Raport Końcowy Projektu z przedmiotu Bazy Danych

System Informacji Turystycznej - Honalolo

Autorzy:
Marcel Lis
Piotr Juszkiewicz
Karol Rozmus
Jan Lelko
Adam Haber

Prowadzący:
dr hab. inż. prof. PŚ Bożena Małysiak-Mrozek

06-02-2026

Spis treści

Ogólne założenia projektu ... 3
Dołączone materiały z poprzednich etapów ... 3
Wybór systemu i technologie ... 6
Dlaczego system webowy? ... 6
Zastosowane narzędzia i technologie ... 6
Specyfikacja zewnętrzna ... 7
Wymagania sprzętowe ... 7
Scenariusz 1: Generowanie raportu PDF (Funkcja dla Administratora) ... 7
Scenariusz 2: Wyszukiwanie i filtrowanie atrakcji ... 9
Specyfikacja wewnętrzna ... 10
Architektura systemu ... 10
Analiza implementacji wybranych funkcjonalności ... 10
Funkcjonalność 1: Dodawanie nowej atrakcji (Polimorfizm) ... 10
Podsumowanie i wnioski ... 11

Ogólne założenia projektu

Celem projektu jest stworzenie systemu informacji turystycznej „Honalolo”, umożliwiającego gromadzenie, przeglądanie oraz zarządzanie bazą atrakcji turystycznych. System ma charakter platformy webowej, obsługującej różne typy obiektów, takie jak hotele, restauracje, szlaki czy wydarzenia kulturalne.

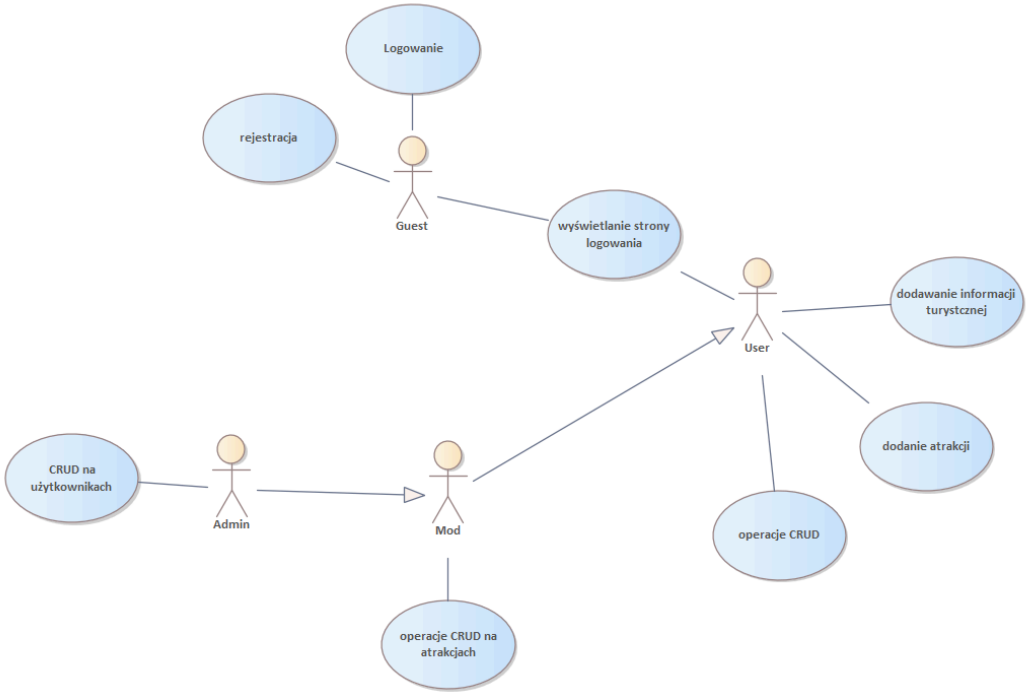
Główne funkcjonalności systemu to:

- Przeglądanie i filtrowanie atrakcji przez turystów.
- Rejestracja i logowanie użytkowników.
- Panel administracyjny do zarządzania treścią i generowania raportów PDF.
- Obsługa multimediów (galerie zdjęć atrakcji).

