of a specific product at a certain point in the future at a predetermined price. Future contracts are negotiated all over the world and cover a broad range of commodities such as agricultural products, stock farming, energy, metals and financial products such as market indexes, interest rates and currencies. The main goal of the futures market is to enable those who want to manage price risk (hedgers) to shift this risk to those who are prepared to take that risk (speculators) in exchange for a profit opportunity.

*Hedging*: companies use a long hedge to lock in the price of a commodity they want to buy in the future. Instead, companies use a short hedge if they want to lock a selling price for a commodity for sale in the future.

*Speculation*: futures speculators take a long futures position when they expect the price of the underlying is going to increase. Instead, they adopt a short futures position when they think that the price of the underlying is going to decrease.

# Chapter 3

## Data sources and technologies

Before start to develop a mobile application, it is necessary to do a several number of choices about data sources and technologies. These choices will determinate the result of the final application.

### 3.1 Data sources

The data sources chosen to retrieve the data needed for the application are web services: a services running on a computer device which are listening for request at a particular port over a network, serving web documents like JSON<sup>1</sup> files and creating web application services. Especially, through the usage of the REST<sup>2</sup> APIs<sup>3</sup>, the access to data exposed by web services is become more easily. The data that the application need, are about the information of finance summary, stocks, quotes and options.

As we said in section 2.3, we are interested in CBOE, CME and EUREX exchanges, so we are going to use the web services provided by them.

#### CBOE web services

CBOE provides different services through its official website that gives us a lot of information about US stocks, indexes options and other market. This information

---

<sup>1</sup>JavaScript Object Notation

<sup>2</sup>REpresentational State Transfer

<sup>3</sup>Application Programming Interface

are updated any 10-15 minutes. The two public REST APIs useful for the the application are the following:

- **GET /symbol\_book/option-roots.json**: permits to download all options catalogue info. The response is an array of JSON objects, using the option-root (the identifier) of the objects is possible to download further information through the other web service.
- **GET /options/option-root.json**: permits to download the information about a single option-root. The response includes information about the price of the underlying and about the options contract.

The first API is useful to have a list of the markets handled by CBOE. The second API provides all information (like options prices for all expiration) about a single market performing only one call. This operation is done every 15 minutes.

### **CME web services**

CME provides information about options and futures contracts of the most important indexes (S&P 500, Nasdaq, Russell 2000, etc.) and its website makes them publicly available. The APIs useful for our purpose are the following:

- **GET /services/product-slate**: allows to download the list of the products or underlying. Each elements contains the information of the product and the product\_id (its identifier) that through the others APIs permits to obtain futures and options related to the product.
- **GET /Quotes/Future/product\_id/G**: allows to download the futures contract related to a particular product for each available expiry date.
- **GET /Quotes/Option/product\_id/G/expiration\_id** : allows to download the options contract related to a particular product and expiration date.
- **GET /Volumes/Details/product\_type/product\_id/last\_trade\_date/P/**: allows to download the statistics data about volumes and open interest of the future or option contract.

The first API is used to download the CME product catalog and initialize the catalog of the application. The other three APIs are called at a regular intervals of 15 minutes during the trading hours for updating markets, futures and options data.

## EUREX web services

Different from the other two exchanges, EUREX does not provides details about options and futures, it only offers the access to this information through their dedicated HTML<sup>4</sup> pages that require an extraction of the data that are needed. Computing this extraction is possible to obtain information about options and futures contracts of the European indexes like EuroStoxx50 and DAX.

## 3.2 Technologies

The technologies selected are mainly divided into three different areas. Django framework is used for the server-side, MongoDB is used for database management and Flutter framework is used for client-side.

### 3.2.1 Django

*Django*[8] is a free and open-source, Python-based web framework with the main goal of ease the creation of complex, database-driven websites. The framework underlines reusability and pluggability of components which lead to write less code and to a rapid development.

*Python*[9] is a high-level, general-purpose and very popular programming language. It is used in different fields like web development and Machine Learning and it is very indicated both for beginners and advanced programmers. Python is widely diffused and allows object orientation programming. The strongest point in favour of Python is the fact that has a huge collection of standard libraries which can be used for several purposes.

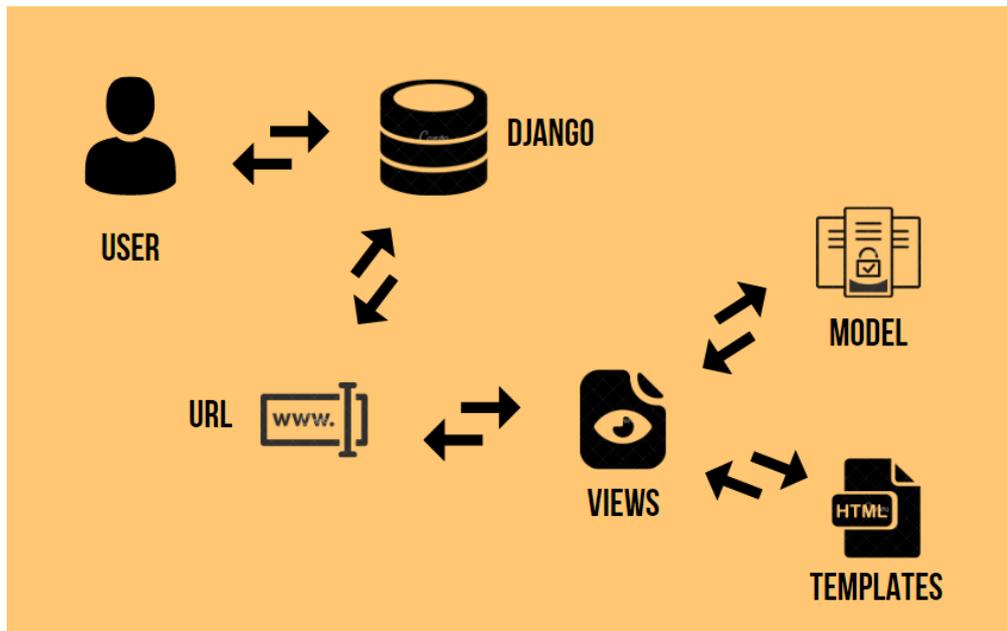
Django is based on MVT (Model-View-Template) architecture (Figure 3.1):

- *Model*: act as the interface of the data (it is represented by a database) and is responsible for maintaining them.
- *View*: is the user interface, it handle the user interaction.
- *Template*: composed by static parts of the HTML output and by some syntax describing how dynamic content will be inserted.

In addition to Django, there are other available frameworks which have different characteristics. Two of the possible alternatives are Flask and Web2py.

---

<sup>4</sup>HyperText Markup Language



**Figure 3.1:** Model-View-Template architecture [10]

*Flask*: a micro web application framework written in Python. The classification as micro framework means that it does not need any tools or libraries. It is built using the Werkzeug WSGI<sup>5</sup>(a standard for Python web application development) toolkit and Jinja2 template engine. The first implements requests, response objects, and other utility operations while the second renders dynamic web pages by combining a template with a specific data source. Simple web applications are best created using Flask, however more complex ones can be sent more quickly with Django because its modules are already set up to provide speedy application development and layout.

*Web2py*: a full-stack web application framework written in Python. The classification as full-stack means that it contains all the components which are necessary to build a complete web applications. Its focus is on agile development and security. It is easy to run, it requires no installation and no configuration. Web2py follows MVC<sup>6</sup> design and is based on Python which is fast and scalable. Web2py was also influenced by Django and shares many of Django's features, including the ability to create forms from database tables and a large number of validators. Due to its smaller size, simpler learning curve, and lack of project-level configuration files,

<sup>5</sup>Web Server Gateway Interface

<sup>6</sup>Model View Controller

web2py differs from Django. Web2py is ideal for a beginning coder or new to web development while Django would be a better choice for a proficient Python user and you have a short timeline.

### 3.2.2 MongoDB

*MongoDB*[11] is the most popular NoSQL<sup>7</sup> DBMS<sup>8</sup>. The NoSQL notation means non-relational, so it is not based on the table-like relational database but use a different mechanism for storing and retrieving data. MongoDB also supports ACID<sup>9</sup> properties. *PyMongo*[12] is a Python distribution which contains tools for interacting with MongoDB databases from Python and it is very useful for our purpose.

MongoDB has several characteristic features. It is document-oriented, the data are stored in BSON<sup>10</sup> format that is similar to JSON format and it uses indexing for efficient searching with huge volumes of data in very short time. It scales horizontally through partitioning data across several server (this operation is called sharding). It increases the availability of data creating more copies of data on different server, so leveraging the redundancy, it protects the database from hardware failures. Aggregation operations handle data records and return the calculated outcomes.

There are relevant differences between SQL<sup>11</sup> databases and NoSQL databases.

- SQL databases have a typical schema showing the tables and the relationships between them while NoSQL databases does not have any type of schema, it is document-oriented.
- Complex join operations are not available in NoSQL databases.
- NoSQL databases allows a flexible and scalable document structure.
- NoSQL databases are faster than SQL databases thanks to the use of indexing technique.
- The terms Table, Tuple and Column of SQL databases are respectively related with the terms Collection, Document and Field of NoSQL databases.

Three possible alternatives of MongoDB are Cassandra, PostgreSQL and MySQL.

---

<sup>7</sup>Non-relational Structured Query Language

<sup>8</sup>DataBase Management System

<sup>9</sup>Atomicity, Consistency, Isolation, Durability

<sup>10</sup>Binary JSON

<sup>11</sup>Structured Query Language

*Cassandra*: an open source NoSQL distributed database developed by Apache that enables operational simplicity and support replication across data centers. Distributed databases are a key feature of the Cassandra system. Both technological and commercial benefits result from that. When an application is under a lot of pressure, Cassandra databases scale readily, and the distribution also shields data from being lost when a datacenter's hardware fails. A distributed architecture also offers technological advantages, such as the ability for developers to individually adjust the read and write query performance. Cassandra is a popularly used wide-column store created for specific use cases where the majority of operations are written using a single primary key while MongoDB is a general-purpose database that, because to its adaptable document format, extensive aggregate language, can handle a variety of use cases.

*PostgreSQL*: a widely popular open source object-relational database management system developed by Oracle, its main characteristics are reliability, data integrity, functionality and scalability. It uses the SQL language and extends it with many functionalities. PostgreSQL runs on all main operating systems and it supports ACID properties. Numerous functions in PostgreSQL are designed to support developers in creating applications, administrators in safeguarding data integrity and creating fault-tolerant systems, and in managing data regardless of the size of the dataset. PostgreSQL has a large ecosystem of SQL skills and tools for a relational data model but MongoDB, if the developer have data that needs to be provided at scale, allows to control the schema with a document database.

