Mastermind

Create a web application to play the game `Mastermind`. You can find more information about this game following this URL: (https://en.wikipedia.org/wiki/Mastermind_(board_game)

Mastermind-Basics

In `Mastermind` usually two human players compete against each other. In our simplified version, the computer generates a random color pattern (using 6 different colors and 4 slots) and you have to try to guess this color pattern (the right colors in the right order).

For example, the computer generated the following pattern:

```
yellow, brown, blue, green
```

Your first attempt to guess the color pattern could be:

```
brown, red, blue, yellow
```

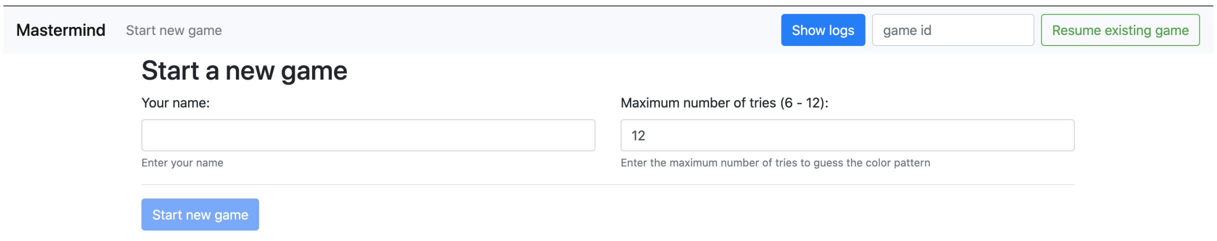
In this case, you have guessed three correct colors: `brown`, `blue` and `yellow`

The computer will tell you `the number of correct colors`, but not their position within the pattern. The computer will also tell you `one` correct slot because `blue` is at position 3, but not which color it is.

It is up to you now to guess the correct color pattern within a defined number of attempts (between 6 and 12).

Task 1

Create a user interface similar to the this screenshot (using Bootstrap):



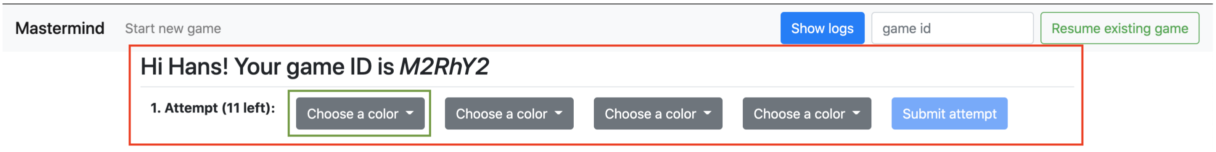
- Use the Bootstap `nav` component for the header.
- Make sure the `Maximum number of tries` cannot be lower then 6 and higher then 12.
- As soon as a valid name is entered, the `Start new game` button shall become available.

- After clicking the `Start new game` button, a `post` request shall be sent to the `backend`. As a result, a new `game id` shall be returned (consisting of `6 random characters -- upper and lower case`).
- Implement this functionality in a separate Angular component.

> Make sure to implement the whole game logic in the backend using services!

Task 2

Create a Angular component `gameboard` (red border below) containing a sub-component `attempt` (green border below):

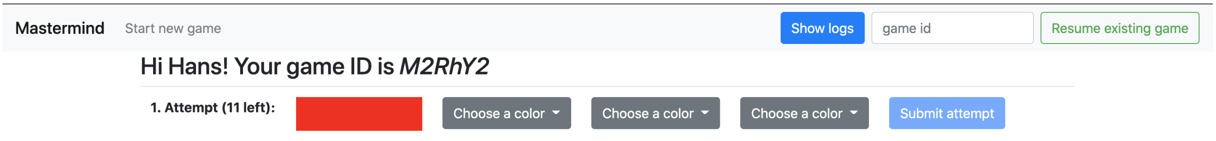


- The user can select a color pattern using `Bootstap Dropdowns`. Implement this functionality in the Angular component `attempt`:

```
<app-attempt [availableColors]="availableColors"
             [selectedColor]="attempt.colors[0]"
             (colorChosen)="colorChosen(1, $event)">
</app-attempt>
```

