Web Architectures
assignment 3

Memory Game - Web version

Sebastiano Chiari - 220527
sebastiano.chiari@studenti.unitn.it

October 31, 2021

1 Introduction

The aim of this assignment is to develop a web version of the memory game, where the user has to discover pair of equal cards, with minimal server side and `HTML + JS + XHR` for the game page.

2 Implementation

2.1 Presentation layer

The **login page** shows a text that greets the new user and a form where the user can put its own username. Validation on client side is performed, so if the user does not input the username field the form will not be submitted. Otherwise, the form sends a POST request to the `LoginServlet` to perform authentication.

The **homepage** is displayed after the user completes the authentication process. This page greets the user by showing its name, shows the ranking (the five best games, with score and the username of the player) and a button "Play game" to start the game. If no game has been played yet, the ranking says "Empty – no game played yet".

2.1.1 Game page

The **game page** is where the user can play it memory game. It presents 16 covered cards in a 4x4 matrix. Each card has an unknown numeric value (1 to 8), so for each value two cards are present. Each card has a unique ID based upon its position and has an `onclick` property, that triggers a javascript function. The game page also shows the remaining number of attempts (each user has 4 attempts to score the best result) and a section where it is record the current points of the game.

All the memory game logic is managed through javascript (`js/memory.js`). The attempts and the points are kept client side, while the value of the cards is known only server side.

When the user clicks on a card, it becomes uncovered and not clickable any more. The script sends an AJAX request to the MemoryServlet to retrieved the value of the card based on its id: a JSON file is returned with the respective value. Then, the card with the given value is shown (`updateCard(id, value)` function).

When the user clicks on a second card, also this card becomes uncovered and unclickable. The same AJAX request is sent in order to retrieve the second card value and the respective card is shown according to the JSON.

At this point, the number of available attempts is decreased by 1. Then, there can be two scenarios (managed by the `updateGameStatus()` function:

- if the **value** of the two **cards** is the **same**, those cards remain uncovered (and unclickable) and the score is increased with an amount established according to the following formula: *value of the card * 2*

- if the **value** of the two **cards** is **different**, the users score is decreased by 1 and after 1 second the two cards are covered, and become clickable again (the two cards get the `onclick` property restored). The delay is manages through a `Promise`.

When the number of available attempts becomes 0, the `endGame()` function is called: all cards become unclickable, a label "Game over" appears replacing the number of attemps and after 1 second the user is redirected (through an AJAX `POST` request sent to the `IndexServlet`) to the initial page with the updated ranking.

2.2 Servlets

2.2.1 `LoginServlet`

The `LoginServlet` manages the authentication process of the user.

The `doGet()` method redirects the user to the login page.

The `doPost()` method gets the username parameter from the request and checks if it corresponds to any existing user: if not, the provided username is added to the list of users (new users though are not persistent). Then, a new session is established and the user gets redirected to the index page.

2.2.2 `IndexServlet`

The `IndexServlet` manages the homepage, where the ranking is displayed.

The `init()` method create an empty `ArrayList` that will collect all the matches, recording the pair username and points in a structure `Pair<Integer, String>`. More, a `emptyScoreboard` variable is set to `true` in order to display the "no match played" scenario.

The `doGet()` method redirect the user to the homepage.

The `doPost()` method is called once a game is finished and retrieves the score and username parameters from the AJAX request. A new `Pair` is build with the given parameters and it is added to the results variable. Then, this array is sorted and reordered in order to retrieve the best 5 games. This pairs are copied to another arraylist, which is set as a ServletContext variable in order to be displayed in the JSP page.

2.2.3 `MemoryServlet`

The `MemoryServlet` handles the server side of the memory game.

The `doGet()` method creates the list with the cards' values. Given the `initParam environment` (which can be `test` or `production`), the arraylist is respectively sorted in ascending order or shuffled. Then, the list is saved as a session attribute (in order to enable concurrent games) and the user is redirected to the game page.

The `doPost()` method is called by the client through an AJAX request and it receives as a parameter the ID of the clicked card. This process retrieves the cards variable from the

session scope and returns as a JSON object to the client the value of the card given the ID.

2.3 Filters

2.3.1 `AuthenticationFilter`

This filter handles the access to all the pages of the web application: to access any page, users must authenticate themselves, while unauthenticated users must be directed to the login page.

