

SW4FED Mandatory assignment 2: ModelManagement

Formål

Af få erfaring med et client side Web framework: React.

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 React JavaScript frameworket, og den skal laves som en single page application, SPA. Du/I må gerne arbejde i Typescript, hvis dette sprog fortrækkes. Du/I 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-access token. Ved succesfuld login returneres et jwt-token, som skal sendes med til alle andre api-kald.

Funktionalitet for managere:

- **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 alle jobs**

En manager kan se en liste med alle jobs.

Funktionalitet for modeller:

- **Se alle jobs**

En model kan se en liste med sine egne jobs.

- **Tilføj en udgift til et job**

En model kan tilføje en udgift til et job.

Api-server

Docker (<https://www.docker.com/products/docker-desktop/>) skal være installeret på din computer for at du kan køre api-serveren.

Download filen compose.yaml og placer den i en mappe. Åbn denne mappe i en terminal og start api-serveren med kommandoen:

```
docker compose up
```

```
PS C:\Courses\SW4FED\Assignments\Opg 2 - React> docker compose up
[+] Running 3/3
 - Network opg2-react_default   Created                                0.0s
 - Container feddb              Created                                0.0s
 - Container api                Created                                0.0s
Attaching to api, feddb
feddb | SQL Server 2022 will run as non-root by default.
```

Første gang api-serveren startes, tager det lang tid! Serveren er klar, når du ser dette:

```
ApplicationName: ModelsApi
Environment: Production
ContentRootPath: /app
info: Microsoft.EntityFrameworkCore.Database.Command[20101]
      Executed DbCommand (19ms) [Parameters=[], CommandType='Text', CommandTimeout='30']
      SELECT 1
info: Microsoft.EntityFrameworkCore.Database.Command[20101]
      Executed DbCommand (28ms) [Parameters=[], CommandType='Text', CommandTimeout='30']
      SELECT OBJECT_ID(N'[__EFMigrationsHistory]');
info: Microsoft.EntityFrameworkCore.Database.Command[20101]
      Executed DbCommand (1ms) [Parameters=[], CommandType='Text', CommandTimeout='30']
      SELECT 1
info: Microsoft.EntityFrameworkCore.Database.Command[20101]
      Executed DbCommand (1ms) [Parameters=[], CommandType='Text', CommandTimeout='30']
      SELECT OBJECT_ID(N'[__EFMigrationsHistory]');
info: Microsoft.EntityFrameworkCore.Database.Command[20101]
      Executed DbCommand (9ms) [Parameters=[], CommandType='Text', CommandTimeout='30']
      SELECT [MigrationId], [ProductVersion]
      FROM [__EFMigrationsHistory]
      ORDER BY [MigrationId];
info: Microsoft.EntityFrameworkCore.Migrations[20405]
      No migrations were applied. The database is already up to date.
info: Microsoft.EntityFrameworkCore.Database.Command[20101]
      Executed DbCommand (10ms) [Parameters=[], CommandType='Text', CommandTimeout='30']
```

I en browser kan du navigere til en Swagger-side, som viser det api som serveren stiller til rådighed. Her kan du se, hvorledes api'et kan kaldes.

Serveren snakker både http og https, men da den bruger et self-signed certificat, så er det ikke sikkert at din browser tillader at snakke med serveren over https.

Du finder swagger-siderne på disse urls:

<http://localhost:8080/swagger/index.html>

eller

<https://localhost:8081/swagger/index.html>

Brug https hvis din browser tillader det!

Serveren seeder databasen med nogle data. Bl.a. nogle brugere du skal bruge for at kunne logge ind.

Disse brugere seedes:

```
// Seed manager
new EfAccount
{
    Email = "boss@m.dk",
    PwHash = HashPassword("asdfQWER", bcryptWorkfactor),
    IsManager = true
},
// Seed some models
new EfAccount
{
    Email = "nc@m.dk",
    PwHash = HashPassword("Pas123", bcryptWorkfactor),
    IsManager = false
},
new EfAccount
{
    Email = "hc@m.dk",
    PwHash = HashPassword("Pas123", bcryptWorkfactor),
    IsManager = false
},
new EfAccount
{
    Email = "al@m.dk",
    PwHash = HashPassword("Pas123", bcryptWorkfactor),
    IsManager = false
},
new EfAccount
{
    Email = "jk@m.dk",
    PwHash = HashPassword("Pas123", bcryptWorkfactor),
    IsManager = false
}
```

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 får du en JWT-token tilbage, som du skal lagre, da den skal sendes med ved alle de efterfølgende kald. Ofte lagres den i localStorage.
- JavaScript eksempel:

```
async login() {  
  let url = "https://localhost:44368/api/account/login";  
  try {  
    let response = await fetch(url, {  
      method: "POST",  
      body: JSON.stringify(this.form), // Assumes data is in an object called form  
      headers: new Headers({  
        "Content-Type": "application/json"  
      })  
    });  
  
    if (response.ok) {  
      let token = await response.json();  
      localStorage.setItem("token", token.jwt);  
      // Change view to some other component  
      // ...  
    } else {  
      alert("Server returned: " + response.statusText);  
    }  
  } catch (err) {  
    alert("Error: " + err);  
  }  
  return;
```

Bemærk at man i klienten kan få adgang til properties i payload (f.eks. role og modelId) ved brug af denne funktion:

<https://stackoverflow.com/questions/38552003/how-to-decode-jwt-token-in-javascript-without-using-a-library/46188039>

Eller du kan installere denne npm package: jwt-decode

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 DELETE
  credentials: 'include',
  headers: {
    'Authorization': 'Bearer ' + localStorage.getItem("token"),
    'Content-Type': 'application/json'
  }
}).then(responseJson => {
  this.response = responseJson;
})
.catch(error => alert('Something bad happened: ' + error));
```

Ved POST og PUT skal data sendes i body:

```
var url = "https://yourUrl";
fetch(url, {
  method: 'POST', // Or PUT
  body: JSON.stringify(this.form), // assumes your data is in a
                                  // form object on your instance.
  credentials: 'include',
  headers: {
    'Authorization': 'Bearer ' + localStorage.getItem("token"),
    'Content-Type': 'application/json'
  }
}).then(responseJson => {
  this.response = responseJson;
})
.catch(error => alert('Something bad happened: ' + error));
```

Brug gerne await i stedet for .then!

Krav til aflevering

Du skal lave en video, der viser, at din app opfylder alle krav, og uploade denne video sammen med en zip-fil af dine kodefiler samt angive adressen på jeres/dit git-repo.