*MySQL*: a open source relational database management system which organizes data into tables which can be in relation each other. SQL is the language used to create, modify and get data from the relational databases. MySQL is quick, scalable, and simple. It was first created to easily work with big datasets. MySQL offers a comprehensive and practical collection of features. Connecting databases via the internet is a task that MySQL is well suited for due to its quickness, and safety. Compared to MySQL, MongoDB makes it possible to construct applications more quickly, manage many types of data, and scale applications extremely well. Also, the intricate ORM<sup>12</sup> layer that converts objects in code to relational tables is removed while using MongoDB.

### 3.2.3 Celery

*Celery*[13] is an open source asynchronous task queue (used as a mechanism to distribute work across threads or machines). There is a continuous process that

---

<sup>12</sup>Object-Relational Mapping

monitor task queues for new work to perform, these works are called tasks. Celery communicates via messages through the usage of a broker that allows to mediate between clients and workers. For our goal a recommended broker is used: *Redis*[14].

Celery does not need any configuration files and it is easy to use and maintain. When a connection loss or failure occurs, workers and clients will automatically retry to communicate. It permits to process millions of tasks by a single process only in a minute. Practically each component in Celery can be extended or used on its own.

### 3.2.4 Docker

*Docker*[15] is an open platform for developing and running applications, it allows to separate applications from infrastructure. Docker uses containers to build, share and run applications. A container is a isolated environment where the applications are packaged and run inside. These containers are very lightweight and have all that is need to run the application. Due to their architecture (isolated and secure), many containers can be run simultaneously on the same host.

Docker utilizes a client-server architecture (Figure 3.2): the Docker client communicate with the Docker daemon using a REST API. Docker daemon is responsible of building, running and distributing the Docker containers. Hosts contain images and containers: the difference is that the first are read-only template with instructions for creating a container while the second are a runnable instance of the images and they can be created, started, stopped, moved or deleted.

To work with applications composed of a set of containers, another Docker client is needed: Docker Compose.

### 3.2.5 Nginx

*Nginx*[17] is free and open-source software, it is a web server that can also be used as a reverse proxy, load balancer, mail proxy and HTTP cache. Nginx has an approach of asynchronous event-driven type to handling the requests. Due to this type of architecture it can provide very good performance under high loads.

Nginx was created with the express intention of surpassing the *Apache* web server. In fact, Nginx uses significantly less memory than Apache, and can handle approximately four times more requests per second. But, this increase in performance results at the price of reduce the flexibility.

There are two different distributions of Nginx: Nginx Open Source and Nginx Plus. As the name suggests, Nginx Open Source is the free and open-source version. Instead, Nginx Plus offers additional features with the payment of a subscription, some of them are: active health checks, session persistence based on cookies and DNS-service-discovery integration.

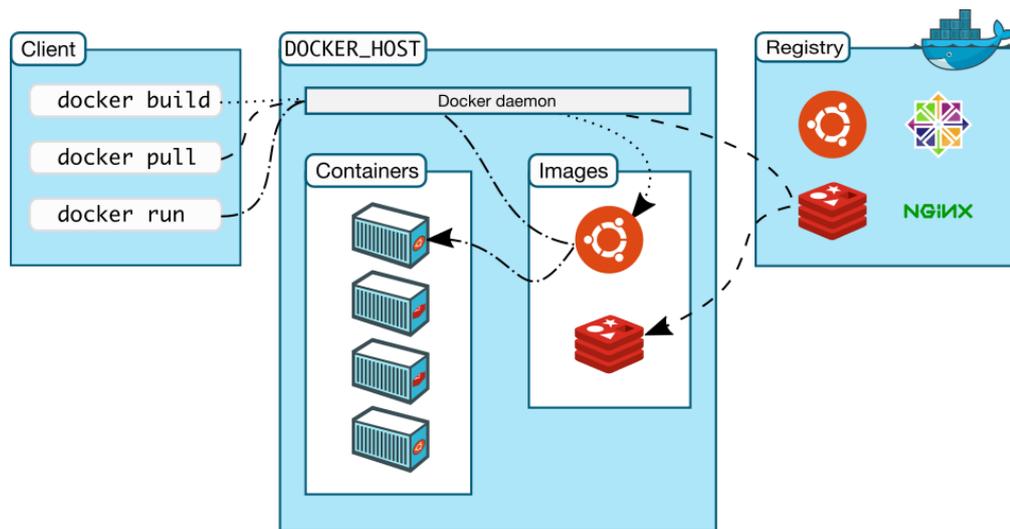


Figure 3.2: Docker architecture [16]

### 3.2.6 Flutter

*Flutter*[18] is a cross-platform UI<sup>13</sup> toolkit created by Google that is designed to allow code reuse across several OS<sup>14</sup> such as iOS, Android, macOS, Windows, Linux, and the web. Its goal is to give the possibility to developers to create high-performance apps on different platforms that seem fluid as native applications while sharing the largest possible portion of code. Flutter, during development, provides a very useful instrument: a stateful hot reload of changes that non required a full recompilation of the code.

*Dart*[19] is a client-optimized language developed by Google for rapid application development across any platform prioritizing both development and high-quality production experiences through a wide range of compilation targets like web, mobile and desktop. It is a type safe language: it checks that the value of the variables always match their static type and it has also sound null safety property: the variables can not have a null value except if you indicate that they can have. Dart offers a large set of core libraries and many useful supplementary packages provided by the Dart team but there is also the possibility to use others packages from third-party publishers and the vast community.

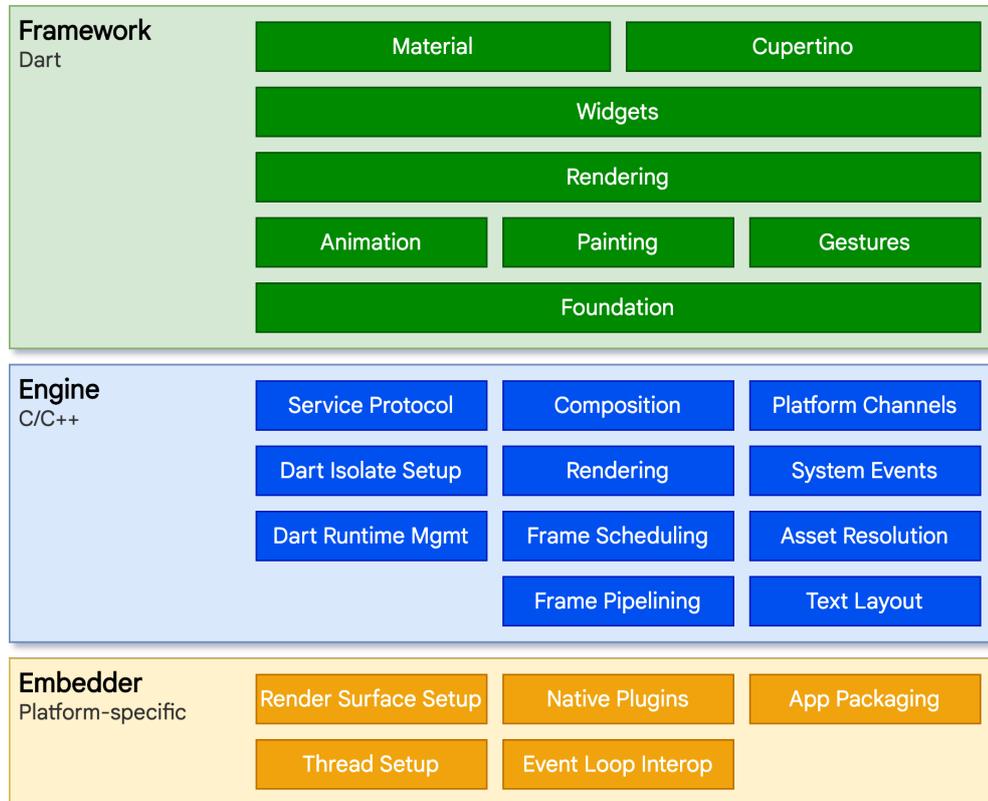
#### Architectural layers

Flutter is structured as a series of layers where each of them depends on the

<sup>13</sup>User Interface

<sup>14</sup>Operating System

underlying layer (Figure 3.3). Developers manage the interaction with Flutter



**Figure 3.3:** Flutter architecture [20]

through the usage of the *Flutter framework*: a modern and reactive framework written in Dart. Visualizing this architecture composition from the bottom to the top, we found:

- *Foundational* classes and basic services like *animation*, *painting* and *gestures*.
- *Rendering layer*: provides an abstraction to manage the layout.
- *Widgets layer*: the layer at which the reactive programming model is introduced.
- *Material* and *Cupertino* libraries allow the implementation of the Material or iOS design languages.

The core of Flutter is composed by the *Flutter engine*: written mainly in C++, it supports the primitives necessary to all Flutter applications. Flutter engine is exposed to the Flutter framework through the library `dart:ui`.

A platform-specific *Flutter embedder* gives an entrypoint and coordinates the access to services with the platform. Flutter applications are packaged as the native applications for the several platforms. The embedder is written in different languages depending for the different OS:

- Android: Java and C++.
- iOS and macOS: Objective-C/Objective-C++.
- Windows and Linux: C++.

### Reactive user interfaces

In Flutter the developers gives a mapping from the application state to the interface state, the framework updates the interface at runtime when the application state changes. This working logic consist in changing several traditional design principles and takes inspiration from React framework. Usually, the initial state of the interface is declared once and then is updated by the user code at runtime as a response to events. But, if the complexity of the application increases, the developers have to know how state changes in a consequently way over the entire user interface. To address this problem, Flutter split the UI from its state. The developers only create the user interface description and the framework manages one configuration to either create and update the UI.

### Widgets

Widgets are the constitutive elements of the UI of Flutter applications. They create a hierarchy based on composition: each of them nests inside its parent and can receive context from the parent. At the top of the structure there is the root widget: the container of the Flutter application. As a consequence of a user interaction, applications update their UI by telling the framework, for example, to replace a widget with another widget.

Flutter has its own implementation for each user interface control and does not utilize those given by the system. Flutter allows unlimited extensibility, the developers can create own variations of the widgets without the limitation of the OS. It also creates the entire scene at once avoiding major performance bottleneck and split application behavior from the operating system dependencies so that the applications seems the same on all versions of the OS.

Possible alternatives to Flutter in mobile app development are React Native, Cordova, Xamarin and Ionic.