Dołączone materiały z poprzednich etapów

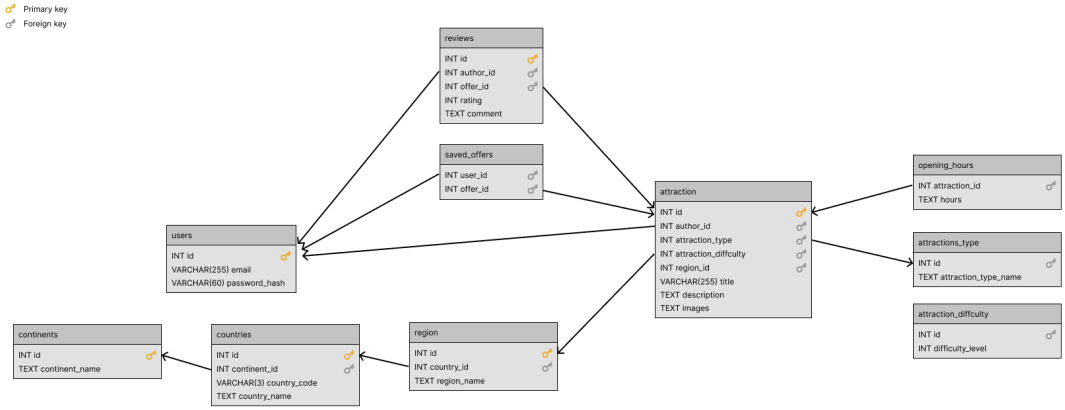
Zgodnie z wymaganiami, do projektu dołączono (jako załączniki):

- Hierarchię ról (Administrator, Zarejestrowany Użytkownik, Gość).



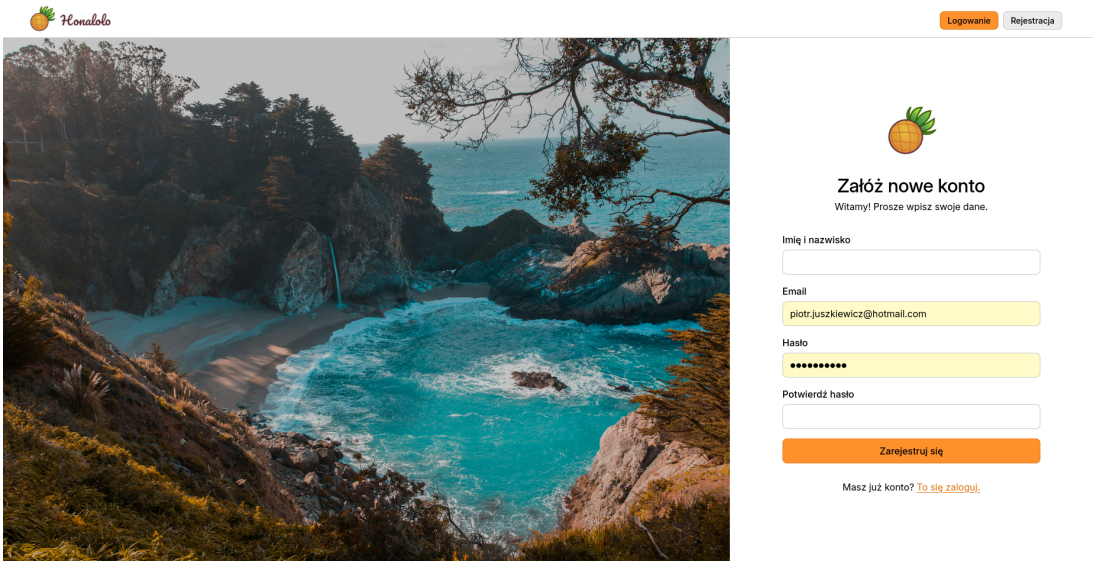
Rysunek 1: Hierarchia ról w systemie

- Model logiczny i fizyczny bazy danych (diagramy ERD).

