University of Pisa
School of Engineering

Large Scale and Multi-Structured Databases

StockSim: Stock Portfolio Simulator

Supervisor
Prof. Pietro Ducange

Students
Marco Pinna
Rambod Rahmani
Yuri Mazzuoli

January 4, 2021

Contents

Part I

Documentation

Chapter 1

Introduction

StockSim is a Java application which, as main feature, allows users to simulate stock market portfolios. The StockSim application is composed by two main programs:

- **StockSim Server**: supposed to be running 24/7 to ensure historical data is always up-to-date;
- **StockSim Client**: can be launched in either `admin` or `user` mode.

The StockSim Server is not thought to be distributed to end users, whereas the StockSim Client can be used by both administrators and normal users. The choice was made to provide the same program to both administrators and normal users with two different running modes. Administrators can add new ticker symbols, new administrator accounts, delete both administrator and normal user accounts. Normal users have access to stocks and ETFs historical data, day by day, starting from 2010. They can create their own stock portfolios, run simulations a visualize the resulting statistics.

Before continuing with what follows, the following terms should be clarified:

- the **stock market** is any exchange that allows people to buy and sell stocks and companies to issue stocks; a stock represents the company's equity, and shares are pieces of the company;
- a collection of investments owned by an investor makes up his or her **portfolio**; you can have as few as one stock in a portfolio, but you can also own an infinite amount of stocks or other securities;
- a **stock symbol** is a one- to four-character alphabetic root symbol that represents a publicly traded company on a stock exchange; Apple's stock symbol is AAPL, while Walmart's is WMT;
- the NYSE and Nasdaq are open from Monday through Friday 9:30 A.M. to 4:00 P.M. (eastern time);

- the NYSE and Nasdaq close at 4 P.M., with after-hours trading continuing until 8 P.M.; the close simply refers to the time at which a stock exchange closes to trading;
- trading stocks after normal market hours through an electronic market, typically between 4:05 and 8:00 P.M., is **after-hours trading**;
- the **high** is the highest price at which a stock traded during a period;
- the **low** is the lowest price of the period;
- **open** means the price at which a stock started trading when the opening bell rang; it can be the same as where the stock closed the night before, but not always; sometimes events such as company earnings reports that happen in after-hours trading can alter a stock's price overnight;
- **close** refers to the price of an individual stock when the stock exchange closed shop for the day; it represents the last buy-sell order executed between two traders; in many cases, this occurs in the final seconds of the trading day;
- the **adjusted closing price** amends a stock's closing price to reflect that stock's value after accounting for any corporate actions; a stock's price is typically affected by some corporate actions, such as stock splits, dividends, and rights offerings; adjustments allow investors to obtain an accurate record of the stock's performance;
- **volume** is the total number of shares traded in a security over a period; every time buyers and sellers exchange shares, the amount gets added to the period's total volume.

Chapter 2

Actors and requirements

The main actors and the functional requirements are defided based on the previous simple description of the application. There is also a particular actor witch is in charge of the authomatic update of the dataset; non functional requirements are the caracteristics that ensure satisfactory interaction with users and real world utility of the product.

2.1 Actors

Based on the application design, four actors can interact with the system:

- A **Guest user** is someone who's not registered on the applicazion; this actor doesn't own private credentials for the login; in order to exploit the application main functionalities, it has to register a new account. The registration it's the only action allowed for a Guest;

- A **Registered user** is someone who's registered on the applicazion; this actor own private credentials (username and password) for the login; this actor can login as a user into the client application and utilize it's main features like watch stoks information, compose portfolios and simulate them.

- An **Admin user** is someone who's registered on the applicazion with administration credentials; this actor can login as an admin into the client and do some maintenance operations; this operations includes check the integrity of the entire dataset and add new stocks to it;

- The **Data Updater** is a thread running on one server; this thread is suppose to run always, and it's in charge of update the dataset with the new information coming from the stock market daily sessions; it's also in charge to find and fix (at list report) integrity issues.

The full use case diagram is provided on chapter 3.

2.2 Requirements

2.2.1 Functional requirements

Something about functional requirements.

