Web Applications

DAT076

FruitPedia

Author:
Linnéa Bark
Matilda Sjöblom
Kevin Brunström

Supervisor:
Robin Adams

Readingperiod 3, 2019

Contents

1 Introduction

We have created a web application called "FruitPedia. The purpose of the website is to allow users to share information about fruits they have tried. The website allows users to collaborate and add fruits together in a central list. The users can review this list and save fruits in their personal list. Perhaps the user wants to save every fruit that he has tried and didn't like.

The website offers functionality for managing your own account on the page. The services offered is registration, maintenance of stored data and the possibility to reset your password. The main design idea is that the website should be useful for different screen-sizes and therefore we have strived towards making pages responsive to the window size.

https://github.com/rackarmattan/DAT076.git

2 Use Cases

- A visitor can register an account.
- A user can login with his or her credentials.
- A visitor cannot register a user-id that already exists.
- A user can update his or her password.
- The logged in user can add fruits to the public fruit list.
- The logged in user can view his or her account details.
- The user can save fruits to their personal list.
- The user can delete from their personal list.
- A user can edit a fruit.
- The user can log out.

3 User Manual

In summary, we have strived towards making the page coherent with regard to color and style. A lesser is more approach have been the design guideline. Colors are limited to shades of gray with green for actions that take user forward and red for actions that bring the user back.

The user first comes to the start page, where the visitor is presented with basic information about the webpage. If interested in the page, the user can choose to click login.

If the user already has an account, he or she can simply log in, otherwise, there is the possibility to create a new account. Furthermore, it might be that the user already has an account but does not remember the password. In that case, there exists a "Forgot password?" link for visiting the reset page.

When logged in, the user can do lots of different things. It can view all of its saved list of fruits, add new fruits, see all of the fruits available and log out.

4 Design

For this web app, a broad various of techniques and frameworks have been used. Here follows a list of everything we used, divided by frameworks, languages and patterns.

Frameworks

- Java server faces 2.2
- Prime faces 6.0
- Java persistance API
- JavaBeans

For the whole application, JSF 2.2 is used. This is mostly because all members of the group were familiar with that framework before the project started. Since JSF works very well with Javabean, this is used for the communication between the database and the web pages. Prime faces is used for the page `ResetPas.xhtml` when restricting the user from writing different passwords in the input fields.

As the web application has a relational database, it uses JPA for managing it.

Languages

- XHTML
- CSS
- Java

All the pages in the web application is written in XHTML. XHTML is more strict than regular HTML, and it is therefor easier to solve and discover bugs before they lead to bigger problems. Along with XHTML, the web pages is styled in CSS. The whole back end is written in Java of the same reason as JSF, the group has previous knowledge in that language.

Design patterns

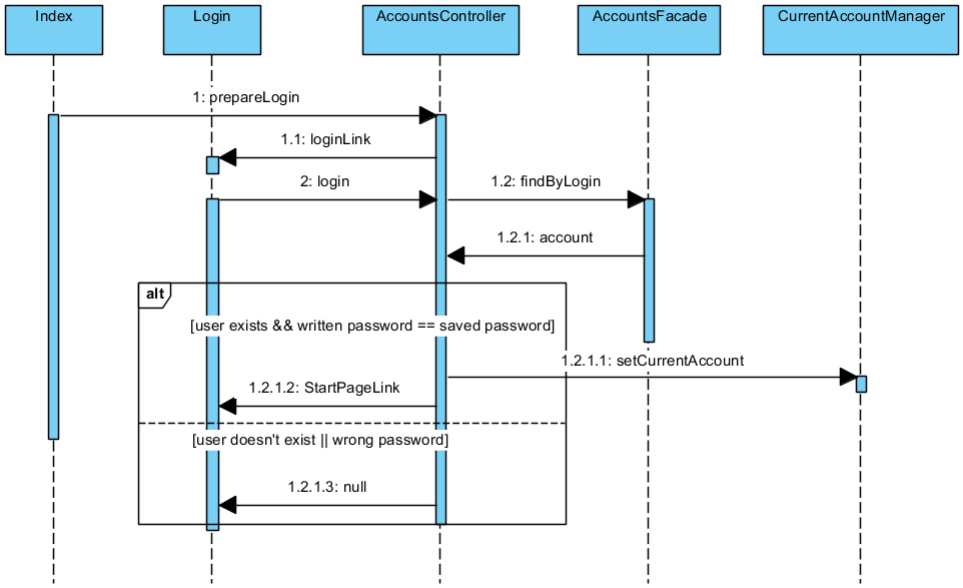
- Singleton

The singleton pattern is used in the class `CurrentAccountManager.java`. As this class is going to be used in many other classes, the singleton pattern suits its purpose well as no new instance can be created. By this, it is sure that every class that uses this class has the same object of it. As many of the controller classes needs to know which user is the current and logged in user, this seemed like a well fitted solution.

