ProjektPortalen

- by Group 18:

Alexander Rol - alrol17

Lasse Fisker - lafis17

Patrick Christoffersen - pachr16

Rasmus Jensen - rasje17

Hand-in 1 for SB5-TEC at SDU, due date 10/11-19

Hand-in 1
SB5-TEC
Due 10/11-19

alrol17
pachr16
rasje17
lafis17

Projektportalen

Description of software

The software is a website for students all over the country, to find exciting bachelor projects posted by companies. Students can browse the offers currently provided, but cannot apply for working with the project, unless they are logged in. Companies can use their company login to view the projects they are currently offering students and create new offers.

How to run it

The process for running the software is as follows:

- Start the server. This is done through cmd, locating the directory in which the file “ProjektPortalen.js” exists, and typing *node ProjektPortalen.js*. It will listen on port 8888.
- Run the ProjektPortalen-directory with HTML files with a WAMP service, and open the index.html file through it.
- You now have a choice between continuing as a student or a company, or you can click the Login button in the top-left corner directly. If you choose that you are a company, you will be asked to log in immediately. If you choose to continue as a student, you will be sent to an overview of current project offers.
- You can also create new users - both for representing a company or a student - by clicking “Opret nyt login” on the login screen.
- If you login as a business, you will have the option for creating new project offers. If you are logged in as a business, and are viewing a project offer you have made, you have the option of deleting it.
- If you are logged in as a student, you can click “Ansøg” while viewing any offer, and you will be added to the list of applicants for that offer - this is not visible to you, but it will be changed in the .json-file that contains the information describing the offer. You will only be added once, no matter how many times you click the button.

Architecture

The site has been set up in a way that separates it into a client and a server. The server is only used to save or load information for the client. The client itself handles showing the .html pages and navigating between them, as well as the functionality on the pages themselves. A page might contact the server to load a project offer so that it can display it to the user on the page, but it only sends a GET-request to the server - the server is not aware what is done with that information. This way, the server never holds user context-specific information, which means that it is a RESTful server.

alrol17
pachr16
rasje17
lafis17

Technologies

- HTML
    - For the project HTML has been used as a way of presenting the product. within the HTML there has been used some Cascading Style Sheets (CSS) for customizing the look of the regular HTML along with some scripts containing JavaScript (JS). These scripts are used for adding actions to buttons or to define which data to load in when loading a page for the client/user of the site.
- JavaScript
    - JS has been used for the backend functionality across the webpage. This includes both regular business logic, and for the scripts for sending and receiving data between the client and the server.
- Express
    - Express is a framework for programming web applications. In this project, it has been used to facilitate the communication between server and client in an easier way than it would be to use regular JS, as Express supports a lot of the functionality needed for websites in one library, rather than using multiple others.
    - To be able to access the body of a request, a body-parser had to be used by the server, because express does not currently support this functionality by default.[1]
- HTTP
    - Different HTTP requests have been used in the software.
        - GET is the request being used the most in the software, since a lot of the functionality of the software relies on the client being able to receive some of the persistent data from the server. Examples of functionality that require a GET request, could be **authenticating user credentials when logging in** or **receiving the data values of an offer, to be shown in an html page**.
        - POST is another request being used in the software. The request is used when we want to **create new users** for the software or when a business user wants to **create a new offer**.
        - PUT is used in the software to **alter the list of users who have applied to work on a specific project**. Each offer has a list of IDs of the users who have asked the company for permission to work on the project.
        - DELETE has been used to **delete offers** from the software. This could be the case when a company does not want to offer a certain project anymore, either because they do not need the project to be worked

[1] https://codeforgeek.com/handle-get-post-request-express-4/

on, or because they have already fulfilled their quota regarding the number of students on a specific project.

- JSON
  - Some of the communication between clients and server is done by sending JSON objects. The server has access to two data files, users.json and offers.json. These files contain the data of users and offers respectively, stored as JSON objects. When extracting data from these objects, the method JSON.parse() is used, and when packing data from a client to be sent to the server for storage, JSON.stringify() method is used.
- REST
  - REST has been used in a way that the server never holds information about anything other than what it is supposed to do when getting the different calls (GET, POST, PUT, DELETE.) This makes the server able to (in theory) handle multiple clients at once as it doesn't need any additional information than the user can provide. Though this has not been tested in this version.
- CSS
  - CSS has been used for formatting and customization across the software, for anything from headers, button and loginboxes to satisfy the visions that we had of the layout of the page. Styling has been customized for both built in types and classes defined by us.
- CORS
  - CORS - Cross Origin Resource sharing. The origin is a combination of protocol, URL, host name and port. For this project the server is using port 8888 and the WAMP server uses port 8080. This makes it necessary to use CORS as a new port will mean a new point of origin. This has been fixed by installing a package (using "npm install cors" in the CMD) this allows for adding one line of code in the server resulting in a default allowance of resource sharing.
- WAMP
  - WAMP is a collection of useful tools for web-development, for example HTML site hosting, databases and SSL support. In this project, we have used it mostly for hosting the different HTML files, so that we are able to host a complete website, even though it is only running on localhost.

Design choices

Server side persistence

Data has been saved on the server by using two .json files. JSON is easy to handle in javascript, which the server is written in, and so has been used for easy loading and saving of data. When the server loads a JSON object, it can ask directly for the value connected to any key in that object, and therefore quickly load the objects that we want to operate on.

alrol17
pachr16
rasje17
lafis17

The data in the .json files is not encrypted - it is stored as readable text, and this method is therefore not very secure.

A better method might have been to store the data in a dedicated database, which would still make it easy to query the information needed. This would ease the server load by avoiding handling the entire list of data at each call, thus making the solution more scalable.

Handling of persistent logins

The way logins are handled is by giving each user its own ID as well as a login value when they log in. These parameters are simply appended to the URL, and then when the user navigates to another page on the site, those other pages can find the parameters from the URL. However, this also means that you can log in to any user by simply modifying your URL when you visit the site, so the level of security with this method is very low.

Known bugs

- Need to pre-fetch JSON files before operating on them
    - Sometimes, if we go directly to trying to create a new user (by sending a POST-request to the server), it doesn’t work. If we ping the server with a GET-request first, we can then afterwards send a POST-request to create a new user, and it works correctly. This workaround can be seen in the code for createUser.html.

Inspiration sources

How to write fetch-requests, as well as how to handle them server-side, have been inspired by this site: http://jsonplaceholder.typicode.com/guide.html

For the offerOverview.html page there have been two major inspiration sources, them being: https://www.w3schools.com/howto/howto_css_cards.asp from which is taken the original styling(css) for creating cards and their shadow.
And: https://www.w3schools.com/js/js_htmldom_nodes.asp From which there has been taken inspiration for dynamically creating new DOMelements for cards and adding them to the existing DOM through JavaScript.