2.2.2 Non-functional requirements

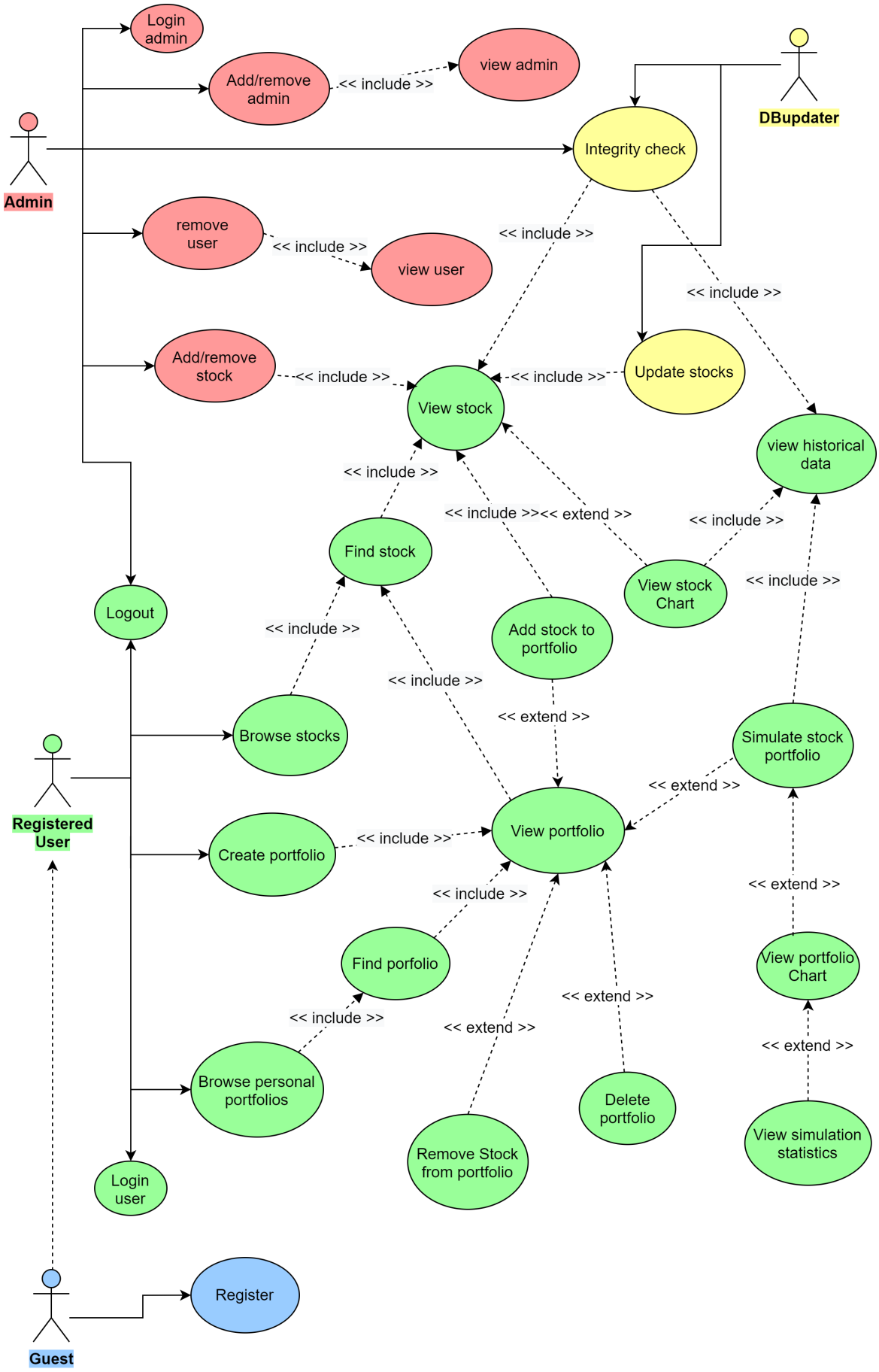
Something about non-functional requirements.

