VRIJE
UNIVERSITEIT
BRUSSEL

SOFTWARE ARCHITECTURES

Assignment 2

Naqi Amine | 0562497

January 6, 2024

ETRO - Department of Informatics and Electronics

Contents

1 Project Description

For this assignment we will implement a social networking website application where users can share their pictures so that other users can see them, like and comment on them. The implementation of the application should follow the Model-View-Controller pattern.

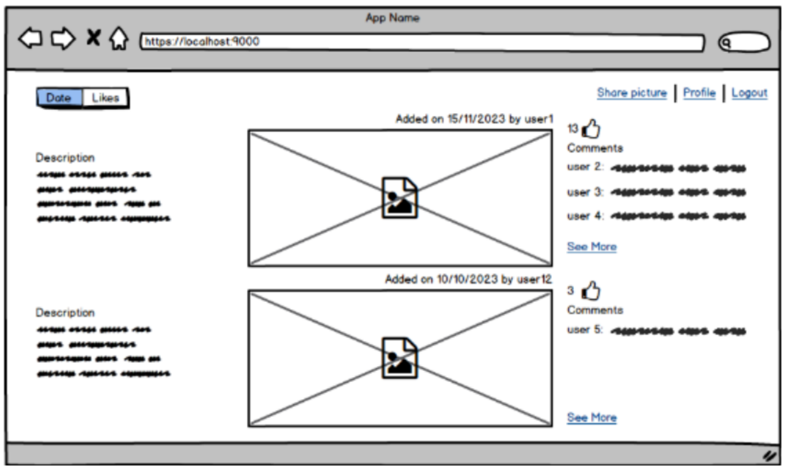


Figure 1: Front page of the website.

As shown, we will have to design the layout of the website in a way that pictures are displayed in the middle of the screen, while description, likes and comments are at the left and right sides of the pictures, respectively. The pictures can be sorted in ascending or descending order by their date of upload or their number of likes. In the comments section, the front page should only show a limited number of comments, to see more comments and see the figure in bigger size, you should create a "See more" link that will redirect the users to the page of the corresponding picture. Basically what we need to do is to create a simple version of facebook. Post, Comment, Like.

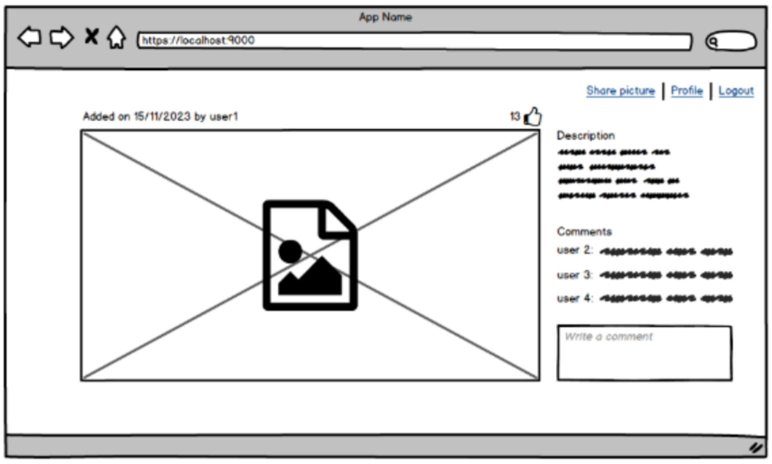


Figure 2: Comment page of the website.

2 Tools and Libraries Used

For our project, we followed an MVC architecture for our website implementation. We have also used the Scala Play API in order to reach do the assignment. Scala Play is a web application framework built on top of the Scala programming language. It's designed to simplify and streamline the development of web applications by providing a set of tools, libraries, and APIs.

(a) Scala Play Library

(b) Scala Build Tool

Figure 3: Tools and libraries used for our project.

In order to compile our project we have used Scala Build Tool (SBT). SBT is a powerful build tool designed explicitly for Scala projects. It's akin to tools like Maven for Java or npm for JavaScript. SBT is not just a build tool but also a task runner and project management tool tailored for Scala development.

For our project the SBT script that is used to build it is as follow :

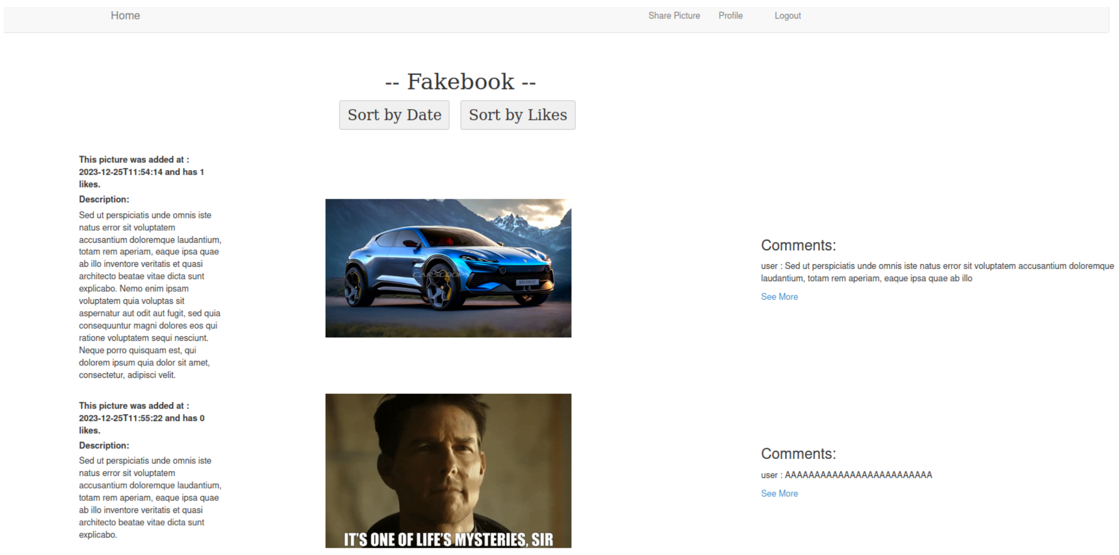
```
ThisBuild / version := ”0.0.1”
ThisBuild / scalaVersion := ”3.3.1”

lazy val root = (project in file(”.”))
  .settings(
    name := ”Assignment 2”
  )

libraryDependencies ++= Seq(
  guice
)
```

For ”ThisBuild”. This line refers to the global settings for the entire build. The version := ”0.0.1” is just the version of the entire build, but the scalaVersion := ”3.3.1” specifies the scala version to be used for the entire build.
When it comes to lazy val root. This just defines the project as root. The (project in file(”.”)) specifies that the project is located in the current directory, with settings having a name := ”Assignment 2”. The last part of the script is the library dependency where we specify an external library ”guice” which is needed for the project.

3 Project Solution



This picture was added at : 2023-12-25T11:54:14 and has 1 likes.

Description:

Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo. Nemo enim ipsam voluptatem quia voluptas sit aspernatur aut odit aut fugit, sed quia consequuntur magni dolores eos qui ratione voluptatem sequi nesciunt. Neque porro quisquam est, qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit.

Comments:

user : Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo

This picture was added at : 2023-12-25T11:55:22 and has 0 likes.

Description:

Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo.

Comments:

user : AAAAAAAAAAAAAAAAAAAAAAAAA

Figure 4: The Main Page

The project starts with being viewing the content of the index html file. In order to access networking website the user has to either log in using an old account or sign up for a new account. By default i have left the account with the username "user" and password "user" as default account for easy accessibility but we can create other accounts if we want. Everything is stored in memory (since we are not required to store them on a SQL database or a different type of database), once the project stops running everything will be lost. Another solution to this would be using JSON file to store username and passwords there but it's still not as secure as using a database for these kind of data. The user in order to log in will have to click on login in the menu bar at the top to log in to his account where he can navigate through content shared by him or other users. The image above shows how the user is going to see the website when he has feed to look at. On the left side of the page you can find the description of the image, in the middle of the page the image itself will be displayed and on the right side of the image, the comments will be displayed with the ability to add more comments simply by pressing "see more".

