**UNIWERSYTET WARMIŃSKO MAZURSKI W OLSZTYNIE**

**WYDZIAŁ MATEMATYKI I INFORMATYKI**

**Dominik Mateusz Lewczyński**

**Kierunek: Informatyka**

**Specjalność: Data Science w Praktyce**

**Aplikacja internetowa do analizy danych pogodowych   
w celu przewidzenia prognozy pogody**

Praca inżynierska wykonana

w Katedrze Metod Matematycznych Informatyki

pod kierunkiem

dr Pawła Drozdę

**Olsztyn 2024.**

**UNIVERSITY OF WARMIA AND MAZURY IN OLSZTYN**

**THE FACULTY OF MATHEMATICS AND COMPUTER SCIENCE**

**Dominik Mateusz Lewczyński**

**Field of study: Computer science**

**Specialty: Data Science in Practice**

**Web application for analysing weather data in order to predict the weather forecast**

Engineering Thesis written in

Chair of Mathematical Methods of Computer Science

under the supervision of

dr hab. Paweł Drozda

**Olsztyn 2024.**

**Spis treści**

[**1.** **Streszczenie** 4](#_Toc125447174)

[**2.** **Abstract** 5](#_Toc125447175)

[**3.** **Wstęp** 6](#_Toc125447176)

[**4.** **Opis podrozdziałów** 7](#_Toc125447177)

[**5.** **Wymagania ogólne** 8](#_Toc125447178)

[**5.1.** **Cel pracy** 8](#_Toc125447179)

[**5.2.** **Podstawowe wymagania użytkownika** 8](#_Toc125447180)

[**6.** **Dokumentacja projektowa** 9](#_Toc125447181)

[**6.1.** **Wymagania sprzętowe** 9](#_Toc125447182)

[**6.2.** **Słownik pojęć systemowych** 9](#_Toc125447183)

[**6.3.** **Aktorzy** 9](#_Toc125447184)

[**6.4.** **Wymagania funkcjonalne** 10](#_Toc125447185)

[**6.5.** **Diagram przypadków użycia** 11](#_Toc125447186)

[**6.6.** **Diagram hierarchii funkcji** 12](#_Toc125447187)

[**6.7.** **Opis funkcjonalności** 13](#_Toc125447188)

[**7.** **Implementacja** 23](#_Toc125447189)

[**7.1.** **Strategia realizacji projektu** 23](#_Toc125447190)

[**7.2.** **Główne moduły aplikacji** 24](#_Toc125447191)

[**7.3.** **Wzorce wykorzystane w projekcie** 24](#_Toc125447192)

[**7.4.** **Przebieg informacji w systemie** 25](#_Toc125447193)

[**7.5.** **Frontend** 25](#_Toc125447194)

[**7.6.** **Backend** 26](#_Toc125447195)

[**7.7.** **Bazy danych**. 26](#_Toc125447196)

[**7.8.** **Testy** 26](#_Toc125447197)

[**8.** **Instrukcja użytkownika** 27](#_Toc125447198)

[**8.1.** **Interfejs strony głównej** 27](#_Toc125447199)

[**8.2.** **Konto użytkownika** 30](#_Toc125447200)

[**8.3.** **Dodawanie artykułu** 33](#_Toc125447201)

[**8.4.** **Przygotowanie do publikacji** 35](#_Toc125447202)

[**9.** **Licencje** 45](#_Toc125447203)

[**9.1.** **Narzędzia** 45](#_Toc125447204)

[**9.2.** **Grafiki** 45](#_Toc125447205)

[**9.3.** **Biblioteki zewnętrzne** 46](#_Toc125447206)

[**9.4.** **Źródła wiedzy programistycznej** 46](#_Toc125447207)

[**10.** **Podsumowanie** 47](#_Toc125447208)

[**11.** **Bibliografia** 48](#_Toc125447209)

# **Streszczenie**

Praca przedstawia proces tworzenia „**Aplikacja internetowa do analizy danych pogodowych w celu przewidzenia prognozy pogody**”. Ma to być aplikacja umożliwiająca **Użytkownikom** na przeglądanie dostępnych danych pogodowych oraz przeprowadzeniu prognozy pogody. **Dane pogodowe** są pobierane z zewnętrznego **API (Application Programming Interface)** i są przechowywane w bazie danych   
w celu dalszej analizy i wizualizacji. Każdy **Użytkownik**, który się zarejestruje ma mieć możliwość przeglądania danych historycznych i aktualnych, a także generować prognozy pogody dla wybranych lokalizacji. **Admin** jest jedynym użytkownikiem, który nie musi się rejestrować, ponieważ jest już dodany do systemu   
i ma możliwość zarządzania danymi pogodowymi i danymi lokalizacyjnymi oraz zarządza kontami użytkowników. Aplikację stworzyłem przy użyciu biblioteki React.js do frontendu i przy użyciu biblioteki Express.js oraz Flask do backendu oraz przy użyciu PostgreSQL do tworzenia bazy danych. Napisałem oraz przedstawiłem dokumentację projektową powstałą na podstawie wymagań użytkownika oraz wstępnych założeń. Dodałem obszerną instrukcję użytkownika wraz z odpowiednimi zrzutami ekranów. Całość wieńczy podsumowanie, mówiące o zrealizowaniu założeń wstępnych wraz ze wskazaniem możliwości dalszego rozwoju pracy.