Chapter 3

UML diagrams

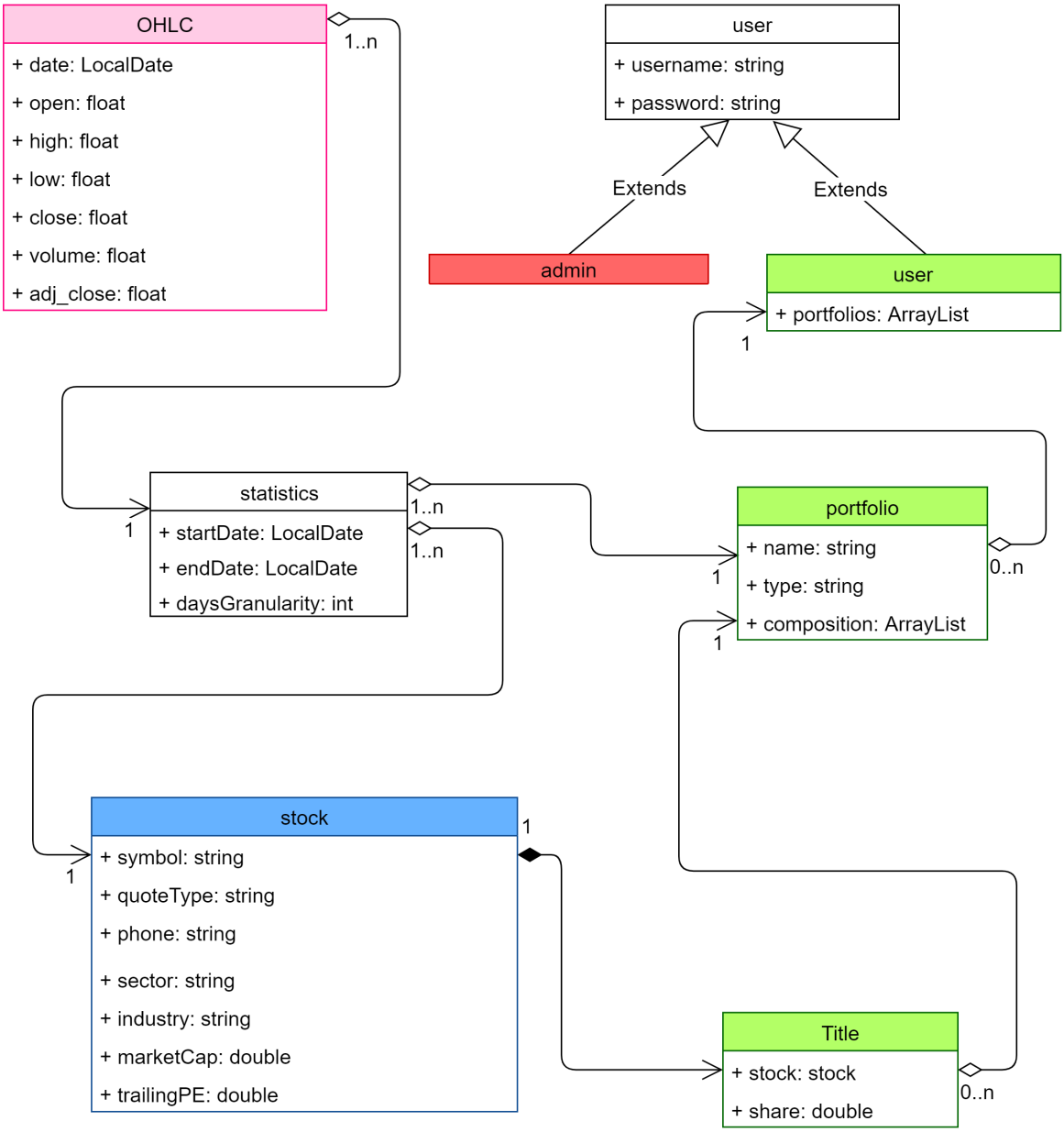
Blablabla.

3.1 Use Case diagram



3.2 Class diagram

Something about functional requirements.



Chapter 4

Database

The dataset is composd by 8236 stocks from the US stock market, along with their general informations and historical data; the application also need to store users' and admins' credential, personal informationof users, composition and details of each user's porfolio. We decided to use a column database for the storage of historical data; those informations represent around the 92.5% of our dataset and they are going to grow very fast during time; aggaregation and financial analytics on these volumes of data will perform better in a colum database where data storage is design to optimize this type of operations; We decided to store every other information in a document database, in order to exploit the schemaless property for save memory; information frequently needed toghether will be stored in the same document and indexes are created to speedup linking beetween documents;