*React Native*: a popular open-source JavaScript-based cross-platform mobile framework developed by Meta which allow to build true native apps for both Android and iOS platforms via a single codebase only. It has the same design as React and enables you to build an advanced mobile user interface with declarative components which are the same as in standard iOS and Android applications. React components encapsulate pre-existing native code and communicate with native APIs using JavaScript and the declarative UI paradigm of React. The fast refresh feature of React Native, thanks to the power of JavaScript, makes possible to see changes as soon as you save. Both Flutter and React Native are top cross-platform mobile application development frameworks, and they share many features. The primary distinction, however, is between the two programming languages: Flutter utilizes Dart while React Native uses JavaScript.

*Cordova*: a useful application development framework developed by Apache which allows to build hybrid applications using CSS<sup>15</sup>, HTML and JavaScript. Hybrid applications means that they are neither completely web-based nor native. These applications are packaged as mobile apps with the possibility to use the native devices' APIs. Cordova is useful to mix native application components with a WebView (special browser window) or to create a plugin interface between native and WebView components. Cordova is a mature cross-platform framework, it is easy to learn and useful for quick prototyping and creating simple apps but it does not provides very high performance and the UI does not look as native. Instead, Flutter can be used for any cross-platforms with higher performance, especially if there are animations and graphics.

*Xamarin*: a open-source C#-based framework provided by Microsoft with the aim to build cross-platform applications for iOS, Android, and Windows with .NET. The use of C# language to develop apps distinguishes Xamarin from the other frameworks. Its applications can be authored on a PC or a Mac and then compiled into native app packages. Developers can write the code in a single language and obtain applications with performance, appearance and feel as the native ones. Xamarin is based on .NET which manages some jobs like memory allocation, garbage collection and interoperability with underlying platforms. The two important differences between Xamarin and Flutter are the programming languages utilized and how the UI is rendered. The user interface in Xamarin is created using XAML with C# support while in Flutter, Dart language is used to handle both logic and the full user interface.

*Ionic*: a open-source user interface toolkit created for easier cross-platform application development with high-performance and high-quality using HTML, CSS

---

<sup>15</sup>Cascading Style Sheets

and JavaScript. It is a front-end SDK<sup>16</sup> framework that permits to develop mobile applications for iOS, Android and Windows, developers can create once and run everywhere. It has integration with popular frameworks like Angular, React and Vue. Ionic has the focus on the user interface: controls, interactions, gestures and animations. It implements techniques like hardware accelerated transitions and touch-optimized gestures to reach great performance on the latest mobile devices. Both Flutter and Ionic are great mobile frameworks, Ionic is very good for web developers that want to build hybrid mobile apps while Flutter guarantee higher performance results.

---

<sup>16</sup>Software Development Kit

# Chapter 4

## Architecture

To organize the technologies used in the applications, an appropriated architecture which permits them to communicate to each other in a proper way is needed. The choice of the architecture is crucial for guarantee the best interaction between the various technologies adopted. Especially because the back-end has to work with different kinds of front-end which operate in a different way, in our case the architecture has to support a web application and a mobile application.

A detailed discussion about the implementation of the back-end of the application is offered in this chapter. Specifically, the topics covered are: the data models used, the asynchronous tasks for retrieve all the financial data need by the application and the APIs provided for the clients.

### 4.1 Client-server model

In the client-server architecture (Figure 4.1), the server provides services to one or many clients. The clients can be of different type (web, mobile, desktop) and can communicate with the same server which responds at the requests done by the clients through specific network protocols like HTTP<sup>1</sup>. This architecture is very common and recommended thanks to its efficient operation through the use of the internet.

The REST API guidelines are followed to make the operations easier: each resource is identified with a unique URL<sup>2</sup> and the HTTP request method has to be specified to indicate the type of operation that you want to do on the resource. The four most utilize HTTP methods are:

---

<sup>1</sup>HyperText Transfer Protocol

<sup>2</sup>Uniform Resource Locator

- **GET**: used to read or retrieve a resource.
- **POST**: used to create a new resource.
- **PUT**: used to modify a resource.
- **DELETE**: used to delete a resource.

Docker containers are used for the development of the server side, they are very useful because they carry all their dependencies with them. Due to this type of architecture two different types of clients have been developed which works with the same single server: the web application and the mobile application.

The web application (Figure 4.2) was developed using React framework, it permits the users to interact with the data and charts related to markets and user strategies. In addition of these operations, the users through the web application can also create, modify and delete the strategies.

In particular, this work has consisted in the development of the mobile application and the next chapter is entirely dedicated to explore it (Chapter 5).

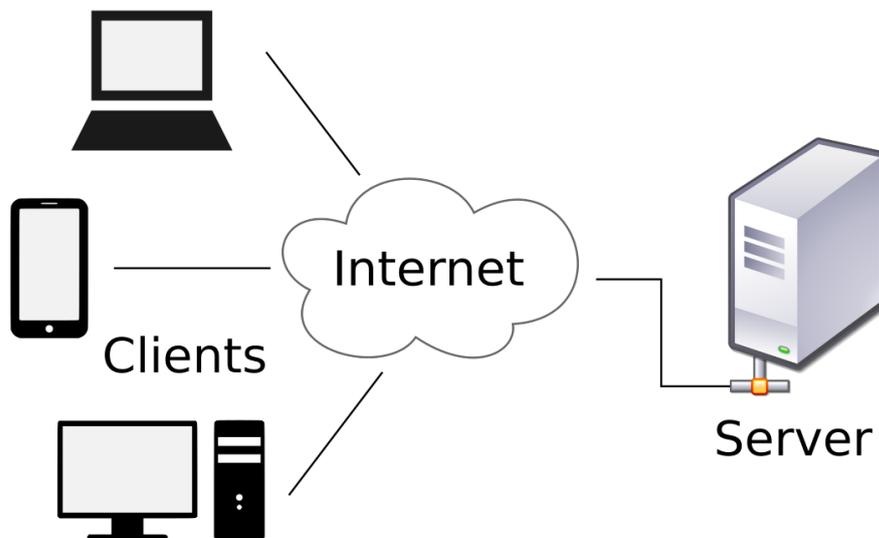


Figure 4.1: Client-server model [21]

## 4.2 Database

To take advantage of the flexibility of the non-relational databases of MongoDB, the information about the users and the financial data are stored in a collections



**Figure 4.2:** Web application - Markets page

through the use of JSON dictionaries. A detailed description of the collections used is provided below.

### User

Collection which contains all information about the users of the application. There are both the information to identify the users and some information about the operations of the users. The structure of the documents is shown in the Table 4.1.

Fields	Description	Type
id	Unique identifier of the user	Number
email/username	Unique identifier for the authentication	String
password	Password for the authentication	String
first_name	Name of the user	String
last_name	Surname of the user	String
is_superuser	Authorization to be superuser	Boolean
is_active	Enabled to operate with the application	Boolean
last_login	Last time the user logged in the application	Date
date_joined	Date in which the user joined the application	Date

**Table 4.1:** User document description

### Market

All the markets' data and the relationships with options and futures are included in this collection, the structure of the documents in this collection is shown in the

Table 4.2. The field *expiration* of the documents is an array of documents with a structure that can be seen in the Table 4.3.

Fields	Description	Type
id	Identifier of the market	String
groupId	Identifier of market group	String
symbol	Symbol of the market	String
label	Name of the market	String
exchange	Symbol of the exchange to which the market belongs	String
country	Symbol of the country to which the market belongs	String
currency	Currency of the market	String
template	Type of the market	String
exposition	Exposition value for related financial options	Number
dividendYield	Dividend yield value of the market	Number
expirations	Array of the expiration documents	Array
underlying	Information about the underlying of the market	Object

**Table 4.2:** Market document description

Fields	Description	Type
symbol	Symbol of the expiration	String
label	Label of the expiration	String
dates	Array of the dates	Array

**Table 4.3:** Market expiration document description

### Chain

Collection that provides all relevant data on the chains of options, each chain refers to the same market and expiration date (Table 4.4). The chains are identified by combining the fields: *symbol*, *expiration* and *date*, for instance the values can be: DAX, EOM and 2022-12-16. Each chain document contains an array of objects with all information about call and put options related to the same strike price (Table 4.5). Each single option is identified by its *contract* field which is formed by combining: exchange symbol, market symbol, expiration symbol, expiration date, option type and strike price (Table 4.6).

Fields	Description	Type
exchange	Symbol of the exchange	String
symbol	Symbol of the market	String
expiration	Symbol of the expiration	String
date	Date of the expiration	Date
options	Array of options	Array

**Table 4.4:** Chain document description

Fields	Description	Type
strike	Value of the strike price	Number
put	Option object of type <i>put</i>	Object
call	Option object of type <i>call</i>	Object

**Table 4.5:** Strike document description

## Future

All futures-related data is included in this collection, each document refers to a specific market and expiration date. To identify the futures the field *contract* is used, it is composed by: exchange symbol, market symbol, expiration symbol and expiration date, for instance these values can be equal to: EUREX, DAX, EOM and 2022-12-16 To maintain the same format as the options contract it is added the values of the *type* and the *strike* in the field *contract*, but these values are always equal to FUTURE and 0. This structure of the documents can be seen in the Table 4.7.

## Group

Collection of data containing every markets groups information, the groups consist of markets with a common reference. For instance, the markets E-mini Russell 2000 and Micro E-mini Russell 2000 refer to the same Russell 2000 index. This organizations of the markets is useful for users that want to operate with markets of the same group. In the Table 4.8 the structure of the group documents is shown.

## Strategy

Collection that is filled with the details regarding users' strategies. Users can create more strategies and each of them is related with a specific group of markets. Within the strategy document (Table 4.9) there is the list of market positions, each market

<b>Fields</b>	<b>Description</b>	<b>Type</b>
price	Price of the option	Number
last	Last price of the option	Number
open	Open price of the option	Number
close	Close price of the option	Number
settle	Settlement price of the option	Number
low	Low price of the option	Number
high	High price of the option	Number
volume	Volume value of the option	Number
openInterest	Open interest value of the option	Number
type	Type of the option	String
state	State of the option	String
contract	Contract identifier of the option	String

**Table 4.6:** Option document description

<b>Fields</b>	<b>Description</b>	<b>Type</b>
exchange	Symbol of the exchange	String
symbol	Symbol of the market	String
expiration	Symbol of the expiration	String
date	Expiration date	Date
price	Price of the future	Number
last	Last price of the future	Number
open	Open price of the future	Number
close	Close price of the future	Number
settle	Settlement price of the future	Number
low	Low price of the future	Number
high	High price of the future	Number
volume	Volume value of the future	Number
openInterest	Open interest value of the future	Number
type	Type of the future	String
contract	Contract identifier of the future	String

**Table 4.7:** Future document description

Fields	Description	Type
type	Type of the group of markets	String
symbol	Symbol of the group of markets	String
name	Name of the group of markets	String
currency	Currency symbol of the group of markets	String

**Table 4.8:** Group document description

position (Table 4.10) is related to a specific option or future, it is indicated if the position has been opened or closed and it is also specified the amount of contracts opened and in which state it is, it can be open, close or temporary. Temporary state means that the position has no effect on portfolio of the user but permits to see the impact of the position on the performance of the selected strategy. The position with a temporary status can be opened, disabled or deleted in any moment. An open position has effect on portfolio and the only operation that can be done is the closing.

