Moapi

# Secondary school of electrotechnical engineering Jecna 30

# 2022-2023

# Matěj Šturma

# C4c

Moapi is an API platform intended for making API request sending easier. Moapi is built using Electron.js and React frameworks and is entirely written in TypeScript.

Development

Pre-requisites

- Node.js
- MySQL Server

Installation and build

1. Clone the GitHub repository.
2. Create a database using the SQL script located at /api/sql/script.sql (See Database for more information).
3. Open terminal and run `npm install` inside the **api**, **app** and **web** folders.
4. Create an .env file according to the Configuration part.
5. Start the API server by running `npm run dev` inside the **api** folder.

Now you can start both the app and the website in development mode by running `npm start` in the **web** folder (for the website) and `npm run electron-dev` in the **app** folder.

Configuration

Before actually running the application, you'll have to specify some information.

To do that, create a file inside all 3 folders (api, app, web) and name each of them **`.env`**.

After that, follow these steps:

1. Inside *api/.env*, input the following:

```
PORT=<your desired port the application will run on (if you don't know which, input 4000)>

DB_HOST=<IP address or domain your database is hosted on. If you're hosting your
database locally, input localhost>

DB_USER=<your database username>

DB_PASSWORD=<your database password>

DB_NAME=<name of your database, api_app if you used the SQL script>
```

SMTP_HOST=<hostname of your SMTP server>

SMTP_PORT=<port of your SMTP server, usually **587**>

SMTP_USERNAME=<your SMTP username>

SMTP_PASSWORD=<your SMTP password>

AUTH=<desired authentication token. This token will be used for request authorizations>

LOG_PATH=<path of the error log file, located in **/api/log/log.txt** (you have to put the whole path here)>

2. Inside *<u>/app/.env</u>*, input the following:

SERVER_URI=<URI the backend server (api) runs on>

AUTH_TOKEN=<the **same AUTH token** inputted in /api/.env>

3. Inside *<u>/web/.env</u>*, input the following:

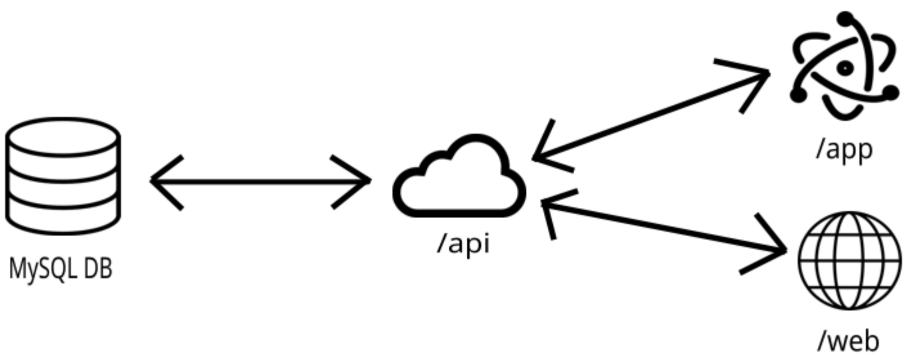
SERVER_URI=<URI the backend server (api) runs on>

AUTH_TOKEN=<the **same AUTH token** inputted in /api/.env>

Architecture

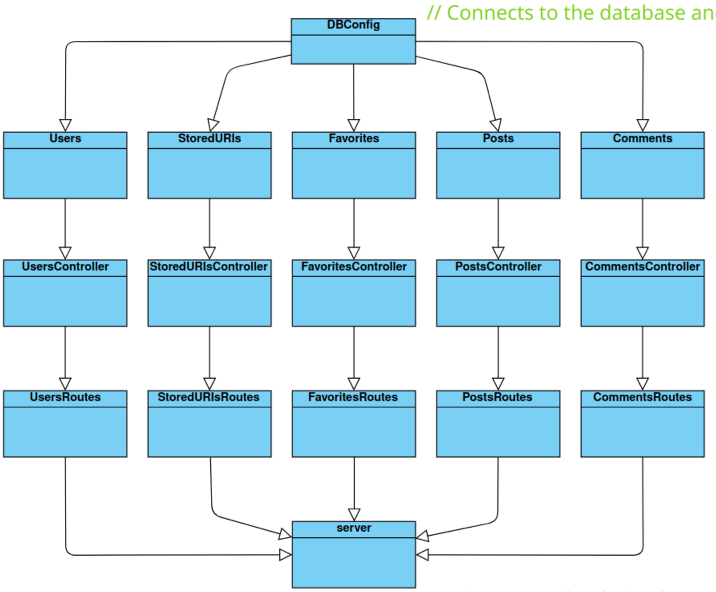
The project is divided into 3 separate programs. The REST API server (in the /api folder), the desktop Electron application (in the /app folder) and the website (in the /web folder).