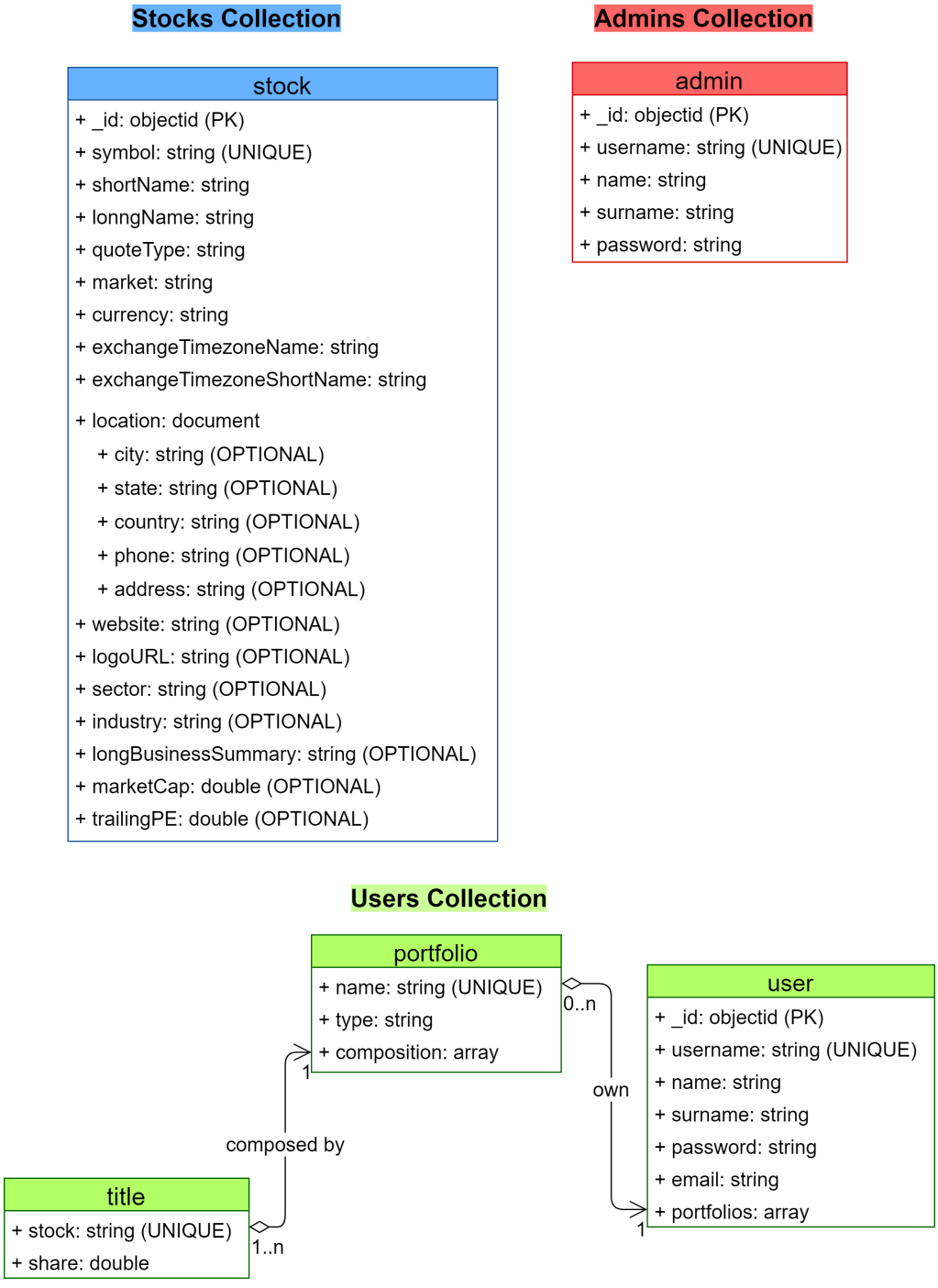
4.1 Dataset

4.1.1 Yahoo! Finance

4.1.2 NasdaqTrader

4.2 MongoDB

"MongoDB is a general purpose, document-based, distributed database built for modern application developers and for the cloud era." Taken from www.mongodb.com. MongoDB is a very famous document database with a great support for cloud operations, witch will improve the avaliability of our application. It also suport a lot of analytics functions and the creation of custom indexes in order to speedup read operations. In order to orgnize data ina a meaningfull and memery-optimal way, we opted for this structure:



This scheme is composed by 3 collections: **stocks**, **users**, **admins**;