The market positions are identified by the field *contract* that is composed by: exchange symbol, market symbol, expiration symbol, expiration date, option type and strike price, and timestamp of the opening position. For instance, the values can be: EUREX, DAX, EOM, 20221216, CALL, 0015250000 and 1630622983463. This data structure allows to open several positions in different moments in the same contract. The field *whatif* is an object that contains values of parameters which can be used to simulate the performance of the strategy in a certain scenario.

Fields	Description	Type
userId	Identifier of the user	String
groupId	symbol of the group of markets	String
name	Name of the strategy	String
positions	Array of position objects	Array
created	Creation date	Date
disabled	Disabled from the view in portfolio	Boolean
closed	No more open positions	Boolean
whatif	Object with the information about the simulation	Object

**Table 4.9:** Strategy document description

<b>Fields</b>	<b>Description</b>	<b>Type</b>
id	Identifier of the position	String
contract	Contract identifier of the option or future	String
active	If the position is active	Boolean
status	Status of the position	String
quantity	Number of contract bought or sold	Number
exchange	Symbol of the exchange	String
symbol	Symbol of the market	String
expiration	Symbol of the expiration	String
date	Date of the expiration	Date
type	Type of the contract	String
strike	Strike price	Number
price	Current price of the contract	Number
whatif	Object with the information about the simulation	Object
startDate	Date of position opening	Number
startPrice	Price on the position opening date	Date
endDate	Date of position closing	Date
endPrice	Price on the position closing date	Number

**Table 4.10:** Position document description

### Portfolio

All the data regarding each user's portfolio is contained in this collection, the portfolio is a virtual account that starts with an initial value of €100,00. The balance of the portfolio is computed using the performance of the strategies of the user. The users can have only one portfolio and the structure of the document is visible in the Table 4.11.

### Market History

Collection that includes all available market pricing data per each day relating to the last two years (Table 4.12). These data are saved in an array where each object corresponds to a different day.

### Chain History

Collection that contains all relevant data on the chains of options per each day relating to the last month (Table 4.13). The documents are identified by the fields:

<b>Fields</b>	<b>Description</b>	<b>Type</b>
userId	Username of the user	String
name	Name of the portfolio	String
value	Total balance of the portfolio	Number
currency	Currency used for the portfolio	String
created	Date of creation	Date
strategies	Array of strategies included in the portfolio	Array

**Table 4.11:** Portfolio document description

<b>Fields</b>	<b>Description</b>	<b>Type</b>
exchange	Symbol of the exchange	String
symbol	Symbol of the market	String
days	Array of objects that refers to a specific date	Array

**Table 4.12:** Market history document description

*exchange*, *symbol* and *date*, and contains an array where each object corresponds to a different day.

Fields	Description	Type
exchange	Symbol of the exchange	String
symbol	Symbol of the market	String
expiration	Symbol of the expiration	String
date	Expiration date	Date
days	Array of objects that refers to a specific date	Array

**Table 4.13:** Chain history document description

### 4.3 Asynchronous tasks

To retrieve the information about the financial markets, several asynchronous tasks are used. In particular, there is one asynchronous task for each different selected exchange. In this way each of them fetch the daily information about the markets that are part of that specific exchange every 15 minutes. The information retrieved concerned: prices, volumes, open interests, futures and chains of options; and are saved in the database. At the end of the day one asynchronous task is used to aggregate the latest information in unique documents divided per each market, futures and chains of options.

#### **update\_cboe**

The asynchronous task *update\_cboe* permits to retrieve the financial data about the CBOE Exchange. This task extract the list of all the markets that are contained in CBOE and calls for each of them the related public API. The markets information collected is about the prices, futures and options with all the available expiration dates. After the processing of the information, the data is stored in the database.

#### **update\_cme**

The CME Exchange's financial information may be retrieved thanks to the asynchronous task *update\_cme*. This task call four different APIs for each market contained in CME and are used to download the information about: options prices, futures prices, volumes and open interests. After the processing of the information, the data is stored in the database.

#### **update\_eurex**

Financial data about the EUREX Exchange are retrieved with the help of the asynchronous task *update\_eurex*. Unlike the other two exchanges, EUREX does not provided any REST API, so the method used as alternative is the web scraping. Starting from the website it is possible to recognize two different path for retrieving

information about futures and options, this pattern is repeated for each market and each available expiration date. After the processing of the HTML files, the data is stored in the database.

### **update\_history**

The *update\_history*, is the asynchronous task which talks with the database at the end of the day to create collections composed by the updated data collected by market and option chains.

## **4.4 REST APIs**

The user, through the client application, can perform various actions with the financial data retrieved from the exchanges. These actions are made possible by specifically created web services. These web services follow the REST guidelines and each of them is characterized by a unique URL and a specific HTTP method. The APIs are divided in three different groups: *user*, *market* and *chain*, and are described below.

### **4.4.1 User**

Before seeing the APIs concerning the users, the explanation of the authentication phase is provided.

#### **Authentication**

To allow the authentication of the user in the mobile client application, the *TokenAuthentication*[22] provided by Django REST framework is used. It makes use of a straightforward token-based HTTP authentication method. For client-server architectures, including native desktop and mobile clients, token authentication is an appropriate choice. The authentication phase is composed by the following steps:

- 1 - The user indicates its username and password in the login page, creating an HTTP request to the server.
- 2 - The server search in the database the username received and using hashing techniques checks if the password corresponds.
- 3 - If the login data is not correct the server responses to the client with a denied permission through an HTTP 401 Unauthorized error. Instead, if the login data is correct the server will return a JSON response which has the key *token* and a corresponding value that is the unique token associated to that user.

- 4 - For all the following request of the user, the token will be automatically send and will permits the server to identify the user.

Knowing the functioning of the authentication, we are going to see the APIs concern the information about the user and all the other things that are related to him like his portfolio and his strategies.

#### **api/users/**

Called with the HTTP GET method returns the information about the logged-in user. Django maintains the information of the user after the login thanks to a unique token and until the user does not logout or close the application, this information is available for the HTTP calls.

#### **api/users/portfolio/**

The information about the logged-in user's portfolio is returned when it is call with the HTTP GET method. The information regards the performance of the portfolio over the time and the details about the strategies of the user that composed the portfolio.

#### **api/users/strategies/**

The list of the user's strategies and some information about them are returned when the HTTP GET method is used to call it. Instead, called with the HTTP POST method, a new strategy can be created by the logged-in user. Before create the new strategy, the server checks if the *groupId* and *name* values passed through the body of the request are valid and if they are acceptable the new strategy is created.

#### **api/users/strategies/:id/**

The information about the user's single strategy that has the *id* parameter taken in input is returned when the HTTP GET method is used to call it. The response of the server contains all the data that are needed to show to the logged-in user in the client application like all the information about the positions opened in that strategy. Instead, called with the HTTP POST method allows to modify an existed strategy of logged-in user. It is possible to change the name of the strategy, change the *whatif* values and insert, modify or delete one or more positions of the strategy. The server checks if the option or future contracts for which a position is opened belongs to the same group of markets of the strategy.

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

The data required to build the strategy charts associated with the strategy for which the *id* was provided as a parameter is returned when called with the HTTP GET method. The parameter *chart-id* can assume the value *profit* or *greeks* and respectively returns the data for the payoff diagram and for the greeks charts. This information is contained in an array of JSON objects with the data per each different price that the market can assume.

**4.4.2 Market**

These APIs are related to market information, they are used to display the data like prices, volumes, open interest and the charts like historical volatility, open interest by maturity, price history in the client application.

**api/markets/**

Called with the HTTP GET method returns the list of the markets with the relative details per each of them.

**api/markets/:symbol/**

The information about the market matching to the *symbol* parameter provided is returned when the HTTP GET method is called. The information regards the market characteristics, market prices and expiration dates for options and futures.

**api/markets/:symbol/futures/**

The list of market-related futures matching to the *symbol* parameter entered as input for all maturities is returned when the HTTP GET method is called.

**api/markets/:symbol/:chart-id/**

The data required to build the market charts associated with the market matching to the *symbol* argument entered is returned when called with the HTTP GET method. The parameter *chart-id* can assume the values: *history*, *volatility* or *open-interest*. The first is a chart about the values assumed by the market during the past, the second is chart about the volatility per each option for the next two expiration chains and the third is a chart about the open interests per each chain expiration. This data is returned in an array of JSON objects with the data per each different price that the market can assume.

**4.4.3 Chain**

These APIs relate to the chains' information and contain the data to show like the list of options and the data needed to create the charts like the open interest for strikes, breakdowns or pressure in the client application.

**api/chains/:symbol/:expiration/:date/**

Called with the HTTP GET method returns the data of a single chain where the related market, the expiration type and the expiration date correspond to the *symbol*, *expiration* and *date* parameters taken in input. In the body of the response there is the data about the characteristics of the chain, the list of the options and the index which indicate the position of the ITM options within the list.

**api/chains/:symbol/:expiration/:date/:chart-id/**

The data required to construct the charts relating to the single chain are returned when the HTTP GET method is called where the related market, the expiration type and the expiration date correspond to the *symbol*, *expiration* and *date* parameters taken in input. The parameter *chart-id* can assume the values: *volatility/variation*, *open-interest* or *open-interest/cumulative*. The first is a chart about the volatility variation per each option contract for the last two days, the second is a chart about the open interests and the third is a chart about the variation of the last two days of the open interests. This data is returned in an array of JSON objects with the data per each different price that the market can assume.

# Chapter 5

## Mobile application

In this chapter, we are going to explore the data management and the UI of the mobile app with the various available functionalities from a technical point of view.

### 5.1 Data management

The data management of the app is based on several Flutter libraries and different data structure. The first step that is required to the user is the login in the application to permits the server to authenticate him, allowing to visualize certain data and perform different operations. When the application is started, the first page that appears is the login page as shown in Figure 5.1, where the user has to insert his email and password and then pressing the *Sign in* button he can send his data to the server through an HTTP POST request. This operation is done thanks to the *http*[23] library, developed by the official Dart Team, which permits to make HTTP requests.

