



**POLITECHNIKA
BYDGOSKA**

Wydział Telekomunikacji,
Informatyki i Elektrotechniki

CHMURY OBLICZENIOWE

Informatyka Stosowana

Instrukcja 02 - Tworzenie aplikacji React

Kacper Krzyżniewski
Filip Pawłowski
Szymon Janiak

Cel laboratorium

Zapoznanie się z procesem inicjalizacji, konfiguracji oraz uruchomienia aplikacji React z wykorzystaniem bibliotek Shadcn UI, React Router, Redux Toolkit oraz Tailwind CSS. Dodatkowym celem jest praktyczne zastosowanie narzędzia v0.dev do budowy intuicyjnego interfejsu użytkownika.

Przykładowe rozwiązanie

Pracę nad utworzeniem aplikacji React należy rozpocząć od zainicjalizowania podstawowej struktury projektu. W tym celu można wykorzystać narzędzie Create React App lub Vite. W tej instrukcji wykorzystamy Vite dla jego szybkości i łatwości konfiguracji.

Inicjalizacja projektu

1. Otwórz terminal i wykonaj następujące polecenie, aby utworzyć nowy projekt React z użyciem Vite:

```
npm create vite@latest my-react-app --template react
```

2. Przejdź do katalogu projektu:

```
cd my-react-app
```

3. Zainstaluj wymagane paczki:

```
npm install react-router-dom @reduxjs/toolkit react-redux  
tailwindcss postcss autoprefixer shadcn-ui
```

4. Uruchom aplikację w trybie deweloperskim:

```
npm run dev
```

Po poprawnym uruchomieniu aplikacja będzie dostępna pod adresem `http://localhost:5173`.

5. Skonfiguruj Tailwind CSS: - Uruchom polecenie do inicjalizacji Tailwind:

```
npx tailwindcss init
```

- W pliku `tailwind.config.js` dodaj ścieżki do plików:

```
content: [  
  "./index.html",  
  "./src/**/*.{js,ts,jsx,tsx}",  
],
```

6. W pliku `src/index.css` dodaj importy Tailwind:

```
@tailwind base;  
@tailwind components;  
@tailwind utilities;
```

Uruchamianie projektu

Po zakończeniu konfiguracji aplikację można uruchomić za pomocą:

```
npm run dev
```

Aplikacja powinna być dostępna pod adresem `http://localhost:5173`.

Tworzenie Layoutu i Podstron

1. Utwórz folder `src/layouts` i dodaj plik `MainLayout.jsx`:

```
import { Outlet } from "react-router-dom";  
  
export default function MainLayout() {  
  return (  
    <div className="min-h-screen bg-gray-50">  
      <header className="bg-blue-600 text-white p-4">  
        <h1>My React App</h1>  
      </header>  
      <main className="p-4">  
        <Outlet />  
      </main>  
    </div>  
  );  
}
```

2. Skonfiguruj routing w pliku src/App.jsx:

```
import { BrowserRouter as Router, Route, Routes } from "react-router-dom";
import MainLayout from "../layouts/MainLayout";
import Home from "../pages/Home";
import About from "../pages/About";

function App() {
  return (
    <Router>
      <Routes>
        <Route path="/" element={<MainLayout />}>
          <Route index element={<Home />} />
          <Route path="about" element={<About />} />
        </Route>
      </Routes>
    </Router>
  );
}

export default App;
```

3. Utwórz podstrony w folderze src/pages: - HomePage.jsx:

```
export default function HomePage() {
  return <h2>Welcome to the Home Page!</h2>;
}
```

- ScriptsPage.jsx:

```
export default function ScriptsPage() {
  return <h2>Scripts Page</h2>;
}
```

Szybkie projektowanie interfejsu użytkownika

Zalecamy korzystanie z narzędzia `v0.dev`, które umożliwia szybkie projektowanie komponentów interfejsu użytkownika w zgodzie z Tailwind CSS. Wygenerowany kod można bezpośrednio wklejać do projektu, co znacznie przyspieszy pracę.

Przykład komponentu wygenerowanego w `v0.dev`:

```
<div className="p-4 bg-white rounded shadow-md">
  <h2 className="text-lg font-bold">Panel</h2>
  <p className="text-gray-600">This is a simple UI component.</p>
</div>
```

Zarządzanie stanem za pomocą Redux Toolkit

1. Utwórz folder `src/store` i dodaj plik `store.js`:

```
import { configureStore } from "@reduxjs/toolkit";
import counterReducer from "../features/counterSlice";

export const store = configureStore({
  reducer: {
    counter: counterReducer,
  },
});
```

2. Utwórz folder src/features i dodaj plik counterSlice.js:

```
import { createSlice } from "@reduxjs/toolkit";

const counterSlice = createSlice({
  name: "counter",
  initialState: { value: 0 },
  reducers: {
    increment: (state) => {
      state.value += 1;
    },
    decrement: (state) => {
      state.value -= 1;
    },
  },
});

export const { increment, decrement } = counterSlice.actions;
export default counterSlice.reducer;
```