- The stocks collection contains one document for each stock; inside this document are stored all the general information about the stock, wich is identified by the

attribute SYMBOL; some basic information are always presents, while others are missing for some stocks; we decided to keep these last type of informations where possible, exploiting the schemaless property of the documentat databse;

- The users collection contains one docuemt for each user regitered on the application; for every user login credentials are stored, along with few personal information; for every user is also stored an array of documents named PORFOLIOS: this array contains the porfolios of the user. Each portfolio has a scheme, witch include an array of TITLEs, named COMPOSITION, witch represent the settlement of the portfolio itsef. This nested structure it's been preferred from splittin data in different collections, because all the information of a user, including his portfolios, are frequently needed toghether; on the other hand, there aren't such operations that involve porfolios owneds by different users.

- The admins collection contains the admins login credentials toghether with few personal informations about them; we decided to crerate a separated collection for administators to improve the security of the administration features: in this way is impossible to inject administration privileges throw the login command.

4.2.1 Aggregations

One of the main features of own application is the possibility to choose some stocks from the market and combine them in to a portfolio. When a user is looking for a stock, he want to know statistic about **industies** and **sectors**, along with classification by **level of capitalization** and **PE ratio**; in ordewr to do so, we will provide these aggregation pipelines:

- the total market capitalization of each sector
- the total market capitalization of each industry
- the total market capitalization of stocks coming from the same country
- the avarage PE ratio of stocks working in the same sectors
- the avarage PE ratio of stocks working in the same industry
- the avarage PE ratio of stocks coming from the same country
- the avarage PE ratio of stocks beeing in a specific range of market capitalization

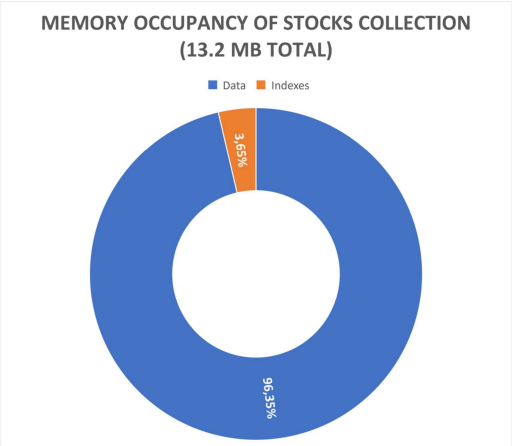
4.2.2 Indexes

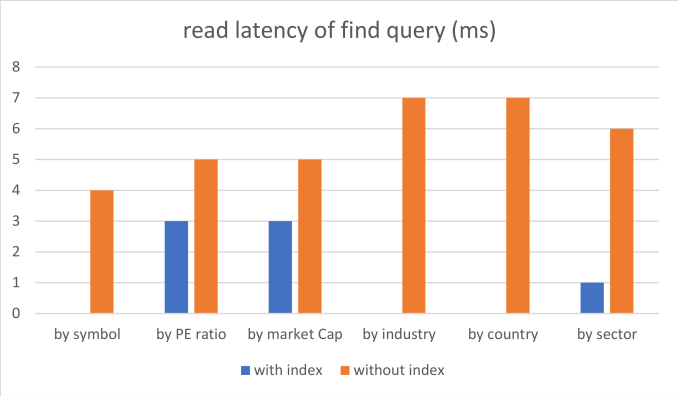
In order to speeup read operation in the document database, we decided to introduce some custom indexes:

- a REGULAR and UNIQUE index on the attribute **symbol** in the collection stocks;

- a REGULAR index on the attribute **marketCAP** in the collection stocks;
- a REGULAR index on the attribute **trailingPE** in the collection stocks;
- a REGULAR index on the attribute **sector** in the collection stocks;
- a REGULAR index on the attribute **industry** in the collection stocks;
- a REGULAR index on the attribute **country** in the collection stocks;
- a REGULAR and UNIQUE index on the attribute **username** in the collection users;

We provide some statistic that endorse our idexes choises





Analog results can be found about the username index in the users collection.

4.3 Apache Cassandra

4.3.1 Aggregations

4.3.2 Indexes

4.4 Sharding and Replicas

4.5 Apache Cassandra vs MongoDB

Chapter 5

Software architecture

5.1 Maven Multi-Module Structure

5.1.1 Library

5.1.2 Server

5.1.3 Client

5.2 Apache Maven Assembly Plugin

The Assembly Plugin for Maven enables developers to combine project output into a single distributable archive that also contains dependencies, modules, site documentation, and other files.