# **Abstract**

The work presents the process of developing a '**Web application for analysing weather data to predict the weather forecast**'. It is intended to be an application that allows **Users** to view available weather data and carry out a weather forecast.   
**The weather data** is retrieved from an external **API (Application Programming Interface)** and is stored in a database for further analysis and visualisation. Any **User** who registers is to be able to view historical and current data, as well as generate weather forecasts for selected locations. **The Admin** is the only user who does not need to register, as he or she is already added to the system and has the ability to manage weather and location data and manages user accounts. I created the application using the React.js library for the frontend and using the Express.js library and Flask for the backend and using PostgreSQL to create the database. I wrote and presented design documentation created from user requirements and initial assumptions. I have added a comprehensive user guide with relevant screenshots. The project is rounded off with a summary of how the initial brief was realised and options for further development.

# **Wstęp**

Prognoza pogody odgrywa kluczową rolę w życiu człowieka, wpływając na wiele aspektów codziennego funkcjonowania oraz kluczowe dziedziny gospodarki, takie jak rolnictwo, transport, turystyka czy lotnictwo. Zapewnienie precyzyjnych   
i niezawodnych prognoz jest niezbędne do podejmowania trafnych decyzji oraz skutecznego planowania działań. W miarę wzrostu zmienności warunków atmosferycznych i złożoności procesów kształtujących pogodę, zapewnienie jak najbardziej precyzyjnych prognoz staje się priorytetem.

Tradycyjne metody prognozowania pogody, oparte głównie na ręcznej analizie danych meteorologicznych oraz doświadczeniu meteorologów, stają się coraz mniej skuteczne w obliczu rosnącej ilości dostępnych danych.

Rozwój technologii, takich jak stacje meteorologiczne, satelity meteorologiczne oraz modele numeryczne, przyczynił się do wzrostu ilości dostępnych danych. Jednakże, efektywne wykorzystanie tych danych do generowania precyzyjnych prognoz wymaga zaawansowanych narzędzi i technik analizy danych.

Jednym z najbardziej obiecujących podejść w prognozowaniu pogody jest wykorzystanie sztucznej inteligencji. Algorytmy uczenia maszynowego mogą efektywnie analizować duże ilości danych pogodowych i identyfikować wzorce, które mogą być wykorzystane do generowania prognoz. Dzięki sztucznej inteligencji możliwe jest także uwzględnianie wielu zmiennych i ich skomplikowanych zależności, co pozwala na bardziej precyzyjne prognozy pogody.

Wraz z rosnącą ilością danych pogodowych dostępnych z różnych źródeł pojawia się pilna potrzeba efektywnych narzędzi do ich analizy i interpretacji. Automatyzacja przetwarzania dużych ilości danych w czasie rzeczywistym oraz generowanie dokładnych prognoz na różne okresy staje się kluczowym wyzwaniem.

W odpowiedzi na to zapotrzebowanie, aplikacje internetowe stają się obiecującym narzędziem, umożliwiając dostęp do danych pogodowych oraz generowanie prognoz za pomocą intuicyjnych interfejsów użytkownika.

# **Opis podrozdziałów**

Niniejsza praca opisuje aplikację internetową stworzoną za pomocą technologii React.js, Express.js, Flask, jej możliwości, budowę, projekt oraz zakres działania. Wszystko to opisane jest w następujących rozdziałach:

* Wymagania ogólne – podstawowe założenia powstającej aplikacji,
  + Cel pracy – określono cel powstania aplikacji,
  + Podstawowe wymagania użytkownika – przedstawienie podstawowych wymagań użytkownika wobec działania aplikacji,
* Dokumentacja projektowa – część poświęcona dokumentacji projektowej
  + Wymagania sprzętowe – opis wymagań dotyczących użytkowania programu,
  + Słownik pojęć systemowych – opis poszczególnych pojęć systemowych
  + Aktorzy – opis poszczególnych Użytkowników, występujących   
    w systemie,
  + Wymagania funkcjonalne – przedstawienie głównych założeń programu,
  + Diagram przypadków użycia – część poświęcona diagramowi przypadków użycia,
  + Diagram hierarchii funkcji – część poświęcona funkcjonalnościom przedstawionym na diagramie hierarchii funkcji,
  + Opis funkcjonalności – część poświęcona szczegółowemu opisaniu wszystkich funkcjonalności.
* Instrukcja Użytkownika – opis działania programu oraz jego użytkowanie,
  + Interfejs strony głównej – część poświęcona przestawieniu strony głównej i innych stron,
  + Konto Użytkownika – czynności związane z kontem Użytkownika,
  + Przeglądanie danych pogodowych – część poświęcona przeglądaniu danych pogodowych po zalogowaniu do konta użytkownika,
  + Przeprowadzenie prognozy – część poświęcona procesowi przeprowadzenia prognozy pogody dla wybranej miejscowości na wybrane przez użytkownika dni.
* Licencje – opis licencji dotyczących technologii oraz elementów wykorzystanych w programie,
  + Narzędzia,
  + Grafika,
  + Biblioteki zewnętrzne,
  + Źródła wiedzy programistycznej.
* Podsumowanie – bilans całej pracy, przedstawienie wyciągniętych wniosków oraz oszacowanie stopnia zrealizowania założeń początkowych.
* Bibliografia – Lista materiałów wykorzystanych w calu zrobienia pracy magisterskiej.

# **Wymagania ogólne**

Celem niniejszego rozdziału jest przedstawienie, jakie były początkowe założenia oraz wymagania użytkownika docelowego wobec tworzonej aplikacji. W podrozdziale 3.1 opisuję, dlaczego tworzona jest aplikacja. W podrozdziale 3.2 wyjaśniam, jakie były wymagania użytkownika wobec tworzonej aplikacji.

## **Cel pracy**

Celem niniejszej pracy magisterskiej jest opracowanie aplikacji internetowej, która umożliwia analizę danych pogodowych w celu przewidzenia prognozy pogody.   
W dobie szybkiego rozwoju technologii informacyjnych i wzrostu dostępności danych meteorologicznych, istnieje coraz większe zapotrzebowanie na narzędzia umożliwiające skuteczną analizę tych danych w celu prognozowania warunków atmosferycznych.

