I4GUI Mandatory assignment 3: ModelsManagement

Formål
Af få erfaring med et client side gui framework: Vue.

Opgaven
Der ønskes udviklet en front-end til en Webapplikation medarbejdere på et modelbureau kan bruge til at holde styr på opgaverne. Back-end api-serveren er ikke fuldt udviklet, der mangler meget funktionalitet, men der er nok til at udviklingen af front-end applikationen kan påbegyndes.

Front-end applikation skal benytte Vue JavaScript frameworket. Du skal selv fastlægge brugergrænsefladen, samt hvilken funktionalitet der eventuelt implementeres ud over den grundlæggende funktionalitet specificeret herunder.

Grundlæggende funktionalitet:

- **Login**
  En medarbejder kan loge ind. Der er to typer af medarbejdere: managere og modeller.
  Login er det eneste api-kald som kan tilgås uden jwt-acces token. Ved succesfuld login returneres et jwt-token, som skal sendes med til alle andre api-kald.
- **Opret ny model**
  En manager kan oprette en ny model.
- **Opret ny manager**
  En manager kan oprette en ny manager.
- **Opret nyt job**
  En manager kan oprette et nyt job.
- **Tilføj model til job**
  En manager kan tilføje en model til et job.
  Bemærk at der godt kan være flere modeller på samme job.
- **Slet model fra job**
  En manager kan fjerne en model fra et job.
- **Se job**
  En manager kan se en list med alle jobs.
  En model kan se en liste med sine egne jobs.
- **Tilføje en udgift til et job**
  En model kan tilføje en udgift til et job.

Api-server

Download opgavens api-server fra Blackboard. Husk at du skal give kommandoen update-database før end du kan starte serveren. Når den startes vises en Swagger-side, som viser det api som serveren stiller til rådighed. Her kan du se hvorledes api'et kan kaldes, men du får også bruge for at kigge i koden for serveren. Du kan f.eks. se hvilke data serveren seeder, da du skal bruge login oplysningerne.

Om brug af jwt-token

Login

- For at logge ind skal du sende et POST request til:
  /api/account/login
  Med et json object som har email og password properties.

- Ved et successful login for du en JWT-token tilbage, som du skal lagre, da den skal sendes med vil alle de efterfølgende kald. Ofte lagres den i localStorage.
- JavaScript eksempel:

```
fetch(url, {
  method: 'POST',
  body: JSON.stringify({
         email: username,
         password: password
    }),
  headers: new Headers({ 'Content-Type': 'application/json'
  })
}).then(res => res.json().then((token) => {
        localStorage.setItem("token", token.jwt);
)
.catch(error => console.error('Error:', error))
```

Kald med jwt i header

Jwt-token skal sendes med i headeren til alle api-kald som kræver at brugeren er logget ind. Dette kan f.eks. gøres sådan i JavaScript.

```
var url = "https://yourUrl";
fetch(url, {
        method: 'GET',  // Or POST, PUT, DELETE
        credentials: 'include',
        headers: {
            'Authorization': 'Bearer ' + localStorage.getItem("token"),
            'Content-Type': 'application/json'
        }
    }).then(responseJson => {
        var items = JSON.parse(responseJson);
    })
    .catch(error => this.setState({
        isLoading: false,
        message: 'Something bad happened ' + error
    }));
```