

# PRAKTYKI

## Dokumentacja projektu

Autor	Dawid Olko
Kierunek, rok	Informatyka, III rok, st. stacjonarne (3,5-l)
Temat projektu	<i>Aplikacja pogodowa wykorzystująca API</i>



**WeatherApp**

**27.01.2025r. - 04.02.2025r.**

## Spis treści

<b>1.</b>	<b><i>Narzędzia i technologie</i></b>	<b>3</b>
<b>2.</b>	<b><i>Baza danych</i></b>	<b>8</b>
<b>3.</b>	<b><i>GUI</i></b>	<b>12</b>
<b>4.</b>	<b><i>Uruchomienie aplikacji</i></b>	<b>17</b>
<b>5.</b>	<b><i>Podsumowanie</i></b>	<b>19</b>

# 1. Narzędzia i technologie

## **1.1 Technologie użyte w projekcie**

Projekt aplikacji pogodowej został zrealizowany z wykorzystaniem technologii webowych oraz backendowych. Poniżej znajduje się lista kluczowych narzędzi oraz technologii użytych w aplikacji:

### **Backend: Laravel (PHP)**

- **Framework:** Laravel 11.4
- **Język programowania:** PHP 8+
- **Baza danych:** MySQL
- **ORM:** Eloquent
- **REST API:** Laravel + OpenWeather API
- **Autoryzacja:** Laravel Sanctum
- **Harmonogram zadań:** Laravel Scheduler (dla odświeżania pogody co 30 min)
- **Obsługa zapytań HTTP:** Laravel HTTP Client
- **Migracje bazy danych:** Laravel Migrations
- **Obsługa plików statycznych:** Laravel Storage

### **Frontend: React + Vite**

- **Framework:** React 18
- **Stan aplikacji:** React Context API
- **Stylowanie:** SASS (SCSS)
- **Routing:** React Router Dom
- **Obsługa API:** Axios
- **Animacje:** Framer Motion
- **Interaktywne komponenty:** React Icons, React CountUp, Slick Carousel
- **Walidacja formularzy:** Formik + Yup
- **Wizualizacja danych:** Recharts

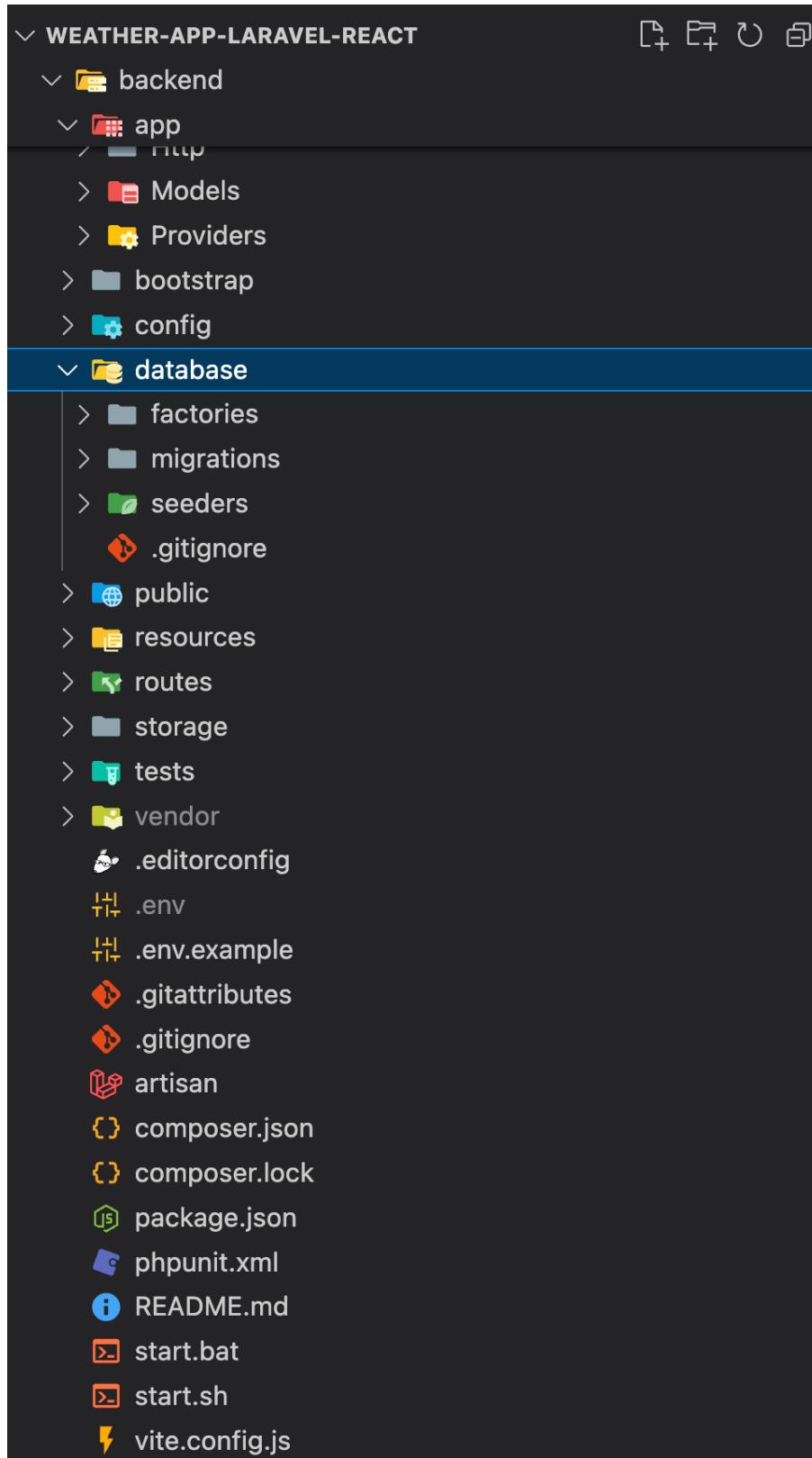
### **Zewnętrzne API: OpenWeather API**

- **Źródło danych pogodowych:** [OpenWeatherMap](#)
- **Dane historyczne i prognozy pogody**
- **ID miast pobierane z pliku JSON ([city.list.json.gz](#))**