3. W pliku src/main.jsx otocz aplikację dostawcą Provider:

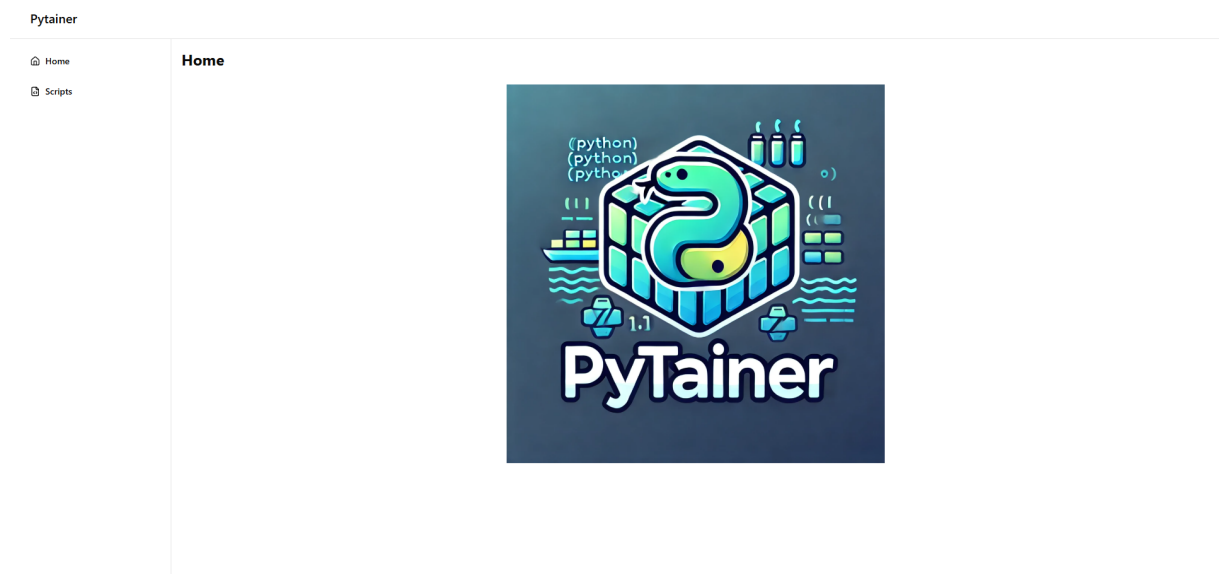
```
import { Provider } from "react-redux";
import { store } from "../store";
import App from "../App";

ReactDOM.createRoot(document.getElementById("root")).render(
  <Provider store={store}>
    <App />
  </Provider>
);
```

Zadania do samodzielnego wykonania

Dopracuj layout strony

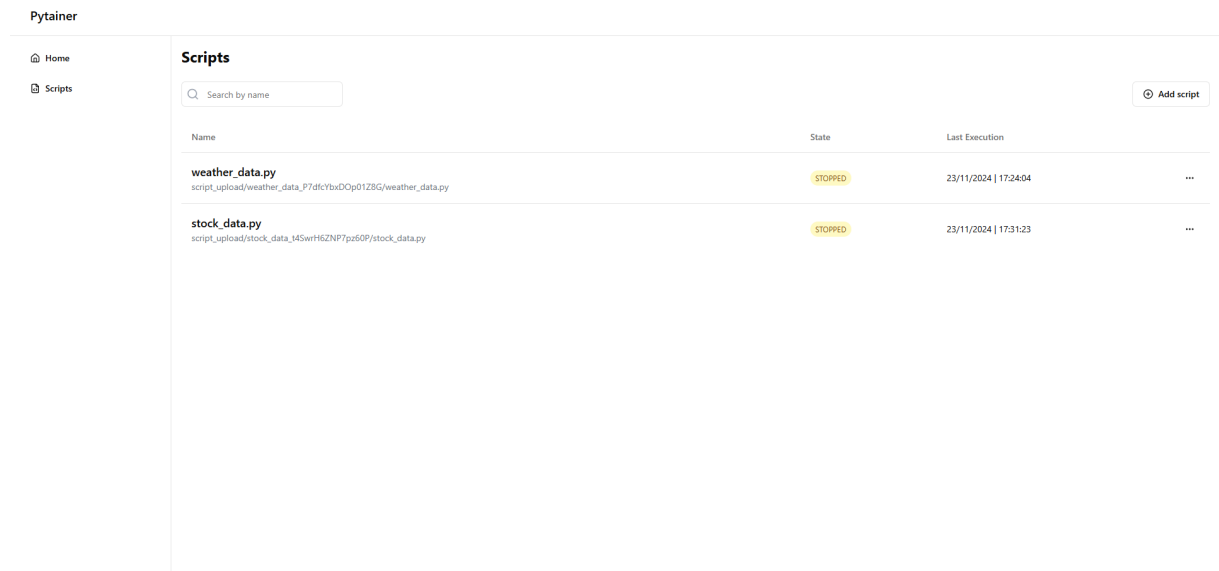
Skorzystaj z narzędzia v0.dev, aby stworzyć bardziej zaawansowaną nawigację i rozbudowany header projektu. Zmień układ strony, dodając elementy takie jak: Menu nawigacyjne z odnośnikami do różnych podstron (np. Home, Scripts), Nagłówek z tytułem strony oraz logo. Wynikowy layout powinien być intuicyjny i estetyczny.



