Software Architectures

Assignment 2: Scala Play

Assistants: Camilo Velazquez, Kennedy Kambona
Email: {cavelazq, kkambona}@vub.be
Office: 10F725,10F732

For this assignment, you need to implement an MVC application using Scala Play.

Assignment

For this assignment you will implement and report on your original solution to an architectural problem. You will find the input file needed for this assignment in *Assignments > Assignment 2*.

Deadline: January 12^{th} 2020 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. When possible, use simple diagrams to illustrate.

The report should be handed in as a single PDF file of no more than 3 pages (excluding diagrams/screenshots/code). 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 or exported from IntelliJ.
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*.

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 appropriate disciplinary action(e.g. expulsion or exclusion from the examination session). If you use any other resources besides those provided in the lectures and in this document, remember to cite them in your report.

Grading Your solution will be graded and can become subject of an additional defense upon your or the assistants' request.

Problem Description

For the second assignment you will implement a blog-like website with different functionalities detailed in the following description:

You are consulted to develop the website of an organisation in the form of a blog. The website should have an interface with posts, comments and votes for the posts. The following mockup could be a motivational example for the front page of your website:

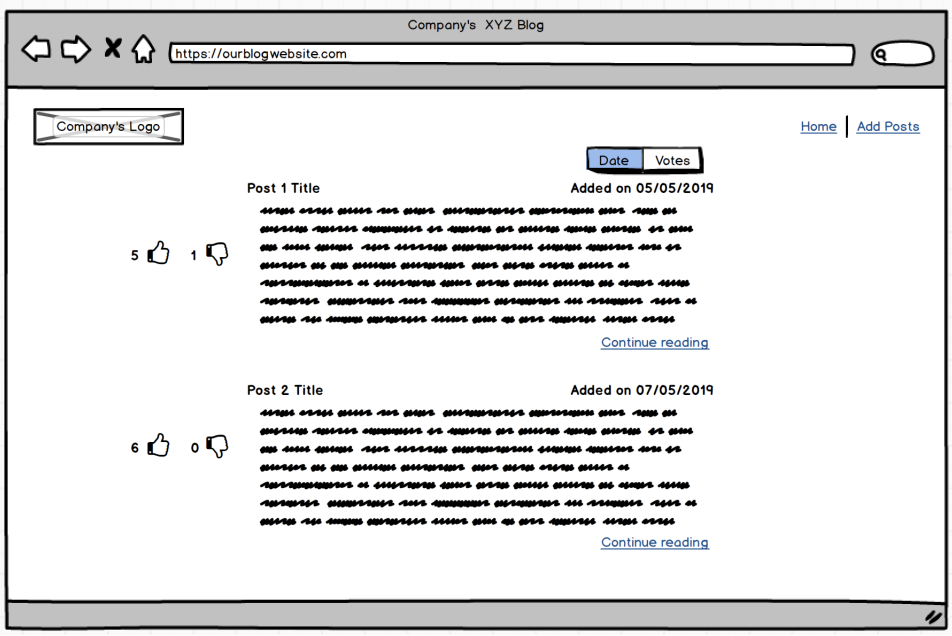

Figure 1: Front page of the website

As shown in Figure 1, you will have to design the layout of the website in a way that posts are displayed in the middle of the screen, showing information such as the *Title*, their *Content* and the number of *likes* and *dislikes* per post. Also, you will have to implement the functionalities of ordering the posts by dates and their number of votes.

The number of votes is composed of the number of *likes* and *dislikes*; hence, the number of votes will be the result of subtracting the *likes* by the *dislikes*. The front page only needs to show a fragment of the *Content* in the posts and the corresponding link to the full reading of the post and other details:

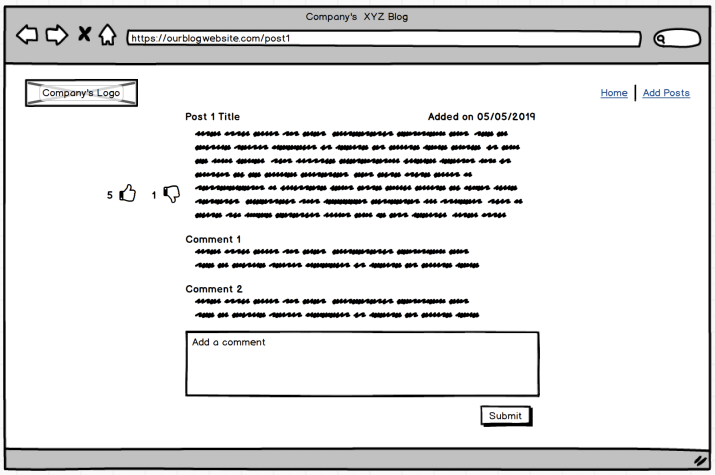

Figure 2: Post details

In the details page in Figure 2, a user can read the full *Content* of the posts and only if it's logged in, should be able to make a comment to a post. Moreover, logged users will also be able to:

1. Like or dislike a vote in any post.
2. Add new posts to the website.

Users that are not logged in should not be able to perform the actions described above. To add new posts on the website, you should implement a form like the following mockup:

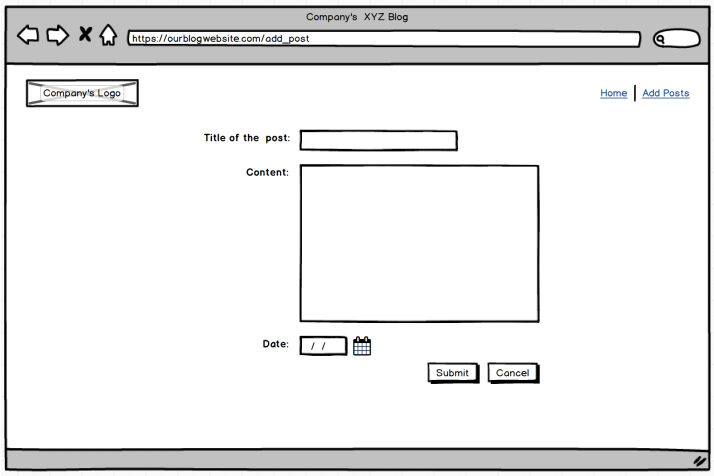

Figure 3: Add a new post to the website

Figure 3 shows the main fields in the addition form, where all of them should be mandatory. Once the post is submitted, any user should be able to read it, comment it and upvote/downvote it.

As a final recommendation, you could use any public template for your webpage design (e.g. Twitter's Bootstrap, Google's Materialize, etc.). Do not use a database for your project, just keep the objects in memory.

The above functionalities should be implemented using MVC constructs in Scala Play. In addition to the general approach, you will be evaluated according to how you implement different functionality on your models, views and controllers. You should take into account issues such as validations, error handling and any other abstractions that you deem fit for the scenario. **Motivate your design decisions** in the report as per the problem description and illustrate your approach using concepts in Scala Play.

More information on Scala Play

- `https://www.playframework.com/`
- `https://www.playframework.com/modules/scala-0.9/controllers`