

Tecnologie e Applicazioni Web

Zane Filippo – 880119

Introduction

We have been asked to develop a web application to be able to play a multiplayer battleship.

Architecture

The application is divided into two main components: “frontend” and “backend”. As the name suggest, the first one contains all the parts of the graphical user interface and all temporary data like chat and game session. In the backend are stored all persistent data such as offline chats and games information.

We developed an Android application which can be used to play. It's developed using Apache Cordova and has the same functionalities as the web one.

We used Socket IO to manage all communications between the players. Some examples regarding its application are: game requests, friend requests, game actions, ecc.

Data Model

We have divided the data model into three collections: match, users and messages.

The user model contains some useful information about a player like his name, username, password and some infos on his match history

```
const UserSchema = new Schema({
  name: {type: String, default: ''},
  username: {type: String, default: ''},
  email: {type: String, default: ''},
  password: {type: String, default: ''},
  wins: {type: Number, default: 0},
  loses: {type: Number, default: 0},
  matches: {type: Number, default: 0},
  isModerator: {type: Boolean, default: false},
  isFirstLogin: {type: Boolean, default: false},
  friends: {type: [Schema.Types.ObjectId], default: []}
});
```

The message model provides a method for representing text messages between users on the application.

```
const MessageSchema = new Schema({
  from: {type: String, default: ''},
  to: {type: String, default: ''},
  message: {type: String, default: ''},
  timestamp: {type: String, default: ''},
  new: {type: Boolean, default: true},
});
```

The match model describes a game between two players.

```
const MatchSchema = new Schema({
  player1: {type: String, default: ''},
  player2: {type: String, default: ''},
  score1: {type: String, default: ''},
  score2: {type: String, default: ''},
  winner: {type: String, default: ''},
});
```

REST APIs

We were asked to create some REST APIs endpoints with NodeJS. Here some:

- **signup**
 - Method: POST
 - Params: name, username, email, password
 - Description: create a new user
- **signin**
 - Method: POST
 - Params: username, password
 - Description: log the user into the application
- **addFriends**
 - Method: POST
 - Params: username, friend
 - Description: add a new friend to the friends list
- **allUsers**
 - Method: GET
 - Params: NaN
 - Description: return all users currently online in the application
- **getAllUsers**
 - Method: POST
 - Params: moderator
 - Description: is the parameter is a moderator, return all users in the application
- **getModerators**
 - Method: GET
 - Params: NaN
 - Description: return all moderators
- **friend**

- Method: POST
 - Params: username
 - Description: return the username's friends list
- **deleteFriend**
 - Method: POST
 - Params: username, friend
 - Description: remove friend from username's friends list
- **logOut**
 - Method: POST
 - Params: username
 - Description: log the user out from the application
- **matchId**
 - Method: POST
 - Params: id
 - Description: if the id is from a match in progress, returns information about the match
- **firstLogin**
 - Method: POST
 - Params: username, password, name, email
 - Description: if a moderator log into the application for the first time, he is asked to change some of his personal data
- **deleteUser**
 - Method: POST
 - Params: moderator, username
 - Description: remove a player from the application
- **addModerator**
 - Method: POST
 - Params: moderator, username, password
 - Description: add an user to the moderators list
- **matches**
 - Method: POST
 - Params: NaN
 - Description: return all in progress games
- **history**
 - Method: POST
 - Params: username
 - Description: return all games about a player
- **chat**
 - Method: POST
 - Params: username, friend, msg
 - Description: create a new message and send it to the receiver
- **getChat**
 - Method: POST
 - Params: username, friend
 - Description: return all messages between two users
- **readChat**
 - Method: POST
 - Params: username
 - Description: read all messages

Authentication

The application handle the authentication method as follows: at login time, if the credentials are correct, a JWT token is generated using JSONWebToken library and the sign function. The token is sent to the user and saved in the frontend in the sessionStorage and then maintained throughout the session.

This token allows the user to access all the endpoints that have “mustAuth” declaration. When a logged in user tries to access a resource with a “mustAuth” declaration, his JWT token is sent over the HTTP request as a Bearer token in the Authentication header.

