

2022

Uwierzytelnianie w React



Szymon Śliwa 155471

ISI GR 1

2022-05-12

Instalacja pakietu React Router

React Router jest pakietem, który umożliwia renderowanie podstron naszej aplikacji w łatwy sposób. Tworzymy nową aplikację React poleceniem:

```
npx create-react-app nazwa_projektu
```

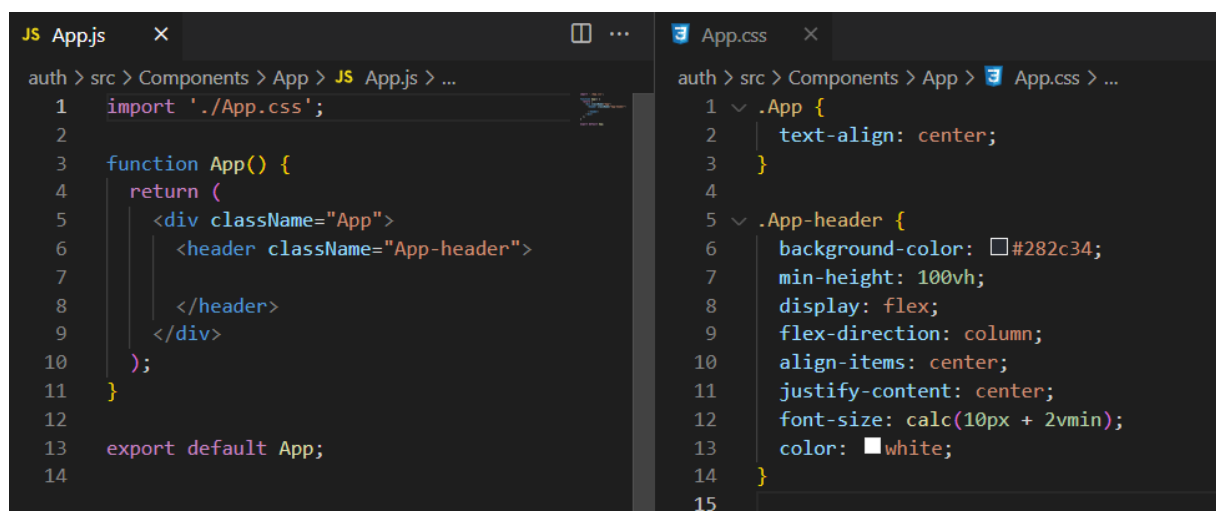
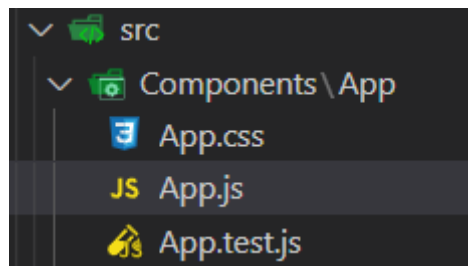
Pobieranie paczki odbywa się przez komendę menadżera pakietów (npm lub yarn, w zależności od tego którego używamy). Należy to zrobić będąc w folderze aplikacji.

```
npm install react-router-dom@5.2.0
```

```
yarn add react-router-dom@5.2.0
```

