Zintegrowany Program Rozwoju Politechniki Lubelskiej - część druga

POLITECHNIKA LUBELSKA
WYDZIAŁ ELEKTROTECHNIKI I INFORMATYKI

KIERUNEK STUDIÓW
INFORMATYKA

MATERIAŁY DO ZAJĘĆ LABORATORYJNYCH

Szkielety programistyczne w aplikacjach internetowych

dr Mariusz Dzieńkowski

Lublin 2022

Fundusze Europejskie Wiedza Edukacja Rozwój
Rzeczpospolita Polska
Unia Europejska Europejski Fundusz Społeczny

LABORATORIUM 7. IMPLEMENTACJA MECHANIZMÓW UWIERZYTELNIANIA I AUTORYZACJI

Cel laboratorium:

Nabycie umiejętności posługiwania się biblioteką React podczas implementacji aplikacji funkcjonującej po stronie klienta.

Zakres tematyczny zajęć:

- Tworzenie i uruchamianie aplikacji opartej na bibliotece React.
- Implementacja komponentów w React za pomocą funkcji oraz klas.
- Sposoby przekazywania i odbierania danych przez komponenty (*props* i *state*).
- Komponenty bezstanowe i ze stanem.
- Obsługa formularzy przy pomocy biblioteki React.

Pytania kontrolne:

a) Czym jest i do czego służy JSX (JavaScript XML)?
b) Przedstaw strukturę aplikacji opartej na bibliotece React.
c) Opisz strukturę komponentu zbudowanego na bazie funkcji oraz klasy.
d) Czym są i do czego służą właściwości (*props)* i stany (*state)*?

Zadanie 7.1. Uwierzytelnianie użytkowników

Etap 1. Konfiguracja szkieletu Express i bazy MongoDB

1. Utworzenie folderu aplikacji oraz podfolderu server, w którym zostanie wygenerowany plik package.json
   npm init --yes
2. Instalacja pakietów: express, dotenv, mongoose, nodemon, cors
3. W części "scripts" pliku package.json skonfigurować uruchamianie serwera:

```
"scripts": {
   "start": "nodemon index.js"
 },
```

4. Utworzenie pliku serwera index.js z poniższą zawartością

```
require('dotenv').config()
const express = require('express')
const app = express()
const cors = require('cors')

//middleware
app.use(express.json())
app.use(cors())
const port = process.env.PORT || 8080
```

```
app.listen(port, () => console.log(`Nasłuchiwanie na porcie ${port}`))
```

5. Utworzenie pliku db.js z następującym kodem:

```
const mongoose = require("mongoose")

module.exports = () => {
	const connectionParams = {
		useNewUrlParser: true,
		useUnifiedTopology: true,
	}
	try {
		mongoose.connect(process.env.DB, connectionParams)
		console.log("Connected to database successfully")
	} catch (error) {
		console.log(error);
		console.log("Could not connect database!")
	}
}
```

6. Utworzenie w katalogu server pliku .env zawierającego ustawienia:

```
DB=mongodb://localhost/auth_react
JWTPRIVATEKEY=abcdefg
SALT=10
```

7. Ustanowienie połączenia z bazą danych w pliku serwera index.js

```
const connection = require('./db')
connection()
```

8. Uruchomienie serwera i sprawdzenie połącznia z bazą danych:
   npm start

Etap 2. Utworzenie modelu User

9. Zainstalowanie pakietów jsonwebtoken, joi, joi-password-complexity

10. Utworzenie katalogu models zawierającego plik user.js

```
const mongoose = require("mongoose")
const jwt = require("jsonwebtoken")
const Joi = require("joi")
const passwordComplexity = require("joi-password-complexity")

const userSchema = new mongoose.Schema({
	firstName: { type: String, required: true },
	lastName: { type: String, required: true },
	email: { type: String, required: true },
	password: { type: String, required: true },
})
userSchema.methods.generateAuthToken = function () {
```

```
        const token = jwt.sign({ _id: this._id }, process.env.JWTPRIVATEKEY, {
                expiresIn: "7d",
        })
        return token
}
const User = mongoose.model("User", userSchema)
```

11. W pliku user.js dodanie walidacji danych z formularza za pomocą joi:

```
const validate = (data) => {
        const schema = Joi.object({
                firstName: Joi.string().required().label("First Name"),
                lastName: Joi.string().required().label("Last Name"),
                email: Joi.string().email().required().label("Email"),
                password: passwordComplexity().required().label("Password"),
        })
        return schema.validate(data)
}

module.exports = { User, validate }
```