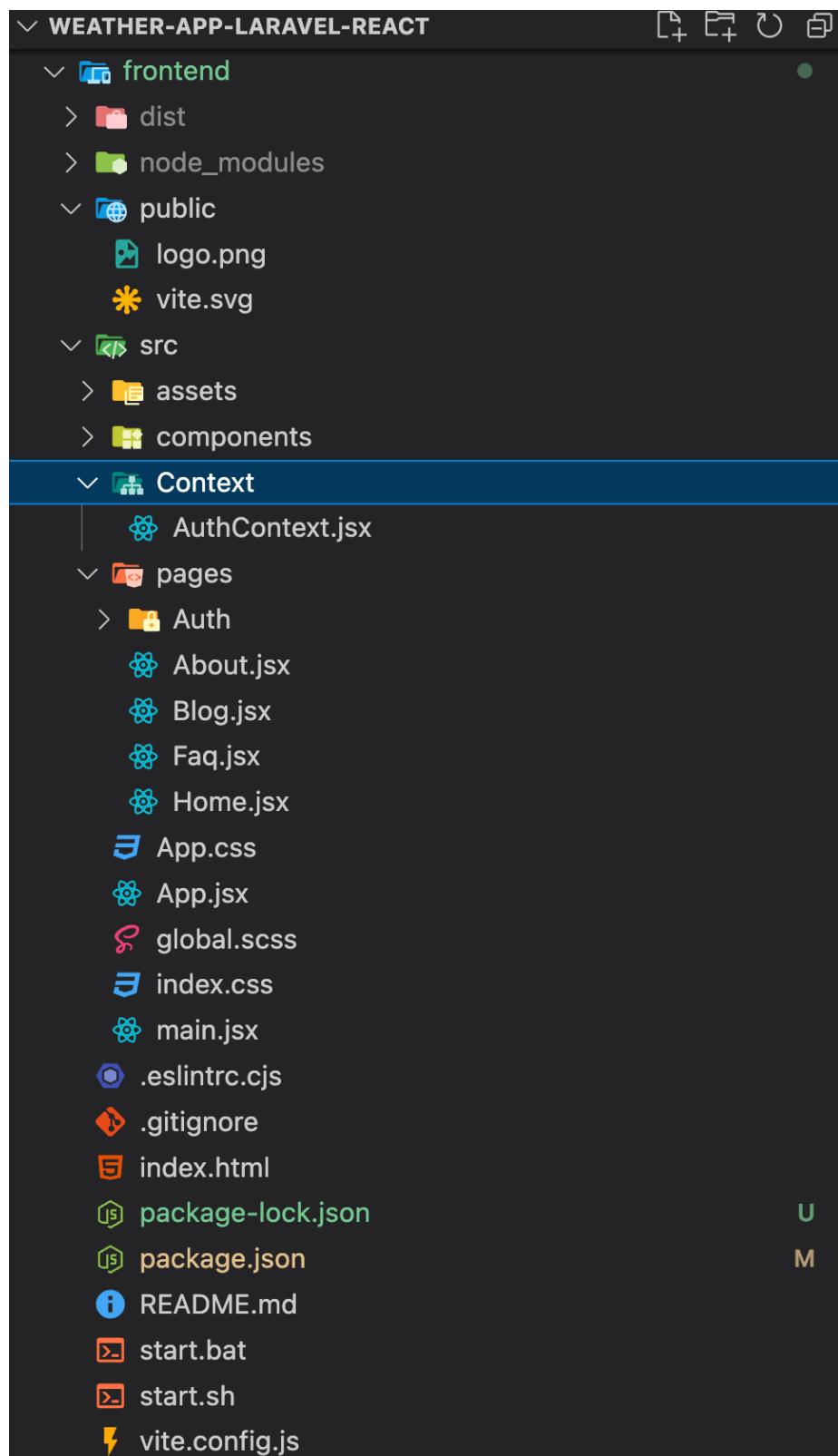
## 1.2 Struktura katalogów w projekcie

Projekt został podzielony na backend (Laravel) oraz frontend (React), a także posiada osobną warstwę bazy danych.

### Backend – Laravel



## Frontend – React



### 1.3 Narzędzia do zarządzania projektem

Projekt wykorzystuje kilka narzędzi wspierających zarządzanie kodem, automatyzację procesów oraz testowanie aplikacji:

#### System kontroli wersji

- **Git** – do śledzenia zmian kodu i wersjonowania.
- **GitLab** – repozytorium kodu źródłowego projektu.

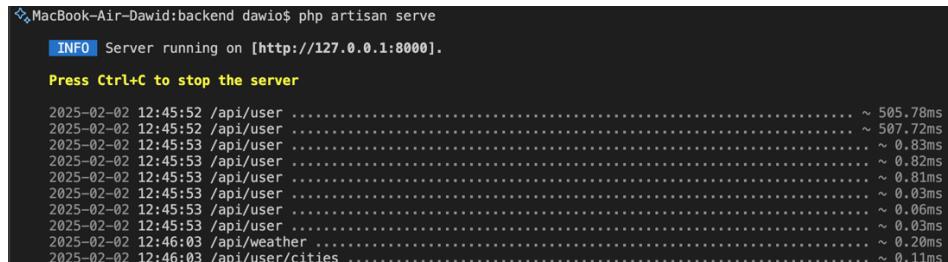
#### Zarządzanie zależnościami

- **Backend:** Composer (Laravel)
- **Frontend:** npm (React + Vite)