Głównym celem aplikacji będzie zapewnienie użytkownikom możliwości dostępu do aktualnych i historycznych danych pogodowych, a także ich analizy za pomocą zaawansowanych algorytmów przetwarzania danych.

## **Podstawowe wymagania użytkownika**

Poniżej przedstawiam wymagania Użytkownika wobec tworzonej aplikacji internetowej:

* Najnowsza wersja przeglądarki internetowej do poprawnego działania aplikacji. Dostępne przeglądarki to np.: Google Chrome, Mozilla Firefox, Microsoft Edge oraz Opera.
* Komputer, na którym powinna działać aplikacja powinien spełniać minimalne wymagania. (procesor Pentium 4, pamięć RAM 4GB, system operacyjny Windows XP, dysk twardy 20GB)
* Połączenie z Internetem jest wymagane, ponieważ serwer musi stać na szybkim i stabilnym łączu.
* Aplikacja ma być intuicyjna oraz estetyczna. Użytkownik nie powinien mieć problemu z korzystaniem z aplikacji.
* Aplikacja musi być odpowiednio zabezpieczona by chronić dane użytkowników.
* Aplikacja powinna działać na dużych jak i mniejszych ekranach.
* Aplikacja musi być kompatybilna z innymi systemami IT.
* Aplikacja powinna być wydajna oraz powinna działać szybko.

# **Dokumentacja projektowa**

## **Wymagania sprzętowe**

Wymagania aplikacji internetowej czasopisma naukowego dotyczące sprzętu wyglądają następująco:

* Komputer lub urządzenie mobilne z dostępem do Internetu,
* Połączenie internetowe: Dowolne (Wi-Fi lub transfer sieci komórkowej),
* Zainstalowana przeglądarka internetowa: Do wyboru jest Google Chrome, Mozilla Firefox, Microsoft Edge, Safari oraz Opera).

Technologie jakich użyto do stworzenia aplikacji zapewnia działanie na każdej przeglądarce na każdym urządzeniu. Aplikacja jest także responsywna przez co jest dostosowana do każdego ekranu o różnej wielkości.

* 1. **Słownik pojęć systemowych:**
* **Admin** = imię + nazwisko + dane użytkownika + status użytkownika. Administrator systemu.
* **Użytkownik** = adres e-mail + hasło + status użytkownika. Osoba podlegająca autoryzacji w systemie.
* **Pogoda** = Miejscowość + Dane pogodowe + Przyszłe daty. Aktualne dane pogodowe
* **Prognoza pogody** = Miejscowość + Dane pogodowe + Przyszłe daty. Dane pogodowe na najbliższe dni
* **Raport** = Miejscowość + Prognoza pogody. Prognoza pogody w formie dokumentu do pobrania

## **Aktorzy**

* **Admin** – Użytkownik systemu zarządzać danymi użytkowników oraz zarządzać danymi lokalizacyjnymi oraz danymi pogodowymi
* **Użytkownik** – Użytkownik systemu, który podlega autoryzacji w systemie oraz ma za zadanie Przeglądać dane pogodowe oraz przeprowadzić prognozę pogody

## **Wymagania funkcjonalne**

* **Autoryzuj użytkownika** - zbiór usług pozwalających na zarządzanie kontem oraz dostęp do niego.
  + Zarejestruj się - usługa pozwalająca Autorowi utworzyć nowe konto.
  + Zaloguj się - usługa pozwalająca, po podaniu odpowiednich danych, uzyskać dostęp do konta danego Użytkownika lub recenzenta.
* **Zarządzanie kontem użytkowników** – usługa pozwalająca Adminowi na zarządzanie kontem użytkowników.
* **Zarządzanie tabelą miejscowości** – usługa pozwalająca Adminowi na Zarządzanie tabelą miejscowości poprzez dodawanie oraz usuwanie.
* **Zarządzanie tabelą danych pogodowych** – usługa pozwalająca Adminowi na Zarządzanie tabelą danych pogodowych poprzez aktualizowanie tabeli o nowe dane.
* **Przeglądanie Aktualnej pogody** – usługa pozwalająca przejrzenie aktualnej pogody dla wybranej miejscowości
* **Przeprowadzenie prognozy** – usługa pozwalająca użytkownikowi na uzyskanie prognozy pogody dla wybranej miejscowości na dane dni
* **Tworzenie raportu** – Usługa pozwalająca użytkownikowi na utworzenie raportu po dokonaniu prognozy

## **Diagram przypadków użycia**

Obraz zawierający tekst, diagram, linia, zrzut ekranu

Opis wygenerowany automatycznie

Rysunek 1 Diagram przypadków użycia dla aplikacji internetowej czasopisma naukowego

Na powyższym diagramie przypadków użycia została przedstawiona funkcjonalność aplikacji, z której mogą korzystać Użytkownicy.

## **Diagram hierarchii funkcji**

Rysunek 2 Diagram FHD

Na powyższym diagramie została umiejscowiona hierarchia poszczególnych funkcjonalności i dodana do odpowiednich Aktorów.

## **Opis funkcjonalności**

W tym podrozdziale opisze szczegółowo wszystkie funkcjonalności w postaci scenariuszy.