Czyszczenie projektu i tworzenie struktury plików

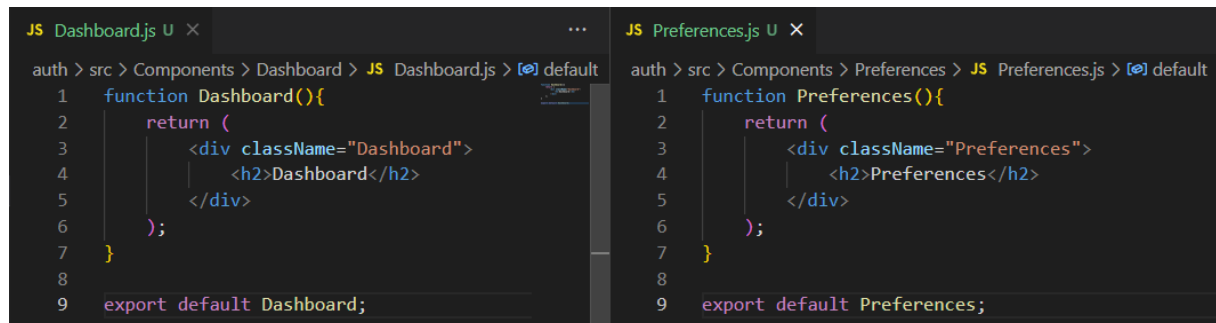
Czyścimy projekt z niepotrzebnych plików. Usuwamy plik logo.svg, oraz czyścimy zbędną zawartość plików App.js oraz App.css, które następnie wraz z plikiem App.test.js przenosimy do folderu „src/Components/App”; uprzednio go tworzymy. Żeby wszystko poprawnie działało, musimy zmienić ścieżkę w index.js do naszego komponentu App.



Projekt ukończony do tego momentu możemy odnaleźć na branchu „main” mojego repozytorium: [szymon242820/react_auth_presentation/tree/main](https://github.com/szymon242820/react_auth_presentation/tree/main)

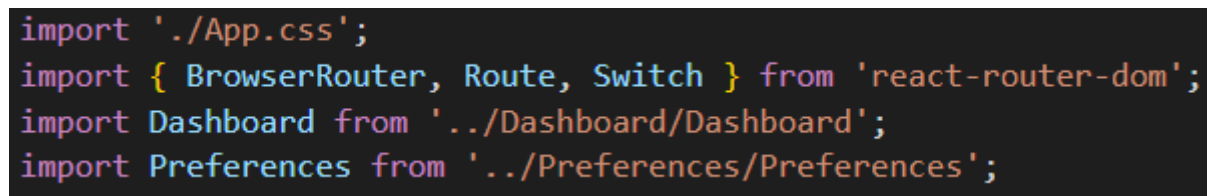
Nowe komponenty

Do folderu Components dodajemy dwa podfoldery Dashboard oraz Preferences, a w nich odpowiadających im nazwą pliki z rozszerzeniem .js. Zawartość plików powinna wyglądać następująco:



```
JS Dashboard.js U x ... JS Preferences.js U x
auth > src > Components > Dashboard > JS Dashboard.js > [e] default auth > src > Components > Preferences > JS Preferences.js > [e] default
1 function Dashboard(){ 1 function Preferences(){
2   return ( 2   return (
3     <div className="Dashboard"> 3     <div className="Preferences">
4       <h2>Dashboard</h2> 4       <h2>Preferences</h2>
5     </div> 5     </div>
6   ); 6   );
7 } 7 }
8 8
9 export default Dashboard; 9 export default Preferences;
```

W następnym kroku importujemy nasze nowe komponenty do naszej aplikacji (App.js). Zaimportujemy również moduły służące do stworzenia systemu routingu z wcześniej zainstalowanej paczki react-router-dom. W taki sposób powinny wyglądać nasze importy:



```
import './App.css';
import { BrowserRouter, Route, Switch } from 'react-router-dom';
import Dashboard from '../Dashboard/Dashboard';
import Preferences from '../Preferences/Preferences';
```

W samej funkcji App dodajemy implementację routera. Wygląda ona następująco:



```
function App() {
  return (
    <BrowserRouter>
      <div className="App">
        <header className="App-header">
          <Switch>
            <Route exact path="/">
              <h2>Main page</h2>
            </Route>
            <Route path="/dashboard">
              <Dashboard />
            </Route>
            <Route path="/preferences">
              <Preferences />
            </Route>
          </Switch>
        </header>
      </div>
    </BrowserRouter>
  );
}
```

Projekt ukończony do tego momentu możemy odnaleźć na branchu „1_new_components” mojego repozytorium: [szymon242820/react_auth_presentation/tree/1_new_components](https://github.com/szymon242820/react_auth_presentation/tree/1_new_components)

