Software Architectures

Assignment 2: MVC using Scala Play

Camilo Velázquez-Rodríguez
camilo.ernesto.velazquez.rodriguez@vub.be
F.10.725

Assignment

The goal of this assignment is to implement a Model-View-Controller (MVC) application using Scala Play.

Deadline: January 8th 2023 by 23:59. The deadline is fixed and will not be extended for any reason.

Deliverables A report and source code. The report (in English) explains your solution to the problem and the main components used in your implementation. Please use simple diagrams to illustrate the architecture of your solution.

The report should be handed in as a single PDF file.
The file should follow the naming schema `FirstName-LastName-SA2.pdf`, for example: `Camilo-Velazquez-SA2.pdf`.

The source code is your implementation of the project zipped in a single file. Do not include files or folders resulting from the compilation process of your project, e.g., target/ or *.class. Submit the *report* and *source code* as a single zip file on the Software Architectures course page in Canvas, by clicking on *Assignments > Assignment 2*.

Plagiarism Note that copying – whether from previous years, from other students, or from the internet – will not be tolerated, and will be reported to the dean as plagiarism, who will decide upon an appropriate disciplinary action (e.g., expulsion or exclusion from the examination session).

Grading Your solution will be graded and can become the subject of an additional defense upon the lecturer's request.

Problem Description

For this assignment, you will need to implement a question and answer (Q&A) web application (similar to Stack Overflow) where users can post their coding questions. Other users can see, answer, like, dislike or comment on the posted entries.

A post in the application is a combination of text and code. The implementation of the application should follow the Model-View-Controller pattern. The mockup in Figure 1 is a motivational example for the frontpage of your website.

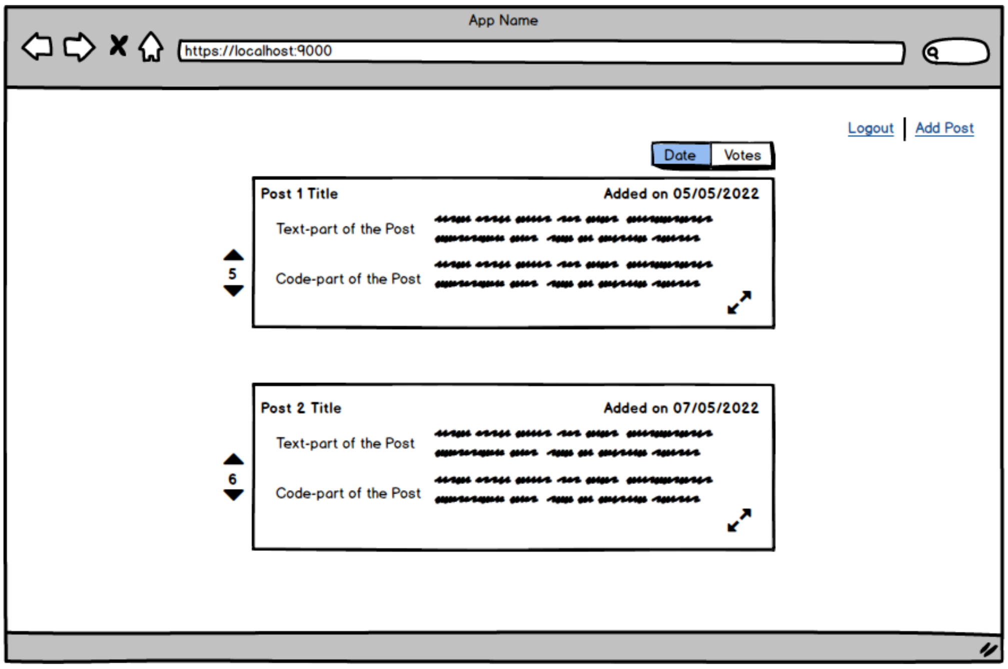

Figure 1: Frontpage of the website.

As shown, you will have to design the layout of the website so that published posts are displayed in the center of the screen. The number of votes for each post is shown on the left, next to each of them. Posts can be sorted by their date of upload or by their number of votes. In Figure 1, the two posts are sorted in chronological order by their date. An expand button in the bottom-right corner of each post enables seeing more details about one specific post.

All users (logged-in or not) can see the shared posts. Only logged-in users can leave comments or modify the posts' votes, as well as add new posts and answers to already-posted content. A logged-in user should be able to add comments to a post on the details page of the post accessible via the expand button. The comments should only be text, and they also have a score that reflects their quality. A possible details page of a post could look like the mockup in Figure 2.

Posts should be added by clicking the "`Add Post`" link in the top-right corner of the web page. This reveals a form with the fields shown in the left sub-figure of Figure 3.