Tabela 1: Dokumentacja przypadku użycia Autoryzacja użytkownika

|  |  |
| --- | --- |
| Nazwa | Autoryzacja użytkownika |
| Numer | 0 |
| Twórca | Dominik Lewczyński |
| Poziom ważności | Wysoki |
| Typ przypadku użycia | Ogólny, niezbędny |
| Aktorzy | Admin, Użytkownik |
| Krótki opis | Przeprowadzenie autoryzacji użytkownika |
| Warunki wstępne | Dostęp do Systemu informatycznego |
| Warunki końcowe | Zalogowanie lub zarejestrowanie się do systemu |
| Główny przepływ zdarzeń | 1) Użytkownik loguje się do systemu. Podaje e-mail oraz hasło. Jeżeli dane są poprawne to Użytkownik zostaje zalogowany.  2) Użytkownik rejestruje się do systemu. Podaje imię, nazwisko oraz e-mail i hasło. Jeżeli dane zostały podane prawidłowo to Użytkownik zostaje zarejestrowany. |
| Alternatywne przepływy zdarzeń | 1a) Błędne dane – system wyświetla błąd. Użytkownik nie jest zalogowany do systemu. System ponawia czynność logowania  2a) Błędne dane – system wyświetla błąd. Konto Użytkownik nie zostaje utworzone. System ponawia czynność rejestracji |
| Wyjątki wymagania | Brak |
| Specjalne wymagania | Brak |
| Notatki i kwestie | Brak |

Tabela 2: Dokumentacja przypadku użycia Zarządzanie kontem użytkowników

|  |  |
| --- | --- |
| Nazwa | Zarządzanie kontem użytkowników |
| Numer | 1 |
| Twórca | Dominik Lewczyński |
| Poziom ważności | Wysoki |
| Typ przypadku użycia | Ogólny, niezbędny |
| Aktorzy | Admin |
| Krótki opis | Funkcja pozwalająca na zarządzanie kontem użytkowników (zmianę hasła, danych itp.) |
| Warunki wstępne | Admin jest zalogowany |
| Warunki końcowe | Admin zarządził danymi użytkowników |
| Główny przepływ zdarzeń | 1. Admin przechodzi do strony „Users” 2. Po wybraniu opcji Adminowi zostanie wyświetlona lista użytkowników 3. Admin ma możliwość zmiany emaila hasła oraz innych danych, a także możliwość usunięcia użytkownika 4. Admin Edytuje dane użytkowników. wybiera opcje „Edit przy wybranym użytkowniku. Wypełnia w formularzu pola jakie chce zaktualizować, a potem klika opcje „Save”. 5. Admin Usuwa użytkownika. Klika przycisk „Delete” przy wybranym użytkowniku. |
| Alternatywne przepływy zdarzeń | 3a) Admin nie musi zmieniać danych. Może w każdej chwili opuścić stronę zarządzania kontem.  4a) Admin może też kliknąć opcje „Anuluj” co anuluje wprowadzone zmiany.  5a) Admin nie musi usuwać użytkownika. Może w każdej chwili opuścić stronę zarządzania kontem. |
| Wyjątki wymagania | Brak |
| Specjalne wymagania | Brak |
| Notatki i kwestie | Brak |

Tabela 3: Dokumentacja przypadku użycia Zarządzanie tabelą miejscowości

|  |  |
| --- | --- |
| Nazwa | Zarządzanie tabelą miejscowości |
| Numer | 1 |
| Twórca | Dominik Lewczyński |
| Poziom ważności | Wysoki |
| Typ przypadku użycia | Ogólny, niezbędny |
| Aktorzy | Admin |
| Krótki opis | Funkcja pozwalająca na zarządzanie tabelą miejscowości (Dodawanie, Usuwanie) |
| Warunki wstępne | Admin jest zalogowany |
| Warunki końcowe | Admin zarządził tabelą miejscowości |
| Główny przepływ zdarzeń | 1. Admin przechodzi do strony „Locations” 2. Admin ma możliwość Dodawania nowej miejscowości oraz możliwość usunięcia wybranej miejscowości 3. Admin, Kiedy chce dodać nowa miejscowość wpisuje w formularz nazwę miejscowości i zatwierdza przyciskiem 4. Po odczekaniu Nowa lokalizacja zostaje dodana do tabeli Locations oraz dane pogodowe do tabeli  „Weather data”. 5. Admin, kiedy chce usunąć miejscowość naciska przycisk „Delete” przy wybranej lokalizacji. 6. Po naciśnięciu dane pogodowe z tabeli Weather data oraz dane lokalizacyjne zostają usunięte. |
| Alternatywne przepływy zdarzeń | 3a) Admin nie musi dodawać nowej lokalizacji. Może w każdej chwili opuścić stronę zarządzania tabelą miejscowości.  5a) Admin nie musi usuwać lokalizacji. Może w każdej chwili opuścić stronę zarządzania tabelą miejscowości. |
| Wyjątki wymagania | Brak |
| Specjalne wymagania | Brak |
| Notatki i kwestie | Brak |

Tabela 4: Dokumentacja przypadku użycia Zarządzanie tabelą danych pogodowych

|  |  |
| --- | --- |
| Nazwa | Zarządzanie tabelą danych pogodowych |
| Numer | 1 |
| Twórca | Dominik Lewczyński |
| Poziom ważności | Wysoki |
| Typ przypadku użycia | Ogólny, niezbędny |
| Aktorzy | Admin |
| Krótki opis | Funkcja pozwalająca na zarządzanie tabelą danych pogodowych (Odświeżenie) |
| Warunki wstępne | Admin jest zalogowany |
| Warunki końcowe | Admin zarządził tabelą danych pogodowych |
| Główny przepływ zdarzeń | 1. Admin przechodzi do strony „Weather data” 2. Admin ma możliwość odświeżenia danych w celu pobrania najnowszych danych pogodowych 3. Admin, Kiedy chce odświeżyć dane naciska na przycisk „Refresh Data”. 4. Po odczekaniu dane pogodowe zawierają najnowsze aktualne dane. |
| Alternatywne przepływy zdarzeń | 3a) Admin nie musi odświeżać danych. Może w każdej chwili opuścić stronę zarządzania tabelą danych pogodowych. |
| Wyjątki wymagania | Brak |
| Specjalne wymagania | Brak |
| Notatki i kwestie | Brak |