Etap 3. Trasa dla rejestracji użytkownika

12. Instalacja pakietu bcrypt do haszowania hasła
13. Utworzenie folderu routes w katalogu głównym serwera zawierającego plik users.js

```
const router = require("express").Router()
const { User, validate } = require("../models/user")
const bcrypt = require("bcrypt")

router.post("/", async (req, res) => {
        try {
                const { error } = validate(req.body)
                if (error)
                        return res.status(400).send({ message: error.details[0].message })

                const user = await User.findOne({ email: req.body.email })
                if (user)
                        return res
                                .status(409)
                                .send({ message: "User with given email already Exist!" })

                const salt = await bcrypt.genSalt(Number(process.env.SALT))
                const hashPassword = await bcrypt.hash(req.body.password, salt)

                await new User({ ...req.body, password: hashPassword }).save()
                res.status(201).send({ message: "User created successfully" })
        } catch (error) {
                res.status(500).send({ message: "Internal Server Error" })
        }
```

```
})
module.exports = router
```

Etap 4. Trasa dla logowania

14. W folderze routes utworzyć plik auth.js

```
const router = require("express").Router()
const { User } = require("../models/user")
const bcrypt = require("bcrypt")
const Joi = require("joi")

router.post("/", async (req, res) => {
	try {
		const { error } = validate(req.body);
		if (error)
			return res.status(400).send({ message: error.details[0].message })

		const user = await User.findOne({ email: req.body.email })
		if (!user)
			return res.status(401).send({ message: "Invalid Email or Password" })

		const validPassword = await bcrypt.compare(
			req.body.password,
			user.password
		)
		if (!validPassword)
			return res.status(401).send({ message: "Invalid Email or Password" })

		const token = user.generateAuthToken();
		res.status(200).send({ data: token, message: "logged in successfully" })
		console.log('asfd')
	} catch (error) {
		res.status(500).send({ message: "Internal Server Error" })
	}
})

const validate = (data) => {
	const schema = Joi.object({
		email: Joi.string().email().required().label("Email"),
		password: Joi.string().required().label("Password"),
	})
	return schema.validate(data)
}

module.exports = router
```

Etap 5. Import tras

15. Do pliku index.js należy zaimportować trasy: users i auth

```
const userRoutes = require("./routes/users")
const authRoutes = require("./routes/auth")
...
// routes
app.use("/api/users", userRoutes)
app.use("/api/auth", authRoutes)
```

Etap 6. Aplikacja strony klienta

16. Utworzenie aplikacji w React za pomocą polecenia:
    npx create-react-app client
17. Instalacja pakietów axios, react-router-dom
18. Usunięcie z nowo utworzonego projektu React plików: App.css, App.test.js, logo.svg, setupTest.js
19. Modyfikacja pliku index.js (entry point) do postaci:

```
import React from 'react'
import ReactDOM from 'react-dom'
import './index.css'
import App from './App'
import { BrowserRouter } from 'react-router-dom'

ReactDOM.render(
 <React.StrictMode>
  <BrowserRouter>
   <App />
  </BrowserRouter>
 </React.StrictMode>,
 document.getElementById('root')
)
```

20. Modyfikacja głównego komponentu App:

```
function App() {
 return (
  <div className="App">
   hello world
  </div>
 )
}

export default App
```

Etap 7. Tworzenie komponentów

21. Utworzenie katalogu components
22. Utworzenie katalogu Signup, a w nim plików: index.jsx oraz pliku ze stylami styles.module.css. Plik komponentu powinien zawierać poniższy kod:

```
import { useState } from "react"
import axios from "axios"
```

```
import { Link, useNavigate } from "react-router-dom"
import styles from "./styles.module.css"

const Signup = () => {
	const [data, setData] = useState({
		firstName: "",
		lastName: "",
		email: "",
		password: "",
	})
	const [error, setError] = useState("")
	const navigate = useNavigate()

	const handleChange = ({ currentTarget: input }) => {
		setData({ ...data, [input.name]: input.value })
	}

	const handleSubmit = async (e) => {
		e.preventDefault()
		try {
			const url = "http://localhost:8080/api/users"
			const { data: res } = await axios.post(url, data)
			navigate("/login")
			console.log(res.message)
		} catch (error) {
			if (
				error.response &&
				error.response.status >= 400 &&
				error.response.status <= 500
			) {
				setError(error.response.data.message)
			}
		}
	}

	return (
		<div className={styles.signup_container}>
			<div className={styles.signup_form_container}>
				<div className={styles.left}>
					<h1>Welcome Back</h1>
					<Link to="/login">
						<button type="button"
className={styles.white_btn}>
							Sing in
						</button>
					</Link>
				</div>
				<div className={styles.right}>
```

```
                                    <form className={styles.form_container}
onSubmit={handleSubmit}>
                                        <h1>Create Account</h1>
                                        <input
                                            type="text"
                                            placeholder="First Name"
                                            name="firstName"
                                            onChange={handleChange}
                                            value={data.firstName}
                                            required
                                            className={styles.input}
                                        />
                                        <input
                                            type="text"
                                            placeholder="Last Name"
                                            name="lastName"
                                            onChange={handleChange}
                                            value={data.lastName}
                                            required
                                            className={styles.input}
                                        />
                                        <input
                                            type="email"
                                            placeholder="Email"
                                            name="email"
                                            onChange={handleChange}
                                            value={data.email}
                                            required
                                            className={styles.input}
                                        />
                                        <input
                                            type="password"
                                            placeholder="Password"
                                            name="password"
                                            onChange={handleChange}
                                            value={data.password}
                                            required
                                            className={styles.input}
                                        />
                                        {error && <div
className={styles.error_msg}>{error}</div>}
                                        <button type="submit"
className={styles.green_btn}>
                                            Sing Up
                                        </button>
                                    </form>
                                </div>
                            </div>
                        </div>
```

```
        );
};

export default Signup
```