5 Application Flow

In this section the application flow for two of the use cases, log in and forgot password will be described by UML sequence diagrams.

Log in

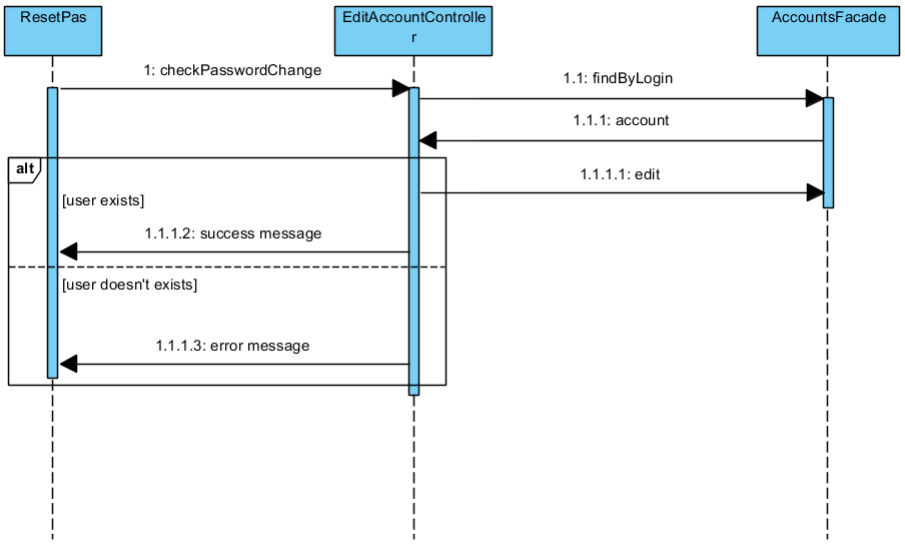


When a user wants to get logged in to the web application, he or she first presses the "Log in" button at `index.xhtml` page top right corner. A request of the `Login.xhtml` page is sent to the server class `AccountsController`'s method `prepareLogin()` which returns the correct direction.

After that, the user begins to fill in the fields "Username" and "Password" and then presses the "Login"-button. When the button is pressed, a request to the `AccountsController`'s method `login()` is sent. This method first sends a request to the class `AccountsFacade`'s method `findByLogin()`. This method creates a named query to get a user from the database with the requested login. If that user exists, it returns the specific user, otherwise it returns null.

If `login()` gets a user and not null, it checks if the written passsword is the password for that specific user. If the password match, `AccountsController` sets the current user in `CurrentAccountManager` and returns a string to the browser which redirects to the page `Startpage.xhtml`. If the user is null or if the password does not match, `login()` returns null, which makes the browser stay at the same page.

Forgot password



If a user has forgotten his or her password, currently, they can change it by remembering their login. For further development, the desirable behavior would be to send a link to the user's email which can then reset the password.

At the page `ResetPas.xhtml` the user fills in his och her login and the new password. By the help of primefaces, the two input fields "New password" and "Repeat new password" are checked if they are equal. After that, the user presses the "Reset password" button and a request is sent to the server class `EditAccountController`.

In the `EditAccountController` the method `checkPasswordChange()` is used. First, the method sends a request to the class `AccountsFacade`'s method `findByLogin()`.

If the user exists, `checkPasswordChange()` checks if the requested password is longer than three, and if it is the new password is set for that user. In order to get it changed in the database, `EditAccountController` makes a request to the `AccountsFacade`'s inherited method `edit()`, which updates that user's password. After that, a success message is displayed for the user at `ResetPas.xhtml`. If the user does not exist, or if the password is shorter than three, an error message is displayed for the user instead.

6 Responsibilities

In general, many of the tasks for this project were done together, either through pair-programming or with all three members working in the same place. This structure aimed to ensure that one person did not get stuck alone with a complex problem. Though each team member had areas of extra interest, they were the following:

Matilda: Database and back-end aspects, some design logic. Kevin: Design and frontend aspects, also back-end aspects. Linnéa: Design and frontend aspects.