

# Laboratorul 09: React Router & Request-uri HTTP

---

## Premisa

---

În cadrul acestui laborator, vom accesa backend-ul creat la Laboratorul 5 [https://gitlab.com/tehnologiiweb/lab5]. Cel mai recomandat este să îl clonați din nou și să dați docker-compose up. Atât serverul cât și baza de date rulează amândouă în Docker.

## 1. React Router

---

React Router DOM este un API creat pentru React pentru a "simula" un browser history pentru aplicația noastră. Deoarece aplicația noastră este un Single Page Application (SPA), browser-ul nostru nu navighează prin fișiere HTML.

### 1.1. Instalare React Router

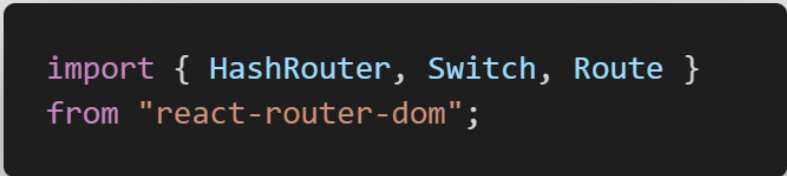
Pentru a instala react-router-dom trebuie să executăm următoarea comandă:

```
npm install react-router-dom
```

### 1.2. Utilizarea React Router

Utilizarea React Router-ului se realizează în 3 pași:

- Adăugarea dependințelor din react-router-dom (**HashRouter**, **Switch** și **Route**)



```
import { HashRouter, Switch, Route }  
from "react-router-dom";
```

- Înglobarea tuturor componentelor din cadrul aplicației în **<HashRouter>** și **<Switch>**
- Definirea rutelor folosind **<Route>**

```
const App = () => {  
  return (  
    <HashRouter basename="/">  
      <Switch>  
        <Route exact path="/" component={() => <h1>Index</h1>} />  
        <Route path="/login" component={Login} />  
  
        <Layout>  
          <Route exact path="/book" component />  
        </Layout>  
      </Switch>  
    </HashRouter>  
  );  
};
```

Schimbarea continutului dinamic se realizeaza prin folosirea componentei **<Link>** din react-router-dom

```
<Link to="/">Go Home</Link>
```

## 2. Axios si Request-uri HTTP

Axios este o biblioteca JavaScript folosita pentru a face **Request-uri HTTP** din **Node.Js** sau **XMLHttpRequest** din browser, cerinta principala fiind compatibilitatea cu **ES6 Promise API**.

### 2.1. Caracteristici principale

- **XMLHttpRequests** [<https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest>] din browser
- Request-uri **HTTP** [<https://nodejs.org/api/http.html>] din **Node.JS**
- Suporta **Promise API** [[https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\\_Objects/Promise](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise)]
- Interceptare request-uri si responses

### 2.2. Instalare axios

```
npm install axios
```

## 2.3. HTTP Get

```
axios.get("localhost/api/v1/authors")
  .then(response => processAuthors(response.data))
  .catch(error => console.error(error));

axios.get(`localhost/api/v1/authors/${id}`)
  .then(response => processAuthor(response.data))
  .catch(error => console.error(error));
```

## 2.4. HTTP Post

```
axios.post("localhost/api/v1/users/register", {
  username: "myUsername",
  password: "myAwesomePassword"
}).then(response => handleSuccess(response.data))
  .catch(error => console.error(error));

axios.post("localhost/api/v1/authors", {
  firstName: "Author First Name",
  lastName: "Author Last Name"
}).then(response => handleAddAuthorSuccess(
  response.data))
  .catch(error => console.error(error));
```

## 2.5. HTTP Put

```
axios.put(`localhost/api/v1/authors/${id}`, {  
  firstName: "Author First Name Updated",  
  lastName: "Author Last Name Updated"  
}).then(response => handleAddAuthorSuccess(  
  response.data))  
.catch(error => console.error(error));
```

## 2.6. HTTP Delete

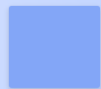
```
axios.delete(`localhost/api/v1/authors/${id}`)  
  .then(response => handleDeletedAuthor(response.  
    data))  
  .catch(error => console.error(error));
```

## 2.7. HTTP Request cu Headers

```
axios.post("url", data, {  
  headers: {  
    Authorization: `Bearer ${token}`,  
  },  
});  
  
axios.get("url", {  
  headers: {  
    Authorization: `Bearer ${token}`,  
  },  
});
```

Pe langa HOST (localhost in cazurile de mai sus), trebuie mentionat si portul si protocolul. Pentru acest laborator o sa fie `http://localhost:3000` [`http://localhost:3000`]

## Exercitii



#83A6F7



#3F3D56



#000000



#ffffff



Border-Radius: 5px

Box-Shadow: 0 3px 6px 0 rgba(0, 0, 0, .16)