While remaining on this page, the user is able to filter the feed based on either it being recent (filter by date) or by being popular (filter by likes). In order to share content, the user can simply navigate to the share picture section of the web app by simply pressing "Share Picture" on the navbar. The user is then required to upload a picture and add a description for it. Nothing will be uploaded if these two elements are not present. The user is also able to check his profile by simply navigating to the profile section where his name will be displayed with a default image. If the user would like to see more images he can simply refresh the page or by clicking on home which is located on the navbar on the left or just press F5 on his keyboard. If the user wishes to disconnect he will then just need to press on log out which is also located on the navbar on the right side.

The project for this assignment is made of 3 parts following the Model-View-Controller architecture. Starting with the models part. This assignment contains 5 models.

- User Model : This part contains the User class with it's parameter (Username, Password, Tag). These informations are used to login to the user account.
- UserDao Model : This part is used to either add a user to our user database during a signup (we only store them in memory) or to look up for the user if he exists or not during the login.
- DataMap Model : This part is related to the image that the user uploads and all the informations that we need in order to use them for our webpage. Our picute data or also called a DataMap contains the username of the user that uploaded the picture, the picture link or url of the image, the picture id, a description, the comments shared that are stored in a Map structure where the key is the username of the person who commented and the value is all of the comments posted by the same user stored in a list. The user is also able to like an image so we need a way to know who commented on the image and who didn't and we do that by storing the likes of the user in a Sequence of Maps that contain the username of the person who liked as a key, and the boolean related if they liked the image or not. This model finally also has a variable for counting the likes and storing the date of when it was posted.
- DataDao Model : This model contains all the functions needed to control the DataMap. We can add a picture to it, add a comment, add a like to the image, and sort the page by likes and by dates.
- Global Model : This part concerns the session info on the user where you can get the username or the tag depending what what we need.

After describing what the model part contains, the View part is made of 8 views for our website.

- Index : This is the first page that a new or old user will see when they access the website while not logged in. It's basically a welcome page with my information and name of the assignment.
- Main : This part is designed to create a consistent layout for web pages, including a header with navigation links that change based on whether a user is logged in or not. It takes in a title for the page and HTML content to be inserted into the body of the page.
- Mainpage : This page is where users will be able to see shared pictures with their descriptions and some comments attached to them with the ability to see more if they wanted, and also comment on them. In this page users are also able to sort the images based on how recent they are or how famous they are (sorted by likes).
- Profile : This page displays the users info such as username and a profile picture.
- Signup : This page registers new users' data so that he will be able to login. This page is used to store username, tag, and password of the new user using a form.
- Login : This page checks if the user already exists in the database and if yes, then he's able to login. If not he will be prompted to the same webpage. This page takes username, and password as data to be compared versus what exists in the database.
- Upload Picture : This view is where the user is able to upload pictures to the website along with a text description of the picture.

Finally our project also contains controllers that will control the data transmission between the models and views. In our assignment we have 8 controllers.

- Authenticated User Controller : It handles authenticated user-related actions, specifically the logout functionality. Inside the logout function the authenticatedUserAction is applied to the action, ensuring that only authenticated users can access this route. When a user logs out, it redirects them to the login page (routes.LoginController.showLoginForm). It sets a flash message to inform the user that they have been logged out. Finally, it calls .withNewSession on the Redirect result, discarding the entire session to ensure the user is completely logged out.

- Comment on Image Controller : It handles functionalities related to commenting on and liking images, ensuring these actions are performed by authenticated users. This controller is in charge of multiple data manipulation. One of them is retrieving data associated with a specific image ID from DataDAO and rendering a view (views.html.comment) for commenting on that image, passing the retrieved data to the view. Another one is handling the submission of comments for an image and then validating the incoming comment text using a form. And finally the controller is also responsible of handling the submission of a 'like' for an image by a user. It also adds the user's like to the image data via DataDAO and redirects back to the image view with a success flash message.

- Home Controllers : This controller renders the homepage (index). This page only includes my name, student ID and the projects' title.