Chapter 6

Conclusions

Content.

Part II

User Manual

Chapter 7

StockSim Server Manual

For an application dealing with the stock market, it is essential to always provide up-to-date, consistent and reliable data. This is the purpose of the **StockSim Server** program. It is not intended to be distributed to end users. It is thought to be running 24/7.

The StockSim Server has two different startup modes: the first one

```
$ java -jar Server.jar

Welcome to the StockSim Server.

DATA CONSISTENCY CHECK SUCCESS. PROCEEDING WITH UPDATE.

Updating historical data for: IRBT.
Last update date: 2020-12-30.
Days since last update: 2.
Historical data updated for IRBT. Moving on.

...

Updating historical data for: EWU.
Last update date: 2020-12-28.
Days since last update: 4.
Historical data updated for EWU. Moving on.

...


Updating historical data for: XRX.
Last update date: 2020-12-28.
Days since last update: 4.
Historical data updated for XRX. Moving on.

Historical data update terminated.
Elapsed time: 1 hrs, 37 mins, 3622 secs.
Exceptions during update process: 8.
```

```
Failed updates: [ARKG, HDS, IWFH, LDEM, WWW, XVV, GINN, AAAU].

Available Commands:

status        check databases status.
update        update databases historical data.
quit          quit Stocksim server.
>
[UPDATER THREAD] Current New York time: 2021-01-02T07:50-[America/New_York]
[UPDATER THREAD] Going to sleep for 13 hours before next update.
>
```

which executes the `historical data update` procedure right after startup. Whereas, the second startup mode can be triggered using the `--no-update` command line argument:

```
$ java -jar Server.jar --no-update

Welcome to the StockSim Server.

Available Commands:
status        check databases status.
update        update databases historical data.
quit          quit Stocksim server.
>
[UPDATER THREAD] Current New York time: 2021-01-02T06:29-[America/New_York]
[UPDATER THREAD] Going to sleep for 14 hours before next update.
> update
DATA CONSISTENCY CHECK SUCCESS. PROCEEDING WITH UPDATE.

Updating historical data for: AUPH.
Last update date: 2020-12-31.
Days since last update: 1.
Historical data for AUPH already up to date. Moving on.

...

Updating historical data for: EWU.
Last update date: 2020-12-31.
Days since last update: 1.
Historical data for EWU already up to date. Moving on.

...

Updating historical data for: XRX.
Last update date: 2020-12-31.
Days since last update: 1.
Historical data for XRX already up to date. Moving on.

Historical data update terminated.
Elapsed time: 0 hrs, 4 mins, 44 secs.
Exceptions during update process: 3.
Failed updates: [ARKG, XVV, DUAL].
```

```
38
39 Available Commands:
40 status        check databases status.
41 update        update databases historical data.
42 quit          quit Stocksim server.
43 >
```

in this mode, no update is executed right after startup, the main menu is shown and the user can decide the action to be performed.

In the previously shown examples, we should pay attention to the following

- `DATA CONSISTENCY CHECK SUCCESS. PROCEEDING WITH UPDATE.`
- `the sotcksim server automatically detects the last update date, the number of days since the last update and fetches the required data using Yahoo Finance for EACH and EVERY ticker symbol.`
- `Historical Data Update logs.`
- `[UPDATER THREAD]`

Chapter 8

StockSim Client Manual

The StockSim Client has two different running modes: the first one

```
1 Welcome to the StockSim Client.
2
3 *** [RUNNING IN USER MODE] ***
4
5 Available Commands:
6 login         login to your user account.
7 quit          quit StockSim client.
```

is the `user mode`. Whereas, the second running mode can be triggered using the `--admin` command line argument:

```
1 $ java -jar Client.jar --admin
2
3 Welcome to the StockSim Client.
4
5 *** [RUNNING IN ADMIN MODE] ***
6
7 Available Commands:
8 login         login to your admin account.
9 quit          quit StockSim client.
```

and is the `admin mode`. Although they might look like the same, the available menu actions differ once the user/admin login has been executed.

8.1 StockSim Client User Mode

8.2 StockSim Client Admin Mode