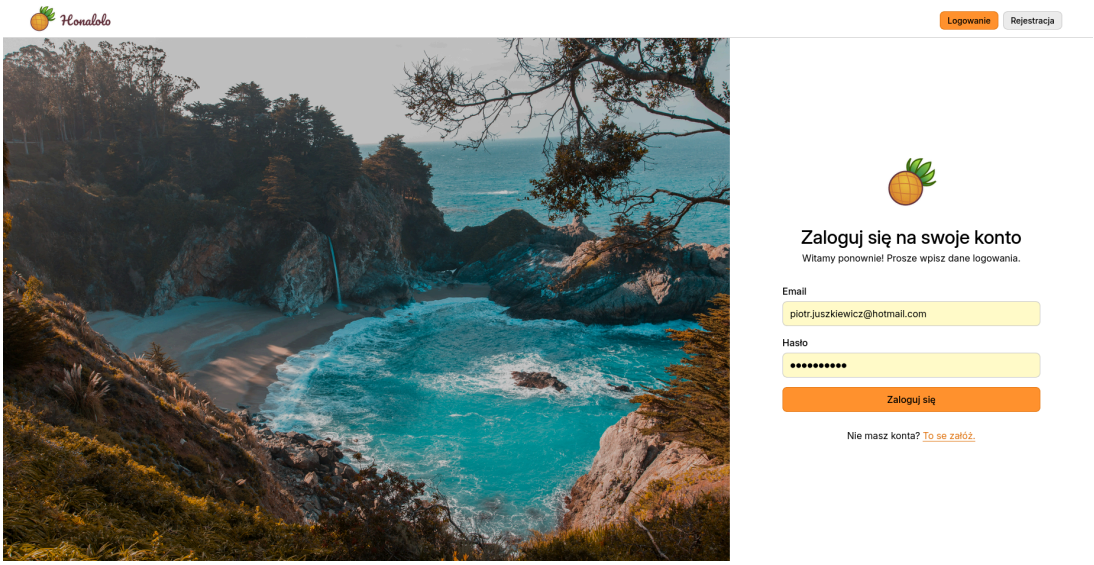


Rysunek 2: Diagram relacji encji (ERD)

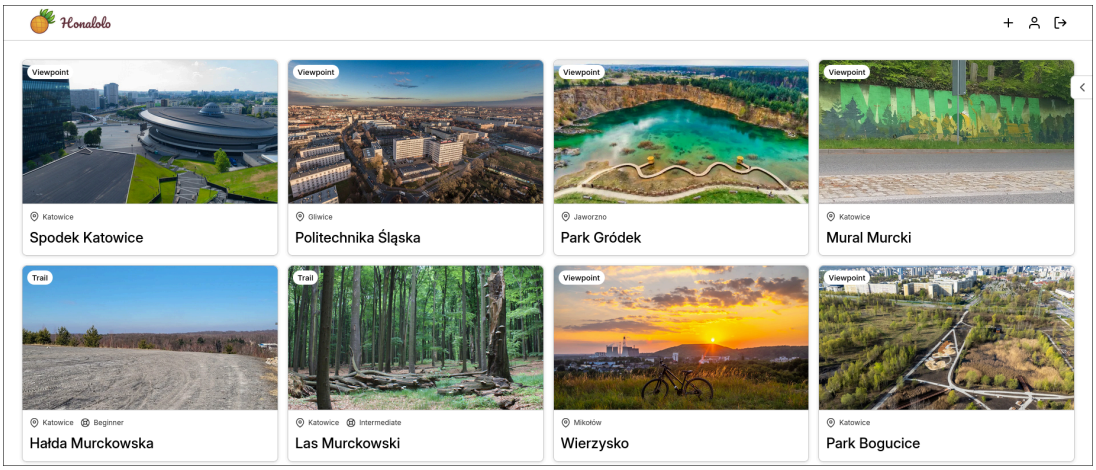
- Szkice interfejsu użytkownika (widoki React).



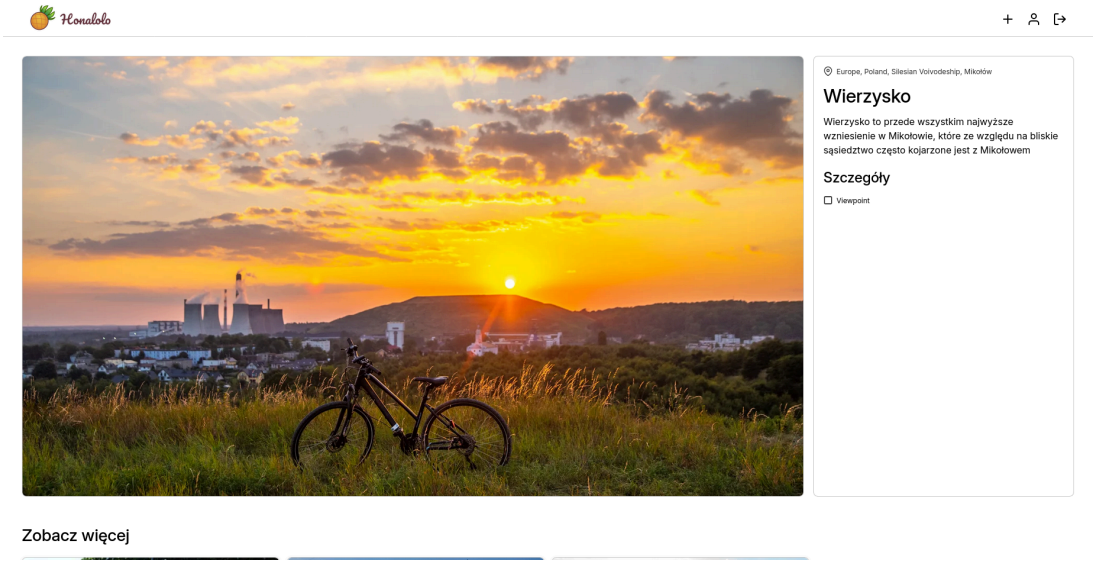
Rysunek 3: Szkic interfejsu rejestracji użytkownika



Rysunek 4: Szkic interfejsu logowania użytkownika



Rysunek 5: Widok główny z listą atrakcji i filtrami



Rysunek 6: Widok szczegółowy pojedynczej atrakcji

Wybór systemu i technologie

Dlaczego system webowy?

Wybrano architekturę aplikacji internetowej (Web App) ze względu na:

- **Dostępność:** Użytkownicy mogą korzystać z systemu na dowolnym urządzeniu z przeglądarką (komputer, tablet, smartfon) bez konieczności instalacji.
- **Centralizacja danych:** Jedna baza danych aktualizowana w czasie rzeczywistym dla wszystkich użytkowników.
- **Łatwość aktualizacji:** Zmiany w logice biznesowej lub interfejsie są widoczne natychmiast po wdrożeniu na serwer.

Zastosowane narzędzia i technologie

W projekcie wykorzystano nowoczesny stos technologiczny, podzielony na warstwę frontendową i backendową:

Frontend (Warstwa Prezentacji):

- **React 19:** Biblioteka do budowania interfejsów użytkownika.
- **Vite:** Narzędzie do budowania i serwowania aplikacji (HMR).
- **TailwindCSS v4:** Framework CSS do stylizacji (Utility-first).
- **Feather Icons:** Zestaw ikon wektorowych.

Backend (Warstwa Logiki i Danych):

- **.NET 8 (C#):** Platforma wykonawcza po stronie serwera.
- **Entity Framework Core:** System ORM do komunikacji z bazą danych.
- **QuestPDF:** Biblioteka do generowania raportów PDF (zgodnie z wymaganiami).
- **MySQL / MariaDB:** Relacyjna baza danych (zależnie od konfiguracji Docker).
- **Docker:** Konteneryzacja środowiska uruchomieniowego bazy danych.