#### Uruchamianie aplikacji lokalnie

- **Backend:**

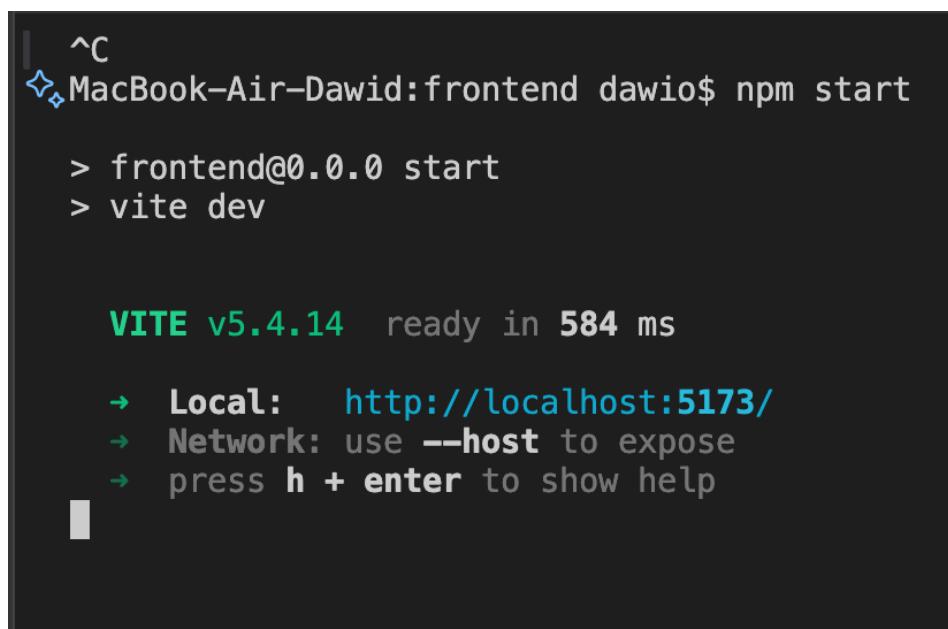
```
php artisan serve
```



```
MacBook-Air-Dawid:backend dawio$ php artisan serve
INFO Server running on [http://127.0.0.1:8000].
Press Ctrl+C to stop the server
2025-02-02 12:45:52 /api/user ..... ~ 505.78ms
2025-02-02 12:45:52 /api/user ..... ~ 507.72ms
2025-02-02 12:45:53 /api/user ..... ~ 0.83ms
2025-02-02 12:45:53 /api/user ..... ~ 0.82ms
2025-02-02 12:45:53 /api/user ..... ~ 0.81ms
2025-02-02 12:45:53 /api/user ..... ~ 0.03ms
2025-02-02 12:45:53 /api/user ..... ~ 0.06ms
2025-02-02 12:45:53 /api/user ..... ~ 0.03ms
2025-02-02 12:46:03 /api/weather ..... ~ 0.20ms
2025-02-02 12:46:03 /api/user/cities ..... ~ 0.11ms
```

- **Frontend:**

```
npm start
```



```
^C
MacBook-Air-Dawid:frontend dawio$ npm start
> frontend@0.0.0 start
> vite dev

VITE v5.4.14 ready in 584 ms

→ Local: http://localhost:5173/
→ Network: use --host to expose
→ press h + enter to show help
```

## 1.4 API OpenWeather

Aplikacja pobiera dane pogodowe za pomocą API OpenWeatherMap. Używane są dwa główne endpointy:

- **Aktualna pogoda:**

`https://api.openweathermap.org/data/2.5/weather?q={city}&appid={API_KEY}`

- **Prognoza na 5 dni:**

`https://api.openweathermap.org/data/2.5/forecast?q={city}&appid={API_KEY}`

Dane pogodowe obejmują:

- Temperaturę, wilgotność, ciśnienie,
- Prędkość i kierunek wiatru,
- Opady deszczu i śniegu,
- Opis warunków pogodowych.

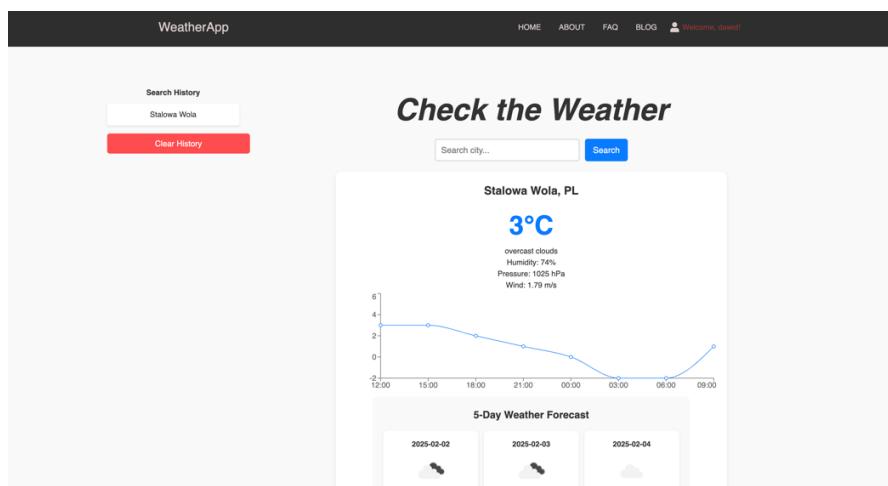
## 1.5 Wykorzystane biblioteki i zależności

### Backend (Laravel)

- laravel/sanctum – autoryzacja użytkowników.
- guzzlehttp/guzzle – obsługa żądań HTTP do API pogodowego.
- laravel/ui – szkielet systemu logowania.

### Frontend (React)

- react-router-dom – zarządzanie trasami.
- axios – wykonywanie zapytań HTTP do API backendowego.
- formik + yup – obsługa formularzy i walidacji.
- recharts – wizualizacja danych pogodowych na wykresach.
- framer-motion – animacje interfejsu.



