



POLITECHNIKA  
LUBELSKA  
WYDZIAŁ ELEKTROTECHNIKI  
I INFORMATYKI

# Programowanie aplikacji w chmurze obliczeniowej

laboratorium

Zadanie 1 + część nieobowiązkowa 1.

Igor Pąsko

grupa dziekańska: 6.10

numer albumu: 99655

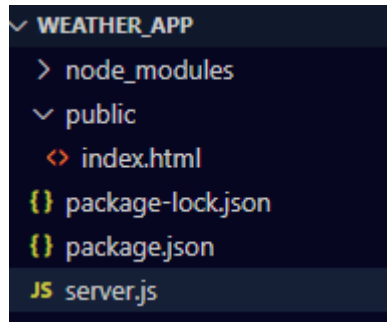
Prowadzący laboratorium mgr. inż. K. Łazaruk

Lublin rok 2025

## Część obowiązkowa

Realizacja aplikacji wyświetlającej prognozę pogody dla wybranych miejscowości w danym kraju

Przygotowane zostały pliki index.html oraz server.js. Do wykonania zadania użyto Node.js oraz Express. Frontend to prosta aplikacja wykorzystująca html, js oraz css. Do wykonania projektu wykorzystano stronę z darmowym api pogodowym: <https://dobrapogoda24.pl/api-pogoda>.



Rys.1 Struktura projektu na poziomie tworzenia aplikacji.

### Plik package.json

```
{
  "name": "weather_app",
  "version": "1.0.0",
  "main": "server.js",
  "scripts": {
    "start": "node server.js"
  },
  "dependencies": {
    "express": "^4.18.2",
    "node-fetch": "^2.6.7"
  }
}
```

### Plik server.js

```
const express = require("express");
const fetch = require("node-fetch");
const path = require("path");

const app = express();
const PORT = 8000;
const AUTHOR = "Igor Pąsko";
const TOKEN = "401cdabee4170dfb0ac7";
const DAY = 0; //aktualny dzień prognozy
//Mapowanie nazwy na format, który zostanie zaakceptowany przez użyte api
const CITY_MAP = {
  Bilbao: "bilbao",
  Sewilla: "sevilla",
  Madryt: "madryt",
  Rzeszów: "rzeszow",
  Tychy: "tychy",
  Szczecin: "szczecin",
  Montpellier: "montpellier",
  Lyon: "lyon",
  Strasburg: "strasburg"
};
```

```

app.use(express.static(path.join(__dirname, "public")));
app.use(express.json());

app.listen(PORT, () => {
  console.log(`[${new Date().toISOString()}] Autor: ${AUTHOR} | Nasłuchuję na :${PORT}`);
});

app.post("/weather", async (req, res) => {
  const uiCity = req.body.city;
  const apiCity = CITY_MAP[uiCity];

  if (!apiCity) {
    return res.status(400).json({ error: "Nieobsługiwane miasto." });
  }

  const url = `https://dobrapogoda24.pl/api/v1/weather/simple?city=${apiCity}&day=${DAY}&token=${TOKEN}`;
  try {
    const resp = await fetch(url);
    if (resp.status === 404) {
      return res.status(404).json({ error: `Brak danych pogodowych dla ${uiCity}.` });
    }
    if (!resp.ok) {
      throw new Error(`HTTP ${resp.status}`);
    }
    const data = await resp.json();
    res.json(data);
  } catch (err) {
    console.error("Błąd API:", err.message);
    res.status(502).json({ error: "Błąd pobierania danych pogodowych." });
  }
});

```

## Plik index.html

```

<!DOCTYPE html>
<html lang="pl">
<head>
  <meta charset="UTF-8" />
  <title>Aplikacja Pogodowa</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      max-width: 600px;
      margin: 2rem auto;
      padding: 1rem;
      line-height: 1.4;
    }
    h2 { text-align: center; }
    select, button {
      padding: 0.5em 1em;
      margin: 0.5em 0.5em 0.5em 0;
      font-size: 1em;
    }
    #controls {
      text-align: center;
    }
    #weatherInfo table {
      width: 100%;
      border-collapse: collapse;
      margin-top: 1em;
    }

```

```

    }
    #weatherInfo th, #weatherInfo td {
        border: 1px solid #ccc;
        padding: 0.5em;
        text-align: center;
    }
    #sun {
        text-align: center;
        margin-top: 1em;
    }
    #error {
        color: crimson;
        margin-top: 1em;
        text-align: center;
    }
}
</style>
</head>
<body>
    <h2>Wybierz kraj i miasto</h2>
    <div id="controls">
        <select id="country">
            <option value="Polska">Polska</option>
            <option value="Hiszpania">Hiszpania</option>
            <option value="Francja">Francja</option>
        </select>
        <select id="city"></select>
        <button id="btn">Pokaż pogodę</button>
    </div>

    <div id="weatherInfo"></div>
    <div id="sun"></div>
    <div id="error"></div>

    <script>
        const COUNTRY_MAP = {
            Polska: ["Rzeszów", "Tychy", "Szczecin"],
            Hiszpania: ["Bilbao", "Sewilla", "Madryt"],
            Francja: ["Montpellier", "Lyon", "Strasburg"]
        };
        //inicjalizacja
        const countrySel = document.getElementById("country");
        const citySel = document.getElementById("city");
        const errorDiv = document.getElementById("error");
        const infoDiv = document.getElementById("weatherInfo");
        const sunDiv = document.getElementById("sun");
        const btn = document.getElementById("btn");

        function updateCities() {
            const k = countrySel.value;
            citySel.innerHTML = COUNTRY_MAP[k]
                .map(c => `<option>${c}</option>`)
                .join("");
        }

        countrySel.addEventListener("change", updateCities);
        updateCities(); // pierwsze załadowanie

        btn.addEventListener("click", getWeather);

        async function getWeather() {
            const city = citySel.value;

