Exam project work

Tecnologie e applicazioni web, a.a. 2020/2021

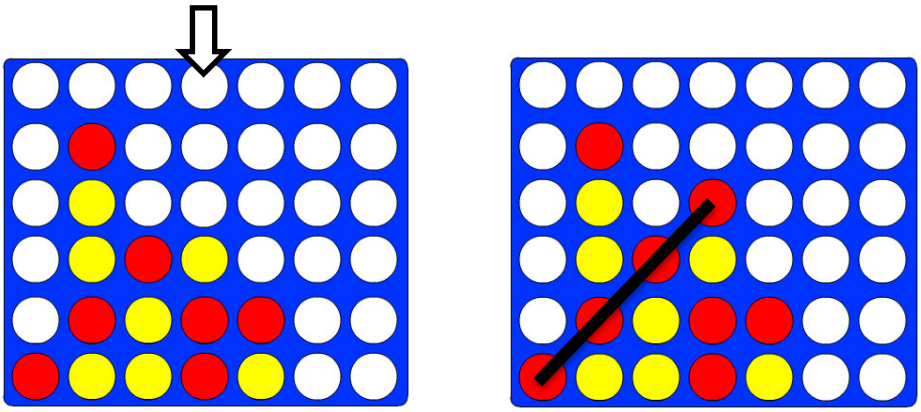
The goal is to develop a full-stack web application, comprising a REST-style API backend and a SPA Angular frontend to let the users play the game of “Connect-4” (See https://en.wikipedia.org/wiki/Connect_Four for the general rules).

A player starts by signing-in to the application. When logged in, she/he can decide to:

- Play against a particular user in the player’s friend list
- Start playing against a random opponent
- Observe a game in progress

In the first case, an invitation is displayed to the challenged player that can decide to accept or refuse the match. In the second case, the system will automatically arrange a match between two waiting players and the game can begin. In the last case, a user can observe a match by choosing it from a list.

Connect-4 is a 2-players turn-based game where the starting player is randomized by the server. In turn, each player will “drop a coloured disk” (red for the first player, yellow for the second) into a grid composed by 7 columns and 6 rows. The first player forming a horizontal, vertical or diagonal line of 4 discs (with the same color) wins the match. During the match, the two players can chat to each other but not with the observers.



The application keeps track of the statistics of each player, including the number of wins and losses, the number of games played, etc.

Users are either standard players or moderators. Moderators can forcefully delete standard players from the system and send messages to any other user. Standard players can only send messages to players in their friend list.

Architecture

The system must comprise:

- A REST-style **backend** webservice, implemented in TypeScript and running on Node.js. The service must use:
    - MongoDB DBMS for data persistence
    - Express.js for routing

- A web **front-end** implemented as a Single Page Application (SPA) using the Angular framework.

Requirements

Your application must (at least) implement the following features:

1. User management
    a. Signup of standard players. A new player can register by choosing a username, password and entering his details (name, surname, etc.). A player must also have an associated avatar image.
    b. Signup of new moderators. Moderators cannot register themselves but must be added by another moderator, specifying a username and a temporary password. At the first login, the new moderator must forcefully set his own credentials (password, name, surname, etc.)
    c. Deletion of standard players from moderators
2. Friend list and chat
    a. A player can add another player in its own friend list. The other player must be notified and accept the request before being added
    b. A player can send a message to another player in its own friend list
3. Gameplay
    a. Once a match is arranged, two players can play the Connect-4 game.
    b. The starting player is randomized.
    c. The color associated to each player is randomized and clearly displayed on the screen
    d. In his turn, a player must choose where to drop the disc by selecting one of the 7 columns (if not full of discs). Then, the opposing player can make his move and the process is repeated until one of the two players wins.
    e. At any time, the two players can chat to each other
    f. Any other user can observe the game in progress and the chat. Observers can only chat among themselves
4. Match arrangement
    a. The system can automatically arrange a match between two players. A user can enter a “wait state”, expressing his will to play against a random opponent. If at least two users are in “wait state”, random pairs are formed, and the corresponding games are started. When matching players, the system should consider the player’s statistics to avoid creating unbalanced

matches. Note that players matched in this way are not required to be friends. Chat is possible in this case, but only during the game.
    b. A player can invite a user in his friend list to play a game. If the other player accepts, the game is started.
5. Statistics
    a. A standard player can see his own statistics and of any other player in his friend list
    b. A moderator can see the statistics of any other user

You are welcome to creatively enrich the project with features not expressly indicated. This does not imply to get honors (Lode) but concur to define a positive grade.

Submission

Projects must be submitted via Moodle at the following page:
https://moodle.unive.it/mod/assign/view.php?id=342913&forceview=1

If working on a group, each group member must submit the same file named:
`<group_name>.zip`.

The package must contain:

- The report of each student, in PDF format, named `<student_firstname>_<student_surname>_<matriculation_number>.pdf`
- A README.txt file with instructions on how to run the entire application. For example, you can briefly summarize the shell commands to compile the sources, start backend and frontend, etc.
- All the application source code, possibly divided in backend and frontend.

NOTES:

- Do not submit any external library. In other words, remember to delete all the `node_modules` directories before submission.
- Students are responsible for group creation. Moodle will not help nor enforce group submission. Individual students can simply use their surname as group name.

Report

Together with the web application, you must submit a report describing the system architecture, the software components used the way in which they contribute to achieving the required functionalities. The report is **strictly individual** and will serve as a basis for the oral discussion of the project.

The report must contain:

- A description of the system **architecture**, listing all the different components and their relationship (i.e. How they work together to implement the application requirements)
- A description of the **data model**, including the collections and the structure of the documents inside each collection
- A precise description of the **REST APIs**. Such description must contain the list of endpoints together with their parameters and what data is exchanged (give examples in JSON format)
- A description of how the user **authentication** is managed, with the associated workflow
- A description of the Angular frontend, listing the of **components, services, routes**, etc.
- Some examples, possibly with **screenshots**, showing the typical application workflow.

Q&A

For any question about the project work (or the course in general) don't hesitate to contact the teacher at:
filippo.bergamasco@unive.it

We can arrange a meeting if needed.

Additional information about the exam is available at:
https://moodle.unive.it/mod/page/view.php?id=283851

Filippo Bergamasco,
Tecnologie e Applicazioni Web, a.a. 2020/2021