## 2. Baza danych

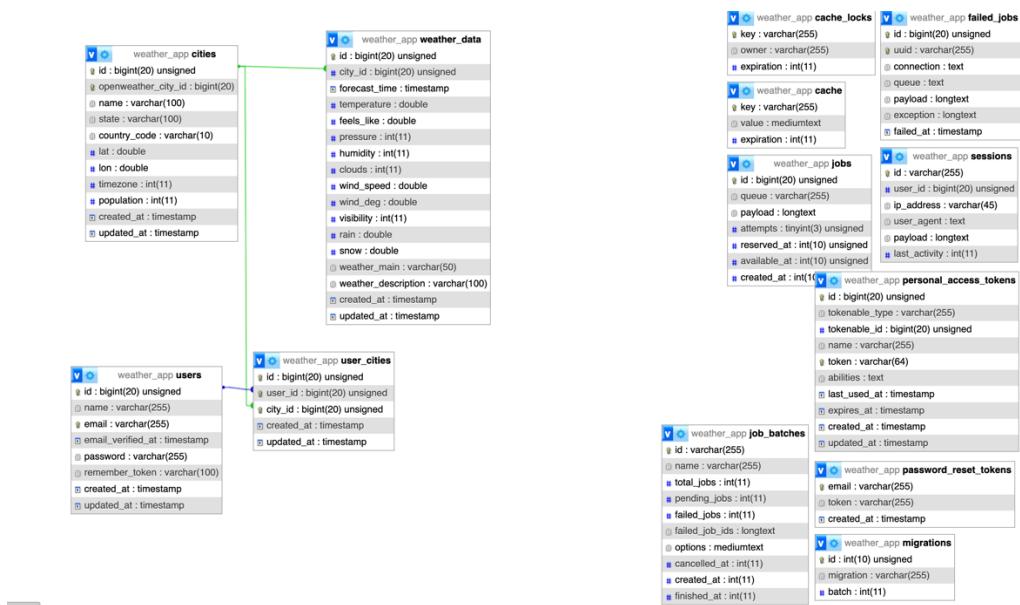
### 2.1 Struktura bazy danych

Aplikacja korzysta z **bazy danych MySQL**, gdzie przechowywane są informacje o użytkownikach, miastach oraz pobranych danych pogodowych. Struktura bazy została zaprojektowana w sposób umożliwiający efektywne zarządzanie danymi i ich relacjami.

#### Główne tabele w bazie danych

Aplikacja posiada następujące kluczowe tabele:

- **users** – przechowuje dane użytkowników
- **cities** – przechowuje miasta dostępne w aplikacji
- **weather\_data** – przechowuje historię pogody dla wybranych miast
- **user\_cities** – tabela pośrednia dla relacji użytkownik-miasto
- **sessions** – przechowuje sesje użytkowników
- **personal\_access\_tokens** – przechowuje tokeny autoryzacyjne dla użytkowników



## 2.2 Opis tabel

Poniżej znajduje się opis najważniejszych tabel bazy danych wraz z ich polami.

### Tabela: users (Użytkownicy)

Przechowuje dane zarejestrowanych użytkowników aplikacji.

Kolumna	Typ danych	Opis
Id	BIGINT UNSIGNED	Klucz główny
name	VARCHAR(255)	Imię i nazwisko użytkownika
email	VARCHAR(255)	Unikalny adres e-mail
password	VARCHAR(255)	Hasło użytkownika (hashowane)
created_at	TIMESTAMP	Data utworzenia
updated_at	TIMESTAMP	Data aktualizacji

### Tabela: cities (Miasta)

Przechowuje listę miast dostępnych w aplikacji. Dane są pobierane z pliku **city.list.json.gz** z OpenWeather API.

Kolumna	Typ danych	Opis
id	BIGINT UNSIGNED	Klucz główny
openweather_city_id	BIGINT	ID miasta w OpenWeather API
name	VARCHAR(100)	Nazwa miasta
state	VARCHAR(100)	Stan/województwo (opcjonalne)
country_code	VARCHAR(10)	Kod kraju (np. PL)
lat	DOUBLE	Szerokość geograficzna
lon	DOUBLE	Długość geograficzna
timezone	INT	Strefa czasowa
population	INT	Liczba ludności (opcjonalne)
created_at	TIMESTAMP	Data utworzenia
updated_at	TIMESTAMP	Data aktualizacji

**Tabela: weather\_data (Dane pogodowe)**

Przechowuje historię pogody dla zapisanych miast.

Kolumna	Typ danych	Opis
id	BIGINT UNSIGNED	Klucz główny
city_id	BIGINT UNSIGNED	Klucz obcy do tabeli cities
forecast_time	TIMESTAMP	Czas prognozy
temperature	FLOAT	Temperatura w °C
feels_like	FLOAT	Odczuwalna temperatura
pressure	INT	Ciśnienie atmosferyczne (hPa)
humidity	INT	Wilgotność (%)
clouds	INT	Zachmurzenie (%)
wind_speed	FLOAT	Prędkość wiatru (m/s)
wind_deg	FLOAT	Kierunek wiatru
visibility	INT	Widoczność (m)
rain	FLOAT	Opady deszczu (mm)
snow	FLOAT	Opady śniegu (mm)
weather_main	VARCHAR(50)	Główna kategoria pogody
weather_description	VARCHAR(100)	Opis warunków pogodowych
created_at	TIMESTAMP	Data utworzenia

**Tabela: user\_cities (Połączenie użytkownika z miastami)**

Tabela pośrednia łącząca użytkowników z miastami, które dodali do swoich ulubionych.

Kolumna	Typ danych	Opis
id	BIGINT UNSIGNED	Klucz główny
user_id	BIGINT UNSIGNED	Klucz obcy do tabeli users
city_id	BIGINT UNSIGNED	Klucz obcy do tabeli cities
created_at	TIMESTAMP	Data dodania miasta do ulubionych

### **2.3 Relacje między tabelami**

Baza danych została zaprojektowana w sposób pozwalający na łatwe zarządzanie użytkownikami i miastami:

- **Użytkownik (users) może dodać do 10 miast (cities)** – relacja wiele-do-wielu poprzez user\_cities.
- **Miasta (cities) mają przypisane dane pogodowe (weather\_data)** – relacja jeden-do-wielu.
- **Każdy użytkownik może śledzić swoje miasta i ich historię pogody.**

### **2.4 Mechanizm zbierania danych pogodowych**

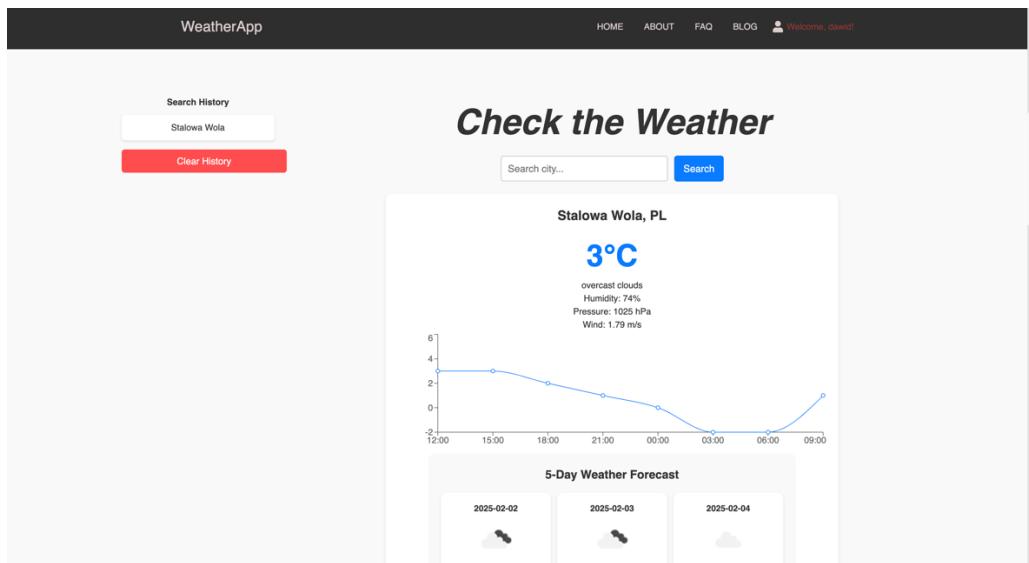
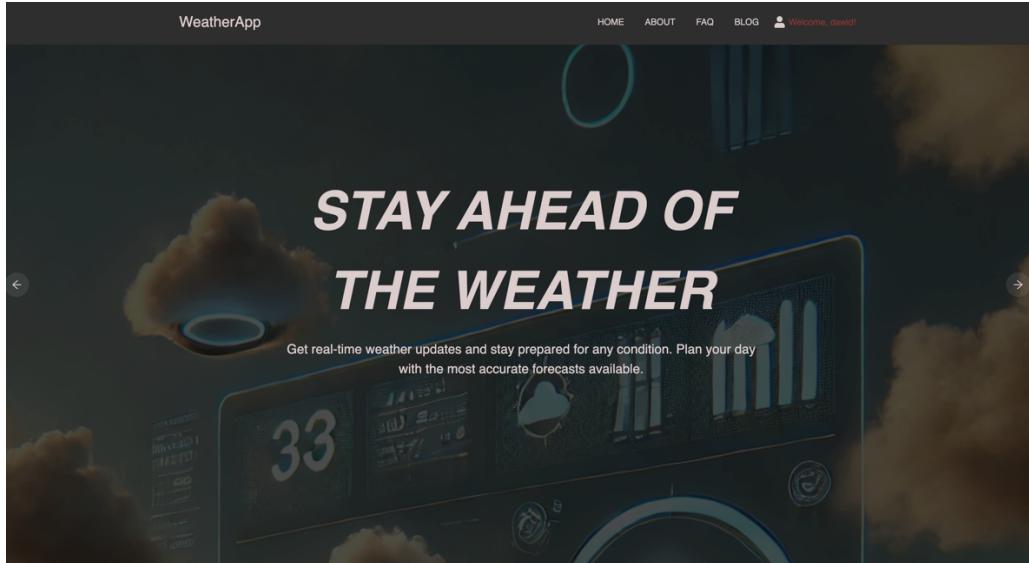
Aplikacja regularnie pobiera dane pogodowe dla miast użytkowników:

1. **Użytkownik dodaje miasto** – system zapisuje je w tabeli user\_cities.
2. **Co 30 minut Laravel Scheduler pobiera pogodę** dla miast zapisanych przez użytkowników.
3. **Dane pogodowe zapisywane są w weather\_data.**
4. **Użytkownik może przeglądać historię pogody** na wykresach i tabelach.

### 3. GUI

#### 3.1 Przegląd interfejsu aplikacji

Aplikacja pogodowa posiada nowoczesny i intuicyjny interfejs użytkownika stworzony w **React + SASS**, zapewniający płynną nawigację oraz wygodę użytkowania. Poniżej przedstawiono kluczowe ekranы aplikacji.



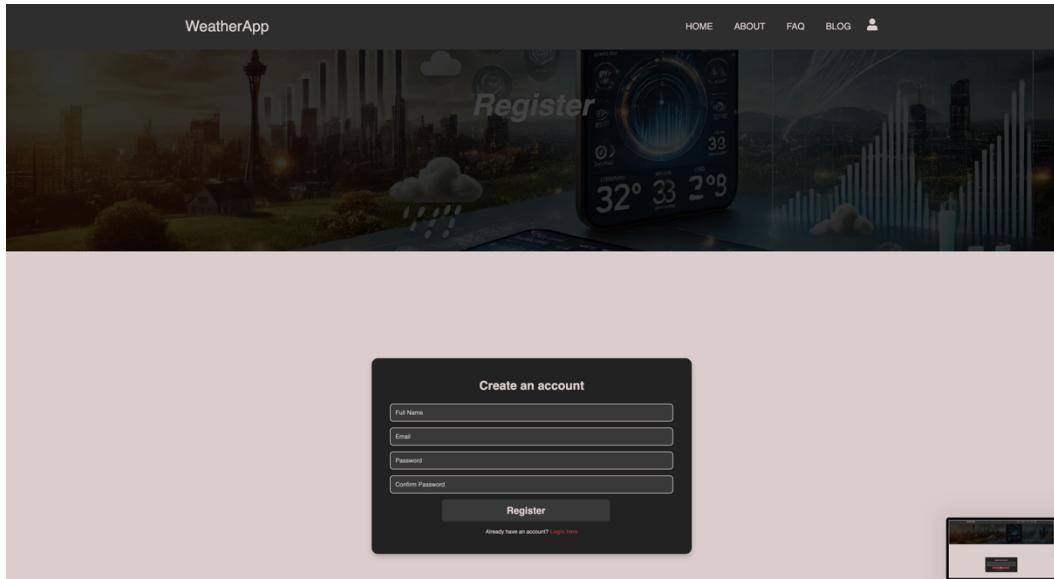
#### Opis ekranu głównego:

- Widok startowy aplikacji prezentuje możliwość wyszukania miasta oraz aktualne warunki pogodowe.
- Główne elementy interfejsu:
  - Pole wyszukiwania miast
  - Przycisk „Szukaj” do pobierania danych pogodowych
  - Wyświetlane podstawowe informacje pogodowe

- Przycisk do dodania miasta do ulubionych (dla zalogowanych użytkowników)

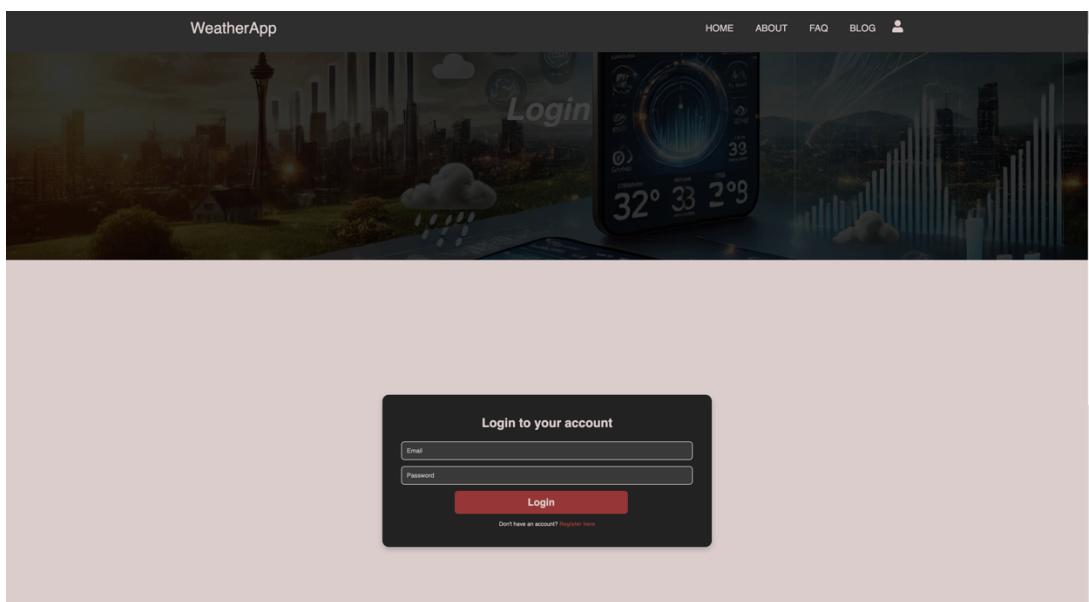
### 3.2 Rejestracja i logowanie

Aplikacja pozwala użytkownikom na założenie konta oraz logowanie się, aby mogli zapisywać ulubione miasta i przeglądać historię pogody.



#### Opis ekranu rejestracji:

- Formularz rejestracyjny zawiera pola:
  - Imię i nazwisko
  - Adres e-mail
  - Hasło i jego potwierdzenie
- Po poprawnym wypełnieniu formularza użytkownik otrzymuje potwierdzenie utworzenia konta.

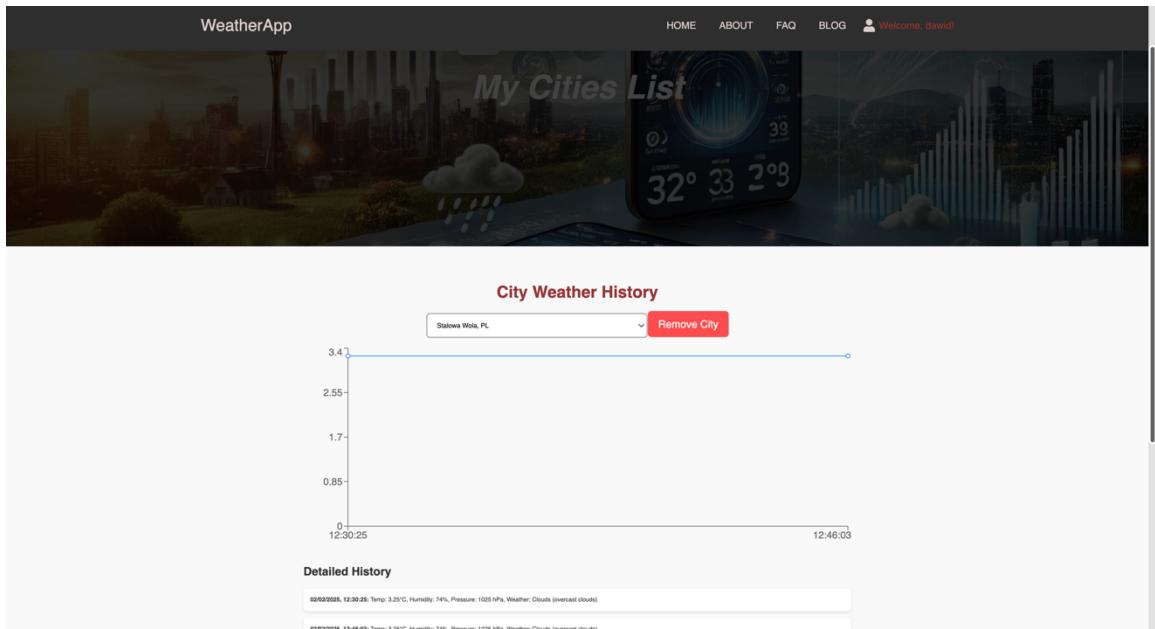


### Opis ekranu logowania:

- Formularz logowania składa się z pól:
  - Adres e-mail
  - Hasło
- Po zalogowaniu użytkownik ma dostęp do funkcji zapisanych miast i historii pogody.

### 3.3 Strona ulubionych miast

Po zalogowaniu użytkownik może dodać do **10 miast** do swojej listy ulubionych, które będą automatycznie aktualizowane co **30 minut**.



### Opis ekranu ulubionych miast:

- Lista miast dodanych przez użytkownika.
- Każde miasto ma przycisk usunięcia.
- Kliknięcie na miasto pozwala przejść do szczegółowego widoku historii pogody.

### 3.4 Historia pogody i wykresy

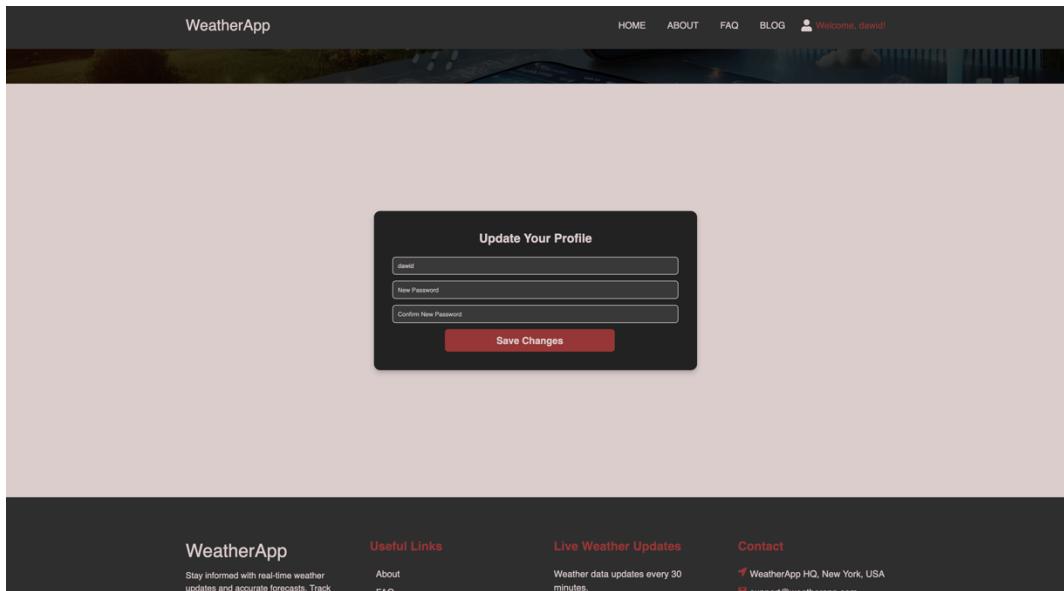
Aplikacja przechowuje historię pogody dla wybranych miast i umożliwia wizualizację danych na wykresach.

### Opis ekranu historii pogody:

- Wykres prezentuje zmiany temperatury, ciśnienia i wilgotności na przestrzeni czasu.
- Dane pochodzą z regularnych zapisów co **30 minut**.
- Możliwość zmiany przedziału czasowego danych pogodowych.

### 3.5 Ekran ustawień użytkownika

Użytkownicy mogą edytować swoje dane, zmieniać hasło i personalizować swoje ustawienia w aplikacji.

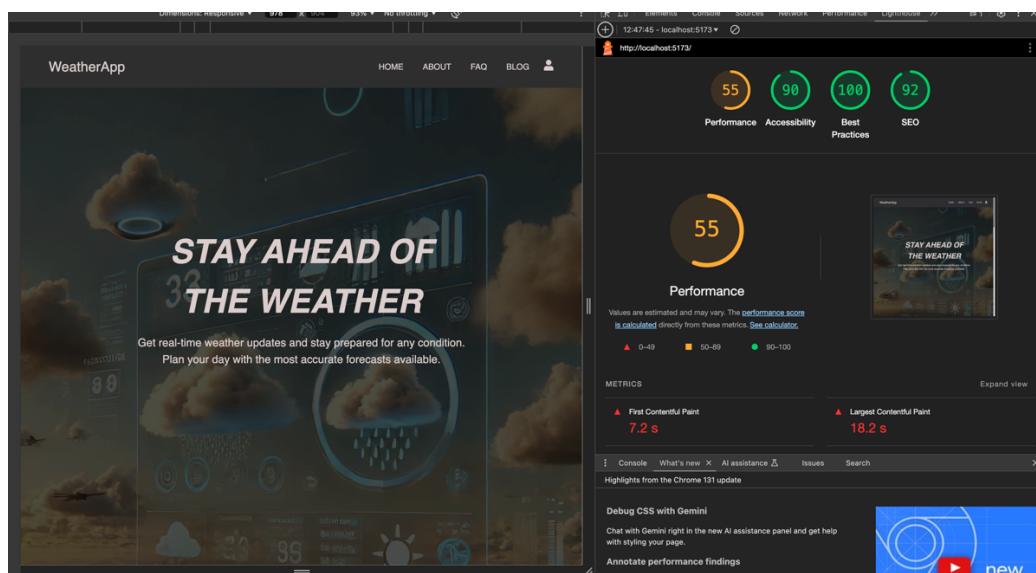


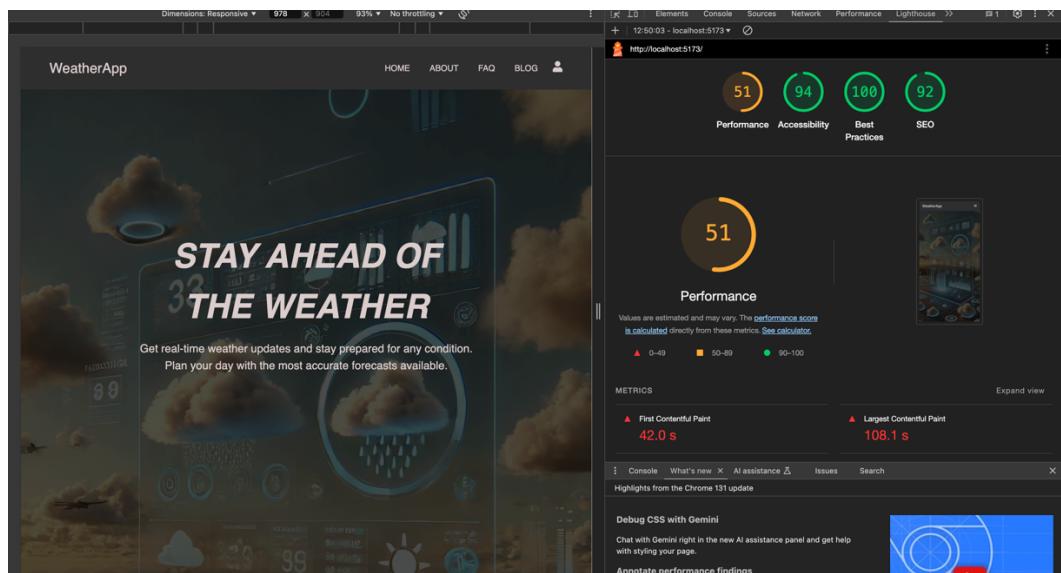
#### Opis ekranu ustawień:

- Możliwość zmiany nazwy użytkownika.
- Opcja aktualizacji hasła.
- Przycisk „Zapisz zmiany” do zatwierdzenia nowego ustawienia.

### 3.6 Responsywność i animacje

Aplikacja jest w pełni responsywna, dostosowując swój układ do różnych rozdzielczości ekranów.