```

```
errorDiv.textContent = "";  
infoDiv.innerHTML = "Ładowanie...";  
sunDiv.textContent = "";
```

```
try {  
    const resp = await fetch("/weather", {  
        method: "POST",  
        headers: { "Content-Type": "application/json" },  
        body: JSON.stringify({ city })  
    });  
    const data = await resp.json();  
    if (data.error) {  
        infoDiv.innerHTML = "";  
        errorDiv.textContent = data.error;  
        return;  
    }  
  
    // dane wschodu i zachodu słońca  
    const sunrise = new Date(data.sunrise).toLocaleTimeString("pl-PL");  
    const sunset = new Date(data.sunset).toLocaleTimeString("pl-PL");  
    sunDiv.innerHTML = `  
        🌅 Wschód słońca: <strong>${sunrise}</strong>  
        &nbsp;&nbsp;&nbsp;&nbsp;& 🌇 Zachód słońca: <strong>${sunset}</strong>  
    `;  
  
    // tabela z danymi pogodowymi  
    const { date, day, night } = data;  
    infoDiv.innerHTML = `  
        <h3>Pogoda w ${city} – ${date}</h3>  
        <table>  
            <tr>  
                <th>Faza</th><th>Temp min</th><th>Temp max</th><th>Opady (mm)</th>  
            </tr>  
            <tr>  
                <td>Dzień</td>  
                <td>${day.temp_min}°C</td>  
                <td>${day.temp_max}°C</td>  
                <td>${day.precipitation}</td>  
            </tr>  
            <tr>  
                <td>Noc</td>  
                <td>${night.temp_min}°C</td>  
                <td>${night.temp_max}°C</td>  
                <td>${night.precipitation}</td>  
            </tr>  
        </table>  
    `;  
  
    } catch (err) {  
        infoDiv.innerHTML = "";  
        errorDiv.textContent = "Błąd sieci lub API.";  
        console.error(err);  
    }  
}  
  
</script>  
</body>  
</html>
```

Aby odpalić aplikację na tym etapie należy użyć następujących komend w celu instalacji Node.js oraz uruchomienia serwera.

1. npm install
2. npm start

Następnie wchodzimy na adres <http://localhost:8000>.

```
PS C:\Users\pasko\Downloads\weather_app> npm start

> weather_app@1.0.0 start
> node server.js

[2025-05-22T19:05:35.741Z] Autor: Igor Pąsko | Nasłuchuję na :8000
Terminate batch job (Y/N)? y
PS C:\Users\pasko\Downloads\weather_app> 
```

Rys.2 Wynik działania aplikacji 1.

## Wybierz kraj i miasto

Polska

▼



Szczecin

▼

Pokaż pogodę

### Pogoda w Szczecin – 2025-05-22

Faza	Temp min	Temp max	Opady (mm)
Dzień	3°C	18°C	0.0
Noc	2°C	11°C	0.0

 Wschód słońca: **04:51:38**  Zachód słońca: **21:06:42**

Rys.3 Wynik działania aplikacji 2.

## Tworzymy plik Dockerfile

//zależności

```
FROM node:18-alpine AS builder
WORKDIR /app
```

```
COPY package.json package-lock.json ./
```

```
RUN npm ci --only=production
```

//tworzenie kopii kodu

```
COPY . .
```

//finalny obraz

```
FROM node:18-alpine
```

//metadane

```
LABEL org.opencontainers.image.authors="Igor Pąsko"
```

```
LABEL org.opencontainers.image.title="Weather App"
```

```
LABEL org.opencontainers.image.version="1.0.0"
```

```
WORKDIR /app
```

//kopiowanie node modules i kodu

```
COPY --from=builder /app/node_modules ./node_modules
```

```

COPY --from=builder /app/server.js ./server.js
COPY --from=builder /app/public ./public
//otworzenie portu
EXPOSE 8000
//healthcheck
HEALTHCHECK --interval=30s --timeout=5s \
  CMD wget --quiet --spider http://localhost:8000/ || exit 1

CMD ["node", "server.js"]

```

Tworzymy kontener, uruchamiamy aplikację i pobieramy informacje.  
Komendy:

wchodzimy w odpowiedni katalog poleceniem cd

1. Budowanie obrazu

```
docker build -t weather_app:1.0 .
```

2. Uruchomienie kontenera

```
docker run -d --name weather_app -p 8000:8000 weather_app:1.0
```

3. Uzyskanie informacji z logów

```
docker logs weather_app
```

4. Lista i rozmiar warstw

```
docker history weather_app:1.0
```

5. Rozmiar obrazu

```
docker image ls weather_app:1.0
```

Logi z terminala docker znajdują się w załączonym pliku logi.txt.

### **Część nieobowiązkowa 1.**

Tworzenie i aktywacja buildera opartego na sterowniku docker container

```
docker buildx create --name multiarch-builder --driver docker-container --use
```

Sprawdzenie działania buildera.

```
docker buildx inspect multiarch-builder --bootstrap
```

Budowa i wypchnięcie obrazu

```
docker buildx build --builder multiarch-builder --platform linux/amd64,linux/arm64 -t  
paskoi/weather_app:multiarch --push .  
--platform – deklaracja architektury.
```

Weryfikacja manifestu

*docker buildx imagetools inspect weather\_app:multiarch*

Logi z terminala docker znajdują się w załączonym pliku logi2.txt.

W logach znajduje się potwierdzenie że manifest zawiera deklaracje obu platform

*Platform: linux/amd64*

oraz

*Platform: linux/arm64*