Stock Market

Axel Louis Brink: CTO, Team Lead and Back- End Developer
Prattya Datta: Front-End Developer
Mani Tahriri: Back-End Lead
André Joyo: Back-End Developer
Cristina Moreno: Design Supervisor, Deploy Officer and Front-End Lead

This project involves designing a simulated stock/equity market trading platform for users. Two applications would be created accessing the same database with the same WEB API. One application will use NODEJS and the other will use VUE. The first application accesses the data only through backend while the other one accesses only through the AJAX in the front end.

- Laravel Rest API
- Node JS client
- Vue client

Contents

Project
Requirements

- Develop a REST API for delivering stock market data (DBMS of our own choice).
- Develop a Vue.js SPA that consumes the data provided by the API (Using Charts.js).
- Develop a Node.js/Express app that consumes the data from the API(Using Charts.js and render engine from our own choice).

The architecture design pattern, the technologies to be used and the final application to be delivered should consider the following requirements:

WEB API (Stock Eye API)

- Implement a DB. (The DBMS to use has to be decided by the team and also if the DB features relational or non relational design)
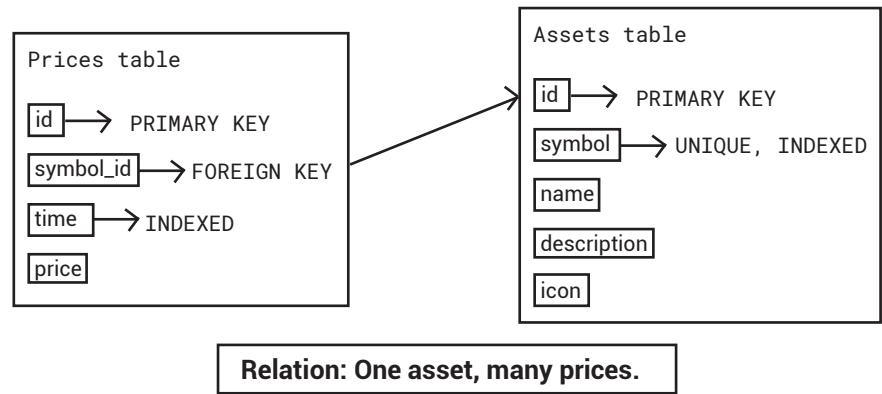- Backend has to be developed in PHP / Laravel v.7.

APP - COMPANY1 (Market Basket)

- This is a hypothetical client for the API.
- It has to be developed in NodeJS to query the API so the responsibility to obtain data relapse on the backend. It's important to properly design Classes and Methods that abstract data the cleaner the better.
- It won't be using AJAX as data will be consumed from the Backend and will be conveyed to the views.
- Views have to be responsive being Bootstrap allowed.
It has to use Chart.js as a requirement.

SPA APP - COMPANY 2 (Visual Market)

- Create a SPA (Single Page Application)
- Implement VueJS and query data from the WEB API using AJAX.
- Using npm, webpack and sass is mandatory
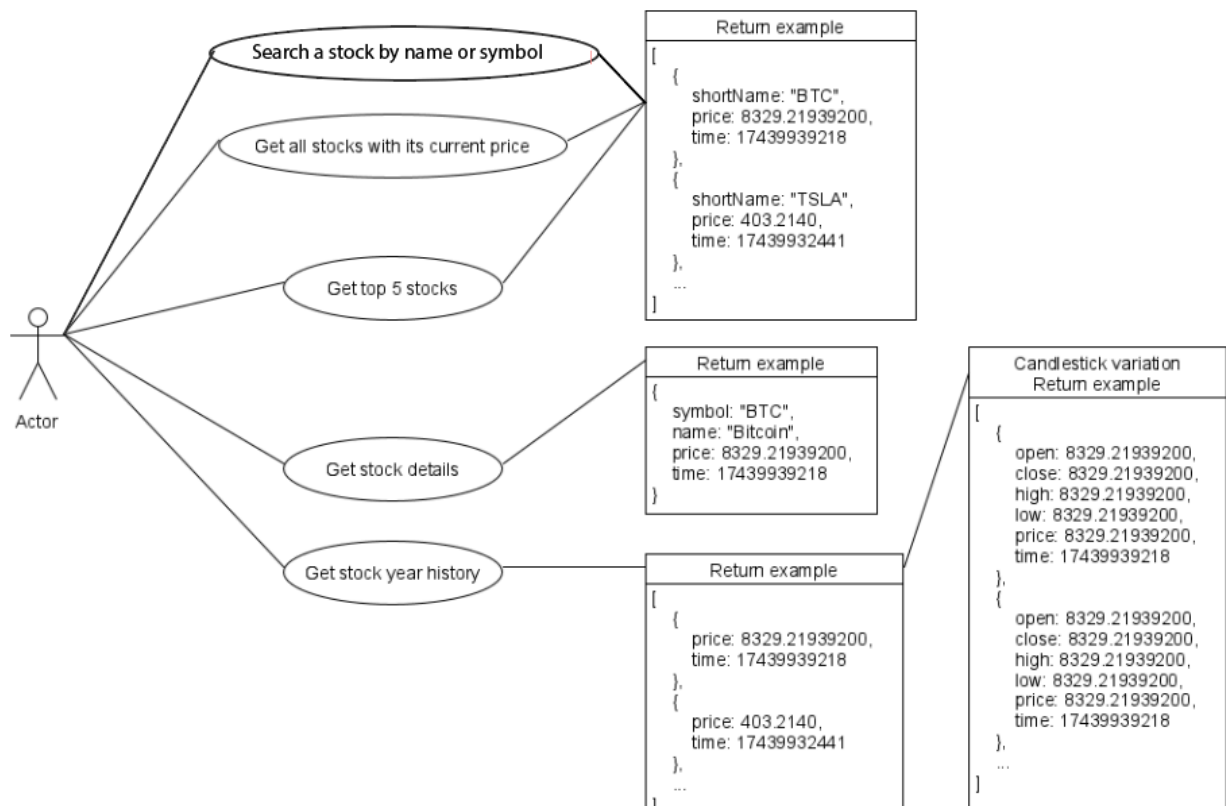- It has to use Chart.js as a requirement.

The database architecture

Prices table

id ⟶ PRIMARY KEY

symbol_id ⟶ FOREIGN KEY

time ⟶ INDEXED

price

Assets table

id ⟶ PRIMARY KEY

symbol ⟶ UNIQUE, INDEXED

name

description

icon

Relation: One asset, many prices.

Using Laravel for the WEB API is mandatory. All the good practices of Laravel (Eloquent, Migrations, Routing, Controllers, etc.) should be used.

API as a client I can...

- Get all stocks with its current price
- Get top 3 stocks
- Get stock details
- Get stock history
- Search stock by symbol

Search a stock by name or symbol

Get all stocks with its current price

Get top 5 stocks

Get stock details

Get stock year history

Actor

Return example

```
[
    {
        shortName: "BTC",
        price: 8329.21939200,
        time: 17439939218
    },
    {
        shortName: "TSLA",
        price: 403.2140,
        time: 17439932441
    },
    ...
]
```

Return example

```
{
    symbol: "BTC",
    name: "Bitcoin",
    price: 8329.21939200,
    time: 17439939218
}
```

Return example

```
[
    {
        price: 8329.21939200,
        time: 17439939218
    },
    {
        price: 403.2140,
        time: 17439932441
    },
    ...
]
```

Candlestick variation
Return example

```
[
    {
        open: 8329.21939200,
        close: 8329.21939200,
        high: 8329.21939200,
        low: 8329.21939200,
        price: 8329.21939200,
        time: 17439939218
    },
    {
        open: 8329.21939200,
        close: 8329.21939200,
        high: 8329.21939200,
        low: 8329.21939200,
        price: 8329.21939200,
        time: 17439939218
    },
    ...
]
```

Market Basket: As a user I can...

- Get all stocks with name, symbol, image, description, latest entry in prices table related to every stock.
- Get history of every stock and enable user to filter by time.
- Search by stock symbol
- Get top 3 stocks

Visual Market: As a user I can...

- Get all stocks with name, symbol, image, description, latest entry in prices table related to every stock.
- Get history of every stock and enable user to filter by time.
- Search by stock symbol
- Get top 3 stocks
- Bookmarks feature (using front-end technologies).

Risk Management

● Risk level: Low   ● Risk level: Medium   ● Risk level: High

Not being able to solve some of the challenge

Members of the team being infected by Coronavirus

Not delivering in time

Having GIT issues

Having issues with a computer

Wireframes

GIT Workflow

Each project will be in its own repository. Having then three repositories with two branches each. All commits will go to the dev branch, and when a release status is reached, it will be pushed into master. Using then the branch master as an stable-release code.

Roles

Axel Louis Brinck Fernández

CTO, Team Lead and Back- End Developer.

Axel is the person responsible for assigning tasks, supervising tasks before being added to the master branch, designing database and decision maker in the team.
Supervisor and highest ranked person in contact with Assembler and other Team Leads.

Prattya Datta

Front-End Developer

Prattya is part of the Front-End team composed by Cristina and himself. They are both in charge of developing the Vue SPA and also conveying same markup, styling and layout to the NodejS App.

Cristina Moreno Medran

Design Supervisor, Deploy Officer and Front-End Lead

Cristina is the design supervisor, it means that she's going to supervise and make sure that design guidelines are being followed. She is also in charge of the research and implementation of the deployment to Heroku. She is part of the Front-End team composed by Prattya and herself and they both are in charge of developing the Vue SPA and also conveying same markup, styling and layout to the NodejS App.

Mani Tahriri

Back-End Lead

Mani is part of the Back-End team composed by André and himself. They are both in charge of developing the NodejS APP and integrate all markup, styling and layout provided by the front-End Team.

André Joyo

Back-End Developer

André is part of the Back-End team composed by Mani and himself. They are both in charge of developing the NodejS APP and integrate all markup, styling and layout provided by the front-End Team.

Tasks

The tasks are all high priority except the ones that finishes in a character.
Tasks are distributed per-day and per-worker so you can see what you need to do.
Tasks represents also the deadline day!

#L: Laravel tasks
#L01: Document DB use cases
#L02: Create DB MySQL creation script
#L03: Create DB MySQL population script
#L04: Create Laravel Project initial state
#L05: Create Laravel API routing according to DB use cases
#L06: Create Laravel Controller files that connect to DB through Eloquent.
#L07: Refactor project.
#L08: Project readme.md
#L09: Deploy to heroku.

#N: Node tasks
#N01: Create initial state project using npm. Install express.
#N02: Make functions that simulate fake data coming from Laravel backend (see document's DB use cases.
#N03: Implement express routing to the following pages: top24, menu ('/'), search, about, market.
#N04: Implement design top.
#N05: Implement design menu.
#N06: Implement design search.
#N07: Implement design about.
#N08: Implement design market.
#N09: Using fake functions from #N02, populate respective chart.js, handle needed routings.
#N10: If laravel backend is on release state (heroku) redirect fake functions to real endpoints.
#N11: Project readme.md
#N12: Testing in IE11. Safari, Opera...
#N13: Deploy to heroku.

#V: Vue tasks
#V01: Create initial state repository.
#V02: Make functions that simulate fake data coming from Laravel backend (see document's DB use cases.
#V03: Implement SPA designed frontend.
#V04: Using fake functions as your source, populate respective chart.js.
#V05: Using localStorage, implement ability to 'star' or follow a market.
#V06: If laravel backend is on release state (heroku) redirect fake functions to real endpoints.
#V07: Project readme.md
#V08: Testing in IE11. Safari, Opera...
#V09: Deploy to heroku.

Investigation tasks
#I01: Provide a documentation about how to use chart.js with financial data (candlesticks)
#L09A: Implement socket realtime data from binance.com
#L09B: Implement docker.
#S: Provide support if needed in other tasks

Heroku init
#H01: Init Laravel
#H02: Init Node
#H03: Init Vue

	Prattya	Cris	Mani	André	Axel
Friday & Monday	V02 S	H01 H02 H03 V01 V03	N01 N03	N02 I01 N03	L01 L02 L03 L04
Tuesday	N04 N06 N09	N05 N07 N08	N03 I01	N03 I01 (Continue)	L05 L06 L07 L08
Wednesday	N10 V07	L09 V05	V04/V06	S	N12 L09A L09B
Thursday	S	N13 V09 N12 V8	S	S	V08 L09A L09B
Friday	S	S	S	S	S

Heroku Docs

What is it?

Heroku is a platform as a service based on a managed container system, with integrated data services and a powerful ecosystem, for deploying and running modern apps. The Heroku developer experience is an app-centric approach for software delivery, integrated with today's most popular developer tools and workflows.

.env

https://devcenter.heroku.com/articles/config-vars

We need to make sure our .env vars (which won't be deployed) are available in Heroku. We can do that by using the Heroku CLI:

What do we need to install?

- HEROKU CLI --> https://devcenter.heroku.com/articles/heroku-cli

- Create Heroku Account

MySQL

https://elements.heroku.com/addons

ClearDB MySQL

The high speed database for your MySQL powered applications.

JawsDB Maria

MariaDB, the open source drop-in replacement for MySQL now available on Heroku

JawsDB MySQL

The database you trust, with the power and reliability you need.