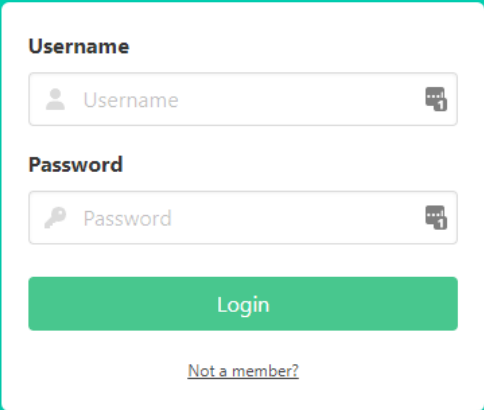
Angular Frontend

Frontend application is divided into eleven components:

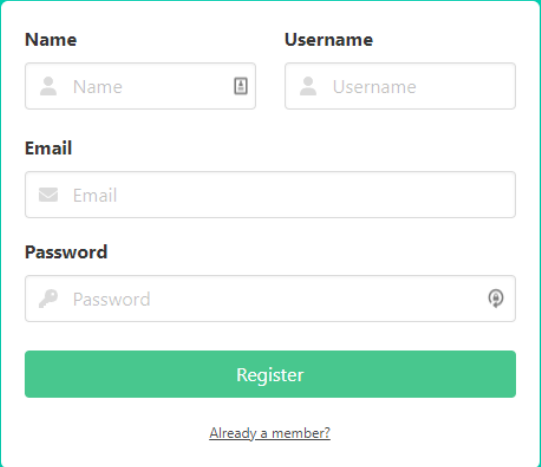
- **battleship-game**: manage all the information about a game
- **chat**: manage offline chats
- **confirmation-dialog**: it is used to show notifications
- **friends**: friends management
- **game**: an user can decide to watch or to play a game
- **home**: navbar and historical information
- **login**: login page
- **moderator**: manage all moderator’s operations
- **register**: sign up page
- **watch-game**: allow a player to watch a game

Application's Workflow

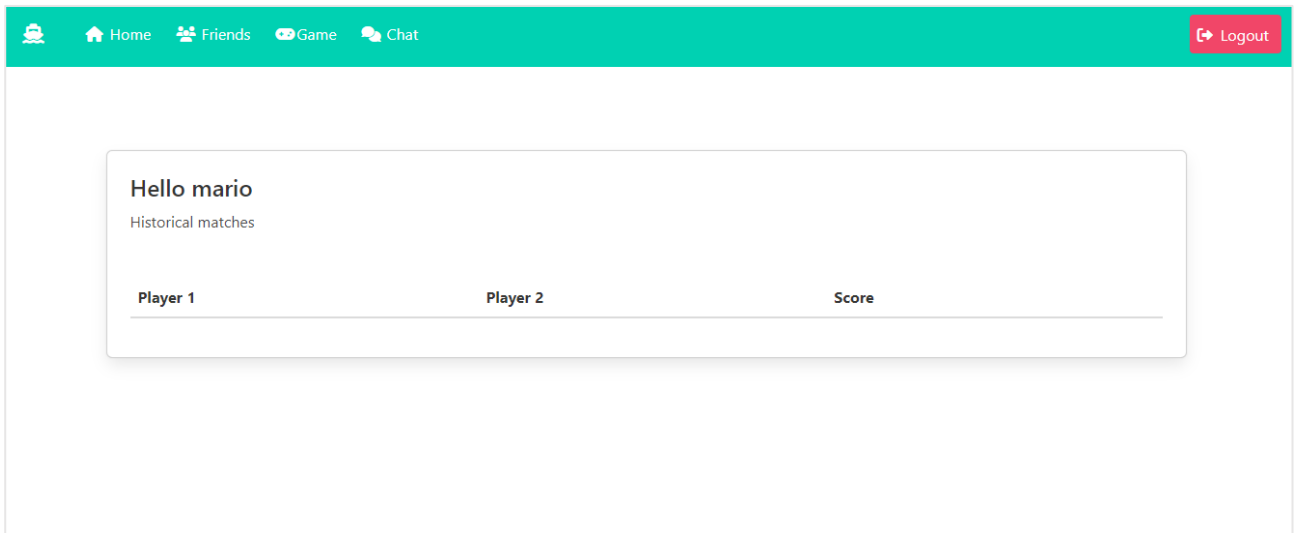
Let's talk about the application's workflow. Let's assume we visit the application for the first time: we are prompted with the login page but we don't know any users credentials. We decide to create an user so we click on the "Not a member?" link.