Raleway

Lorem ipsum dolor sit amet,

Lorem ipsum dolor sit amet,

Lorem ipsum dolor sit amet,

Raleway

Lorem ipsum dolor sit amet,

Lorem ipsum dolor sit amet,

Lorem ipsum dolor sit amet,

Xd

**BOOKISH**


Already have an account? [Login](#)

**Create account**

Username

Email

Password



UPB 2021

Laborator #9 PWEB


**BOOKISH**

Don't have an account? [Register](#)

**Login**

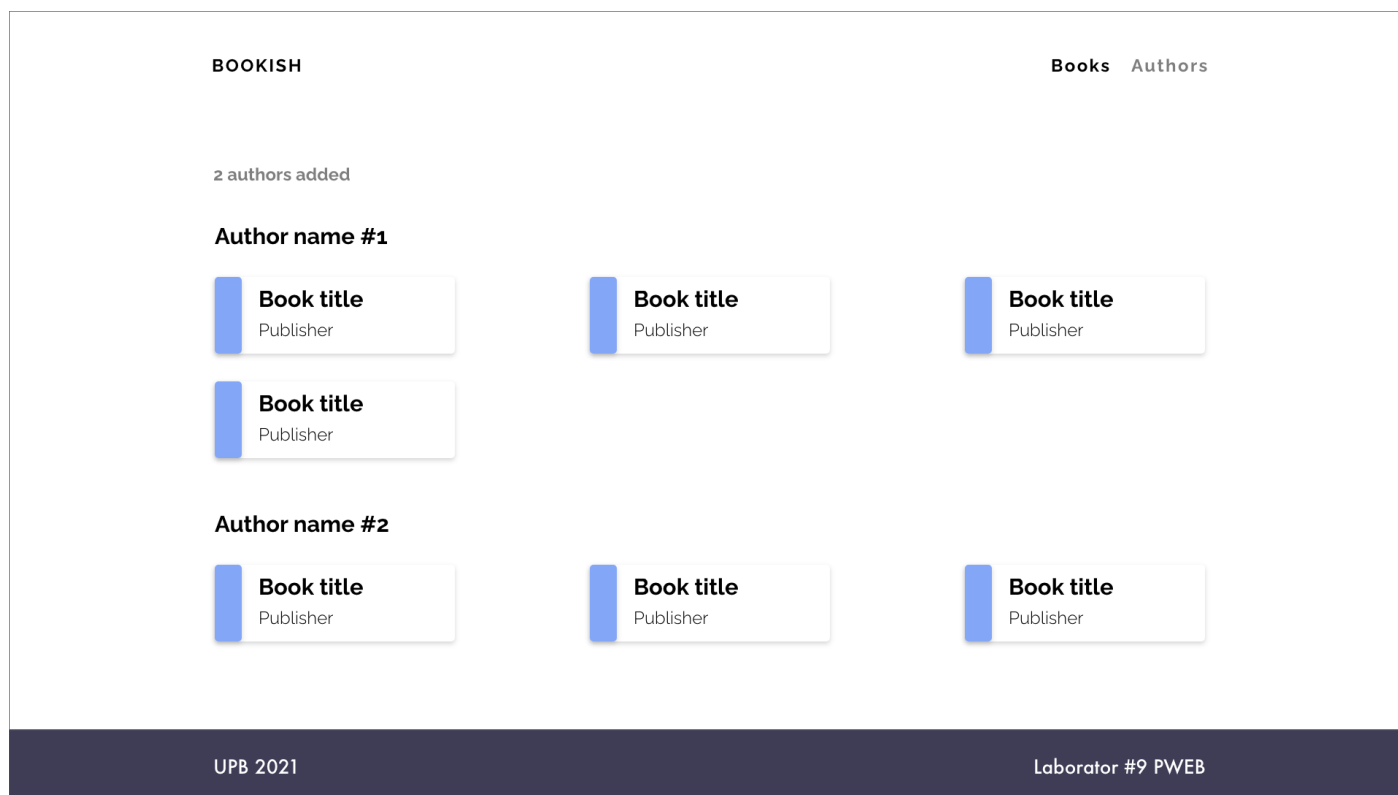
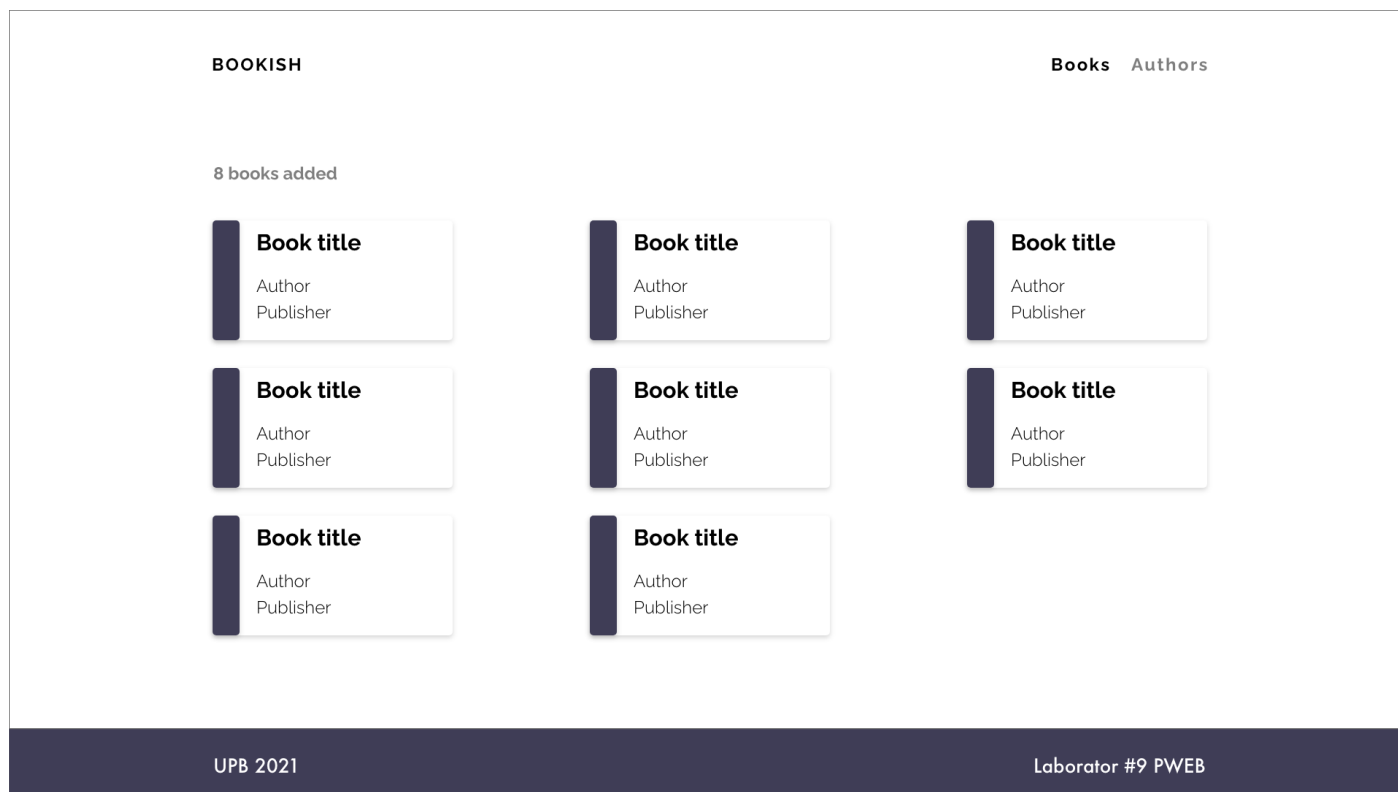
Email