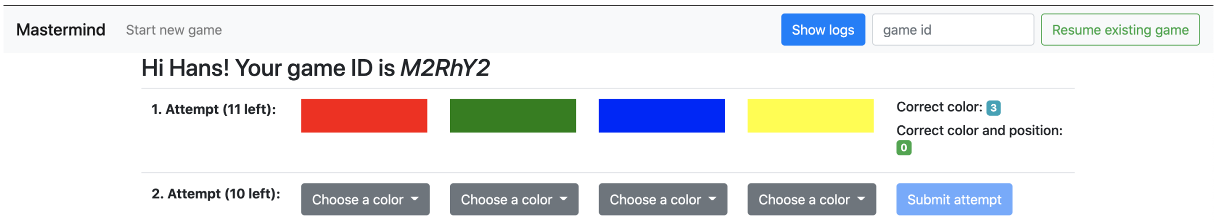
- `availableColors` … contains a string-Array with all available color; retrieve this Array from the backend. Do not hardcode it!
- `selectedColor` … when restoring a game (details see later) use this attribute to pass the color from the previous attempt
- `colorChosen` … this event is triggered when the user has selected a color from the dropdown

As soon as the user has chosen a color, the selected color shall be displayed and cannot be changed any longer:



If all four colors are chosen, the `Submit attempt` button shall become available. If this button is pushed, a `post` request containing the guessed color pattern shall be sent to the backend. The backend shall calculate the number of correct colors and correct slots and send this info back to the frontend.

The `Submit attempt` button shall be hidden and the following information should be displayed instead:



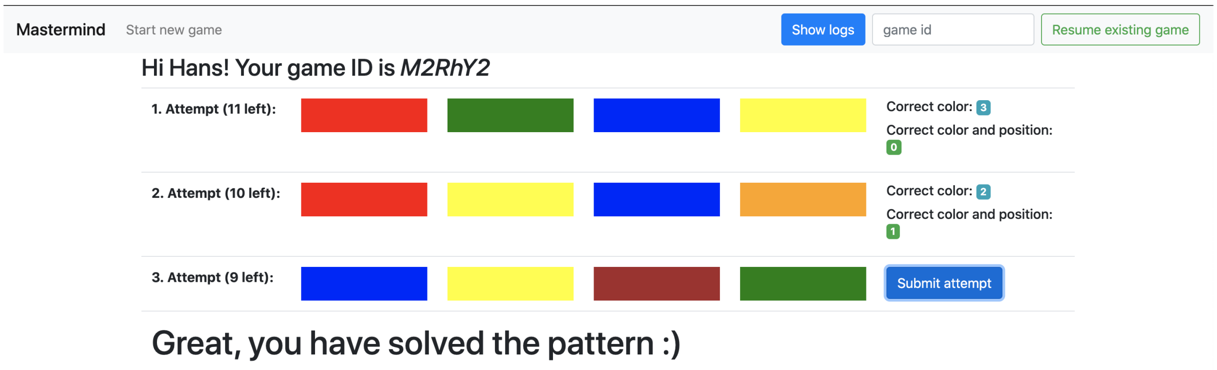
Hi Hans! Your game ID is *M2RhY2*

1. Attempt (11 left): Correct color: 3 Correct color and position: 0

2. Attempt (10 left):

As soon as no more attempts are left, the game is over and the message `Sorry, no more tries left` :(shall be displayed.

If the user guessed the correct color pattern, the following text should be shown:



Hi Hans! Your game ID is *M2RhY2*

1. Attempt (11 left): Correct color: 3 Correct color and position: 0

2. Attempt (10 left): Correct color: 2 Correct color and position: 1

3. Attempt (9 left):

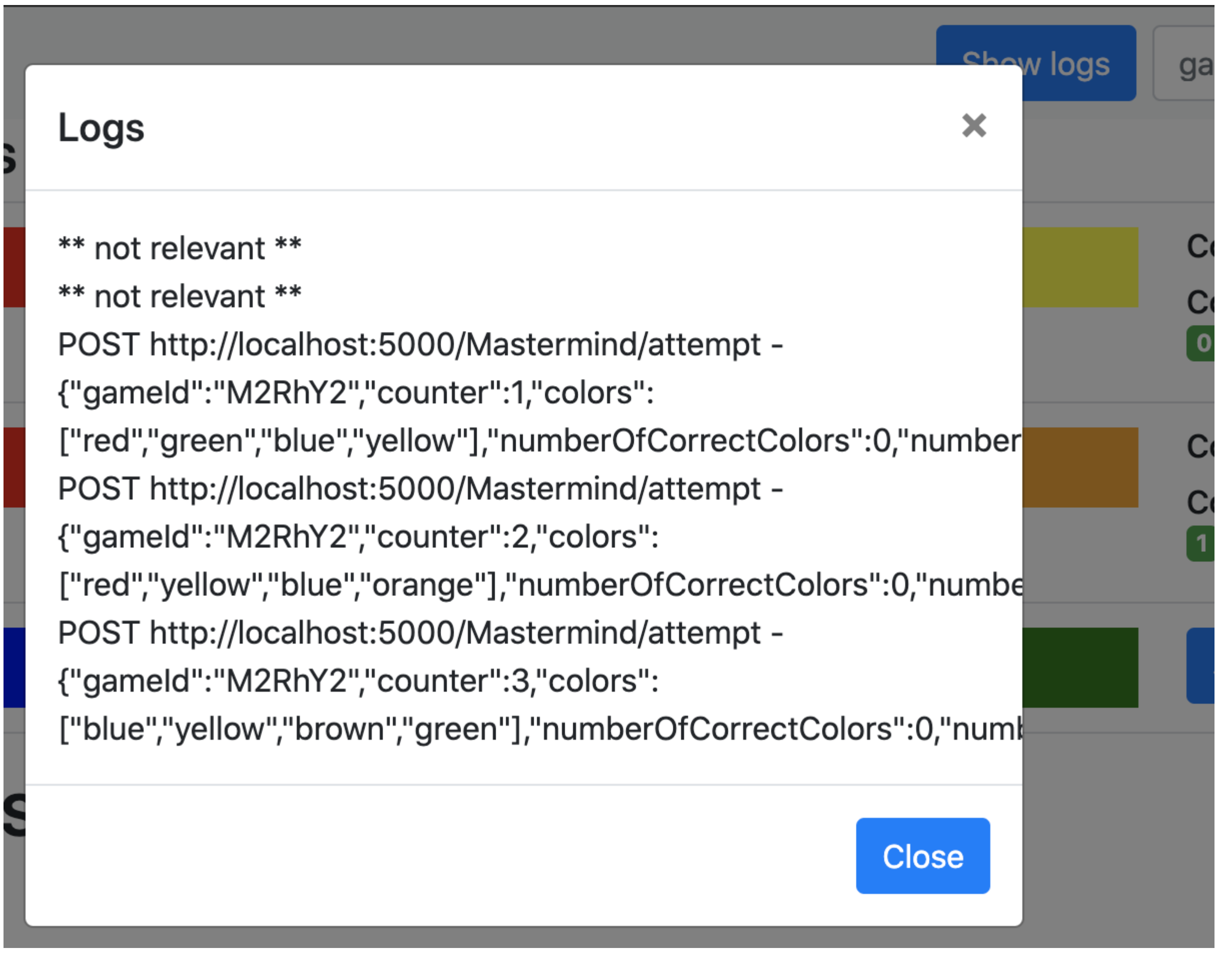
Great, you have solved the pattern :)

Task 4

Create an `interceptor` to log the outgoing HTTP requests. Create a `Subject<string>` in the `MasterMind-Service` and notify the subscribers if a new HTTP request was intercepted containing request method, target and body.

Create a new Angular component `logs` which listens to those notifications. Track all received notifications in a string-Array and display this Array in the component body.

As soon as the used clicks the `Show logs` button, an Bootstrap `modal dialog` shall pop up (see https://getbootstrap.com/docs/4.5/components/modal/). Display the `logs component` in the body of this dialog:



```
** not relevant **
** not relevant **
POST http://localhost:5000/Mastermind/attempt -
{"gameId":"M2RhY2","counter":1,"colors":
["red","green","blue","yellow"],"numberOfCorrectColors":0,"number
POST http://localhost:5000/Mastermind/attempt -
{"gameId":"M2RhY2","counter":2,"colors":
["red","yellow","blue","orange"],"numberOfCorrectColors":0,"numbe
POST http://localhost:5000/Mastermind/attempt -
{"gameId":"M2RhY2","counter":3,"colors":
["blue","yellow","brown","green"],"numberOfCorrectColors":0,"numb
```

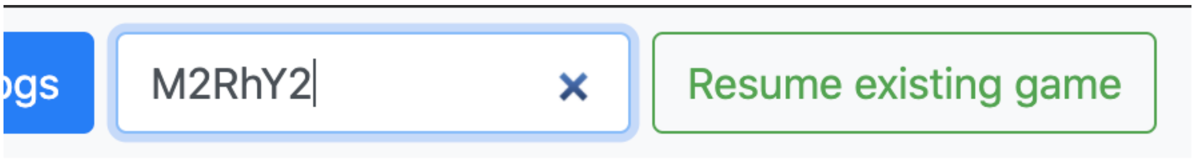
Create a `pipe` to filter irrelevant HTTP requests. The pipe shall be used like this:

```
<div *ngFor="let log of logs">{{log | requestFilter:'attempt'}}</div>
```

Irrelevant requests shall be replaced with the text `** not relevant **`.

Optional Task 5

Enable the user to continue the game at a later time by using the `game id`. When closing the browser window, relaunching the app and entering the `game id` in the field on the top right corner, a previous game shall be resumed at the exact same position as the user has left before:


ogs
M2RhY2
Resume existing game