A login form with a white background and rounded corners, centered on a solid teal background. It features two input fields: 'Username' with a person icon and 'Password' with a key icon. Both fields have a small '1' in a square on the right. Below the fields is a green 'Login' button. At the bottom, there is a blue link labeled 'Not a member?'.

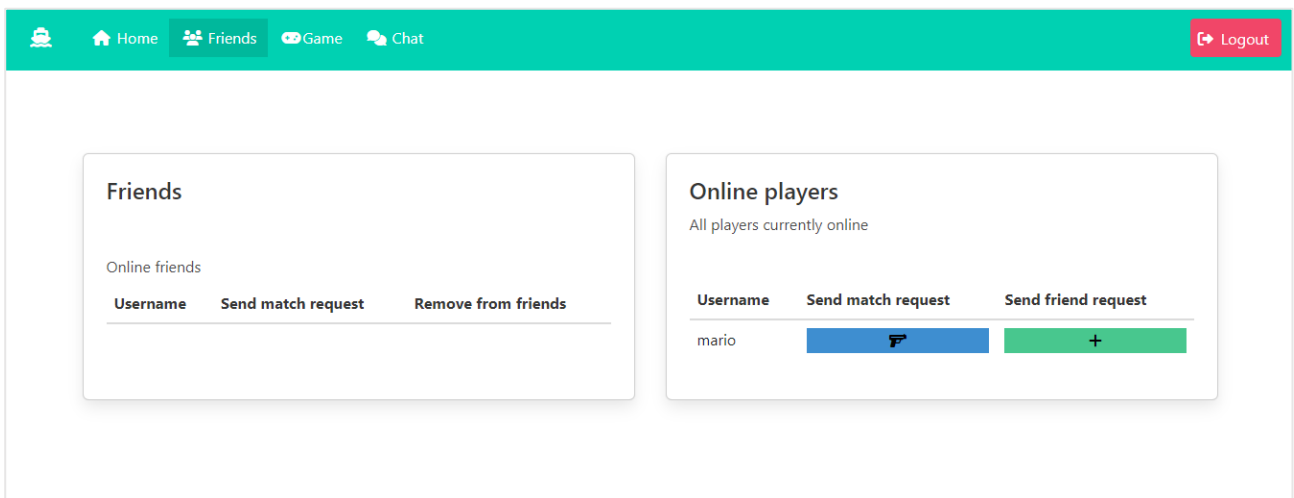
We are now prompted with the registration form where we can insert our information. After that we can click "Register" and we are redirected to the login page. This time we can successfully log into the application.

A registration form with a white background and rounded corners, centered on a solid teal background. It has four input fields: 'Name' (with a person icon), 'Username' (with a person icon), 'Email' (with an envelope icon), and 'Password' (with a key icon). The 'Password' field has a small eye icon on the right. Below the fields is a green 'Register' button. At the bottom, there is a blue link labeled 'Already a member?'.