Password



UPB 2021

Laborator #9 PWEB



Pentru vizualizarea detaliilor, puteti intra pe acest link [<https://xd.adobe.com/view/7ba0a1d7-2e62-45d7-b27b-f74a16e55876-2ef3/screen/47f00dd6-0679-4cbb-b4ae-abe03f8445fc/specs/>] si alegand optiunea </> din meniul vertical aflat in dreapta.

Pentru ilustratie puteti folosi Undraw [<https://undraw.co>]. Ilustratia folosita in design poate fi gasita cautand 'Tree swing'.

1. Creati o componenta **Autentificare** care sa contina un formular cu **doua campuri text si un buton de Submit** respectiv **Register**.

Componenta Autentificare va fi responsabila de a trimite un request de tip POST catre API, avand in body datele necesare autentificarii respectiv inregistrarii utilizatorului.

```
{
  username: "admin",
  password: "admin"
}
```

Raspunsul primit de la server, in cazul login, va contine token-ul JWT. Stocarea token-ului se face in browser, folosind **localStorage**.

Stocare token in **localStorage**

```
localStorage.setItem("token", jwt_token);
```

Citirea token-ului din **localStorage**

```
localStorage.getItem("token")
```

2. Creati doua componente, **Book** si **Author**, care sa afiseze detaliile despre fiecare carte, respectiv autor. Stilizarea poate sa urmeze design-ul de mai sus. Nu este obligatorie copierea exacta a design-ului, importanta este afisarea datelor.

3. Creati o componenta **BooksList** care sa contina o lista cu toate cartile existente, respectiv **AuthorsList** care sa contina o lista cu toti autorii existenti in baza de date. Pentru afisarea tuturor cartilor, respectiv autorilor, se vor folosi componentele create anterior, **Book** si **Author**. Aceste componente pot fi reprezentate de paginile Books si Authors din design. Din nou, nu este obligatorie respectarea design-ului.

4. Componentele **BooksList** si **AuthorsList** trebuie sa trimita cate un request de HTTP Get pentru a aduce toate intrarile din baza de date.

Cererile catre server pot fi date fie la apasarea pe un buton, fie atunci cand se incarca componenta (hint, `useEffect`)