Specyfikacja zewnętrzna

Wymagania sprzętowe

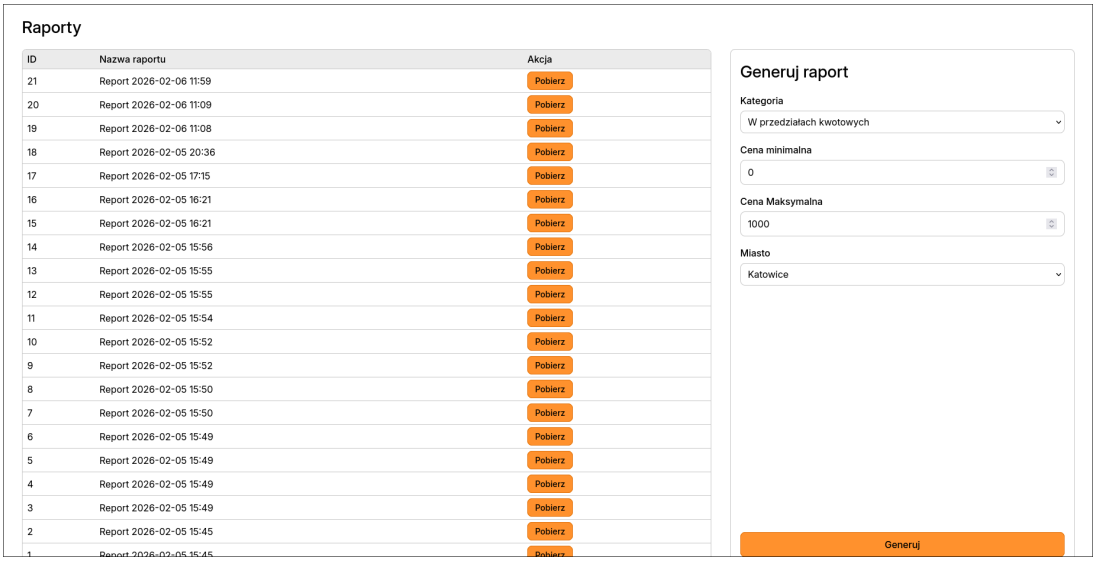
Do poprawnego działania systemu wymagane są:

- Serwer aplikacji: CPU 2-rdzeniowy, 4GB RAM, obsługa konteneru dla bazy danych w Docker oraz środowiska .NET 8.
- Klient: Dowolne urządzenie z przeglądarką internetową (Chrome, Firefox, Edge) obsługującą JavaScript i HTML5.

Scenariusz 1: Generowanie raportu PDF (Funkcja dla Administratora)

Jest to kluczowa funkcjonalność wymagana w projekcie.

1. Użytkownik loguje się na konto z uprawnieniami Administratora.
2. Przechodzi do zakładki „Raporty”.
3. Wybiera parametry raportu (np. zakres dat, miasto, zakres cenowy).
4. System przetwarza dane i generuje dokument PDF zawierający statystyki oraz listę atrakcji.
5. Następuje automatyczne pobranie pliku PDF na urządzenie użytkownika.



Rysunek 7: Interfejs wyboru parametrów raportu PDF

Honalolo Admin Report: Report 2026-02-06 11:59

Generated At: 2026-02-06 11:59
Total Attractions: 8
Average Price: 15.38 PLN

Filter Parameters

Parameter	Value
Min Price	0 PLN
Max Price	1000 PLN
City	Katowice

Attractions by Type

Type	Count
Trail	2
Hotel	1
Restaurant	1
Viewpoint	4

Top 5 Most Expensive

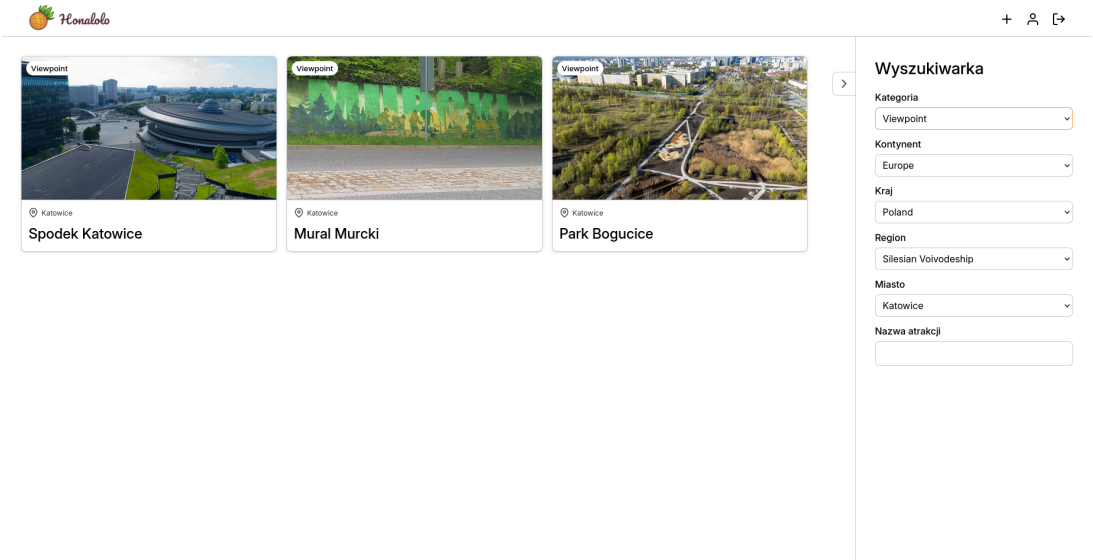
Attraction Name	Event Date	Price
Dzień dobry	-	123.00 PLN
Hałda Murckowska	-	0.00 PLN
Las Murckowski	-	0.00 PLN
Aparamenty Madera na Murckach	-	0.00 PLN
Bar Selene Murcki	-	0.00 PLN

