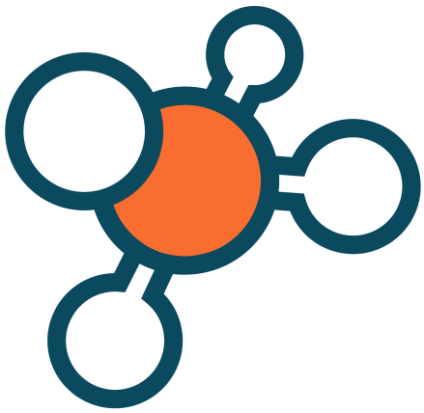
Advanced WEB Programming

4A Semester 1

PROJECT MANAGEMENT APP



Report

Participants: Ugo TAGNATI and Vincent KERHARO

Project description

We had the idea of a project management app, which would allow users to set up a list of their ongoing projects by detailing important information, such as deadline, priority, number of people involved and more. The users would then be able to check their list or search for a specific project by name to check its specificities. That way, they would be able to manage their time effectively and focus on the most urgent/important projects.

The app would be built with HTML, CSS and Vue.js for the front-end, and would use Node.js and Express for the back-end, and more specifically the handling of the projects and their data (CRUD interactions).

Since this theme allowed us to meet every requirement that were specified, we decided to work on it without further deliberation.

An example of an employee not using our app

Project features

The project currently features an authentication system, a user can either register themselves or login if they're already known by the system. They are then greeted by the project list, where they can see all the ongoing projects they have. They add more project with a button that takes them to a "submit project'' page where they have to fill in the project's info. The current fields to fill in are:

- "Name" (a string)
- ''Priority'' (a selection)
- "Starting date" and "Deadline" (two dates)
- "Number of participants" (number)
- "Project description" (a large string typed in a textarea)

Once the project is saved, it will appear in the previously mentioned project list. Entering a specific project name in the search field will remove all but the specific project from the view. Clicking the delete button a row of the project list's table will remove the project from the view. Users and projects are saved in two separate JSON files.

Project's timeline

Design

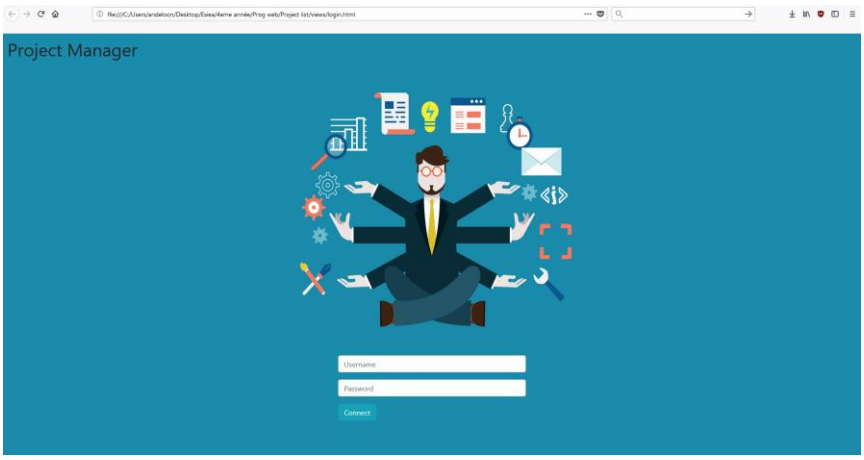
The clearly defined features of the project made it easy to imagine what elements were necessary for the project to work: 1 html page for the login, 1 for registration, 1 for the list, and 1 for adding a new project. We knew both the login and project list pages needed to read data that was created and added to by the registration and project creation list respectively.

Since we were learning how to use tools such as Vue.js and Nodes.js in parallel with the coding of the project, we added features as we learnt to implement them.

Production

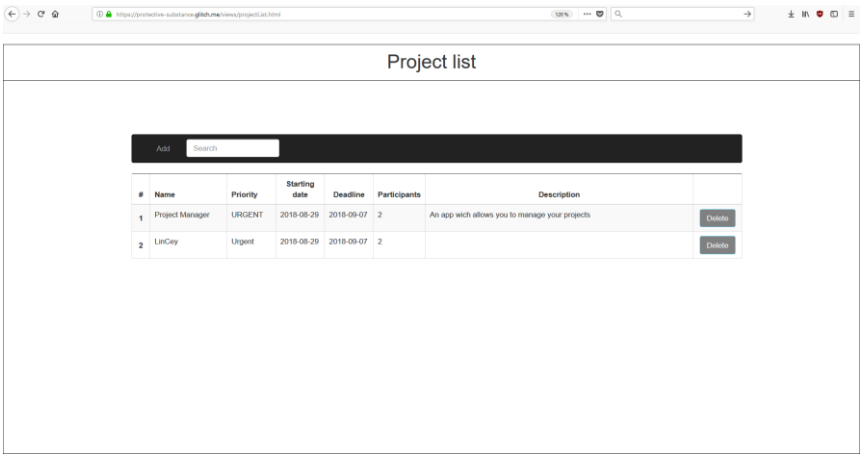
We started with the login page, which was at first just a HTML and a CSS.

Once we worked on Bootstrap, JavaScript and Vue.js, we added more page that contained elements such as buttons and links between them.



Login page: first attempt

Finally, after we read the tutorial in Node.js and Express that taught us to use POST and GET requests to handle data, we implemented the backend of the app, with the ability to actually use the info from a project registration in another page so that it could show it in a list.



Current project list page

In the end, a few design decisions were modified, for example the registration page is a now a pop-up that appears when clicking “sign-up”, and the project’s looks were changed.

The project is currently made of:

- 3 HTML pages described earlier:
    - login.html, the first page, where the user logs in or signs up
    - projectList.html, where the user can see his projects, add a new one, delete one from the view, or search for one
    - newProject.html, where the user enters the info of the new project

- 3 JSONs:
    - users.json, where users are saved
    - projects.json, where projects are saved
    - package.json where the dependencies are specified

- Server.js that handles Node.js, working with port 8080

- Style.css, to make things pretty

Shipping

Once the project was nearly finished, we started looking at how to ship the application. We decided to use Glitch.com as it was said to be easier and more straightforward than Google Cloud Platform. After setting up the server.js and adding our sources to the Glitch project, the app was up and running. There is a default account created that has admin as username and admin as password.

Difficulties

We faced multiple problems while creating this app. Most solved themselves naturally as we progressed in the tutorials and learned to use new tools, but some took a bit of time to solve.

The first main issue we had was with the table, creating it was not too difficult, but we then had to make it so it could be dynamically updated by adding or removing a row. The issue was solved by using two JavaScript functions that used splice and an index (to target a specific row in the case of deletion, and to use the right index number in the leftmost column of the table in the case of adding a row).

The second most time-consuming problem we faced was sending the data from the project registration page to the project list, even if Node.js was to be used for exactly this purpose. Luckily, the great documentation available online allowed to understand and use the app.get and app.post methods provided by this run-time environment.

Eventual Improvements

Even though the project requirements were met, there are still ways we thought the app could be improved in different ways:

- If the description text is too long, it will vertically stretch the row the project is in. There could be a scrollbar in the cell to allow for consistent size throughout the table.
- If more info were to be necessary when adding a project, for example technologies and tools used, budget and more a simple table would not suffice. To solve this problem maybe we make a row clickable, so the user would be sent on a page dedicated to the project with an extensive amount of details.
- Besides being able to search for a project with the search bar, there is little the user can do to help him organize the project list. Sort buttons could be added to show projects by priority or deadline for example.
- Currently, the authentication system does not make every users files inaccessible from each other.