Formularz logowania

Z racji tego, że okropnie nie lubię standardowego wyglądu formularzy, a jednocześnie nie jestem wybitnie uzdolnioną w temacie designu osobą, postanowiłem dodać od siebie synchronizację Bootstrapa z Reactem 😊. W tym celu musimy doinstalować dwie paczki: bootstrap i react-bootstrap.

```
npm install bootstrap@4.5.2
```

```
npm install react-bootstrap@1.3.0
```

Dodatkowo do pliku index.js dodajemy import pliku .css aby wszystkie klasy bootstrapa działały poprawnie!

```
import 'bootstrap/dist/css/bootstrap.min.css';
```

Kiedy już to zrobimy, należy stworzyć nowy komponent o nazwie Login. (w folderze Components dodajemy folder Login, a w nim plik Login.js). Struktura tego pliku niewiele będzie się różniła od poprzednich klas, jedyną różnicą jest to, że użyjemy kontenerów bootstrapa oraz będziemy zwracać formularz. Zawartość pliku Login.js wygląda następująco:

```
JS Login.js U X
auth > src > Components > Login > JS Login.js > ...
1  import { Form, Col, Button } from 'react-bootstrap';
2
3  function Login(){
4    return (
5      <div className="Login">
6        <h1>Please Log In</h1>
7        <Form>
8          <Form.Group controlId="username">
9            <Form.Label column>
10             Username
11            </Form.Label>
12            <Col>
13              <Form.Control type="text" placeholder="Username" />
14            </Col>
15          </Form.Group>
16          <Form.Group controlId="password">
17            <Form.Label column>
18             Password
19            </Form.Label>
20            <Col>
21              <Form.Control type="password" placeholder="Password" />
22            </Col>
23          </Form.Group>
24          <Form.Group>
25            <Col>
26              <Button type="submit">Sign in</Button>
27            </Col>
28          </Form.Group>
29        </Form>
30      </div>
31    );
32  }
33
34  export default Login;
```

Tak samo jak w przypadku pozostałych komponentów importujemy go w pliku App.js. Importujemy również React z paczki react, ponieważ będziemy używali funkcji useState (importowanie Reacta nie było wstępnie potrzebne do działania aplikacji, ponieważ jest on importowany w pliku index.js).

```
import './App.css';
import React from 'react';
import { BrowserRouter, Route, Switch } from 'react-router-dom';
import Dashboard from '../Dashboard/Dashboard';
import Preferences from '../Preferences/Preferences';
import Login from '../Login/Login';
```

Na początku funkcji tworzymy nasz useState, który będzie przechowywał token logowania, oraz funkcję sprawdzającą czy jesteśmy zalogowani.

```
const [token, setToken] = React.useState();

if(!token){
  return (
    <div className="App">
      <header className="App-header">
        <Login setToken={setToken} />
      </header>
    </div>
  );
}
```

Projekt ukończony do tego momentu możemy odnaleźć na branchu „2_login_form_and_token” mojego repozytorium: [szymon242820/react_auth_presentation/tree/2_login_form_and_token](https://github.com/szymon242820/react_auth_presentation/tree/2_login_form_and_token)

```
npm install --save-dev express cors
```

Podpięcie pod backend (express server)

W normalnym przypadku podpiinalibyśmy się teraz do backendu napisanego w Django, SpringBoot lub innym frameworku, ale na potrzeby tego poradnika zrobimy to „po najmniejszej linii oporu” robiąc lokalne API przy użyciu express. W tym celu dodajemy nową paczkę komendą:

W folderze głównym projektu (/auth) tworzymy plik server.js o następującej zawartości:

```
JS server.js U X
auth > JS server.js > ...
1  const express = require('express');
2  const cors = require('cors');
3  const app = express();
4
5  app.use(cors());
6
7  app.use('/login', (req, res) => {
8    res.send({
9      token: 'test123'
10   });
11 });
12
13 app.listen(8080, () => console.log('API is running on http://localhost:8080/login'));
```

Z racji tego, że w poradniku skupiamy się na frontendzie, nie będę opisywał linia po linii co dany kod robi. Ważne jest to, że na porcie 8080 naszego lokalnego hosta udostępnia przy użyciu cors json zawierający nasz przykładowy token.