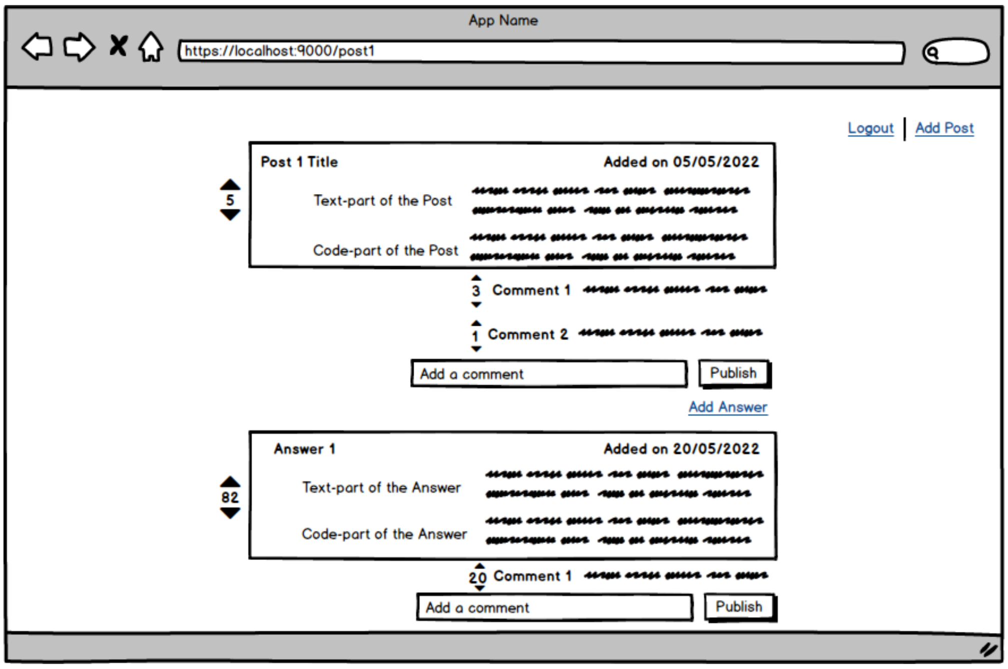

Figure 2: Details page of a post with its comments and answers.

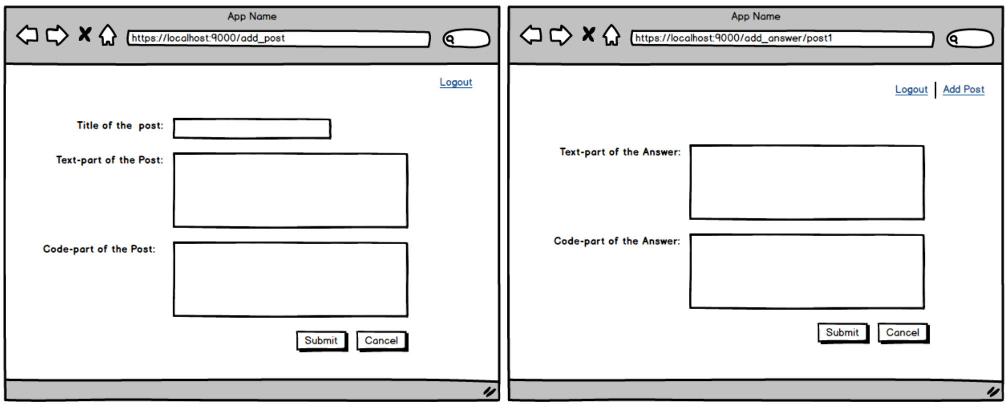

Figure 3: Left: Adding posts. Right: Adding answers to a question in a post.

Here, mandatory fields should be filled-in with the information related to a post. Once the information is filled, users can choose to submit the post or to cancel it. The date of the post should be taken automatically from the server where the website has been deployed. Similarly to the post addition form, the right sub-figure of Figure 3 shows a form for adding answers. Once again, required fields should be filled-in before an answer can be submitted to a post. Answers should be linked to the post from which the form was accessed (i.e., please do note the URL of the mockup in this case).

The functionalities “`Add Comment`”, “`Add Post`”, “`Add Answer`”, and “`Vote`”, should be inaccessible to logged-out users. You are also required to implement a page where unregistered users can sign up.

As a final recommendation, you can use any public template for your web page design (e.g., Twitter's Bootstrap). Do not use a database for your project, just keep the data in memory or store them in a JSON file.

The above web application should be implemented using the Model-View-Controller (MVC) pattern with the Play framework in Scala 2. In addition to the general functionality, you will be evaluated according to how well you implemented the model, views and controllers. You should take into account issues such as data validation and error handling. Motivate your design decisions in the report as per the problem description and illustrate your approach using concepts from Scala Play.

More information on Scala Play

- `https://www.playframework.com/`