Project Group 9

Jonathan Gildevall, Johannes Mattsson & Viktor Lindblom

March 2019

1 Introduction

The goal was to create a website which would allow it's users to create a professional looking profile page that could serve as an introduction of you when applying for a job or similar. One could call it an extended CV in the form of a website. In addition to this a homepage that would encourage new users to create their own profile page and allow the promotion of already made pages so that a potential recruiter might catch a glimpse of your page when visiting the site.

Link to github: `https://github.com/Baloo1/DAT076`

List of use cases:

- Register account
- Login to account
- Logout
- View User-page
- View Admin-page
- Add skills
- Add work experiences
- Add projects
- Edit profile picture
- Edit contact information
- Edit skills
- Edit work experiences
- Edit projects
- Generate link to page

2 User manual

How to create a profile page: The first step is to create an account, this is done by pressing the register button in the top right. When pressed you will be presented with a popup that displays fields for email and password. After entering this information, press the submit button and an alert welcoming you will appear. You are now logged in and have your own profile page ready to edit.

How to edit a profile page: When you are logged in, press the button called Edit my page. You are now presented with a pretty sparse user page, but using the edit and add buttons below each segment you will be able to create content that describes you to a potential employer. You can edit your contact information and profile picture, quick facts about yourself, experiences such as work and projects that you have done. When you press the add button for each item you are presented with all the relevant fields for each component. Each item can be edited after it has been created using the edit button. Images from your own computer can be uploaded to the site using the Browse button when adding or editing your contact information or a project. When you're satisfied with the way your page looks you can click the get link button and the link to your page will be copied to the clipboard.

Access the admin interface When you are logged in as an administrator for the site, you can easily access the admin interface from the top navigation bar. While in the admin interface, you can easily view every user on the site, get some quick information about them and access their page.

3 Design

The web app follows the architectural design pattern Model-View-ViewModel (MVVM).

3.1 Back-end

The web application is built with a Node.js server as back-end. The Node.js server uses the Next.js and Express.js frameworks to provide the web application. Our ORM layer was built with Objection.js as the ORM. The database was built using Postgresql.

Next.js is used as it is easy to get started with, friendly for new Node.js developers and works great with Reactjs as front-end. Next.js provided us with server-side rendering for our pages so they load fast even for slower devices. And easy to create routes for navigation between pages.

We used express.js to build our RESTful API, and our model layer with objection.js and postgresql. Express.js was used as it is the mosty widely used Node.js and has several built in HTTP utility methods.

Objection.js and PostgreSQL is part of our model layer where all the data about our users and such are stored. We used Postgresql as a database it is a solid relational-database and we all had prior knowledge of it.

3.2 Front-end

Reactjs framework is used for the front-end. To ease the styling of React components the framework React-Bootstrap was used. We have separated React into three folders in our project. The Pages part which is served by Next.js as the actual web pages a user visits. The components which are presentational and reusable React components, representing our View layer. Finally there is the containers which act as our ViewModel, they get the data from the model and they manipulate it so that the view layer can display the data. Due to the nature of React the ViewModels has to be a part the React component tree. Therefor components, the view, includes Containers which then include their components.

Towards the end of the project a decisions was made to restructure the entire code-base to a containerized version instead of the regular React version where there isn't any separation between the JavaScript code and the HTML. In the containerized version a lot of attention has been put into separating the two.

4 Application flow

The application generally uses a component that is displayed to the user, backed up by a container that contains the logic for that component or multiple similar components.

The main pages, admin, edit, index, user, viewuser have responsibility to place the components on the page and populate them with the data they need.

On the backend the application uses a RESTful API backed by a PostgreSQL database to handle all data. Below follows a few examples of use cases and how they are handles by the application.

4.1 Login

The user accesses the login component and fills in their email and password. When the user then presses login, the mainNavBarContainer sends a POST to */api/login* which checks that the user exists, and that the password matches the stored one. If login in successful, the api creates a JSON web token and returns it. Upon getting the token, the mainNavBarContainer saves it and updates the state so the user is now logged in.

