Full-stack software development

Lab 07: Front-end Security & i18n

J. Pieck, J. Rombouts, B. Van Impe

Contents

1 Introduction

1.1 Assignment

Accept the assignment on https://classroom.github.com/a/AgqM74Vc

1.2 Back-end

By now you should have a postgres database running.

Add an .env file in the backend containing:

```
DATABASE_URL="postgresql://postgres:postgres@localhost:5432/courses-lab7?schema=public"
APP_PORT=3000
JWT_SECRET="d2ViNC1ub3Qtc28tc2VjcmV0LWFjY2Vzcy1zZWNyZXQ="
JWT_EXPIRES_HOURS=8
```

Note that **this is again a different database name**, this way the other labs continue to function. You can use a different JWT secret if you prefer.

Setup the back-end and database with these commands:

```
npm install
npx prisma migrate dev
npx ts-node .\util\seed.ts (or mac/linux: npx ts-node ./util/seed.ts )
```

Start up with npm start. This is the only thing that we'll be doing in the back-end.

1.3 Front-end

For the front-end, first create a new *.env* file in the root folder. Add the following content:

```
NEXT_PUBLIC_API_URL = http://localhost:3000
```

And run with npm start

1.4 General Remark

This lab will tackle security and i18 together, not one before the other. This way you won’t have to do certain parts twice.

2 Set-up for i18n

2.1 Install dependencies

We need 3 packages installed. Add these to the front-end package.json as dependencies:

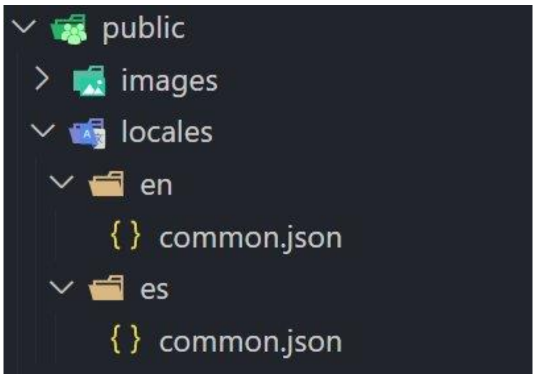
```
    "i18next": "^23.7.6",
    "next-i18next": "^15.0.0",
    "react-i18next": "^13.4.1"
```

And run npm install to install them. This will install the specific versions of these packages that are compatible with the rest of our dependencies.

Side note: the ^ sign in front of the version number allows npm to install versions that are higher in minor version. In this example i18next would upgrade to version 23.8.x or 23.9.x, but not to version 24.0.x

2.2 Configuration

- Check in next-i18next.config.js that we are going to use the English and Spanish locales in this app and that the default one is English.
- The other config file (next.config.js) is also already present and has the correct contents.
- Wrap the App in a i18n provider, as shown in slide 8 of the Internationalisation slides.
- Create this file structure:



```
public
  images
  locales
    en
      common.json
    es
      common.json
```

In the Spanish file, add this:

```
{
  "app.title": "Cursos App"
}
```

In the English one:

```
{
  "app.title": "Courses App"
}
```

In the index.tsx file, add get the translate function from the useTranslation hook and the necessary import. Make sure to import useTranslation from next.js and not from react.

```
const { t } = useTranslation();
```

- Change the title from fixed text to

```
{t('app.title')}
```

Load the translations using server-side rendering. Look in the internationalisation slides to see how this is done.