23. Utworzenie katalogu Login oraz plików index.jsx i styles.module.css. Plik index.jsx powinien zawierać kod:

```
import { useState } from "react"
import axios from "axios"
import { Link } from "react-router-dom"
import styles from "./styles.module.css"

const Login = () => {
        const [data, setData] = useState({ email: "", password: "" })
        const [error, setError] = useState("")

        const handleChange = ({ currentTarget: input }) => {
                setData({ ...data, [input.name]: input.value })
        };

        const handleSubmit = async (e) => {
                e.preventDefault()
                try {
                        const url = "http://localhost:8080/api/auth"
                        const { data: res } = await axios.post(url, data)
                        localStorage.setItem("token", res.data)
                        window.location = "/"
                } catch (error) {
                        if (
                                error.response &&
                                error.response.status >= 400 &&
                                error.response.status <= 500
                        ) {
                                setError(error.response.data.message)
                        }
                }
        }

        return (
                <div className={styles.login_container}>
                        <div className={styles.login_form_container}>
                                <div className={styles.left}>
                                        <form className={styles.form_container}
onSubmit={handleSubmit}>
                                                <h1>Login to Your Account</h1>
                                                <input
                                                        type="email"
                                                        placeholder="Email"
                                                        name="email"
```

```
                                        onChange={handleChange}
                                        value={data.email}
                                        required
                                        className={styles.input}
                                    />
                                    <input
                                        type="password"
                                        placeholder="Password"
                                        name="password"
                                        onChange={handleChange}
                                        value={data.password}
                                        required
                                        className={styles.input}
                                    />
                                    {error && <div
className={styles.error_msg}>{error}</div>}
                                    <button type="submit"
className={styles.green_btn}>
                                        Sing In
                                    </button>
                                </form>
                            </div>
                            <div className={styles.right}>
                                <h1>New Here ?</h1>
                                <Link to="/signup">
                                    <button type="button"
className={styles.white_btn}>
                                        Sing Up
                                    </button>
                                </Link>
                            </div>
                        </div>
                    </div>
                )
}

export default Login;
```

24. Utworzenie katalogu Main oraz plików index.jsx i styles.module.css. Plik index.jsx powinien zawierać kod:

```
import styles from "./styles.module.css"

const Main = () => {
    const handleLogout = () => {
        localStorage.removeItem("token")
        window.location.reload()
    }

    return (
```

```
                <div className={styles.main_container}>
                        <nav className={styles.navbar}>
                                <h1>MySite</h1>
                                <button className={styles.white_btn} onClick={handleLogout}>
                                        Logout
                                </button>
                        </nav>
                </div>
        )
}
export default Main
```

25. W celu ostylowania komponentów należy pobrać gotowe pliki styles.module.css dla odpowiednich komponentów (mdz.cs.pollub.pl/pai/style.zip).

26. Modyfikacja głównego komponentu App:

```
import { Route, Routes, Navigate } from "react-router-dom"
import Main from "./components/Main"
import Signup from "./components/Signup"
import Login from "./components/Login"

function App() {
        const user = localStorage.getItem("token")

        return (
                <Routes>
                        {user && <Route path="/" exact element={<Main />} />}
                        <Route path="/signup" exact element={<Signup />} />
                        <Route path="/login" exact element={<Login />} />
                        <Route path="/" element={<Navigate replace to="/login" />} />
                </Routes>
        )
}

export default App
```

27. Struktura aplikacji oraz zrzuty ekranowe działającej aplikacji: