# Fianance Guru

TBD TBD TBD

July 2023

## 1 Introduction

The goal of this project to build a database of stock informations. This includes stock prices as well as general information about the respective stock companies. Therefor we build a web server that acts as in interface between the user and the AlphaVantage API. While using our web server to query information from the API, the server stores all queried informations in a MongoDB and thereby builds our database. An overview of our project in shown in image TBD.
The main page of our webserver is the index page `http://localhost/`, see TBD. Here a user can insert any kind of query to find a company of them choice, eg. one can search of a ticker or a name. This query is forwarded to the AlphaVantage API and a list of possible matching results is displayed on the search result page page `http://localhost/stock`, see TBD. After selecting the company of choice, the webpage redirects you to the stock details `http://localhost/:symbol:` page, where `:symbol:` is the ticker that identifies the requested company. Here all relevant informations that were queried form the AlphaVantage API are displayed. At the same time, the webserver will store the company information in our own database. For each company that a user can inspect, the server also offers to download the data in the XML format. We also provide a XML schema file `http://localhost/schema.xsd` that can be used to validate all downloaded XML files.

## 2 Usage

The webserver is build with JavaScript and node. As prerequisites you need to

1. install node as well as its package manager npm
2. install python3
3. have a running MongoDB, either as a local server or by using an online cluster
4. create an account for using the AlphaVantage API.

If all prerequisites are satisfied, clone the repository and install all needed packages by running `npm install` in the main folder.
Now you need to setup the port on which you want your webserver to be available as well as the URI to your MongoDB database. Therefor create a file called `.env` in the main folder (see also in the next section) and adding the following two lines

```
MONGODB_URI = "<URI-TO-SERVER>"
PORT = <YOUR-PORT-NO>
API_KEY = "<YOUR-API-KEY>"
```

Now you should be able to start the server by running the command `node server` in the main folder.

## 3 Code description

The `./Controllers/stockController.js` file contains the code that runs the queries to the AlphaVantage API and stores items in our database. This file defines an object called `stockController` which defines four methods.
The search page `http://localhost/` is connected to the `stockController.fetchStockList` method, which sends the user query to the API via a GET request. The API response with a list of potentially matching results, which are rendered via an embedded-javascript template on the search result page `http://localhost/stock`.
If a user clicks on a company of them choice, a link redirects to `http://localhost/:symbol:/save`. The selected company is passed via the URL argument `:symbol:` to the `stockController.saveStockDetails` method. This method queries the relevant information about the selected company from the AlphaVantage API and stores them into our MongoDB. Afterwards the method redirects the users browser to the stock details page `http://localhost/:symbol:`.
Again the company of interest is passed as the URL argument `:symbol:` to the method behind, which is `stockController.display`
This method queries the requested symbol from our MongoDB and displays the result via an embedded-javascript template.

If a user clicks the download button on this page, the `createJSONFile` method, which can be also found in the `stockController.js` file, creates a local json file from the stock information currently displayed. Afterwards the `triggerPythonFunction` method calls the `converter.py` python file, that creates an XML file from it. This XML file is then served as a download to the users browser.

## 4 Files

| Path | Description |
|---:|---|
| `views/` | Contains the Embedded-JavaScript templates for rendering the html pages. |
| `views/index.ejs` | Template for the search page. |
| `views/stock.ejs` | Template for the result page, showing the matching results. |
| `views/stock_results.ejs` | Template for the page, showing the stock details. |
| `models/stock.js` | The database schema for an entry in our database. |
| `Controllers/stockController.js` | Contains the backend for the three pages of our webserver. Here the API requests as well as adding/updating entries in our database is done. |
| `.env` | Contains two environment variables that must be set to start the webserver and connect to the MongoDB database. |
| `server.js` | Entry script for the webserver. Start this file with node. Contains the code for assembling the webserver and defining the available pages. |
| `converter.py` | Contains the code for converting json to xml. Will be called by the server script automatically. |
| `schema.xsd` | Contains the XML schema, which we offer to validate downloaded XML files. |