In the `init()` method, the filter imports all the users from the configuration file (in .txt format): all users are added to a list and this collection is set as a `ServletContext` attribute. The path to the configuration file is given to the filter as `initParams`, specifically as `usersFilePath` in the configuration of the filter.

In the `doFilter()` method, this filter checks if a session exists and if the session has a user attribute (this means that the current user is logged in or has a previous session open). If the users is logged in and the request coincides with the login page, then, the user is redirected to the homepage. If the user is not logged in, the filter will redirect the user to the login page. Any other case will forward the user's request.

3 Comments and notes

Since most of the game logic (such as points and attempts) are handled client side, also in this case ensuring security during the game can be an issue. Variables can be changed but, most important, the AJAX requests to the servlets can be edited.

4 Results

In the following section, there are screenshots of the running application with descriptions that documents the various steps and scenarios.

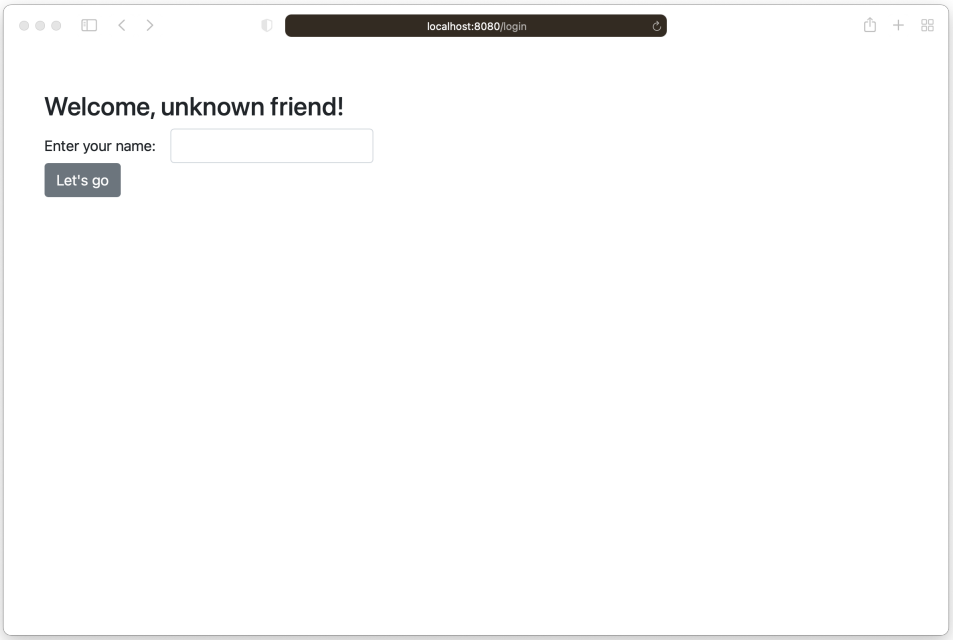

Figure 1: Login page

Figure 2: Welcome page with no played game

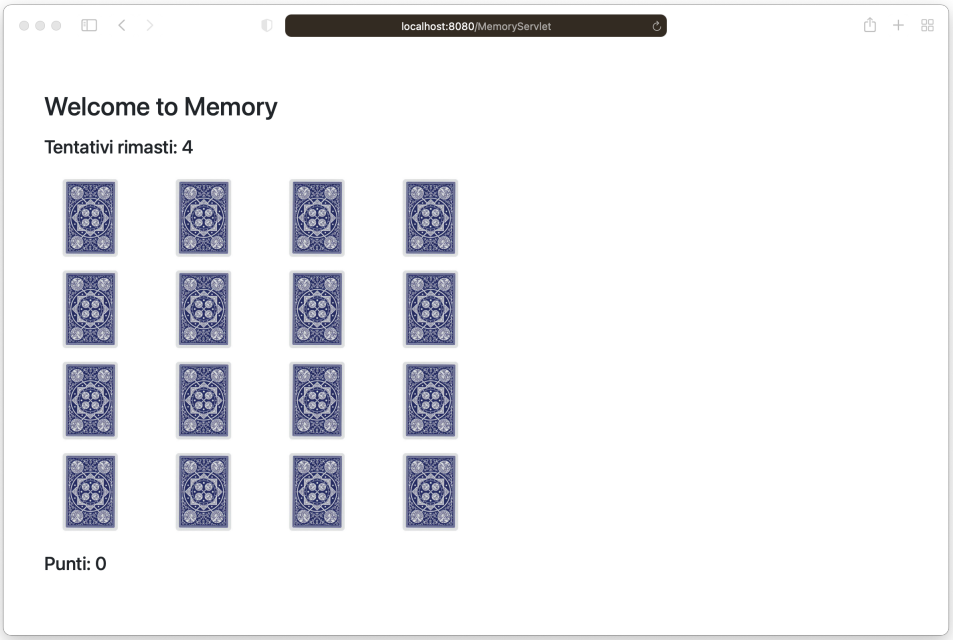

Figure 3: Memory game page

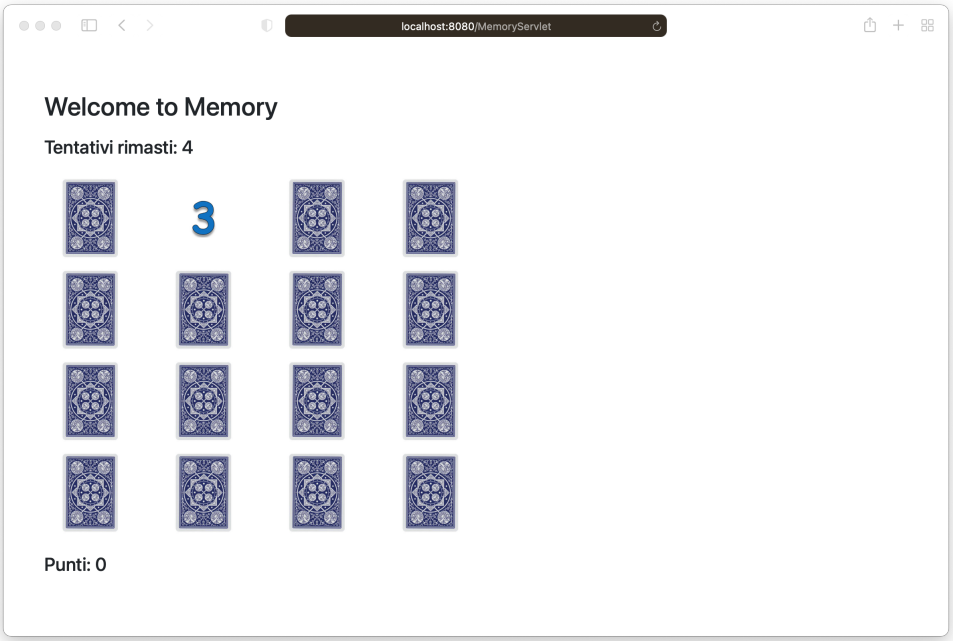

Figure 4: Player turns the first card

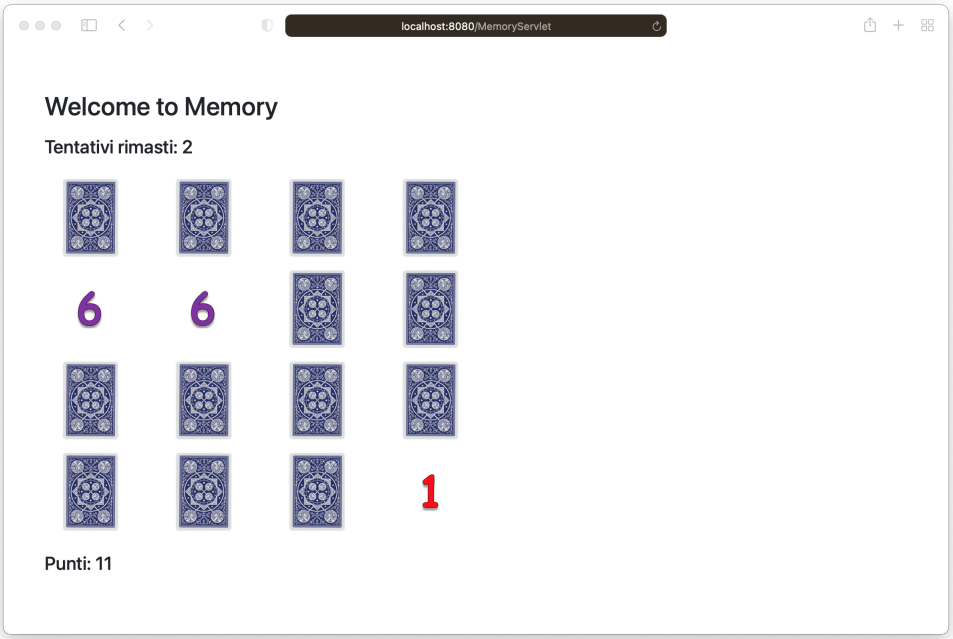

Figure 5: Player gets a couple in and turns the next card on the following attempt

