SEW IV/V - Assignment: *Survey McSurveyface*

Objective

Create an *Angular* application for a simple quiz game.

Things To Learn

- Utilizing *Panache*.
- Creating interface using *Angular Material*.
- Establishing direct communication using *WebSockets*.

Submission Guidelines

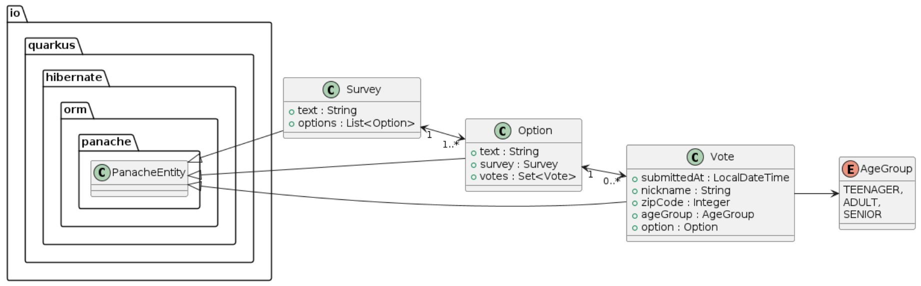
- Your implemented solution as ***zipped*** *IntelliJ*-project.

Task

The objective of this assignment is to develop a complete *backend-frontend*-application for a simple survey tool. The tool will allow users to vote on polls and store the results. A key feature is the real-time automatic refresh of survey data in the frontend, achieved using `WebSocket`s.

Backend

Create a *Quarkus* backend using *Panache*. Your *data model* could look something like this:



Implement endpoints to ensure that at a minimum, the following functionality is provided:

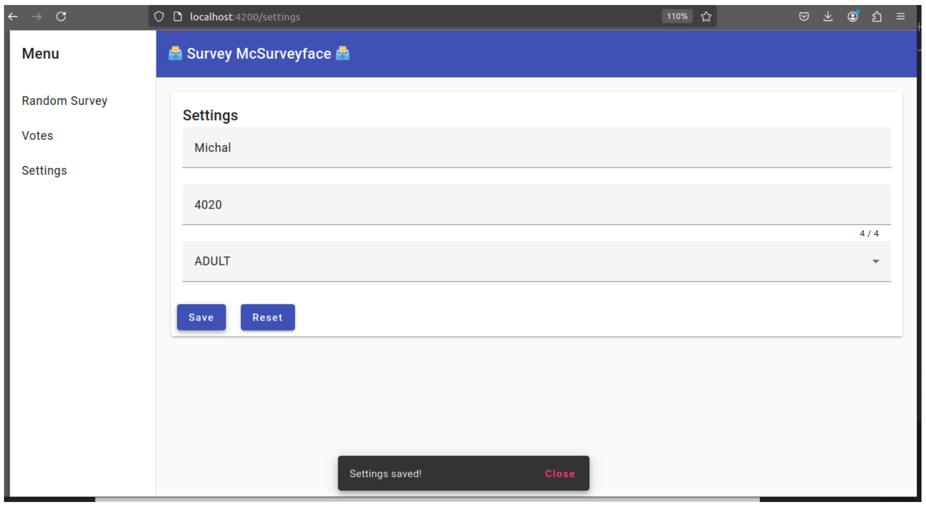
- Sending a `POST` request to `/api/surveys` should create a survey along with its options (refer to the provided `.http` file for specifics).
- `GET`ting from `/api/surveys/random` should return a randomly selected survey.
- By `POST`ing to `api/votes` should add a `Vote` to the appropriate option (again, see the supplied `.http`-file for reference).

- `GET`ting from `/api/votes/list` should return a complete list of all votes, sorted descendingly by their submission date.
  - Hint: Using a *DTO* might come in handy here - take a look at the data you need in the frontend!

Frontend

For the frontend, set up an *Angular* project and install *Angular Material*, to be able to use its useful *schematics*. Set up a navigation bar using the `navigation` schematic and configure *routing* to the following pages:

`/settings`



Settings

Michal

4020

4 / 4

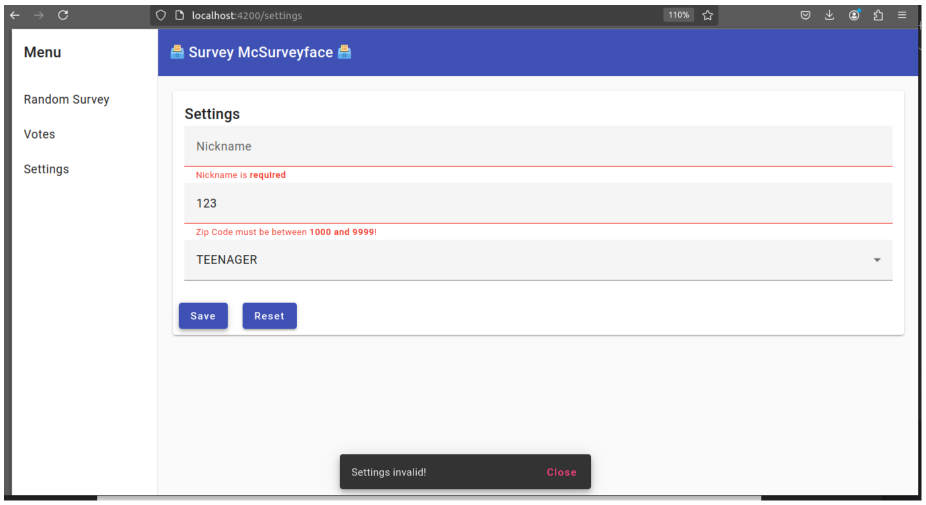
ADULT

Settings saved!

Use the `address-form` schematic to create simple page for user settings and a `Service` for (re-)storing them:

- A nickname, `Anonymous` by default.
- A zip code, `4060` by default.
  - Create a custom validator for Austrian zip codes (1000-9999)!
- An age group, `TEENAGER` by default.
  - Use an `enum` to store the possible values.

All values are *required* - use *Material's Snack Bar* to display a notification, when a value is missing or invalid!



Settings

Nickname

Nickname is **required**

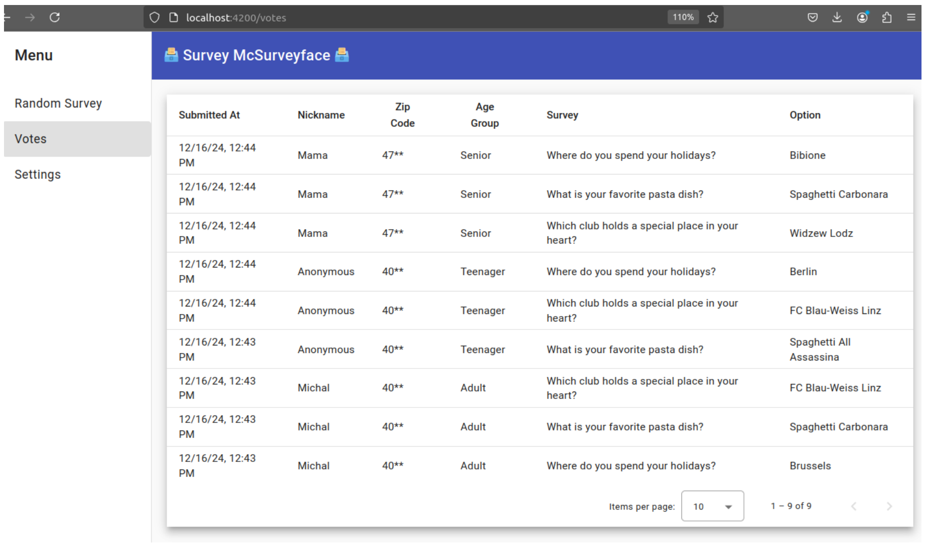
123

Zip Code must be between **1000 and 9999**!

TEENAGER

Settings invalid!