Tabela 5: Dokumentacja przypadku użycia Przeglądanie aktualnej pogody

|  |  |
| --- | --- |
| Nazwa | Przeglądanie aktualnej pogody |
| Numer | 2 |
| Twórca | Dominik Lewczyński |
| Poziom ważności | Wysoki |
| Typ przypadku użycia | Ogólny, niezbędny |
| Aktorzy | Admin, Użytkownik |
| Krótki opis | Przegląd pogody dla wybranej miejscowości |
| Warunki wstępne | Użytkownik przegląda listę publikacji |
| Warunki końcowe | Użytkownik po wybraniu publikacji może ja przeczytać |
| Główny przepływ zdarzeń | 1. Użytkownik na stronie głównej ma wyświetloną listą miejscowości. 2. Użytkownik ma możliwość wybrania nazwy miejscowości by zobaczyć aktualną pogodę. 3. Po wybraniu nazwy miejscowości Użytkownikowi pokazuje się aktualna pogoda dla wybranej miejscowości. |
| Alternatywne przepływy zdarzeń | Brak |
| Wyjątki wymagania | Brak |
| Specjalne wymagania | Brak |
| Notatki i kwestie | Użytkownik może w każdej chwili opuścić stronę |

Tabela 6: Dokumentacja przypadku użycia Przeprowadzenie prognozy

|  |  |
| --- | --- |
| Nazwa | Przeprowadzenie prognozy |
| Numer | 3 |
| Twórca | Dominik Lewczyński |
| Poziom ważności | Wysoki |
| Typ przypadku użycia | Ogólny, niezbędny |
| Aktorzy | Użytkownik, Admin |
| Krótki opis | Przeprowadzenie prognozy pogody dla wybranej miejscowości |
| Warunki wstępne | Użytkownik musi być zalogowany |
| Warunki końcowe | Użytkownik otrzymał prognozę pogody |
| Główny przepływ zdarzeń | 1) Użytkownik po zalogowaniu na stronie głównej wybiera opcje „Forecast Weather”  2) Po wybraniu opcji Użytkownik jest przekierowany do formularza, w którym wybiera nazwę miejscowości oraz na ile dni ma być prognoza. Następnie klika przycisk  „Predict Forecast Weather”.  3) Po wypełnieniu formularza system przeprowadzi prognozę. Po przeprowadzeniu prognozy Użytkownik otrzymuje wynik w postaci danych pogodowych na najbliższe dni jakie wybrano |
| Alternatywne przepływy zdarzeń | 4a) jeśli formularz nie zostanie wypełniony w całości pojawi się odpowiedni komunikat. |
| Wyjątki wymagania | Brak |
| Specjalne wymagania | Brak |
| Notatki i kwestie | Brak |

Tabela 7: Dokumentacja przypadku użycia Tworzenie raportu

|  |  |
| --- | --- |
| Nazwa | Tworzenie raportu |
| Numer | 4 |
| Twórca | Dominik Lewczyński |
| Poziom ważności | Wysoki |
| Typ przypadku użycia | Ogólny, niezbędny |
| Aktorzy | Użytkownik, Admin |
| Krótki opis | Wygenerowanie raportu po przeprowadzeniu prognozy |
| Warunki wstępne | Prognoza pogody została wykonana |
| Warunki końcowe | Raport został wygenerowany |
| Główny przepływ zdarzeń | 1. Po przeprowadzeniu prognozy Użytkownikowi pokazują się wyniki działania systemu z dostępnym przyciskiem wygeneruj raport 2. Po kliknięciu w przycisk W nowej karcie przeglądarki pokaże się raport w formie pliku pdf zawierający dane  z przeprowadzonej prognozy. Użytkownik może mieć możliwość pobrania raportu |
| Alternatywne przepływy zdarzeń | Brak |
| Wyjątki wymagania | Brak |
| Specjalne wymagania | Brak |
| Notatki i kwestie | Użytkownik nie musi tworzyć raportu. Może w każdej chwili opuścić stronę z wynikami |

# **Implementacja**

W niniejszym rozdziale opisałem szczegóły dotyczące implementacji aplikacji internetowej do analizy danych pogodowych. W kolejnych podrozdziałach przedstawiłem jaka była strategia realizacji projektu, z jakich głównych modułów składa się aplikacja, jakie wzorce wykorzystano w projekcie, jaki jest przebieg informacji a także szczegółowo opisałem technologie zastosowane na frontendzie   
i backendzie oraz jaką bazę danych wykorzystałem i w jaki sposób przeprowadziłem testy. Całą aplikację napisałem za pomocą języka programowania JavaScript   
i Pythona oraz frameworków opartych o ten język.