Rysunek 1: Dopracowany layout strony z użyciem narzędzia v0.dev

Stwórz widok pozwalający na wyświetlanie listy skryptów

Dodaj nową podstronę `ScriptsPage`, która wyświetli listę skryptów. Każdy element listy powinien zawierać: Tytuł skryptu, Ścieżkę do szczegółowego widoku skryptu. Wykorzystaj dane statyczne lub załaduj je z pliku JSON.



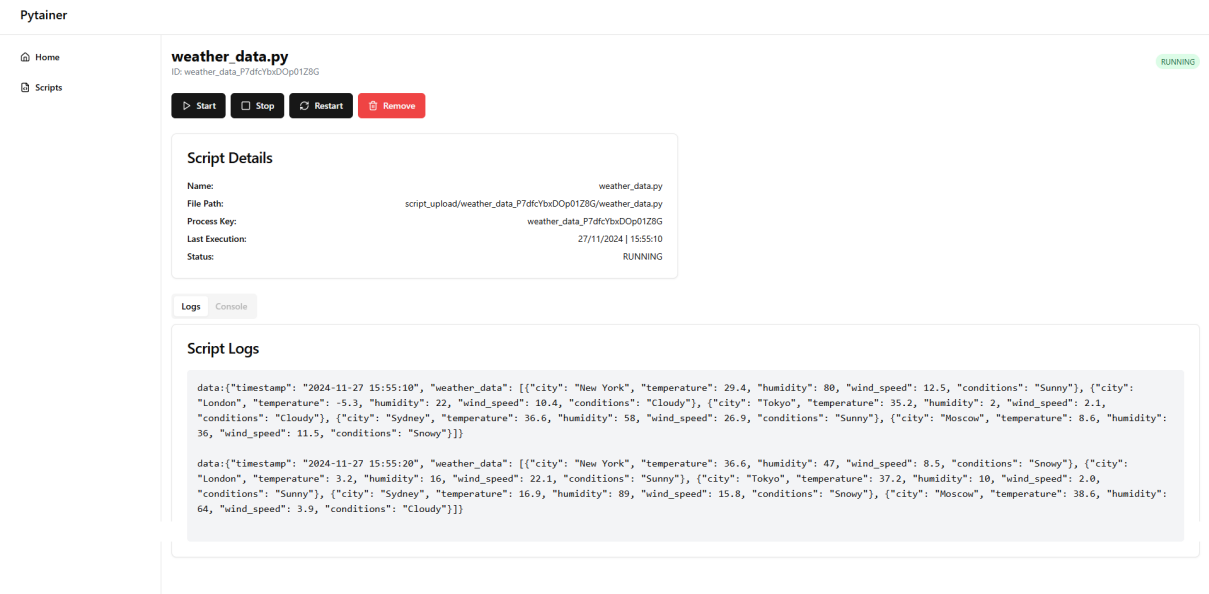
The screenshot shows a web application titled "Pytainer" with a sidebar containing "Home" and "Scripts" links. The "Scripts" page is active, displaying a table of scripts. The table has columns for Name, State, Last Execution, and an ellipsis menu. Two scripts are listed: "weather_data.py" and "stock_data.py", both in a "STOPPED" state. A search bar and an "Add script" button are at the top right of the table.

Name	State	Last Execution	
weather_data.py script_upload/weather_data_97dfcybOCy0128G/weather_data.py	STOPPED	23/11/2024 17:24:04	...
stock_data.py script_upload/stock_data_4SwtH6ZNP7pz50P/stock_data.py	STOPPED	23/11/2024 17:31:23	...

Rysunek 2: Widok listy skryptów z tytułami i ścieżkami

Stwórz widok pozwalający na wyświetlenie wybranego skryptu

Dodaj podstronę ScriptPage, która wyświetli szczegółowe informacje o skrypcie. Widok powinien zawierać: Tytuł skryptu, Szczegółowe informacje o skrypcie, Panel konsolowy do wyświetlania wyników skryptu. Wykorzystaj useParams z React Router do dynamicznego pobierania ID skryptu i wyświetlania jego danych.



Rysunek 3: Widok szczegółowy wybranego skryptu