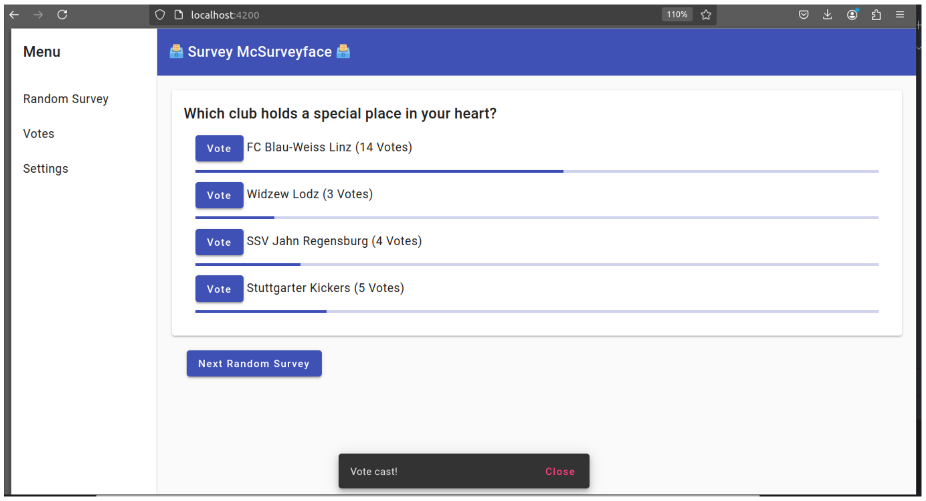
`/votes`



Submitted At	Nickname	Zip Code	Age Group	Survey	Option
12/16/24, 12:44 PM	Mama	47**	Senior	Where do you spend your holidays?	Bibione
12/16/24, 12:44 PM	Mama	47**	Senior	What is your favorite pasta dish?	Spaghetti Carbonara
12/16/24, 12:44 PM	Mama	47**	Senior	Which club holds a special place in your heart?	Widzew Lodz
12/16/24, 12:44 PM	Anonymous	40**	Teenager	Where do you spend your holidays?	Berlin
12/16/24, 12:44 PM	Anonymous	40**	Teenager	Which club holds a special place in your heart?	FC Blau-Weiss Linz
12/16/24, 12:43 PM	Anonymous	40**	Teenager	What is your favorite pasta dish?	Spaghetti All Assassina
12/16/24, 12:43 PM	Michal	40**	Adult	Which club holds a special place in your heart?	FC Blau-Weiss Linz
12/16/24, 12:43 PM	Michal	40**	Adult	What is your favorite pasta dish?	Spaghetti Carbonara
12/16/24, 12:43 PM	Michal	40**	Adult	Where do you spend your holidays?	Brussels

Use the `table` schematic to create a view of all submitted votes, sorted by their date. Utilize a *date pipe* for a neat representation of the submission date and censor the last two digits of the zip code for privacy reasons.

/ (*Home*)



Which club holds a special place in your heart?

FC Blau-Weiss Linz (14 Votes)

Widzew Lodz (3 Votes)

SSV Jahn Regensburg (4 Votes)

Stuttgarter Kickers (5 Votes)

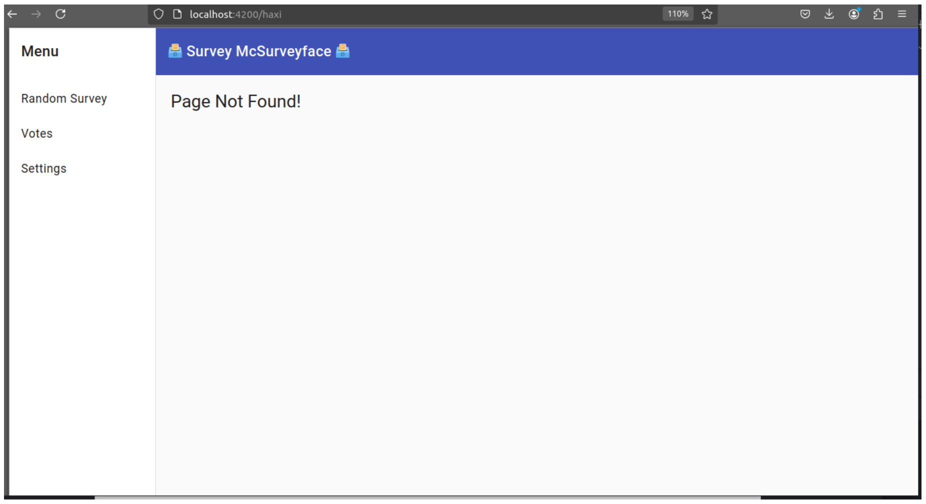
Vote cast!

The home page should display a random survey and a button to load another one.

Each option should display its text, the absolute number of votes and a visualization - e.g. as an *Angular Material progress bar* - depicting its percentage share of the total votes. Additionally, a button should allow the user to cast a vote, which also triggers a *snack bar* notification.

***Not-Found*-Page**

Don't forget to include a simple *not-found*-page for invalid paths:



Page Not Found!

`WebSocket`

To keep the displayed survey up to date at all times, you will use `WebSockets` to broadcast a message every time a new vote is cast. Here's how you can approach this:

1. Set up a `WebSocket` connection between the backend and frontend (in a separate `Service`). This will allow the backend to push updates to all connected clients in real time.
2. When a new vote is cast, the backend should *broadcast* a message to all connected clients via the WebSocket.
3. Decide what to broadcast:
   - You can broadcast the details of the individual vote. This minimizes the amount of data sent but requires the frontend to update the survey state accordingly.
   - Alternatively, you can broadcast the entire survey object with the updated votes. This simplifies frontend handling but increases the size of the message being transmitted.

N.B.: Finding the right balance is key. Consider factors like how frequently votes are cast, the size of the survey object, and the effort required for frontend updates.

4. On receiving the `WebSocket` message, the frontend should update the survey display dynamically. If you broadcast single votes, ensure the frontend correctly updates only the affected option. If you broadcast the full survey, simply replace the existing survey data with the updated object, if the *IDs* match.
5. Verify that all connected clients see the updated survey data immediately after a vote is cast, ensuring a seamless and responsive user experience.

By implementing `WebSockets` in this way, your survey tool will remain synchronized across all users in real time.