IRCChat

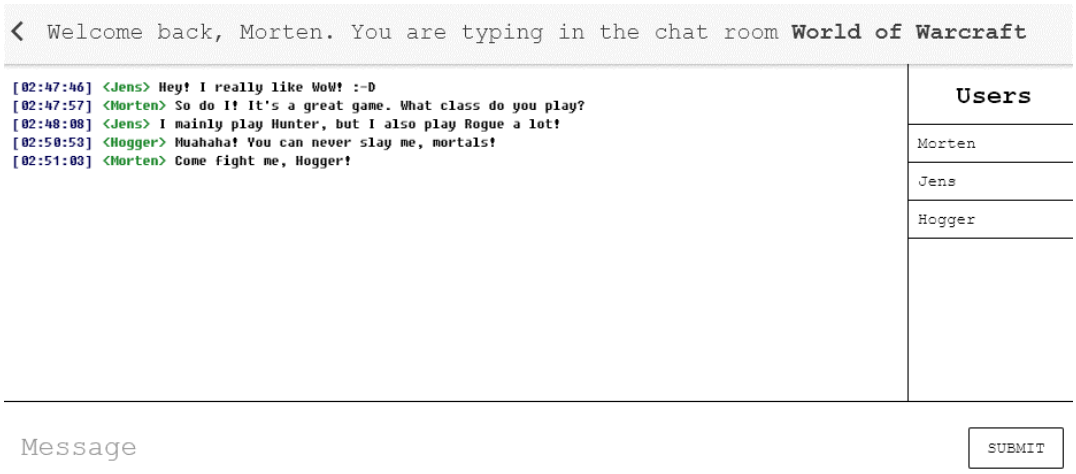


Table of contents

How to run it

To run my project, you need to do the following things:

1. Extract the two folders (Client and Server) into a folder
2. Open up two CLIs and cd into both Client/ and Server/
3. Start the server by doing:
    a. **npm install**
    b. **node app.js**
4. To start the client, make sure ionic is installed first, then do:
    a. **npm install**
    b. **ionic serve**
5. It should now open a browser on http://localhost:8100

How to test it

1. Make sure the project can be run (see "How to run it")
2. Open up another CLI and run this command:
    a. **npm test**
3. A browser window should open up with Jasmine, which should return all the tests as passed

The tech

I went with the following tech:

- Frontend
    - Ionic 2
    - Angular 5
    - TypeScript
    - Socket.IO (ng-socket-io package)
- Backend
    - NodeJS
    - Mongoose
    - Socket.IO (socket.io package)
- Database
    - MongoDB

How it's all setup

Frontend

I have three pages/views, all with their own controller/component. I have three models on the frontend: User, Room, and Message. These all have their own page and microservice, which connects to the NodeJS backend with HTTP requests.

When someone creates a new room, it will broadcast to **everyone** that a new room has been added, then prepend that to the list of rooms.

When someone connects to a new chat room, it will broadcast to that room, that a new user has connected. That user is then be added to the list of connected users (on each client in that room).

When someone writes a new message, it sends a POST request to the backend, which broadcasts to the entire room, that a new message has been sent. The new message is then appended to the list of messages in that room.

Backend

The backend receives GET and POST requests. It is also used for polling, using sockets, for every client connected. I chose to use HTTP requests, to show I know how to use Express and how to setup a neat environment using microservices.

I have three models on the backend: User, Room, and Message. Each of these are set up in Mongoose, where they are also referenced to each other. For example: A room was created by an user. That user is then referenced in the room, so we can see who created it. It then gets populated and returned to the user. The serialization between the frontend and backend works wonderfully, so we always work with pure objects instead of objects with IDs (which is how they are stored in the database, though).

Github

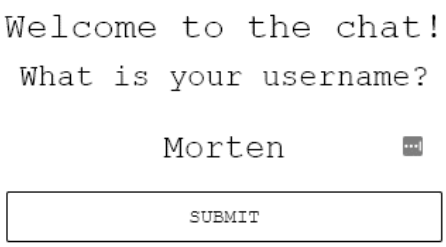
I obviously made a Git repository, just because ☺ Here is a link:

https://github.com/mortenmoulder/EAAA-Chatroom

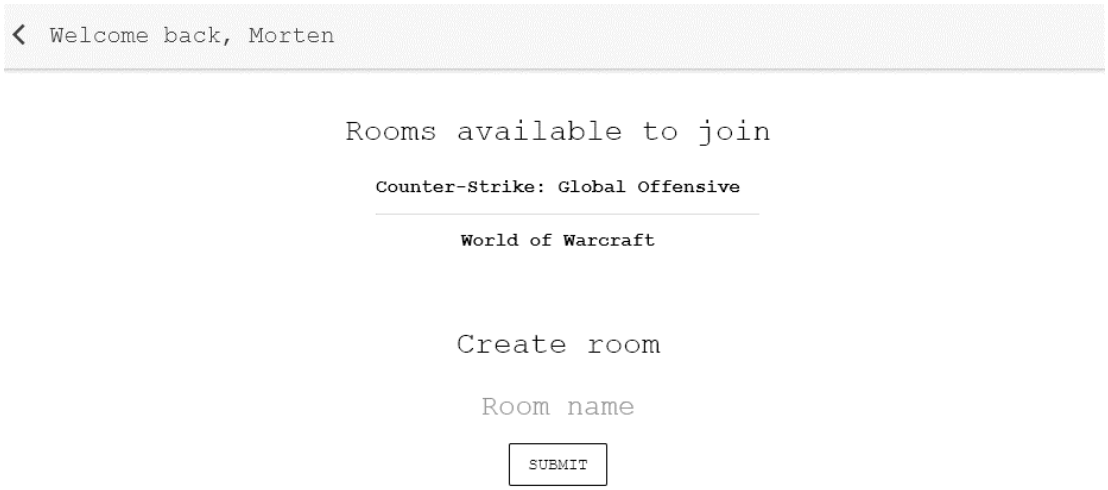
Design considerations

I went with a very minimalistic design. I tried to replicate the mIRC client for IRC chats (old chat program), which is why I decided to import the default font in mIRC and their syntax for messages. I tried to make use of Ionics styling as much as possible, but I did most of the style myself (using flexboxes most of the way through).

Login page



Room selection



Chat