## **Strategia realizacji projektu**

Implementowanie aplikacji internetowej czasopisma naukowego realizowałem zgodnie z następującą strategią:

* **Projektowanie bazy danych**: W pierwszej kolejności została zaprojektowana baza danych PostgreSQL, która będzie przechowywać dane dotyczące Lokalizacji, Pogody, Użytkowników, oraz inne dane potrzebne do działania aplikacji. Szczegóły odnośnie do tego co wykorzystałem do stworzenia bazy danych znajduje się w folderze [**7.7 Bazy danych**](#_Bazy_danych.),
* **Tworzenie backendu**: Stworzyłem backend za pomocą frameworka Express.js oraz Flask (Framework Pythona) który odpowiada ze komunikację z bazą danych, przetwarzanie danych oraz udostępnianie danych do części frontendowej. Więcej szczegółów o tym co użyłem do stworzenia frontendu znajduje się w podrozdziale [**7.6 Backend**](#_Backend),
* **Testowanie backendu**: Po stworzeniu backendu zacząłem testowanie poprzez dodawanie nowych użytkowników oraz Dodanie nowych Lokalizacji, aby upewnić się, że wszystko działa prawidłowo oraz że integracja z bazą danych istnieje. Więcej szczegółów o tym czego użyłem do testowania znajduje się w podrozdziale [**7.8 Testy**](#_Testy).
* **Tworzenie frontendu**: Opracowałem frontend aplikacji przy użyciu React.js ze zdefiniowanym szablonem Redux. Do wyglądu aplikacji zastosowałem framework CSS o nazwie Bulma, natomiast do wysyłania wiadomości e-mail wykorzystałem EmailJS. Więcej szczegółów o tym co użyłem do stworzenia frontendu znajduje się w podrozdziale [**7.5 Frontend**](#_Frontend),
* **Integracja**. Następnie frontend i backend połączyłem ze sobą, aby umożliwić komunikacje między nimi,
* **Testowanie aplikacji**: Na koniec testowałem aplikację korzystając z niej, aby upewnić się, że całość działa prawidłowo i nie występują problemy   
  w czasie użytkowania.

## **Główne moduły aplikacji**

Aplikacja składa się z kilku głównych modułów:

* **Panel administracyjny**: Umożliwia Adminowi na zarzadzanie Lokalizacjami oraz danymi pogodowymi, a także zapewnia zarządzenie kontami Użytkowników.
* **System Prognozowania pogody**: Pozwala Użytkownikom na przeprowadzenie prognozy pogody poprzez wybranie miejscowości z listy oraz wybranie ilości dni na jakie chce się przeprowadzić prognozę.
* **Witryna publiczna**: Ten moduł zawiera stronę internetowa, która jest dostępna dla wszystkich użytkowników i umożliwia przeglądanie pogody na aktualną porę dnia oraz sprawdzenia informacji na temat projektu

## **Wzorce wykorzystane w projekcie**

W aplikacji zastosowano następujące wzorce projektowe:

* **Wzorzec Model-View-Controller (MVC)**: Wzorzec ten pozwolił na oddzielenie logiki biznesowej od logiki prezentacji. W aplikacji, backend odpowiada za logikę biznesową, natomiast frontend zajmuje się prezentacją danych.
* **Wzorzec Store**: Wzorzec ten został wykorzystany w połączeniu z React   
  i Redux. Umożliwia on centralizację stanu aplikacji i ułatwia jego zarządzanie.
* **Wzorzec Singleton**: Wzorzec ten został użyty do klasy odpowiedzialnej za połączenie z bazą danych, zapewnił on, że w całej aplikacji jest tylko jedna instancja tej klasy, co umożliwia łatwiejsze zarządzanie połączeniem z bazą danych.

## **Frontend**

Frontend aplikacji został stworzony z użyciem biblioteki React. Jest ona oparta   
o język JavaScript. Komenda „**create-react-app**” z argumentem **template redux** pozwoliła na stworzenie aplikacji w React z predefiniowanym szablonem Redux. Zawiera on niezbędne pliki i konfiguracje do implementacji stanu aplikacji za pomocą biblioteki Redux. Biblioteki wymienione poniżej były najbardziej przydatne w czasie tworzenia interfejsu aplikacji oraz do integracji z backendem.

* + **React-router-dom** – pozwala na implementację nawigacji aplikacji.
  + **Axios** – służy do komunikacji z API.
  + **Moment** – pozwala na wyświetlenie daty w odpowiednim formacie.
  + **Bulma** – framework CSS daje gotowe komponenty, które pozwoliły na stworzenie solidnego i responsywnego interfejsu.
  + **@Redux.js/tollkit** – ułatwia tworzenie aplikacji z wykorzystaniem biblioteki Redux. Zawiera on skróty dla najczęściej stosowanych operacji, takich jak konfiguracja i tworzenie store'a, tworzenie akcji i reduktorów oraz ułatwia pracę z typami danych.
  + **Emailjs** – pozwala na wysyłanie wiadomości e-mail z poziomu aplikacji.

## **Backend**

Backend aplikacji został stworzony z użyciem frameworka Express.js oraz dodatkowego frameworka Flask do przeprowadzania prognoz pogody. Wykorzystałem dodatkowe biblioteki, które pomogły w napisaniu backendu:

* + **Dotenv** – pozwala na ładowanie pliku .env z konfiguracją do projektu.
  + **Cors** – umożliwia obsługę mechanizmu CORS (Cross-Origin Resource Sharing) w aplikacji express. Jest to potrzebne, jeśli aplikacja korzysta   
    z zasobów z innych domen.
  + **Pg** – zezwala na połączenie z bazą danych PostgreSQL za pomocą Node.js.
  + **Sequelize** – segment ORM (Object-Relational Mapping) dla Node.js, który ułatwia pracę z bazami danych, bez konieczności pisania zapytań SQL.
  + **Connect-session-sequelize** – przyzwala na trzymanie sesji użytkownika   
    w bazie danych za pomocą ORM Sequelize.
  + **Express-session** – umożliwia obsługę sesji użytkownika w aplikacji express.
  + **Argon2** – daje możliwość hashowanie haseł za pomocą algorytmu Argon2, który jest uważany za jeden z bezpieczniejszych sposobów hashowania.
  + **Scikit-learn** – Biblioteka do uczenia maszynowego w Pythonie.
  + **Numpy** – Biblioteka do obliczeń naukowych w Pythonie.
  + **Pandas** – Biblioteka do manipulacji danymi i analizy danych w Pythonie
  + **Flask-SQLAlchemy** – Rozszerzenie do Flask'a ułatwiające integrację z bazą danych za pomocą SQLAlchemy.