When the server receives the data, it takes the email and password and find a match in the document of users saved in the database, the passwords of the users in the document are encrypted and so an encryption technique is used to compare them. If it finds a match, returns to the client the authentication token corresponded to the user. The app through another library called *flutter\_secure\_storage*[24], stores the token in a secure storage and will use it for all the following HTTP requests by passing it in the header *Authorization*. In this way the user will be allowed to navigate in the app and he is recognized in order to provide the data belonging to him. After the login, the email of the user is stored using the *shared\_preferences*[25] library, by doing so the email is available for all the files of the app and they can retrieve this value using the same library. In practice, the implemented class of the navigation drawer menu, which consist of a *Drawer* widget of Flutter, read the email of the user and display it in the welcome message (Figure 5.2).

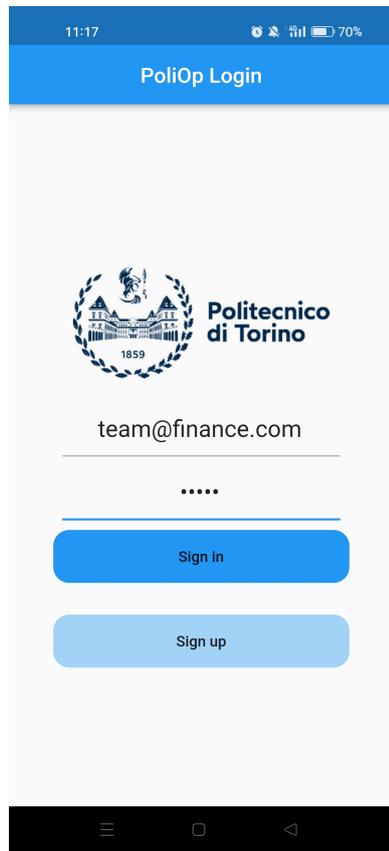


Figure 5.1: Login page

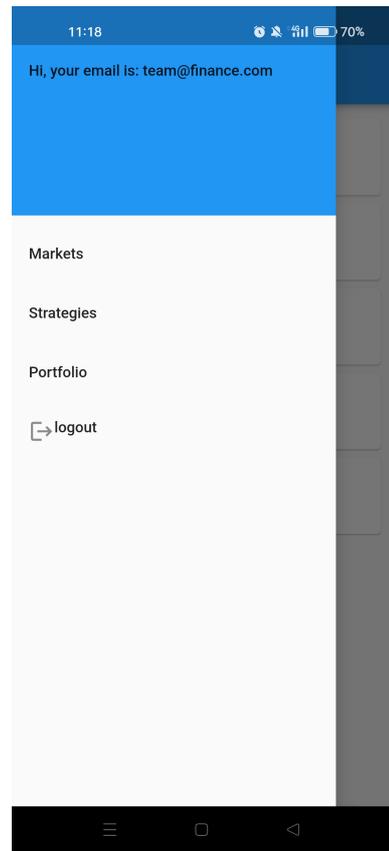


Figure 5.2: Navigation drawer

To manage better the HTTP requests, the DAO (Data Access Object) pattern is implemented, a pattern that provides an abstract interface to the database. Essentially, there is a file which contains as many functions as there are different requests, each of them corresponds with one of the APIs presented in Chapter 4.4. All these functions have *Future* return type, it means that they return the result of an asynchronous computation. An asynchronous computation may need to wait for retrieving data which takes time. Instead of blocking all computations until the result is available, the asynchronous computation immediately returns a *Future* which will eventually "complete" with the result. The data retrieved from the APIs are saved in structures of types *List*, an indexable collection of objects with a length, and *Map*, a collection of key/value pairs, from which you retrieve a value using its associated key. All the classes of the different pages of the application import the DAO file and call only the functions that they need. The pages which compute asynchronous tasks are *StatefulWidget*, a widget that has mutable state,

and implement the *initState()* and *setState()* functions, that are used to initialize and notify the framework that the internal state of the object has changed. In this way the user interface is rebuild when all the data is retrieved from the APIs. In the Figure 5.3, one example of this mechanism is displayed.

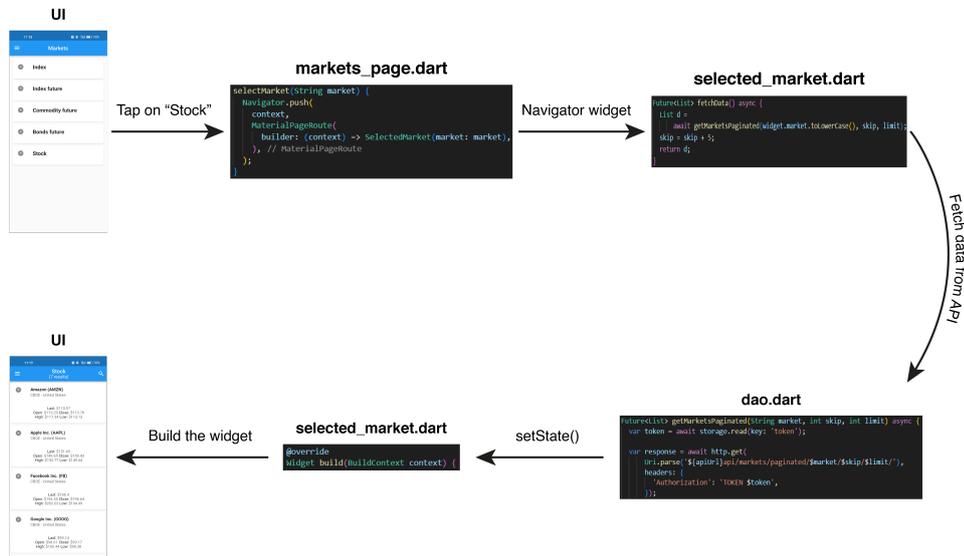


Figure 5.3: Example of how the app works

## 5.2 User interface

The user interface is one of the most important elements of a mobile application. In financial mobile applications, like in this case, the data and charts to show are a lot and complicated. A detailed explanations of the different parts of the UI is provided below.

The app maintains a common style within the different pages, for all of them the Flutter widget parent is always a *Scaffold*. It will expand to fill the available space that usually means that it will occupy its entire window or device screen. This widget allows to insert an app bar to display at the top of it, so the *AppBar* widget of Flutter is used to implement the app bar, it is always shown and change the title based on the page. The *AppBar* widget allows also to implement some actions, a list of widgets to display in a row after the title of the widget, these widgets are *IconButton* and representing different operations and change based on the page. One element which is always shown is the icon of the drawer menu, located on the left of the app bar. The drawer used in our *Scaffold* is a customize

*Drawer* widget and as can be seen in the previously shown Figure 5.2, it allows the user to navigate through the various sections of the app or to do the logout, from each page of the application.

The navigation between the pages is managed with the Flutter widget *Navigator*, it permits to move from one page to another. These pages, in reality, are customized Flutter widgets which inside contain other widgets. This type of navigation allows to send data from the starting page to the arrival page and it is very useful.

### 5.2.1 Markets

The first page that appears after the login is the *Markets* page (Figure 5.4), a simple page composed by a *Column* widget which allows to insert more widgets one under the other. In particular, there is a list of widget of type *Card*, each of them indicate one of the different markets available. When one of these cards is tapped, the *Navigator* widget will send the user to the *Market* page (Figure 5.5).

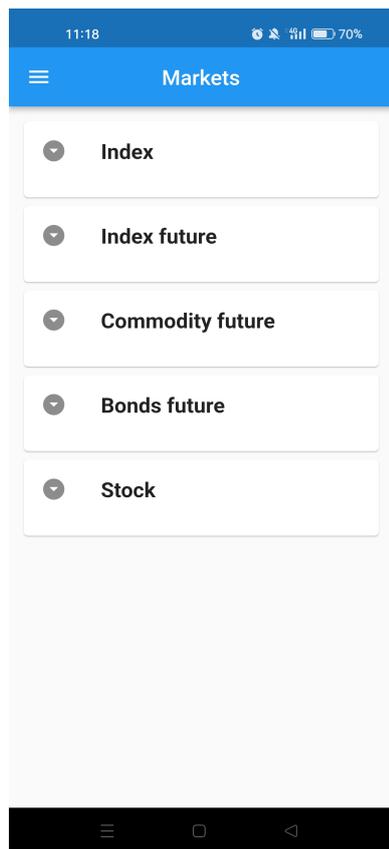


Figure 5.4: Markets page

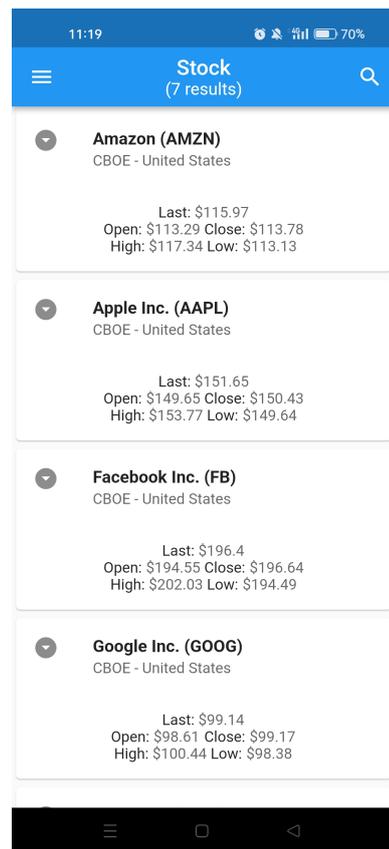


Figure 5.5: Stocks page

This new page, receive from the *Navigator* the type of the chosen market by the user and consequently it demands to the server the corresponding data. It is similar to the previous page but, the cards contain more information, in the app bar is indicated the number of results and there is a new *IconButton* in the actions of the app bar. This *IconButton* allows the user to search the items by the name, this is possible thanks to the *app\_bar\_with\_search\_switch*[26] library which modifies the standard *AppBar* implementing the search function. In addition, a pagination technique is here implemented, thanks to a dedicated API the results are gradually retrieved while the user scroll down.

