Oblig 2

RESTful chat application

with bad bots

Authors:

- Adrian Tokle Storset, s341859
- Erik Storås Sommer, s341870
- Mats Sommervold, s341829

Our solution:

The core of our solution is a RESTful `nodejs/express` server. Our frontend is a `svelte application` (frontend framework) that runs in the browser and is statically hosted at the server root. All API endpoints are hosted at `/api`. Further API documentation can be found in `DOCS.md`.

How to run:

There are multiple run configurations, all of which can be found in `README.md`, but the easiest way is to build and run a docker image like this:

Make sure you are in the root directory "/"

```
docker build -t rest-chat-app .
docker run -p 5000:5000 -d rest-chat-app
```

The option `-p` maps a port in the docker container to your local machine. Now go to `localhost:5000` in your browser to view the application.

Run `docker stop $(docker ps -a -q)` to stop the docker container.

Try our online version:

Our solution is also hosted at: http://rest-bots.herokuapp.com/
When visiting for the first time, the server might do a cold start, which means it can take up to 20-30 seconds to load initially.

How to use:

Our solution uses an in-memory store. Therefore any user and message data won't persist between restarts. The application is designed and tested in the newest stable version of google chrome, but will most likely run fine in any modern browser.

Recipe:

- Go to `localhost:5000` in your browser
- Create a user (it will log you in automatically)
- The login is valid for 24 hours and the browser will remember your token between refreshes.
- Create how many rooms you want from the left side panel
- Click a room to view and post messages.
- When viewing a room, feel free to post messages or add bots to the room from the right sidebar.
- Since you are the creator of the room you also have access to delete the room.
- If you log out and back in with another user, you will see that you have the option to join rooms created by the first user.

Bots:

- The bots are designed to interact with you, the user, but will occasionally interact with each other if there are multiple bots in the room at the same time.
- In the background, both the svelte application and the bots use the same. For every bot you add, you will create another instance of the client with its own user and login information.
- When a bot has sent between 10 and 30 messages it can randomly decide to leave the room and delete its own user, but it will send a goodbye message of some sort first.
- A bot can also leave prematurely if it does not receive any responses.
- If you refresh the page when a bot is running, it will cut off the bot. The bot won't be able to exit and delete its user gracefully and it will not start up again when the page reloads.

Source code:

The source code is structured into three main folders. `server`, `client` and `app`. The `client.js` file in the `client` folder exports a new instance of the client. It is designed to be able to run in both the browser and nodejs. The app folder contains the source code for the svelte frontend application. As mentioned before, it uses the client module to interact with the server. Since the client also can be run in a node environment, you could technically run a bot from the terminal by the code snippet below when standing in the client directory:

```
npm install
USERNAME=Andreas ROOM=Testroom node bot.js
```

The environment variables USERNAME and ROOM will let you chose the name of the bot and which room it should join. If the room does not exist, it will create it. This will make the bot admin in the room. Since the bot is admin it will avoid deleting its own user, since that will mean deleting the room when leaving as well. Running the bot from the terminal is considered an extra, largely untested feature.