1 Introduction

StocksPlayground is a web application whose aim is to provide a tool for users to view and analyze the stock and cryptocurrency market, and it also provides the possibility to make an account to use for saving favorite stocks and for keeping them all in one place. It is mainly built like a dashboard, composed of a series of views that group together relevant information, like the hottest sector of the market or the biggest gainer of the moment. It is also possible to view a treemap of the market, that is, a structure composed of block where each block represents a ticker, and whose size depends on the stock's market capitalization and whose color varies based on how much that company is gaining or losing at that moment in time. In addition, for every ticker there is a dedicated page where a candlestick graph is displayed, together with some relevant information about the symbol. These are just some examples of the features of StocksPlayground, as we will present them all in the next pages.

2 Infrastructure

The application is built on top of an infrastructure that was implemented using Docker and Docker Compose, as we can see in the following figure:

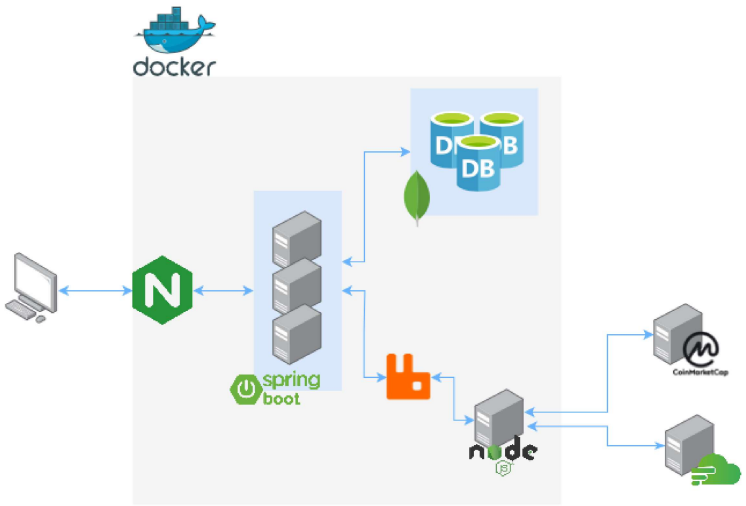

Figure 1: Infrastructure

- The `NGINX` is used as a reverse proxy for the application, as it permits to implement useful feature like caching and load balancing. We decided to use a round-robin load balancing strategy for distributing requests among the different instances of the server.

- The server itself is implemented through `Spring Boot`, a framework for building applications in Java. It makes it simple to create stand-alone, production-grade Spring based Applications that you can "just run". We decided to use this framework because it is very simple to use and it provides a lot of useful features, like the possibility to use the `Spring Security` module for managing the authentication and authorization of the users.
- The `NodeJS` server is used for performing API requests, as Javascript makes it very simple to perform HTTP requests and handle JSON objects.
- The `RabbitMQ` is used to build an RPC system, so that it behaves as a middleware that allows the communication between the instances of Spring Boot and the one of NodeJS.
- The `MongoDB` database is used for storing the data of the users, like their username, password and favorite stocks. We decided to use this database because it is production ready, it is very simple to use and it is very flexible, as it is a NoSQL database. It is also composed of JSON documents, a format that is easy parsable and not too verbose. Moreover, we deployed it as a replica set to ensure dependability.
- For the API to use for retrieving the data about the stocks, we decided to use `Finnhub`, a free API that provides real-time information about the stock market, and `CoinMarketCap`, since the latter had some more information we needed.

3 Deployement

The use of Docker Compose makes it straightforward to deploy the application, since we rely on a `docker-compose.yml` file that specifies all the services that compose the application, and how they should be dispatched. In particular, in the file we specify things like container names, ports that should be exposed, subnetworks and so on. Notably, in this file we also make a part of the configuration needed to make `MongoDB` work in replicaset mode.
The deployement itself is very simple. We just need to run `docker-compose build` to build the images, and then `docker-compose up` to start the containers. At a certain point, we created a script to automatize this process, and also to perform some steps needed for the set up of the application, which we will dive into later.

4 Feature Overview

After starting the application, we can navigate to the homepage: