Technical University of Moldova

Faculty of Computers, Informatics and Microelectronics

Software Engineering and Automatics Department

Technologies and Tools in Web Development

Twitter Clone App

Authors:
Vasile Drumea
Namașco Petru
Chetrușca Dumitru

Supervisor:
Mihail Gavrilița

Chișinău 2019

1 About the project

The project represents a twitter-like web application, with the features described below.

We implemented the application using 2 JS frameworks. For the back-end (API) we used AdonisJS which is a Node.js MVC framework, and for the front-end Vue, a progressive JavaScript framework.

2 Project Requirements

- **Mandatory requirements**
  - Version Control (Git);
  - Containerization (Docker);
  - Testing
- **Mandatory features**
  - User Registration;
  - User Login/Logout;
  - User should be able to tweet;
  - User should be able to follow other users;
  - User should see tweets from users he follows;
- **Optional features**
  - User should be able to react to tweets;
  - User should be able to retweet a tweet from other users;
  - User should receive an email notification when a person likes/retweets his tweets;
  - User should be able to bookmark/save tweets and there should be a dedicated page where user can see them;
  - Any other features;

3 Technology Stack

- **Frontend**
  - Vue - Progressive JS Framework;
  - Axios - HTTP client;
  - Semantic UI - CSS Framework;
  - VeeValidate - Input Validation for Vue.js;
- **Backend**
  - AdonisJS - Node.js MVC Framework;
  - Node.js 10.15.0;
  - NPM 6.4.1;
  - MySQL;

4 Project implementation

4.1 Backend

As mentioned above for the backend we used AdonisJS, a Node.js MVC framework. Here are some characteristics of it :

- Solid MVC arhitecture;
- Active Record based ORM;
- Unit testing API;

First we need to instal the Adonis CLI for future usages with the command:

```
npm install -g @adonis/cli
```

Then, to create the application we used the command:

```
adonis new tviter_api --api-only
```

Here was used the **–api-only** flag because we don't need any views from adonis, only the blueprint for the api.

To launch the app run commands:

```
cd tviter_api
adonis serve --dev
```

It will be accessible at **localhost:3333**.

Next comes the database setup. For the DBMS we used MySQL. To interact with the app we need to install the coresponding package:

```
npm install mysql --save
```

The details for the DB connection are located in the environment variables from .env:

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_USER=root
DB_PASSWORD=$%#@&
DB_DATABASE=tviter
```

After the database is configured, to create all the necessary tables and all of it we need migrations.

To create a migration we use the command:

```
adonis make:migration name
```

In the current app there are 6 migrations all identified by an id and the table or object it creates/modifies: *id_name.js*

To keep the form of the data an the relations between tables we use models. The models used in the API are:

– Favorite;

– Reply;

– Tweet;

– User;

In those are specified relations with methods from adonis API.

For the Controller we have 3 Http request/response controllers created with the command:

```
adonis make:controller name --type=http
```

These are:

– FavoriteController;

– TweetController;

– UserController;

Also, the routes are defined in the routes.js file from start folder.

4.2 Frontend

The frontend was implemented in a Vue.js app. Initially we need Vue cli ofcourse and it can be installed with the command:

```
npm install -g vue-cli
```

Then to create the app we used the command:

```
vue init webpack tviter-frontend
```

To launch the app use the commands:

```
cd tviter-frontend
npm install
npm run dev
```

To connect the frontend to the API we need axios which is an HTTP client. Install it with:

```
npm install axios --save
```

Then to configure it go in src/main.js file and update as follows:

```
import axios from 'axios'

// add these before Vue is instantiated
window.axios = axios // reference axios globally
axios.defaults.baseURL = 'http://localhost:3333' // default URL for API
```

As CSS framwork we used SemanticUI which was include in index.html by CDN:

```
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/semantic-ui/2.2.13/s
```

After this comes the UI. It was implemented with components. Each component contains the markup and the script. The used components can be arranged as a hierarchy in the following way:

- **Components**
- - **Auth**
    - LoginForm
    - SignupForm
- - **Tweet**
    - Replies
    - SingleTweet
    - Tweet
    - TweetRecations

  - Tweets
- - **User**
  - * **Profile**
    * FavoriteTweets
    * UserCard
    * UserFollowers
    * UserProfile
    * UserProfileHeader
    * UserProfileSidebar
    * UsersFollowing
  - * **Settings**
    * UserPasswordSettings
    * UserProfileSettings
    * UserSettingsMenu
  - UserSidebar
  - WhoToFollow
- Home
- Notification

The routes are kept in router/index.js file.