Full-stack software development

Lab 03: React and Next.js introduction

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

Contents

1 Introduction

To get started, accept the assignment on https://classroom.github.com/a/bCpKsNE9.

Install and start the back-end in a terminal.

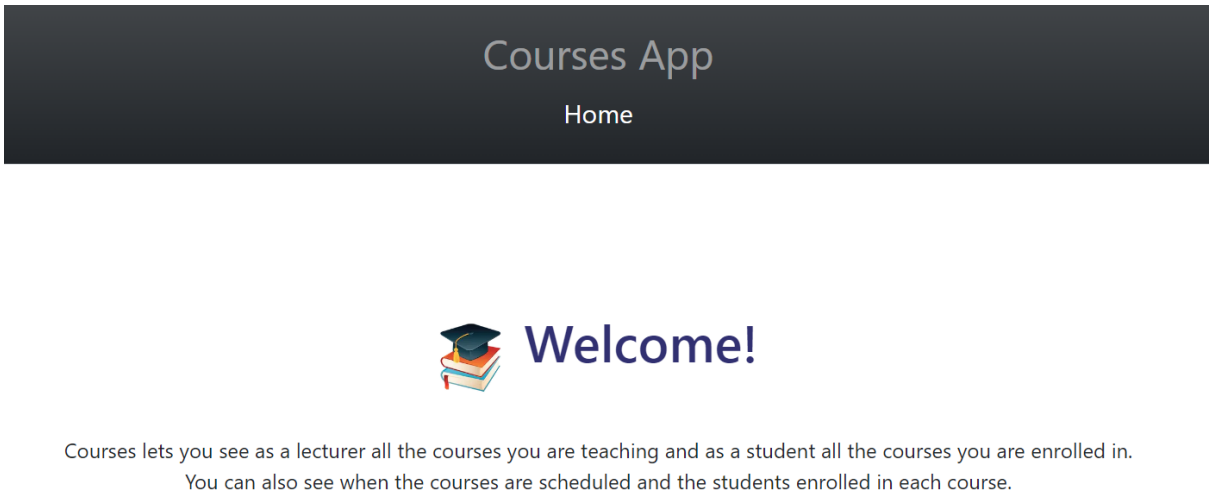
For the front-end, first create a new *.env* file in the root folder. Add the following content:

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

In a second terminal, install and start the front-end:

```
npm install
npm start
```

In your browser, navigate to http://localhost:8080, which should look like this:



2 Lecturers overview

2.1 Create a new page

1) Create a lecturer overview page that is accessible on http://localhost:8080/lecturers. Since we are still using the Pages Router in Next.js (not the App router!) it is sufficient to create a folder “lecturers” with an *index.tsx* in “pages” to register this new route.
2) Copy-paste the code below in this file, so you have a basic skeleton of a React component.

```
const Lecturers: React.FC = () => {

  return (
    <>
      <Head>
        <title>Lecturers</title>
      </Head>
      <main className="d-flex flex-column justify-content-center align-items-center">
        <h1>Lecturers</h1>
        <section>
          <h2>Lecturers overview</h2>
        </section>
      </main>
    </>
  );
};

export default Lecturers;
```

3) Add a link “Lecturers” in the navigation menu that leads to the new lecturers page. The navigation menu is rendered by a React component. Find and extend this component. Start by looking at the main index file of our app.
4) Add this same component to the lecturers overview page, right above the main content. The front-end automatically restarts with every change. Check your changes in the browser.

2.2 Fetch and render lecturer data

In this section, we'll add an overview table to our overview page that lists all lecturers.

1) For readability and maintainability, we encapsulate fetch logic that interacts with a Rest API in separate services. Open *LecturerService.ts* and implement the *getAllLecturers* function. Our back-end is running on http://localhost:3000. This path should be configured in a variable in the *.env* file, which you access by calling:

```
process.env.NEXT_PUBLIC_API_URL
```

2) On top of the Lecturers overview component, add new state for storing retrieved lecturer data. This variable should be initialized with an empty array of lecturers.
3) Implement a function *getLecturers* that calls the LecturerService to retrieve all lecturers. Call the setLecturers state setter function to store the result in state.
4) Implement a *useEffect* that is only executed after the first render of this component. Call the *getLecturers* function.
5) When the lecturers state field is not empty, a table must be rendered that lists all the lecturers. It is best practice to encapsulate this in a separate reusable component. We have already prepared this for you in *components/lecturers/LecturerOverviewTable*. Call this component (after checking the lecturers state field) and pass the lecturers as a prop to this component.
6) Test your changes and make sure the lecturers table is rendered correctly.

2.3 Render courses of the selected lecturer

If we click on an lecturer (a row) in the lecturer table, a second table should be displayed below it with the courses the lecturer teaches.

1) In *components/courses* create a new CourseOverviewTable component that expects one lecturer as a Prop. For each course of the lecturer, render a table row (<tr>) containing the name, description, phase and credits of the course.
2) The courses table can only be rendered when a lecturer is selected in the lecturers table. We don't have this information in the overview component, since all logic concerning the lecturers table is encapsulated in the LecturerOverviewTable

component. We need a “notification system” so that the table component can notify the overview component when a lecture is selected. To start, add an extra state field “selectedLecturer” in the lecturer overview component. This state can contain either one Lecturer or *null* (when no lecturer is selected).

3) We need to pass a *callback function* to LecturerOverviewTable. This component can call that callback function whenever a lecturer is selected. Extend the props of LecturerOverviewTable so that it receives a *selectLecturer* function:

```
selectLecturer: (lecturer: Lecturer) => void;
```

4) In the onClick event of a lecturer table row, replace the empty placeholder function (()=>{}) with a call to the *selectLecturer* function and pass the Lecturer that is currently rendered as a parameter. Don’t forget to destructure the callback function in the Props parameter of the component.
5) In the lecturers overview component, the declaration of LecturerOverviewTable now expects an extra attribute “selectLecturer”, to pass the callback function as a prop. Simply pass the *setSelectedLecturer* state setter function, so that state is immediately updated with the lecturer that was clicked on in the table component.
6) Now render the CourseOverviewTable below the LecturersOverviewTable. Only render this table when *selectedLecturer* contains a value. Pass the *selectedLecturer* as a prop to the CourseOverviewTable. Above the table, add an <h2> title with the text: “Courses taught by …” (replace … with the first name of the lecturer).
7) Test your changes.

3 Dynamic routes

1) Create a page component that supports requests on http://locahost:8080/lecturers/{lecturerId}.
2) This component doesn't have any props. Instead, it must lookup the lecturerId in the URL (a path variable) using the Next.js router.
3) In LecturerService, implement a *getLecturerById* function that calls the correct API endpoint in the back-end.
4) In the component, implement a *useEffect* that calls the service whenever the *lecturerId* changes.
5) If there exists a lecturer with the given ID, render the component *LecturerDetails* on this page, which can be found in *components/lecturers.*
6) Test your code by directly navigating in the browser to http://locahost:8080/lecturers/1 or http://locahost:8080/lecturers/2.