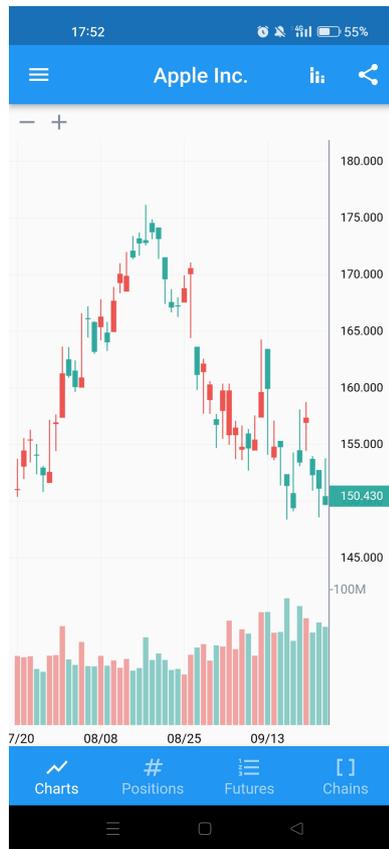


Figure 5.6: Price history chart

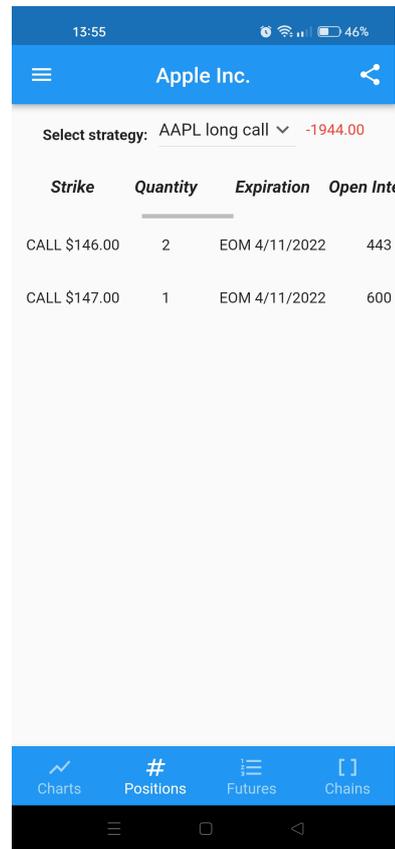


Figure 5.7: Positions page

### Market page

In the same way as before, when one card of the page is tapped, the *Navigator* widget send the user to another page and pass to it the data relative to the item selected. This page is more complicated compared to the previous, so to navigate internally is added a bottom navigation bar using the *BottomNavigationBar* widget.

There are four different tabs, all of them implement the share button, located on the right of the app bar that allows the user to share the content of the page with other applications. The library used to make this operation is called *share\_plus*[27] and it is implemented in different way according to the type of content to share. In the case of charts, it do a screenshot of the page but removing the app bar and the bottom navigation bar and save it as image. Instead, for the other tabs, where the content is textual and contained in tables, the share button permits to take the values in the tables and share it in a CSV<sup>1</sup> file.

Expiration	Price	Open Price	Close
EOM 31/7/2022	\$26.90	\$26.70	\$26.70
EOM 31/8/2022	\$21.15	\$21.20	\$21.20
EOM 30/9/2022	\$27.40	\$27.75	\$27.75
EOM 31/10/2022	\$31.00	\$32.85	\$31.00
EOM 30/11/2022	\$30.80	\$32.05	\$30.80
EOM 31/12/2022	\$30.15	\$30.65	\$30.15
EOM 31/1/2023	\$30.55	\$31.10	\$30.55
EOM 28/2/2023	\$30.10	\$30.50	\$30.10
EOM 31/3/2023	\$29.45	\$29.40	\$29.45
EOM 30/4/2023	\$29.20	\$0.00	\$0.00
EOM 31/5/2023	\$29.35	\$0.00	\$0.00

Figure 5.8: Futures page

Strike Price	Volume	Open Interest	Bid-As-Spread
\$143.00	76	0	\$0.20
\$144.00	1	13	\$0.20
\$145.00	238	199	\$0.20
\$146.00	110	244	\$0.20
\$147.00	104	407	\$0.15
\$148.00	112	403	\$0.20
\$149.00	173	359	\$0.20
\$150.00	288	738	\$0.20
\$152.50	243	1502	\$0.25
\$155.00	107	1105	\$0.15

Figure 5.9: Chains page

In the *Charts* tab (Figure 5.6), different interactive graphs are shown, beside the share button there is a button which allows to change the chart visualized by choosing from a dropdown menu. The graphs are implemented using the *candlesticks*[28] and *syncfusion\_flutter\_charts*[29] libraries.

<sup>1</sup>Comma-Separated Values

The *Positions* tab (Figure 5.7) is composed by a row where the user can select different strategies through a dropdown menu implemented with the *DropDownButton* widget and a table below the row. This table is created using the *flutter\_expandable\_table*[30] library because it give the possibility to maintain the first column fixed on the left and scroll only on the other columns.

Similarly, in *Futures* tab (Figure 5.8), there is a table created with the same library as before in a way that the first column can be fixed on the left.

The *Chains* tab (Figure 5.9), in addition to contains the table created with the same library as before, includes also a row above the table with a dropdown menu implemented with the *DropDownButton* widget and a *ToggleButtons* widget composed by two buttons, enabling one button disables the other automatically.

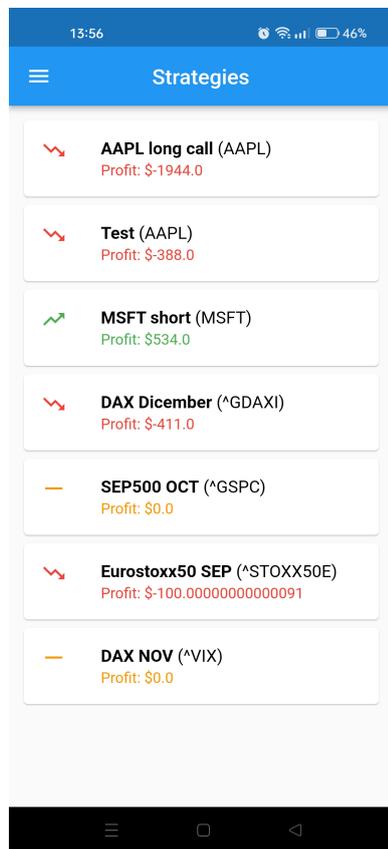


Figure 5.10: Strategies page

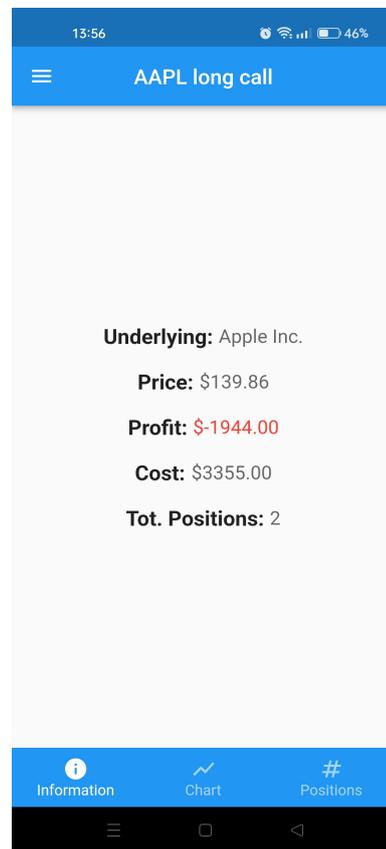


Figure 5.11: Strategy information

## 5.2.2 Strategies

The structure of the *Strategies* page is similar to that of the *Markets* page (Figure 5.10), a *Column* widget with inside many *Card* widgets. When a card is tapped, the *Navigator* widget sends the user to the *Strategy* page passing the data of the selected card.

### Strategy page

This page implements the bottom navigation bar using the *BottomNavigationBar* widget. The *Information* tab (Figure 5.11) is made up only by *Text* widgets. Instead, the *Chart* tab (Figure 5.12) contains a interactive graph created with the *syncfusion\_flutter\_charts* library and implement the share button using the *share\_plus* library. Finally, the *Positions* tab contains the same table shown in the *Positions* tab of *Market* page (Figure 5.7).

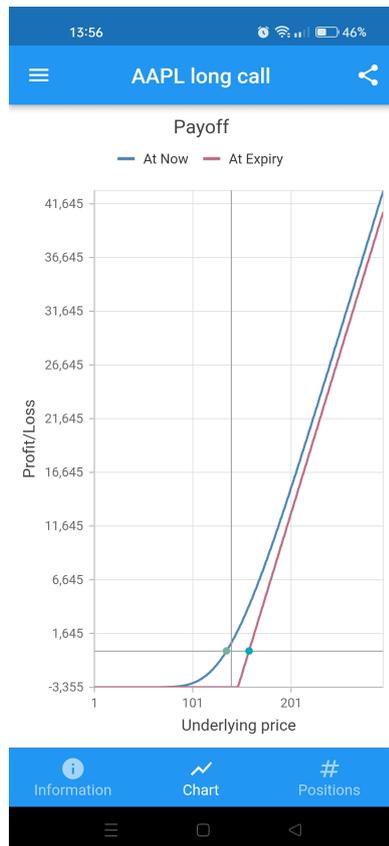


Figure 5.12: Strategy chart

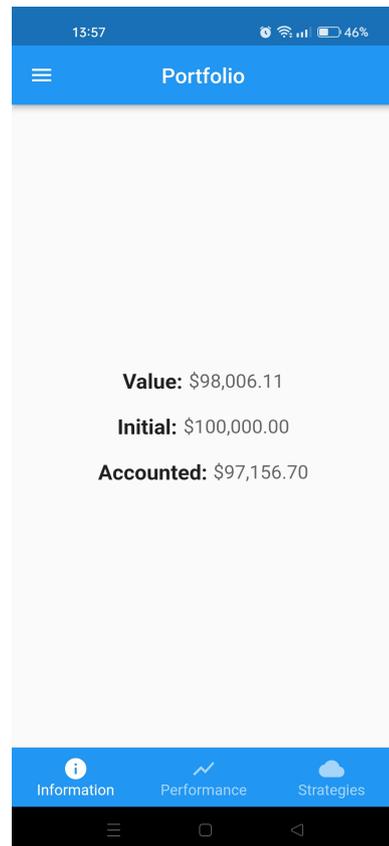


Figure 5.13: Portfolio information

### 5.2.3 Portfolio

The structure of the *Portfolio* page is similar to that of the *Strategy* page, it includes the bottom navigation bar using the *BottomNavigationBar* widget. The *Information* tab (Figure 5.13) includes only *Text* widgets. The *Performance* tab (Figure 5.14), instead, contains an interactive graph constructed using the *syncfusion\_flutter\_charts* library. Lastly, the *Strategies* tab (Figure 5.15) contains a table implement through the *flutter\_expandable\_table* library. In this way, the first column is maintained fixed and the other scroll normally. The last column contain *IconButton* widgets, the tap on them triggers an API call and change the icon displayed.