The following image describes how these 3 programs work together.



API

Lets now talk about the API. As mentioned before, the program is written according to the **REST** architectural style, allowing it to perform basic **CRUD** operations.



// Connects to the database and returns a pool

// Database entities mapped into objects.
// Includes CRUD functions.

// Creates an object from the classes
// above and calls the CRUD
// functions with the request data.
// Returns a response status with a JSON.

// Provides a router for each of the
// access functions.

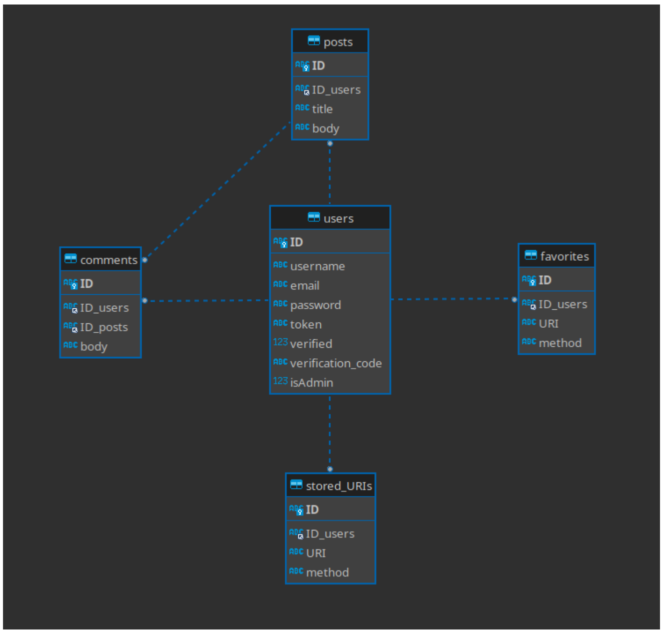
// Listens on the desired port

Data Access Object (DAO) design pattern is being used.

More details about the individual functions is summarized in the in-file documentation.

Database

Moapi uses MySQL as its RDBMS. Below, you can see the E-R model and description for each table.



Moapi’s database consists of 5 tables.

1. **Users** table
    - When a user registers, his credentials (username, email, password) are stored into this particular table.
    - Passwords are stored hashed.
    - Token is a 36 characters long code that is being stored in cookies.
    - Verified is a boolean value.
    - Verification_code is being sent to the user's email after registration and before verification.
    - isAdmin is a boolean value.

2. **Stored_URIs** and **Favorites** tables
    - Both of these tables have the same columns. Data is being inserted into the stored_uris table after every request, while the favorites table is being inserted only after user favorites an URI.

3. **Posts** table
    - This table is being accessed only on the website. It stores the posts from /help.

4. **Comments** table
    - Users can send a comment under every post.

Third-party Libraries

Moapi uses a lot of NPM packages. The most notable ones are:

- **React**: the most popular library/framework amongst frontend developers. Both the app and the website are written using React.

- **Express**: a backend framework for building REST APIs.

- **Electron**: a framework designed to create desktop applications.

- **Sass**: a popular CSS framework.

Other third-party libraries:

- **MySQL2** - Node.js MySQL library, allowed to perform queries and prepared statements.
- **SMTP-client** - a Simple Mailing Protocol client.
- **UUIDv4** - library for generating UUIDs.