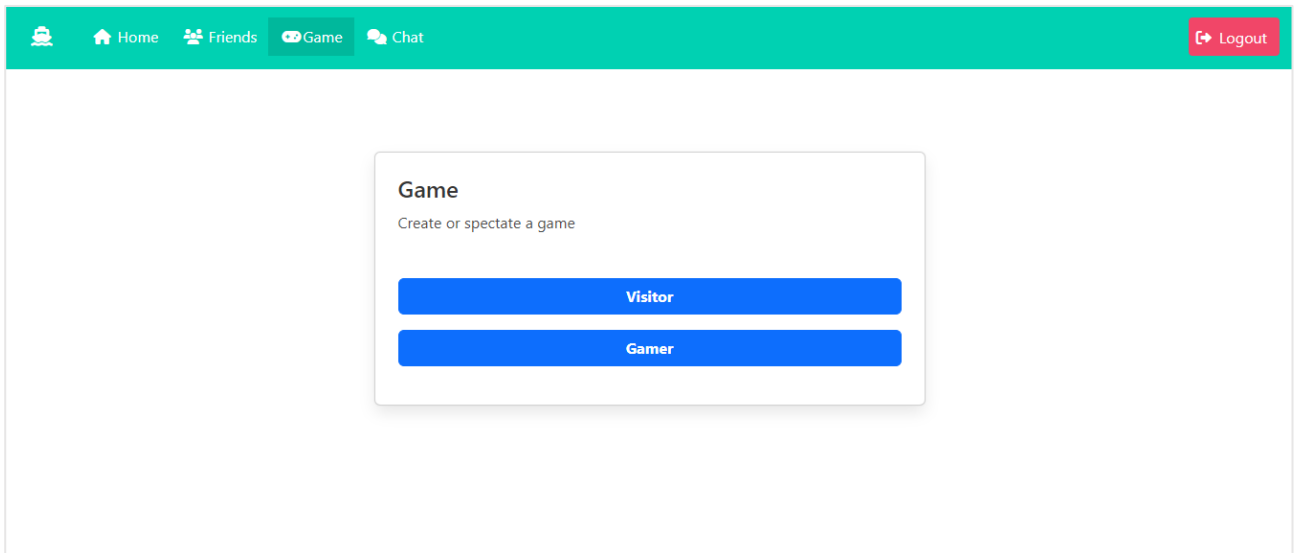
We can see our homepage where we would see our games history.



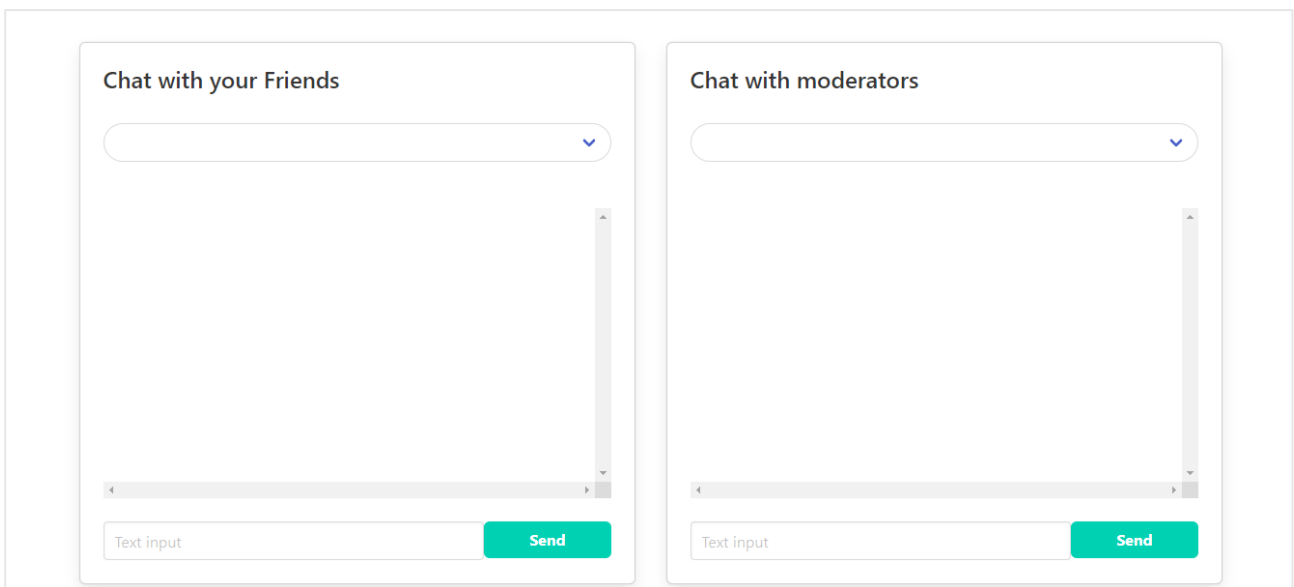
In the "Friends" section we can see some useful information about our friends and other players online. We can send match and friends requests.