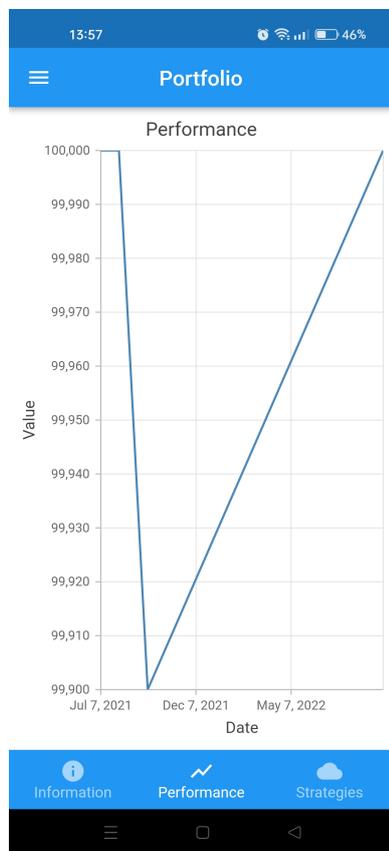


Figure 5.14: Portfolio performance

The screenshot shows the 'Strategies' tab of the Portfolio page. It displays a table with columns for 'Name', 'Accounted Revenue', 'Profit', and an 'IconButton' for each row. The bottom navigation bar shows 'Information', 'Performance', and 'Strategies' (selected).

Name	Accounted Revenue	Profit	IconButton
AAPL long call	\$0.00	\$-1944.00	Eye icon
DAX December	\$0.00	\$-411.00	Eye icon
DAX NOV	\$0.00	\$0.00	Eye with slash icon
Eurostoxx50 SEP	\$1750.00	\$-100.00	Eye icon
MSFT short	\$0.00	\$534.00	Eye icon
SEP500 OCT	\$0.00	\$0.00	Eye with slash icon
Test	\$0.00	\$-388.00	Eye icon

Figure 5.15: Portfolio strategies

# Chapter 6

## Use cases

Three use cases are provided in this chapter, they includes the most common operations which the user can perform and are explained from a functional point of view.

### 6.1 Analysis of a position

In this first use case, we are going to do an analysis of a position in relation to the market of the underlying. In the meantime, we will discover the content of the *Markets* and *Market* pages, and the available features provided.

#### Markets page

The markets page of the application shows the list of the available markets (Figure 5.4). By pressing one of them is possible to see the items that belong to that specific market, for instance, pressing on the *Stock* market, will appear the list of stocks available in the app (Figure 5.5). Under the name of the market it is displayed the number of items that belong to it. This list of items is sorted alphabetically by name and does not only provide the names but also other important information like the exchange, the symbol and the most recent details about the price (last, open, close, low and high). To avoid the overloading of the application and wasting the time of the user, only a few results are initially downloaded, and when the user scroll down within the list the other results are gradually retrieved. In addition of this pagination technique, the user can directly search an item using the search button located on the top right of the page. Selecting an item, the user can visualize all the data and charts that concern that specific item, for example, if the user select the *Apple Inc.* stock, a new *Market* page will appear.

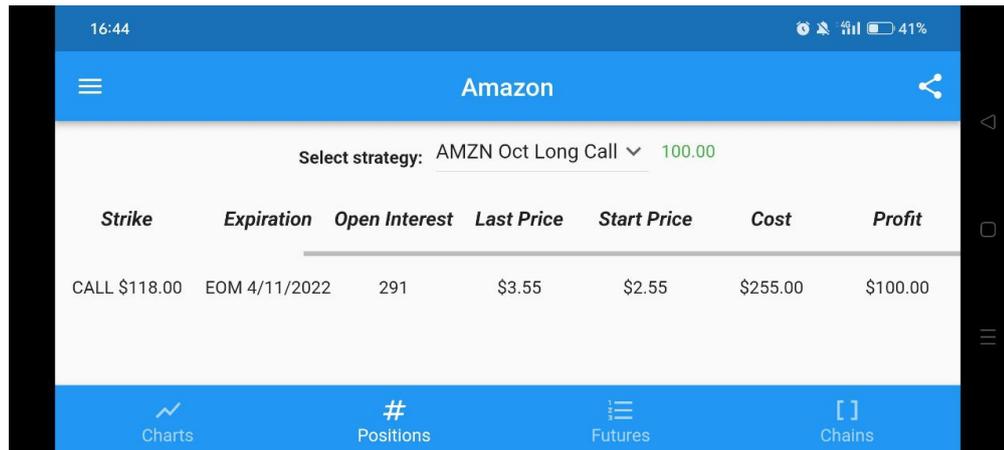
## Market page

This new page has an internal tab menu which allows the user to internally navigate, the tab menu is positioned in the bottom of the page and has four tabs: Charts, Positions, Futures and Chains.

- *Charts*: this tab allows to visualize and interact with several charts that shown different financial data. Each chart gives the possibility to zoom in and out, scroll in horizontal and vertical way, and pressing on the screen the information about that specific point will appear. To change the visualized chart the button with three bars in the top-right of the screen has to be pressed. Instead, pressing the button beside, it is possible to share the chart as an image through the social applications installed in the smartphone like WhatsApp, Telegram, Facebook and Twitter. For example the Figures 5.6 and shown the price history chart and what happens when the user press in one point of the screen.
- *Positions*: in this second tab, the user can see his strategies on the selected market (Figure 5.7). He can choose which to see through a drop-down menu on the top of the page. At the right of the name of the strategy there is the profit value of the strategy. In the center of the page there is a table where each row corresponds to a different open position of the strategy, the column of the strike price remain fixed on the left of the page while the other (quantity, expiration date, open interest, last price, start price, cost and profit) can be horizontally scrolled. The share button on the top right of the page permits to share the strategy as a CSV file through other applications installed on the smartphone like WhatsApp, Telegram and Gmail.
- *Futures*: as the name said, is the tab where there are all the available futures related to the selected market. The first column is relative to the expiration date and is blocked while on the other it is possible to horizontally scroll (quantity, price, open price, close price, open interest and volume). Also in this case, the button positioned in the top right of the page allows to share the futures as a CSV file with other applications. The Figure 5.8 shows this tab but of another market (*VSTOXX Mini Futures*) because the market *Apple Inc.* does not have any futures available.
- *Chains*: the tab of the options, in the upper part of the page the user can select the expiration date and the type of option (call or put) of the available options. The column of the strike price is locked on the left and the value is highlighted in yellow color if it is an ITM option, instead the other columns (volume, open interest, bid-ask spread, average price) can be horizontally scrolled. Also for the options it possible to share them as a CSV file using

the button on the top right of the page. The Figures 5.9 and show the chains page and what happens when the share button is pressed.

If the user want to analyze a position opened on the *Amazon* stock, he has to tap on the *Amazon* card in *Markets* page and then go the the *Positions* tab and see the available strategies. The *AMZN Oct Long Call* (Figure 6.1) adopts a long call strategy and has a temporary positive profit, this because the price of the opened positions has increased since it was bought.



**Figure 6.1:** AMZN Oct Long Call strategy

## 6.2 Exploration of a strategy

The second use case consist of explore a strategy to understand its performance, in the specific we want to visualize the performance of the strategy seen in the previous use case.

### Strategies page

In the strategies page of the application, the user can visualize the list of the strategies created by him. In addition of the name of the strategy, it is also indicated the symbol of group markets of which the strategy belongs and the profit (Figure 5.10). Selecting one strategy of the list, the user can see more information about it, for example, if the user select the *AAPL long call* strategy, a new strategy page will appear.

## Strategy page

The page just opened permits to navigate in three different tabs using a tab menu, the tab menu is positioned in the bottom of the page and has three tabs: Information, Chart and Positions.

- *Information*: in this first page, the general details about the strategy are provided (Figure 5.11): the underlying asset, the price, the profit, the cost and the number of positions.
- *Chart*: in the chart page there is the payoff diagram that represents the potential outcomes of the strategy. In particular there are two payoffs shown: at now and at expiry, the payoff at expiry shows what the profit will be on the contract expiration date while the payoff at now shows the trend of the profit in each instant. The chart is interactive and the Figure 5.12 shows it.
- *Positions*: this last page provides the information about the position opened in the strategy. The information is displayed using a table as in the positions tab of the market page seen previously.

Viewing the chart displayed in Figure 6.2 it is possible to understand the profit of the strategy depending on the variation of the price of the underlying. As can be seen the loss of the strategy is equal to the cost of the opened position until the underlying price does not reach the strike price. From this value the loss will begin to decrease until it becomes a profit. If the user wants to share this chart, he can tap the share button on the top right of the screen, the screen that will display is shown in Figure 6.3.

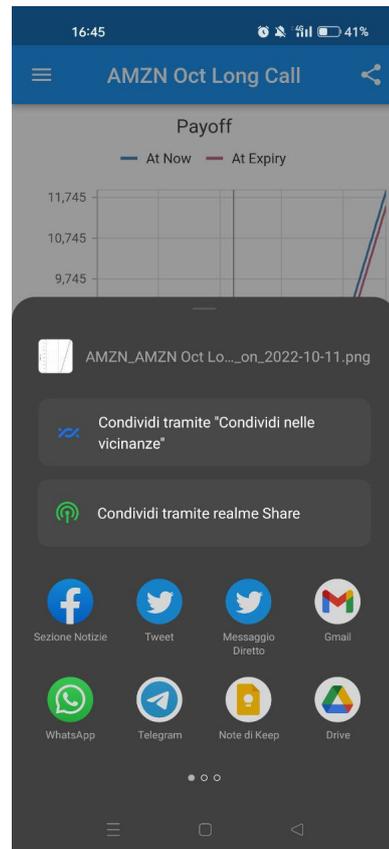
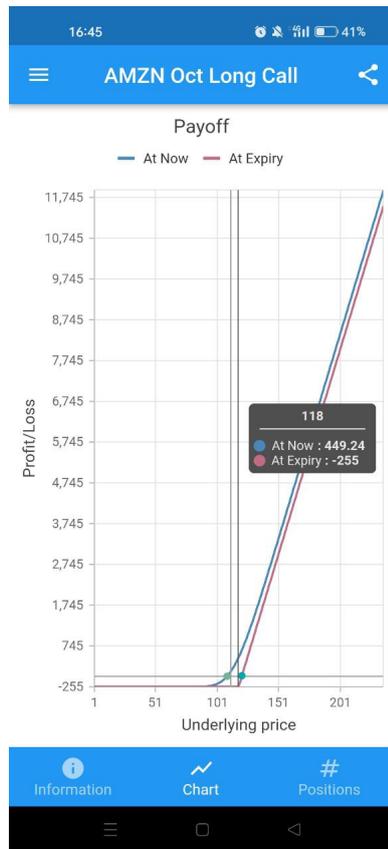
## 6.3 Exploration of the composition of portfolio

Explore the composition of the portfolio and visualize how the strategy seen in previous use cases will change the performance of the portfolio.