Aby uruchomić serwer należy w nowym terminalu wpisać polecenie ‘node server.js’.

Teraz, gdy posiadamy „backend” możemy rozbudować nasze logowanie. Posłużymy się w tym celu PropTypem i ustalimy, że logowanie jest wymagane do używania aplikacji. W kolejnych krokach importujemy w pliku Login.js PropTypes z paczki prop-types, jako argument funkcji dodajemy {setToken} oraz pomiędzy funkcją login, a exportem defaultowego loginu ustawiamy Login.propTypes.

```
import PropTypes from 'prop-types';

function Login({setToken}){
```

```
    </Form>
  </div>
);
}

Login.propTypes = {
  setToken: PropTypes.func.isRequired
}

export default Login;
```

Importujemy również Reacta z paczki react i tworzymy useState dla username i password na początku naszego komponentu. Implementujemy setery na onChange naszych inputów.

```
const [username, setUsername] = React.useState();
const [password, setPassword] = React.useState();
```

```
<Form.Control type="text" placeholder="Username" onChange={e=>setUsername(e.target.value)} />
```

```
<Form.Control type="password" placeholder="Password" onChange={e=>setPassword(e.target.value)} />
```

Nad komponentem tworzymy asynchroniczną funkcję imitującą logowanie do prawdziwego backendu, której prawdziwym celem jest pobranie tokenu logowania z naszego lokalnego serwera Node, który na początku tego punktu uruchomiliśmy.

```
async function loginUser(credentials){
  return fetch('http://localhost:8080/login', {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json'
    },
    body: JSON.stringify(credentials)
  }).then(data => data.json())
}
```

W komponencie, pod naszymi useState tworzymy funkcję, która obsłuży przycisk logowania oraz implementujemy ją jako onSubmit naszego formularza.

```
const handleSubmit = async e => {
  e.preventDefault();
  const token = await loginUser({
    username,
    password
  });
  setToken(token);
}

return (
  <div className="Login">
    <h1>Please Log In</h1>
    <Form onSubmit={ handleSubmit }>
```

Jak wcześniej wspomniałem, takie logowanie jest jedynie prowizoryczne (bo pobiera token bez sprawdzania poprawności naszego loginu i hasła).

Projekt ukończony do tego momentu możemy odnaleźć na branchu „3_express_server” mojego repozytorium: [szymon242820/react_auth_presentation/tree/3_express_server](https://github.com/szymon242820/react_auth_presentation/tree/3_express_server)

Przechowywanie tokenu w sesji

Temat zaczniemy od usunięcia `useState` dla tokenu z pliku `App.js`, a w jego miejsce stworzymy nowy customowy token hook. W tym celu tworzymy nowy plik `useToken.js` w folderze `src/Components/App`, obok pliku `App.js`. Zawartość tego pliku wygląda następująco:

```
JS useToken.js U X
auth > src > Components > App > JS useToken.js > ...
1  import React from 'react';
2
3  function useToken(){
4      const getToken = () => {
5          const tokenString = sessionStorage.getItem('token');
6          const userToken = JSON.parse(tokenString);
7          return userToken?.token
8      }
9
10     const [token, setToken] = React.useState(getToken());
11
12     const saveToken = userToken => {
13         sessionStorage.setItem('token', JSON.stringify(userToken));
14         setToken(userToken.token);
15     }
16
17     return {
18         setToken: saveToken,
19         token
20     }
21 }
22
23 export default useToken;
```

W pliku `App.js` importujemy nasz `useToken` i używamy go do stworzenia tokenu w Aplikacji.

```
import useToken from './useToken';

function App() {
    const { token, setToken } = useToken();
```

Dzięki temu, nasz token będzie przechowywany w sesji, co spowoduje „zapamiętanie” naszego logowania i pozwoli na używanie „aplikacji”.

Projekt ukończony do tego momentu możemy odnaleźć na branchu

„4_session_storage_and_local_storage” mojego repozytorium:

[szymon242820/react_auth_presentation/tree/4_session_storage_and_local_storage](https://github.com/szymon242820/react_auth_presentation/tree/4_session_storage_and_local_storage)