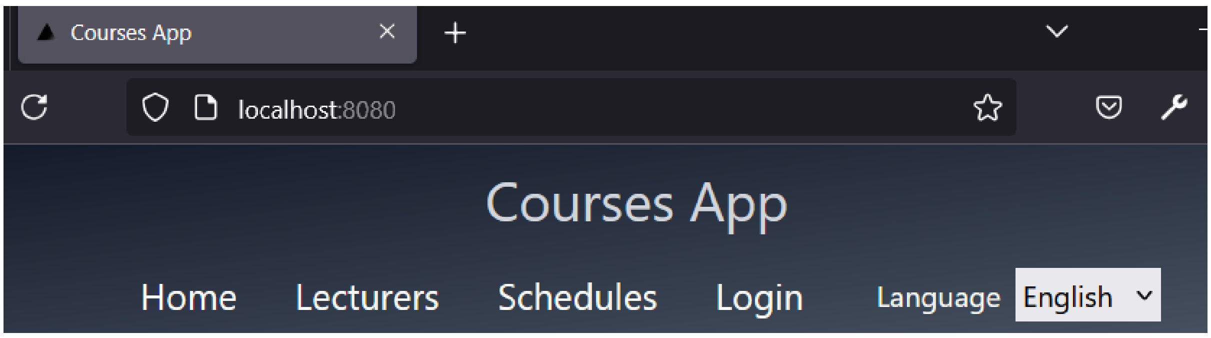
2.3 Language picker

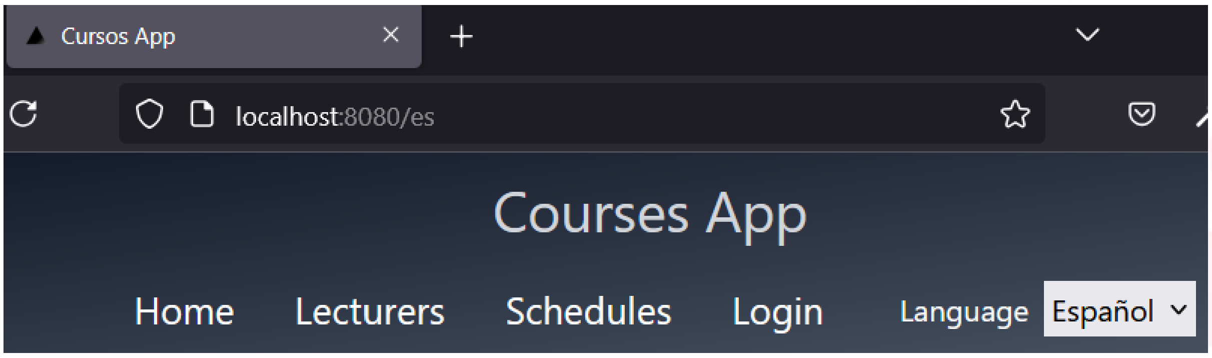
There is the basis of a language picker component in the file *components/language/Language.tsx*

Complete that file by writing the handleLanguageChange function. Then add that Language component to the Header component.

2.4 Test it out

The title of the page (as seen in the tab of your browser) should change when you change languages.




Cursos App
localhost:8080/es
Courses App
Home
Lecturers
Schedules
Login
Language
Español

3 Logging in

3.1 localStorage

Unlike in the theory or the clips we will be using the localStorage to store the token. This means that a user does not have to log in again after the browser has been closed. Up until the expiry of the token, at least.

3.2 Login page

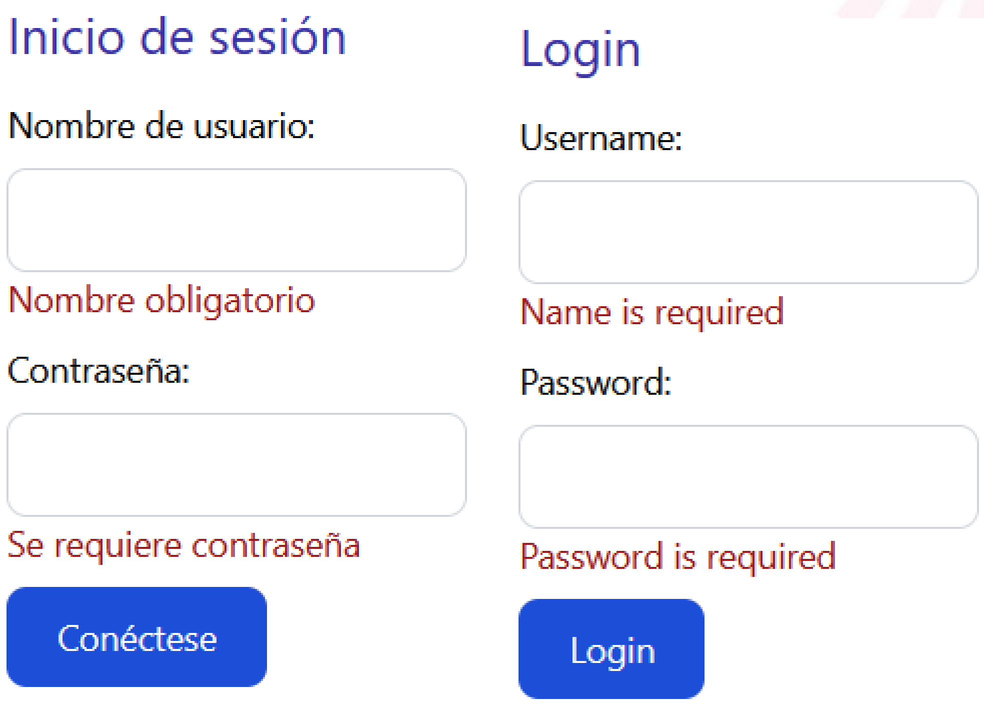
On the page itself: add the getServerSideProps method that loads the translations.

On the UserLoginForm component:

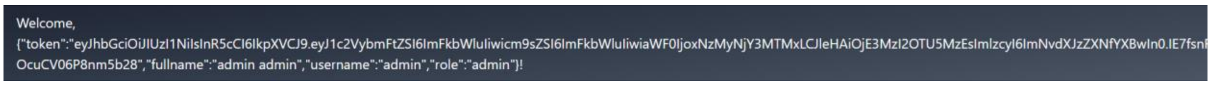
- in the handleSubmit function, use the UserService.logInUser() function to log in
- if the response has the status 200: save User object in the localStorage , with the same key as before, that contains: token, fullname, username, and role of the logged in user
- replace hardcoded text with translations. To save you typing , copy and paste these translations:

```
"general.error": "An error has occurred. Please try again later.",
"login": {
    "title": "Login",
    "button": "Login",
    "success": "Login succesful. Redirecting to homepage...",
    "validate": {
      "name": "Name is required",
      "password": "Password is required"
    },
    "label": { "username": "Username:", "password": "Password:" }
  },
```

```
"general.error": "Se ha producido un error. Vuelva a intentarlo más tarde.",
"login": {
    "title": "Inicio de sesión",
    "button": "Conéctese",
    "success": "Inicio de sesión exitoso. Redirigiendo a la página de inicio...",
    "validate": {
      "name": "Nombre obligatorio",
      "password": "Se requiere contraseña"
    },
    "label": { "username": "Nombre de usuario:", "password": "Contraseña:" }
  },
```



Try logging in:



```
Welcome,
{"token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6ImFkbWluIiwicm9sZSI6ImFkbWluIiwiaWF0IjoxNzMyNjY3MTMxLCJleHAiOjE3MzI2OTU5MzEsImlzcyI6ImNvdXJzZXNfYXBwIn0.IE7fsn
OcuCV06P8nm5b28","fullname":"admin admin","username":"admin","role":"admin"}!
```

Earlier, the key referenced a string. Now it is referencing an object. In every useState that reads from the localstorage we need to change how it is being read, starting with fixing the header.

3.3 Fix the Header

To prevent carpal tunnel, translations to copy-paste:

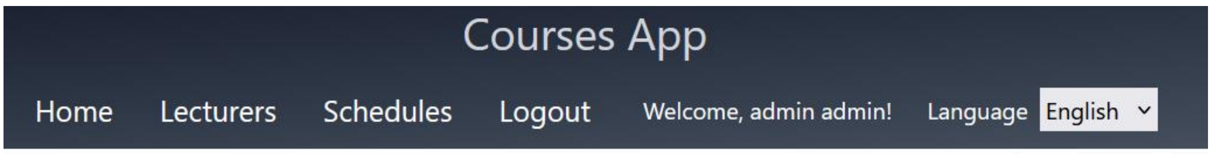
```
  "header": {
    "welcome": "Welcome",
    "nav": {
      "home": "Home",
      "lecturers": "Lecturers",
      "schedules": "Schedules",
      "login": "Login",
      "logout": "Logout"
    }
```

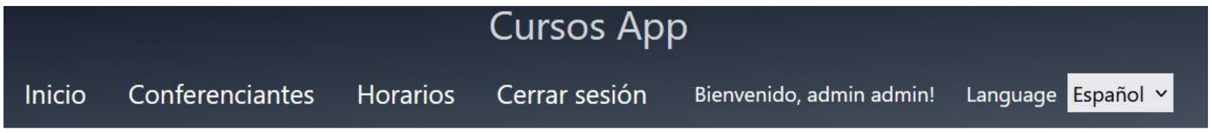
```
  "header": {
    "welcome": "Bienvenido",
    "nav": {
      "home": "Inicio",
      "lecturers": "Conferenciantes",
      "schedules": "Horarios",
      "login": "Conéctese",
      "logout": "Cerrar sesión"
    }
```

Change the hard-coded text to translations.

For the useState hook: change the type to the Usertype, from @types.

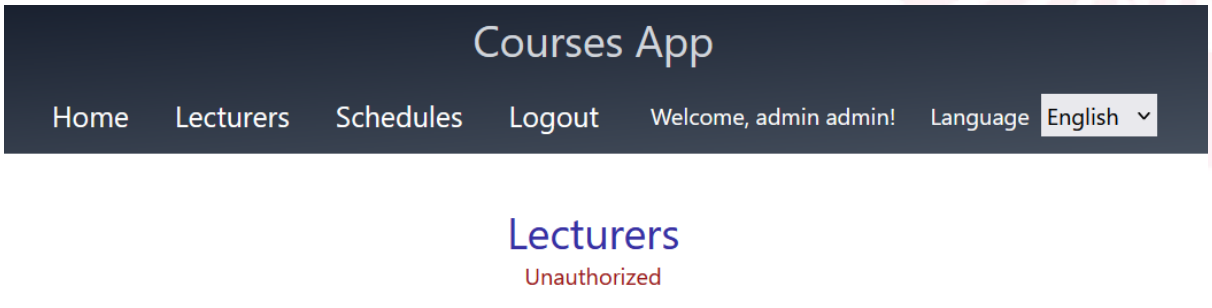
As mentioned earlier, the method getItem(key) on localStorage returns a string, a json string in this case. This means that in the useEffect() hook, you will need to parse this string.





4 Using a protected route

Even logged in, the lecturers page looks like this:



Courses App

Home Lecturers Schedules Logout Welcome, admin admin! Language English

Lecturers

Unauthorized

Fix this page:

- fix all hardcoded tekst so it can be translated
- make sure a request is done to the backend containing the token (from localStorage)
- handle a possible 401 response by showing an error message

5 Extra challenges:

This part is not mandatory, but it is a good way to prepare you for the exam.

- Try adding a new language to the app. (Dutch locale is ‘nl’)
- fix the schedules page so it takes the role of the user into account.