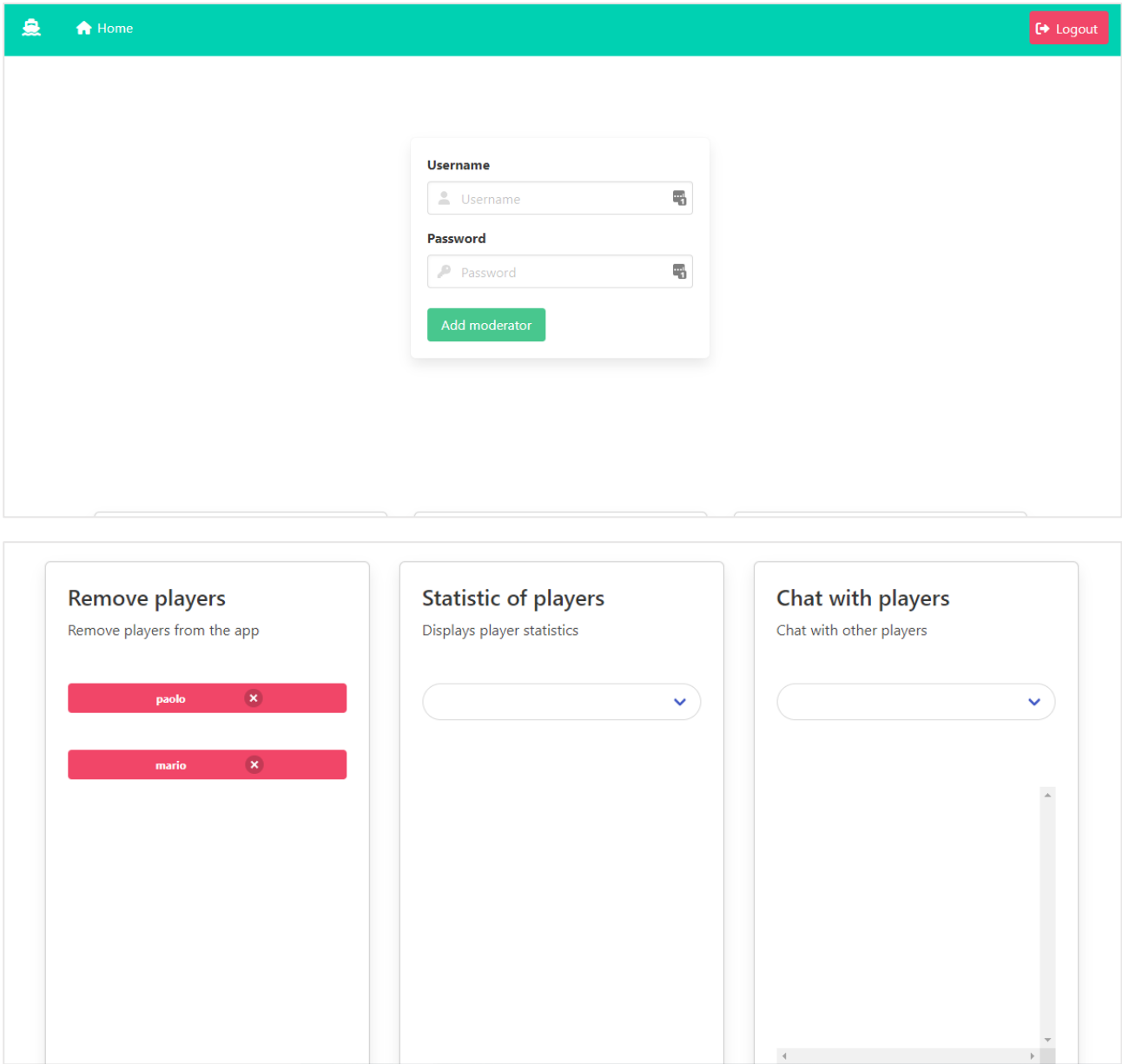
In the “Games” section we can start a new game or view a game already in progress.



In the “Chat” section we can send messages to friends and to moderators.



Here some screenshots of the moderator homepage



Some games screenshots

Ready to sink some battleships?

Battleship

PLAYER mario You SCORE: 1

You

SCORE: 1

PLAYER paolo **SCORE: 0**

COR

Ready to sink some battleships?

Battleship

PLAYER paolo **You** SCORE: 0

You

SCORE: 0

PLAYER mario SCORE: 1

SCO



OOPS! YOU MISSED THIS TIME

Keep trying.