### Opis wersji mobilnej:

- Menu nawigacyjne dostosowane do ekranów dotykowych.
- Wygodne przyciski i pola tekstowe do interakcji.
- Optymalizacja wyświetlania wykresów i tabel na małych ekranach.

## 4. Uruchomienie aplikacji

### **4.1 Wymagania systemowe**

Aby uruchomić aplikację pogodową, należy spełnić następujące wymagania systemowe:

#### **Backend (Laravel)**

- **PHP 8.0+**
- **Composer 2.0+**
- **MySQL 5.7+ lub MariaDB 10+**
- **Node.js 16+ (dla narzędzi frontendowych)**
- **OpenWeather API Key**

#### **Frontend (React)**

- **Node.js 16+**
- **npm 8+ lub yarn**
- Przeglądarka obsługująca nowoczesne standardy JavaScript (Chrome, Firefox, Edge)

### **4.2 Konfiguracja backendu (Laravel)**

#### **Krok 1: Klonowanie repozytorium**

Pobierz kod źródłowy aplikacji z repozytorium:

```
git clone https://github.com/uzytkownik/weather-app.git  
cd weather-app/backend
```

#### **Krok 2: Instalacja zależności PHP**

Zainstaluj zależności backendu za pomocą **Composera**:

```
composer install
```

#### **Krok 3: Konfiguracja pliku .env**

Skopiuj przykładowy plik konfiguracji środowiska i uzupełnij kluczami API oraz danymi dostępowymi do bazy danych:

```
cp .env.example .env
```

Następnie edytuj plik .env i ustaw poprawne wartości:

```
DB_CONNECTION=mysql  
DB_HOST=127.0.0.1  
DB_PORT=3306  
DB_DATABASE=weather_app  
OPENWEATHER_API_KEY=twoj_klucz_api
```

#### **Krok 4: Generowanie klucza aplikacji**

```
php artisan key:generate
```

#### **Krok 5: Migracja bazy danych**

Aby utworzyć struktury tabel w bazie danych, wykonaj migrację i zapełnij bazę danymi testowymi:

```
php artisan migrate --seed
```

#### **Krok 6: Uruchomienie serwera aplikacji**

```
php artisan serve
```

Aplikacja powinna działać na adresie: **http://127.0.0.1:8000**

### **4.3 Konfiguracja frontend (React)**

#### **Krok 1: Przejście do katalogu frontend**

```
cd ../frontend
```

#### **Krok 2: Instalacja zależności frontendowych**

```
npm install
```

#### **Krok 3: Uruchomienie aplikacji frontendowej**

Aby uruchomić frontend w trybie deweloperskim, wykonaj:

```
npm start
```

Aplikacja będzie dostępna pod adresem **http://localhost:5173**.

### **Kompilacja produkcyjna**

Aby przygotować wersję produkcyjną aplikacji, wykonaj:

```
npm run build
```

### **4.4 Testowanie API**

Aby sprawdzić, czy backend poprawnie działa, można użyć **Postman** lub narzędzia curl.

Przykładowe zapytanie do sprawdzenia pogody dla miasta:

```
curl -X GET "http://127.0.0.1:8000/api/weather?q=Warsaw"
```

Przykładowa odpowiedź:

```
{
  "city": "Warszawa",
  "temperature": 12.3,
  "humidity": 80,
  "wind_speed": 3.2
}
```

#### **4.5 Uruchomienie aplikacji na serwerze produkcyjnym**

Aby wdrożyć aplikację na serwerze, należy:

1. Skonfigurować środowisko **Linux (Ubuntu 20.04+)**.
2. Zainstalować **Nginx / Apache** oraz **PHP 8.0+**.
3. Skopiować pliki projektu i skonfigurować `.env`.
4. Skonfigurować **cron job** do uruchamiania harmonogramu pobierania danych pogodowych:  
`crontab -e`

### **5. Podsumowanie**

Projekt aplikacji pogodowej został zrealizowany w oparciu o nowoczesne technologie backendowe i frontendowe, zapewniając użytkownikom intuicyjny i funkcjonalny interfejs do monitorowania warunków pogodowych.

#### **6.1 Kluczowe funkcjonalności**

Aplikacja umożliwia użytkownikom:

- Wyszukiwanie miast i sprawdzanie ich aktualnej pogody.
- Rejestrację i logowanie, co pozwala na zapisywanie do **10 ulubionych miast**.
- Automatyczne pobieranie danych pogodowych co **30 minut** dla zapisanych miast.
- Przeglądanie historii pogody oraz analizowanie danych na interaktywnych wykresach.
- Personalizację ustawień konta użytkownika.

#### **6.2 Technologie wykorzystane w projekcie**

Podczas realizacji projektu zastosowano:

- **Backend:** Laravel (PHP) + MySQL
- **Frontend:** React + SASS
- **Baza danych:** MySQL
- **Autoryzacja użytkowników:** Laravel Sanctum
- **Zewnętrzne API:** OpenWeather API do pobierania danych pogodowych
- **Biblioteki frontendowe:** Axios, React Router, Recharts, Formik, Yup, Framer Motion

#### **6.3 Wnioski i dalsze możliwości rozwoju**

Aplikacja jest w pełni funkcjonalna, jednak istnieje kilka możliwych rozszerzeń: **Prognoza pogody na kilka dni** – obecnie wyświetlana jest tylko bieżąca pogoda, można dodać prognozę 5-7 dniową. **Powiadomienia o zmianach pogodowych** – użytkownicy mogliby otrzymywać alerty o nagłych zmianach temperatury czy opadach. **Optymalizacja pobierania danych** – zastosowanie cache'owania wyników API w bazie danych lub Redis. **Wersja mobilna** – dedykowana aplikacja mobilna z powiadomieniami push.

#### **6.4 Podsumowanie końcowe**

Projekt aplikacji pogodowej łączy nowoczesne technologie webowe i bazodanowe, zapewniając użytkownikom wygodne narzędzie do monitorowania warunków atmosferycznych. Dzięki zastosowaniu **React + Laravel** udało się stworzyć **wydajny, skalowalny i intuicyjny system**. Aplikacja została przetestowana pod kątem funkcjonalności i wydajności, a jej dalszy rozwój może obejmować kolejne usprawnienia i nowe funkcjonalności.