Każdy z tych modułów pozwolił na stworzenie działającego zaplecza, które łączy się   
z bazą danych. Dają one możliwość tworzenia konta, logowania się na nie oraz umożliwiają pobieranie danych z zewnętrznego API do bazy danych oraz przeprowadzenie prognozy pogody.

## **Bazy danych**.

Do przechowywania danych Użytkowników oraz danych lokalizacyjnych oraz danych pogodowych została wykorzystana baza danych PostgreSQL. Aby ją uruchomić zastosowałem narzędzie pgAdmin dostarczone wraz z instalatorem PostgreSQL co pozwoliło na proste stworzenie bazy danych potrzebnej do stworzenia aplikacji. Jako danych przekazywanych do bazy posłużyłem się zewnętrznym API Open-Meteo które pozwala na dostęp do danych pogodowych na podstawie współrzędnych geograficznych.

## **Testy**

Testowanie aplikacji przeprowadziłem za pomocą narzędzia Postman. Realizowałem je na różnych etapach tworzenia aplikacji, aby upewnić się, że każda   
z funkcjonalności działa poprawnie. Dodatkowo testowanie wykonałem również poprzez używanie aplikacji przez Użytkowników końcowych.

# **Instrukcja użytkownika**

W niniejszym rozdziale widnieje instrukcja użytkowania z **Aplikacji internetowej czasopisma naukowego**. Na opisanych zrzutach ekranu przedstawiony został interfejs graficzny.

## **Interfejs strony głównej**

Po uruchomieniu aplikacji w wybranej przeglądarce internetowej pierwszym widokiem jakie użytkownik zobaczy to jest widok strony głównej, który wygląda następująco:

To jest strona główna aplikacji, gdzie jest wyświetlana pogoda na aktualna porę dzisiejszego dnia. Wyżej znajduje się lista z dostępnymi lokalizacjami. Po wybraniu innej miejscowości zobaczymy inne dane pogodowe.

Na górze strony znajduje się nawigacja, za pomocą której zostaniemy przekierowani do odpowiednich stron:

* „About us” – strona, w której jest zawarte informacje o właścicielach aplikacji
* „About App” – strona, w której jest zawarte informacje na temat aplikacji (Na temat działania, użytych technologiach itd.)
* „Contact” – strona, na której umieszczony jest formularz kontaktowy. Za jego pomocą można wysłać wiadomość np. można zapytać się o szczegółowe informacje.
* „Register” – strona, na której jest formularz do rejestracji nowego konta
* „Login” – strona, z formularzem do logowania się do swojego konta

## **Konto użytkownika**

Jeżeli Użytkownik zdecyduje się na utworzenie konta na przykład konto „Autor” do dodawania publikacji, Użytkownik wybiera na stronie głównej przycisk „Register”. Zostanie wówczas otwarty formularz do rejestracji konta.

Następnie w formularzu tym wypełnia on niezbędne dane. Na poniższym przykładzie pokazałem utworzenie konta dla Użytkownika Jan Kowalski

.

Kiedy konto zostanie utworzone Użytkownik Jan Kowalski wybiera na górze   
z prawej strony przycisk „Login” i zostaje przekierowany do strony z formularzem do logowania na konto.

Po naciśnięciu przycisku „Login” na formularzu, Użytkownik zostaje zalogowany na swoje konto.

W ten oto sposób konto Użytkownika zostało utworzone oraz zalogowano się   
na nie. Na powyższym zdjęciu znajduje się główny panel konta Użytkownika.   
Po lewej stronie jest nawigacja, która zmienia się w zależności od roli jaką wybrano   
w czasie rejestracji. W tym przykładzie znajduje się link do strony Panelu Głównego   
i link do strony z listą publikacji. Dla nowo utworzonego konta „Autor” strona   
z publikacjami wygląda następująco:

# **Licencje**

Celem niniejszego rozdziału jest przestawienie środowisk, grafiki, bibliotek zewnętrznych oraz źródeł wiedzy programistycznej jakie zostały wykorzystane w celu stworzenia aplikacji.

## **Narzędzia**

* [**Visual Studio Code**](https://code.visualstudio.com/)– darmowy edytor kodu z kolorowaniem stworzony przez firmę Microsoft, o otwartym kodzie źródłowym na licencji MIT.
* [**Google Chrome**](https://www.google.com/intl/pl_pl/chrome/)– darmowa przeglądarka internetowa na urządzenia stacjonarne i mobilne, za pomocą, której wpisując hasło w wyszukiwarce Google, możemy wyszukiwać i poruszać się po stronach internetowych oraz przeglądać na nich treści w celu znalezienia potrzebnych dla nas informacji.
* [**Postman**](https://www.postman.com/)–darmowe narzędzie, które służy do testowania i debugowania interfejsów API (Application Programming Interface). Pozwala na tworzenie, edytowanie i wysyłanie żądań HTTP oraz przeglądanie odpowiedzi   
  z serwera. Może być używane do testowania różnych typów żądań, takich jak GET, POST, PUT i DELETE, a także do wysyłania danych w różnych formatach, takich jak JSON czy XML.
* [**Node.js**](https://nodejs.org/en/) – darmowa platforma programistyczna oparta na języku JavaScript, która pozwala na tworzenie aplikacji serwerowych. Dzięki Node.js, programiści mogą korzystać z jednego języka (JavaScript) na obu stronach aplikacji (frontend i backend) co pozwala na zwiększenie produktywności   
  i jednolity styl kodowania.
* [**Open-Meteo**](https://nodejs.org/en/) – Darmowe API pogodowe o otwartym kodzie źródłowym, z którego może korzystać każdy bez potrzeby rejestracji czy uzyskiwania klucza API. Platforma kieruje się ideą, że dostęp do rzetelnych informacji pogodowych powinien być powszechnie dostępny

## **Grafiki**

* [**Font awesome**](https://fontawesome.com/) - źródło darmowych ikon w dowolnym formacie.