4.2 Logout

The logout process is very simple. Since the JSON web tokens can not be invalidated on the server side, we simply remove them from storage on the

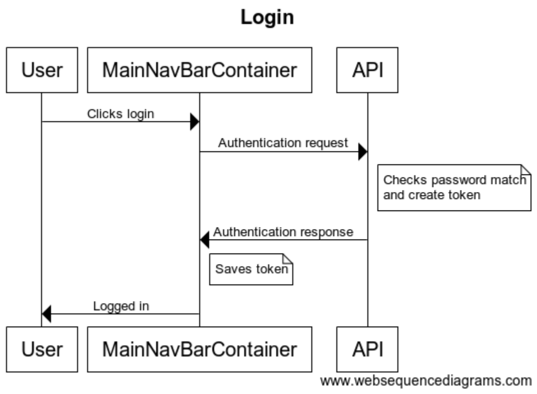

Figure 1: An UML sequence chart of the login process

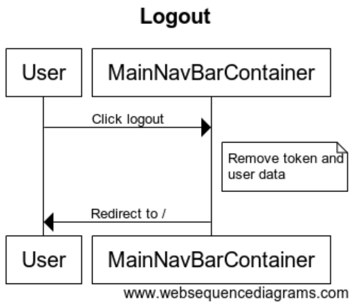

Figure 2: An UML sequence chart of the logout process

client side, together with the user and the user roles. Since the user often is on a restricted page when logged in, we also redirect the user to the home page for the site when logging out.

4.3 Add a new project

When adding a new project to the users page, a new image has to be uploaded as well. This is done in two steps, first the image is uploaded and saved. After that, the image id is returned to the container and sent along with the other data to save the added project. Here, the api also checks if the sending the edit is the owner of the item it wants to edit.

4.4 Edit an "about me" item

The edit process is similar to the process for adding a new item with the main difference being that we have to select which item to edit. This is done between the user and the container on client side.

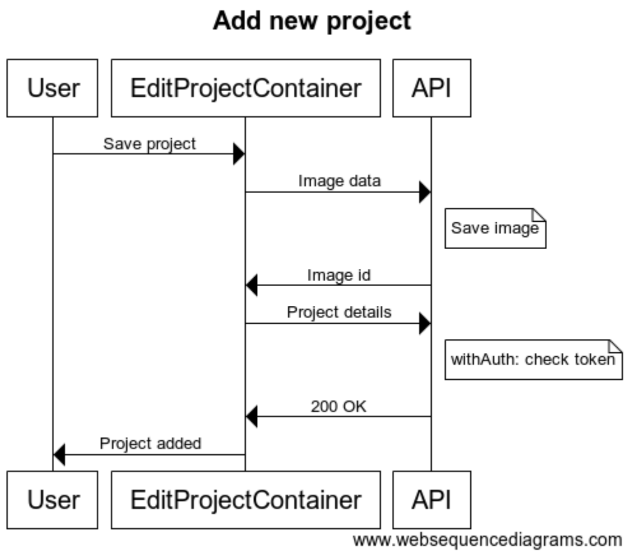

Figure 3: An UML sequence chart of the Add project process

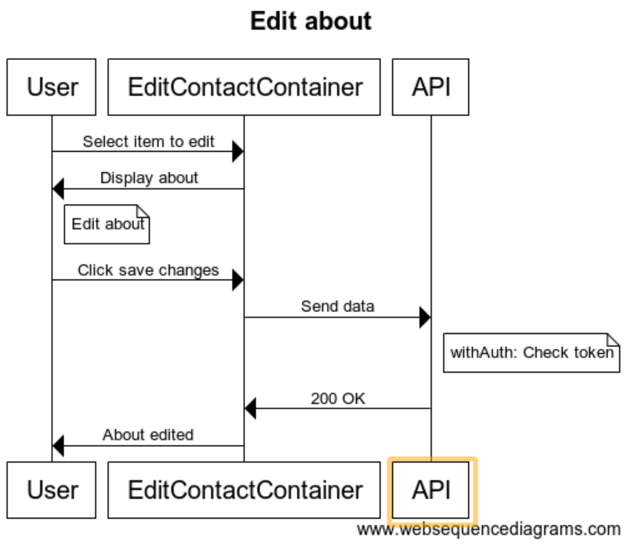

Figure 4: An UML sequence chart of the Edit about process

5 Responsibilities

Viktor was responsible for most of the database and the API, as well as a lot of the authentication. Also did the admin view on the client side.

Johannes was responsible for the login, setting up forms, part of the authentication, tests and general code structure.

Jonathan was responsible for most of the front-end.