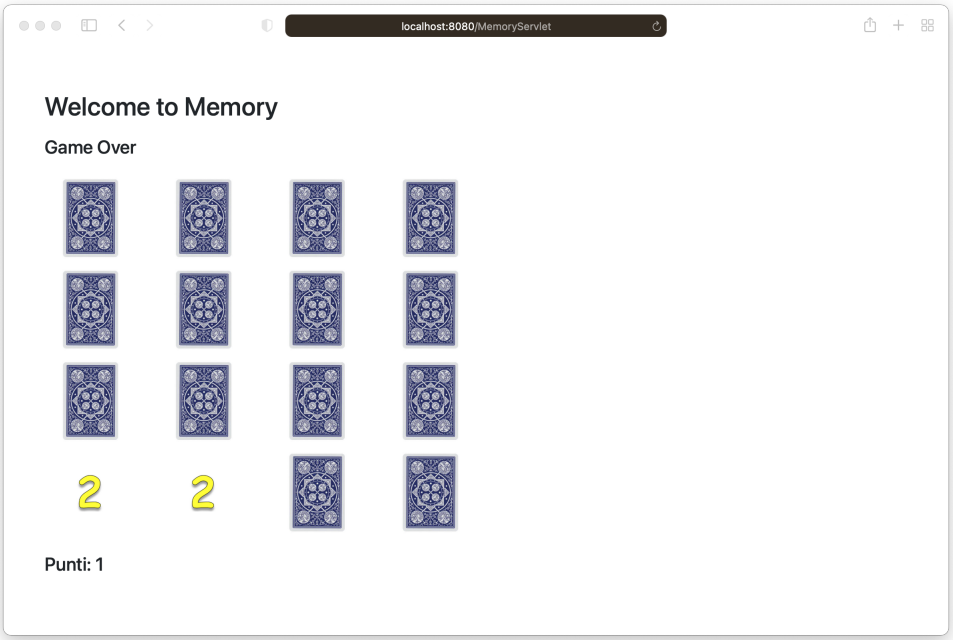

Figure 6: Gameover - the user is then redirected to the home page

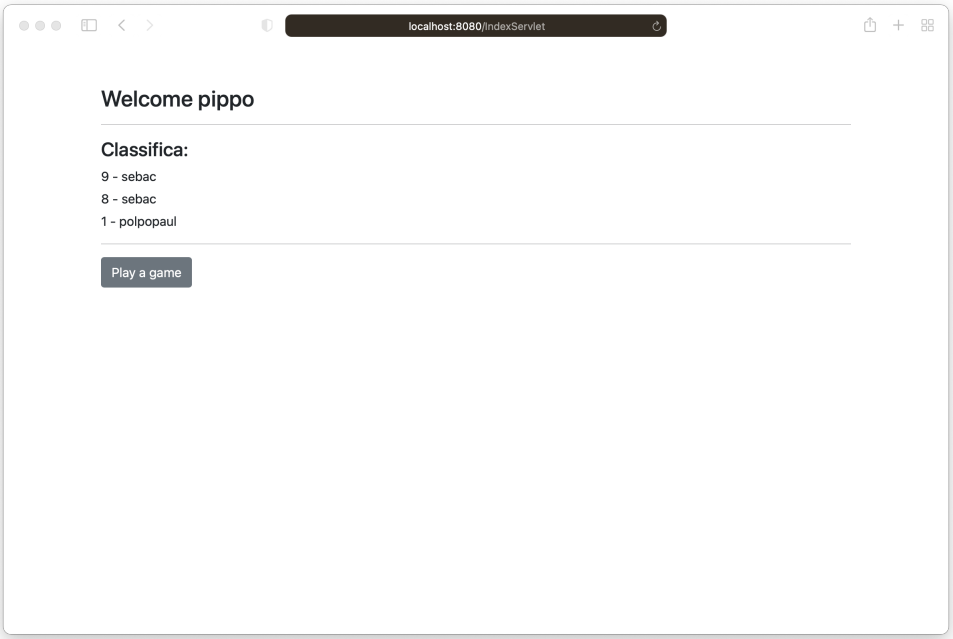

Figure 7: Another user logs in and can see the top 5 matches