Page 1

Rysunek 8: Widok wygenerowanego raportu PDF z listą atrakcji i statystykami

Scenariusz 2: Wyszukiwanie i filtrowanie atrakcji

1. Użytkownik wchodzi na stronę główną (HomePage).
2. W pasku wyszukiwania wprowadza kryteria:
   - Nazwa miasta (np. „Katowice”).
   - Typ atrakcji (np. „Viewpoint”).
3. System dynamicznie odpytuje API.
4. Użytkownik otrzymuje listę kafelków ze zdjęciami i cenami spełniającymi kryteria.
5. Kliknięcie na kafelek przekierowuje do szczegółowego widoku atrakcji.



Rysunek 9: Widok główny z listą atrakcji po zastosowaniu filtrów

Specyfikacja wewnętrzna

Architektura systemu

System zrealizowano w architekturze warstwowej (Clean Architecture/Onion Architecture):

1. **Domain:** Zawiera encje (np. `Attraction`, `User`, `Report`) i interfejsy repozytoriów. Jest niezależna od innych warstw.
2. **Application:** Zawiera logikę biznesową, serwisy (`AttractionService`, `PdfService`) oraz obiekty DTO.
3. **Infrastructure:** Implementacja dostępu do danych (EF Core, `TouristInfoDbContext`) oraz serwisów zewnętrznych.
4. **WebAPI:** Kontrolery REST API (`AttractionsController`) udostępniające punkty końcowe dla frontendu.

Analiza implementacji wybranych funkcjonalności

Funkcjonalność 1: Dodawanie nowej atrakcji (Polimorfizm)

Proces tworzenia atrakcji jest obsługiwany przez `AttractionService.CreateAsync`. Ze względu na różne typy atrakcji (Hotel, Szlak, Wydarzenie), system stosuje logikę warunkową do budowania odpowiednich obiektów.

Działanie od wewnątrz:

1. Kontroler odbiera obiekt `CreateAttractionDto`.
2. Serwis najpierw rozwiązuje powiązania geograficzne (Kraj -> Region -> Miasto) metodą `GetOrCreateLocationChainAsync`, aby uniknąć duplikacji w bazie.
3. Następnie, na podstawie pola `TypeName`, tworzony jest obiekt bazowy `Attraction`.
4. System dołącza szczegółowe dane w zależności od typu (Pattern Factory/Builder wewnątrz switcha):
   - Dla **Event**: dodawane są daty rozpoczęcia/zakończenia.
   - Dla **Trail**: dodawany jest dystans i poziom trudności.
   - Dla **Hotel**: dodawana jest lista udogodnień (zapisywana jako JSON).
5. Całość jest zapisywana w jednej transakcji do bazy danych.

Fragment kodu odpowiedzialny za logikę typów (`AttractionService.cs`):

```
switch (attractionType.TypeName)
{
    case "Event":
        attraction.EventDetails = new Event { ... };
        break;
    case "Trail":
        attraction.TrailDetails = new Trail { ... };
        break;
    // ...
}
```

Podsumowanie i wnioski

Realizując projekt „Honalolo” zrealizowaliśmy założone przez nas cele w zakresie stworzenia webowego systemu informacji turystycznej. Zastosowanie architektury warstwowej w środowisku .NET pozwoliło na łatwą rozbudowę o nowe typy atrakcji bez konieczności przebudowy całego systemu. Wykorzystanie biblioteki QuestPDF umożliwiło spełnienie wymogu generowania dynamicznych raportów po stronie serwera.

Aplikacja jest gotowa do dalszego rozwoju, np. o moduł płatności za rezerwacje czy integrację z mapami Google.