### Portfolio page

The portfolio page provides the information about the portfolio of the logged-in user, users have only one portfolio each and it is composed by the strategies owned by them. The user may navigate in three different tabs in this page: Information, Performance and Strategies.

- *Information*: in this page there are indicated the information about the balance of the portfolio of the user (Figure 5.13).



**Figure 6.2:** AMZN Oct Long Call pay-off diagram      **Figure 6.3:** AMZN Oct Long Call share function

- *Performance*: in the performance page the user can visualize the interactive chart where he can see the different values that the portfolio has assumed over the time (Figure 5.14).
- *Strategies*: this tab shows the list of all the strategies of the user that composed the portfolio. The information are provided in a scrollable table which shows the following details: name, creation date, markets group, number of positions, cost, possible revenue, accounted revenue and profit. The name column remains fixed on the left while the other columns can be horizontally scrolled. At the end of each row there is an icon button which permits the user to enable or disable that specific strategy in the visualization of the performance in the chart (Figure 5.15).

The user, using the buttons implemented in the last column of the table that contains all the strategies that are part of the portfolio, can enable and disable the

corresponded strategy in the computation of the performance of the portfolio. For instance, if the user enable the strategy *AMZN Oct Long Call*, the performance of the portfolio change in the following way, before the enabling it was as in the Figure 6.4 while after it is as in the Figure 6.5.

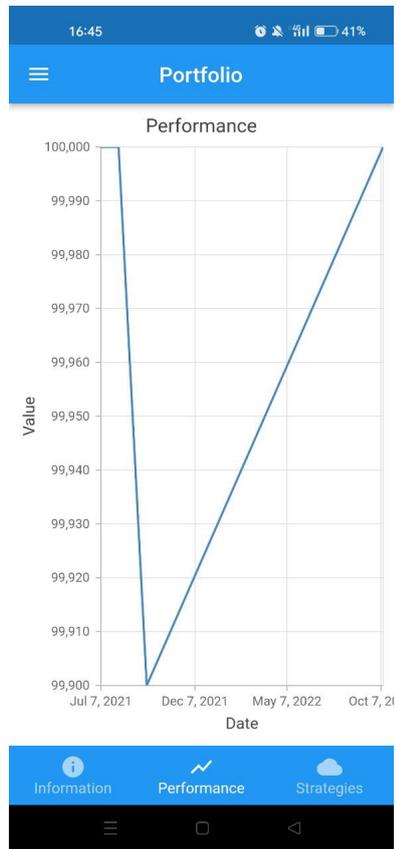


Figure 6.4: Portfolio performance

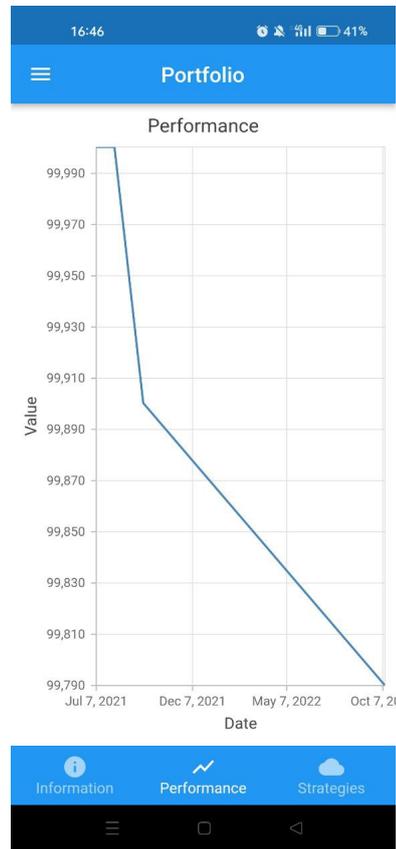


Figure 6.5: Portfolio performance

## Chapter 7

# Conclusions and future works

The mobile application created meets the requirements indicated by a professional trader. All the functionalities previously described work in a proper way through an user interface which allows the user to navigate within the application in a simple and intuitive manner. As said in the introduction, the mobile app has the main purpose of being a consultation tool, this is an important piece that will be added to the great project that previously consisted only in the web application. This allows the user to analyze and study data and statistics about options and futures at any moment. In addition, the share function implemented in the mobile app, which is available for market analysis data, charts and strategies, allows the user to share the information that he considers useful with other people through external social apps.

As mentioned before, this mobile app is part of a great project, the future works that can be developed are several and different. One possibility, for example, can be the implementation of an artificial intelligence system based on time series of historical financial data that suggests to the user some strategies potentially profitable. Another improvement which could be done regards the data retrieved from the exchanges. At this moment data is retrieved through the public APIs that provides it only 15 minutes, but in this way is not possible to perform intraday trading. The solution of this problem is to adopt APIs which provides real-time data for a fee. In parallel with this work, is being developed an extension of the web application. This extension consist in the classification of the strategies, they can be public or private and may be owned by a single user or a group. With this classification, the user has the possibility to browse within the web application to explore strategies of other users and groups. Subsequently, one future work could be the implementation of this extension also in the mobile app.

# Bibliography

- [1] Option Alpha. *Long Call*. URL: <https://optionalpha.com/strategies/long-call> (visited on 11/10/2022) (cit. on p. 10).
- [2] Option Alpha. *Short Call*. URL: <https://optionalpha.com/strategies/short-call> (visited on 11/10/2022) (cit. on p. 12).
- [3] Option Alpha. *Long Put*. URL: <https://optionalpha.com/strategies/long-put> (visited on 11/10/2022) (cit. on p. 14).
- [4] Option Alpha. *Short Put*. URL: <https://optionalpha.com/strategies/short-put> (visited on 11/10/2022) (cit. on p. 15).
- [5] Option Alpha. *Covered Call*. URL: <https://optionalpha.com/strategies/covered-call> (visited on 11/10/2022) (cit. on p. 17).
- [6] Option Alpha. *Covered Put*. URL: <https://optionalpha.com/strategies/covered-put> (visited on 11/10/2022) (cit. on p. 18).
- [7] Option Alpha. *Protective Put*. URL: <https://optionalpha.com/strategies/protective-put> (visited on 11/10/2022) (cit. on p. 20).
- [8] Django Software Foundation. *Django*. URL: <https://www.djangoproject.com/> (visited on 11/10/2022) (cit. on p. 24).
- [9] Python Software Foundation. *Python*. URL: <https://www.python.org/> (visited on 11/10/2022) (cit. on p. 24).
- [10] AskPython. *Django MVT Architecture*. URL: <https://www.askpython.com/django/django-mvt-architecture> (visited on 11/10/2022) (cit. on p. 25).
- [11] MongoDB Inc. *MongoDB*. URL: <https://www.mongodb.com/> (visited on 11/10/2022) (cit. on p. 26).
- [12] MongoDB Inc. *Pymongo*. URL: <https://pymongo.readthedocs.io/en/stable/> (visited on 11/10/2022) (cit. on p. 26).
- [13] Ask Solem and contributors. *Celery*. URL: <https://docs.celeryq.dev/> (visited on 11/10/2022) (cit. on p. 27).
- [14] Redis Ltd. *Redis*. URL: <https://redis.io/> (visited on 11/10/2022) (cit. on p. 28).

- [15] Docker Inc. *Docker*. URL: <https://www.docker.com/> (visited on 11/10/2022) (cit. on p. 28).
- [16] Docker Inc. *Docker overview*. URL: <https://docs.docker.com/get-started/overview/> (visited on 11/10/2022) (cit. on p. 29).
- [17] Inc. F5. *Nginx*. URL: <https://www.nginx.com/> (visited on 11/10/2022) (cit. on p. 28).
- [18] Google and community. *Flutter*. URL: <https://flutter.dev/> (visited on 11/10/2022) (cit. on p. 29).
- [19] Google. *Dart*. URL: <https://dart.dev/> (visited on 11/10/2022) (cit. on p. 29).
- [20] Google and community. *Flutter architectural overview*. URL: <https://docs.flutter.dev/resources/architectural-overview> (visited on 11/10/2022) (cit. on p. 30).
- [21] Wikipedia. *Client-server model*. URL: [https://en.wikipedia.org/wiki/Client-server\\_model](https://en.wikipedia.org/wiki/Client-server_model) (visited on 11/10/2022) (cit. on p. 35).
- [22] Django Rest Framework. *TokenAuthentication*. URL: <https://www.django-rest-framework.org/api-guide/authentication/#tokenauthentication> (visited on 11/10/2022) (cit. on p. 44).
- [23] Dart.dev. *Http*. URL: <https://pub.dev/packages/http/> (visited on 11/10/2022) (cit. on p. 48).
- [24] Steenbakker.dev. *Flutter\_secure\_storage*. URL: [https://pub.dev/packages/flutter\\_secure\\_storage](https://pub.dev/packages/flutter_secure_storage) (visited on 11/10/2022) (cit. on p. 48).
- [25] Flutter.dev. *Shared\_preferences*. URL: [https://pub.dev/packages/shared\\_preferences](https://pub.dev/packages/shared_preferences) (visited on 11/10/2022) (cit. on p. 48).
- [26] Unknown publisher. *App\_bar\_with\_search\_switch*. URL: [https://pub.dev/packages/app\\_bar\\_with\\_search\\_switch](https://pub.dev/packages/app_bar_with_search_switch) (visited on 11/10/2022) (cit. on p. 52).
- [27] Fluttercommunity.dev. *Share\_plus*. URL: [https://pub.dev/packages/share\\_plus](https://pub.dev/packages/share_plus) (visited on 11/10/2022) (cit. on p. 53).
- [28] Rmzy.dev. *Candlesticks*. URL: <https://pub.dev/packages/candlesticks> (visited on 11/10/2022) (cit. on p. 53).
- [29] Syncfusion.com. *Syncfusion\_flutter\_charts*. URL: [https://pub.dev/packages/syncfusion\\_flutter\\_charts](https://pub.dev/packages/syncfusion_flutter_charts) (visited on 11/10/2022) (cit. on p. 53).
- [30] Rprogrammer.net. *Flutter\_expandable\_table*. URL: [https://pub.dev/packages/flutter\\_expandable\\_table](https://pub.dev/packages/flutter_expandable_table) (visited on 11/10/2022) (cit. on p. 54).