- Login Controller : This controller takes care of the login attempt that is done by the user. It handles user login operations. It uses a form to validate user input for username, password, and a tag. The showLoginForm function renders the login view with the form, while processLoginAttempt handles form submission and redirects based on validation :
  - It validates the form data. If successful and the user exists in the userDao, it redirects to the main page with a success message and stores user session information.
  - If the user is not found or the credentials are invalid, it redirects back to the login form with an error message. The code utilizes helper functions to check the length of strings.

- Main Page Controller : This controller is responsible for displaying and sorting images on a main page. It's annotated with @Singleton and uses dependency injection to receive MessagesControllerComponents, a DataDAO, and an AuthenticatedUserAction. The controller contains 3 functions :
  - The function showSharePicture() fetches picture data from dataDao and renders the main page view with the retrieved data.
  - The function sortByDate() sorts images by date, fetching and displaying the sorted data from the dataDao.
  - The function sortByLikes() sorts images by likes, retrieving and displaying the sorted data from the dataDao.

  Each function uses an authenticatedUserAction to ensure that only authenticated users can access these endpoints, and it responds with the appropriate sorted data displayed on the main page view.

- Profile Controller : This controller contains a single function which is showProfile(). It renders the user's profile by fetching an image URL and passing it to the profile view. It utilizes authenticatedUserAction to ensure only authenticated users can access this endpoint and responds with an HTML view that includes the user's image.
- Share Picture : This Controller has 2 functions.
  - uploadPicture() renders a page where users can upload an image and add comments.
  - upPicture() handles the actual uploading of a picture file along with additional information. It saves the image to a specified directory, and then it adds details about the image to a data repository (presumably for application display/management).
- Sign Up Controller : This controller contains 2 functions.
  - showSignUpForm() renders the signup form view, passing the signup form and its submission URL.
  - processSignUpAttempt() handles form submission during user signup. It validates form data and redirects based on success or failure:
    - If the user is not found and the selected tag is available, it redirects to the main page with a success message and stores user session information.
    - If the user is found or the tag is already taken, it redirects back to the signup form with an appropriate error message.

4 Conclusion

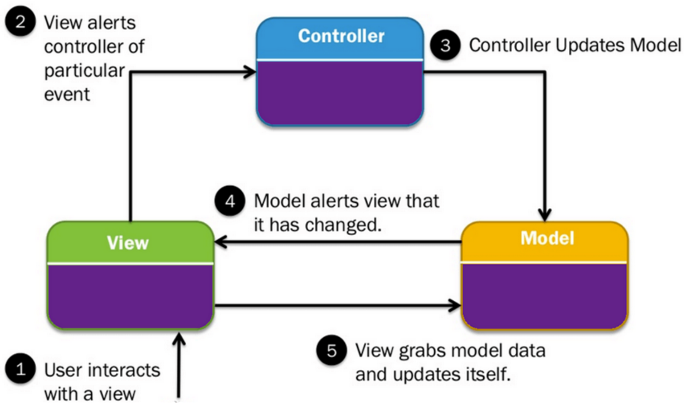

2 View alerts controller of particular event
Controller
3 Controller Updates Model
4 Model alerts view that it has changed.
View
Model
1 User interacts with a view
5 View grabs model data and updates itself.

Figure 5: The MVC pattern

The MVC pattern is known for being easy to extend and easy to grow as well. It provides clean separation of concerns and all classes and objects are independent from each other which makes testing pretty simple but the Model View Controller pattern also has disadvantages. One of the biggest disadvantages this pattern is code maintenance. In order to make it work is to have a controller for every view. It is the recommended practice for this pattern and this brings alot of challenges when a programmer is working on a big website with a lot of pages. MVC can also introduce performance overhead due to the interplay between different layers and the communication between the components can create a tight coupling making the code harder to maintain and modify.

One way to fix this is to use other variants of the MVC pattern. Such as :

- Model View ViewModel (MVVM) : Which is a popular in frontend development, MVVM further separates the view and model through a ViewModel that binds the view and model.
- Model View Presenter (MVP) : This pattern is similar to MVC but focuses more on the presenter to mediate between the model and view, reducing dependencies.
- Model View Intent (MVI) : A reactive pattern that emphasizes unidirectional data flow and immutability, popular in frontend and mobile app development.