## **Biblioteki zewnętrzne**

* [**React.js**](https://reactjs.org/) – biblioteka języka programowania JavaScript, która umożliwia tworzenie interfejsów graficznych w aplikacjach internetowych.
* [**Express.js**](http://expressjs.com/)– back-endowy framework aplikacji internetowych do budowania interfejsów API RESTful z Node.js, wydanym jako darmowe oprogramowanie o otwartym kodzie źródłowym na licencji MIT. Przeznaczony jest do budowania aplikacji internetowych i API.
* **Flask** – back-endowy framework dla języka Python, który umożliwia szybkie tworzenie aplikacji internetowych. Jest lekki i łatwy w użyciu, co czyni go popularnym wyborem dla projektów, które wymagają szybkiego prototypowania i prostego zarządzania.
* [**Bulma**](https://bulma.io/) - bezpłatna platforma typu open source, która zapewnia gotowe do użycia komponenty frontendowe, które można łatwo łączyć   
  w celu tworzenia responsywnych interfejsów internetowych.
* [**EmailJS**](https://www.emailjs.com/) - biblioteka JavaScript, która pomaga wysyłać wiadomości e-mail przy użyciu technologii po stronie klienta. Zaletą EmailJS jest to, że nie są wymagane żadne serwery; wszystko, co należy zrobić, to podłączyć go do jednej z obsługiwanych usług e-mail, utworzyć szablon wiadomości e-mail   
  i użyć EmailJS do uruchomienia wiadomości e-mail.

## **Źródła wiedzy programistycznej**

* **StackOverflow** – internetowa społeczność programistyczna z całego świata. Opisane tam artykuły stanowiły pomoc przy rozwiązywaniu problemów programistycznych jak i przy nauce pisania czytelnego kodu.

# **Podsumowanie**

Niniejsza praca magisterska dotyczyła stworzenia aplikacji internetowej do analizy danych pogodowych i prognozowania pogody. Celem projektu było opracowanie nowoczesnego i funkcjonalnego narzędzia, które umożliwi użytkownikom łatwe przeglądanie danych pogodowych oraz samodzielne prognozowanie pogody.

Aplikacja została stworzona zgodnie z początkowymi założeniami. Wszystkie aspekty jej funkcjonalności realizują początkowe założenia. Aplikacja Pozwala na przeglądanie danych pogodowych. Aplikacja pobiera dane pogodowe z zewnętrznego źródła i prezentuje je w przejrzysty sposób, umożliwiając użytkownikom łatwe sprawdzenie pogody w dowolnym miejscu na świecie. Za pomocą aplikacji można Analizować dane pogodowe. Aplikacja posiada narzędzia do analizy danych pogodowych, takie jak wykresy i tabele, które pozwalają użytkownikom na identyfikację trendów i zależności w danych. Użytkownik ma także możliwość prognozowania pogody. Aplikacja wykorzystuje modele uczenia maszynowego do prognozowania pogody na różne okresy. Użytkownicy mogą generować prognozy dla konkretnych lokalizacji i sprawdzać szczegółowe informacje, takie jak temperatura, opady, wiatr.

Aplikacja spełnia wszystkie założone cele i stanowi funkcjonalne narzędzie   
do analizy danych pogodowych i prognozowania pogody. Aplikacja internetowa   
do analizy danych pogodowych może zostać rozbudowana na wiele sposobów. Można na przykład zintegrować aplikacje z dodatkowymi źródłami danych pogodowych,   
np. z lokalnymi stacjami meteorologicznymi lub satelitami. Można także rozszerzyć możliwości analizy, czyli Zaimplementowanie bardziej zaawansowanych narzędzi do analizy danych, takich jak modele statystyczne i algorytmy uczenia maszynowego. Istnieje także możliwość Udoskonalenia modeli prognozowania, czyli opracowanie bardziej precyzyjnych modeli prognozowania pogody wykorzystując zaawansowane techniki uczenia maszynowego.

W trakcie realizacji projektu zastosowałem najnowsze technologie po stronie frontendu, takie jak React.js z narzędziami Redux Toolkit oraz frameworkiem Bulma, oraz technologie po stronie backendu takie jak Express.js oraz Flask, co pozwoliło na stworzenie szybkiej i responsywnej w pełni działającej aplikacji. Przeprowadzone testy poprzez korzystanie z aplikacji oraz narzędzia Postman wykazały, że aplikacja jest stabilna i spełnia wszystkie wymagania funkcjonalne.

Proces stworzenia aplikacji internetowej do analizy danych pogodowych zakończył się sukcesem. Otrzymano nowoczesne i funkcjonalne narzędzie, które umożliwia użytkownikom łatwy dostęp do danych pogodowych i samodzielne prognozowanie pogody. Aplikacja ta jest stabilna i spełnia wszystkie założone wymagania funkcjonalne.

Aplikacja stanowi wartościowe narzędzie dla osób, które potrzebują dostępu   
do rzetelnych danych pogodowych i chcą samodzielnie prognozować pogodę. Może być ona wykorzystywana przez osoby prywatne, jak również przez firmy i instytucje działające w różnych branżach, np. rolnictwie, turystyce, lotnictwie czy transporcie.

# **Bibliografia**