



**Wyższa Szkoła
Ekonomii i Informatyki
w Krakowie**

Kierunek: Informatyka Stosowana

Ruslan Pidhainyi

13787

**Project i implementacja aplikacji webowej pod tytułem
„Travel App”**

Praca inżynierska
napisana pod kierunkiem
dr. inż. Zbigniewa Handzla, prof. WSEI

Kraków 2025

Streszczenie pracy

Celem niniejszej pracy jest przedstawienie projektu inżynierskiego - aplikacji webowej pod tytułem „Travel App” oraz omówienie najważniejszych zagadnień teoretycznych, dotyczących technologii i narzędzi wykorzystanych do jej stworzenia.

Omawiany w pracy projekt został zrealizowany przy użyciu sprawdzonych technologii do projektowania aplikacji, z wykorzystaniem współczesnego podejścia oraz jasno określonymi przypadkami zastosowania, które zapewniły zgodność z wymaganiami użytkowników i efektywność realizacji zaplanowanych celów.

Słowa kluczowe: *TravelApp, Media społecznościowe, ASP.Net web API, Angular, C#, TypeScript, Microsoft SQL Server, SQLite, RESTful API*

Abstract

The purpose of this thesis is to present an engineering project - a web application entitled 'Travel App' - and to discuss the key theoretical issues regarding the technologies and tools used to create it.

The project discussed in the thesis was carried out using proven technologies for application design, with a contemporary approach and clearly defined use cases that ensured compliance with user requirements and efficiency in achieving planned goals.

Keywords: *TravelApp, Social media, ASP.Net web API, Angular, C#, TypeScript, Microsoft SQL Server, SQLite, RESTful API*

Spis Treści

Streszczenie pracy	2
Abstract	2
Wstęp	5
1. Analiza założeń i wymagań projektowych	7
1.1. Wprowadzenie do problematyki mediów społecznościowych.....	7
1.2. Cel projektu	8
1.3. Wymagania funkcjonalne.....	8
1.4. Wymagania pozafunkcjonalne.....	12
1.5. Wizja realizacji projektu	12
1.5.1. Makiety	12
1.6. Struktura baz danych	13
1.6.1. Diagram przypadków użycia / Use Case Diagram	13
1.6.2. Diagram Klas / Class Diagram	15
1.6.3. Diagram relacji encji / Entity–Relationship Diagram.....	16
1.7. Technologie i narzędzia	17
2. Analiza technologii i narzędzi wykorzystanych w projekcie.....	20
2.1. Opis Technologii:.....	20
2.1.1. Framework Angular.....	20
2.1.2. Język programowania TypeScript.....	22
2.1.3. Język znaczników HTML.....	24
2.1.4. Język arkusza stylów Sass (SCSS).....	25
2.1.5. Framework ASP.NET Core.....	27
2.1.6. Język programowania C#	29
2.2. System Kontroli Wersji	31
2.2.1. Git.....	31
2.2.2. GitHub	31
2.3. Bazy danych	33
2.3.1. SQLite	33
2.3.2. Microsoft SQL Server	33

2.4.	Bezpieczeństwo	35
2.5.	Testowanie aplikacji.....	36
2.5.1.	Testy jednostkowe i integracyjne	36
2.5.2.	Postman.....	36
2.5.3.	Development Tool.....	37
3.	Dokumentacja procesu i implementacji.....	38
3.1.	Opis procesu realizacji projektu	38
3.2.	Problemy napotkane w trakcie realizacji projektu.....	41
3.3.	Przedstawienie gotowej aplikacji	42
3.4.	Opis testów skalowalności oraz wydajności zrealizowanego projektu.....	43
	Zakończenie.....	47
	Bibliografia.....	48
	Spis ilustracji.....	49
	Spis tabel.....	51
	Aneks.....	52
	Instrukcja instalacji aplikacji lokalnie	52
	Screeny z web-aplikacji.....	55

Wstęp

We współczesnym świecie większość ludzi lubi podróżować po różnych zakątkach świata, a także korzystać z mediów społecznościowych, dzielić się postami i zbierać polubienia. Niektórzy poszukują mniej znanych miejsc oraz zapierających dech w piersiach widoków przyrody.

Niestety, w dzisiejszych czasach brakuje aplikacji, która koncentruje się na problemach osób lubiących zajmować się turystyką skupiającej się na zwiedzaniu dzikiej przyrody. Najczęściej występujący problem - brak dostarczania ważnych informacji, zapewniających turystom krótkie i jasne dane o konkretnych miejscach.

Ze względu na wymienione argumenty oraz przeanalizowanie aplikacji, które są dostępne w danej chwili, utworzyła się idea o stworzeniu aplikacji turystycznej. Głównym celem było zaprojektowanie w nowoczesnych technologiach aplikacji webowej dla turystów zachwycających się turystyką przygodową. Aplikacja umożliwi użytkownikom udostępnianie postów z wyświetleniem jasne informacji dla innych użytkowników. Każdy użytkownik może polubić wybrane posty oprócz własnych publikacji.

Celem pracy jest przedstawienie projektu wspomnianej aplikacji oraz omówienie najważniejszych technologii i narzędzi wykorzystanych do jej stworzenia oraz pokazanie procesu implementacji i wdrożenia projektu.

Omawiany w pracy projekt został zrealizowany przy użyciu sprawdzonych technologii do projektowania aplikacji, z wykorzystaniem współczesnego podejścia oraz jasno określonymi przypadkami zastosowania, które zapewniły zgodność z wymaganiami użytkowników i efektywność realizacji zaplanowanych celów. W ramach implementacji projektu Travel App, wykorzystałem przedstawione poniżej technologie:

- W trakcie rozwoju projektu wykorzystałem relacyjną bazę danych – SQLite.
- Po zakończeniu realizowania projektu przenieśli projekt na inną bazę danych - Microsoft SQL Server.
- W back-endowej części aplikacji wykorzystałem frameworki: .Net Core, ASP.Net Core web API.
- Jako język programowania wykorzystałem C#.
- Po stronie frontendowej zastosowałem Node.js oraz framework Angular.
- Wykorzystałem TypeScript - rozszerzenie języka programowania JavaScript.

Praca składa się z trzech rozdziałów. W pierwszym zostały omówione cele i założenia projektowe, wskazano wymagania funkcjonalne i pozafunkcjonalne, dokonano analizy

istniejących rozwiązań. Ponadto pokazano wizję realizacji projektu. W rozdziale drugim omówiono najważniejsze technologie i narzędzia wybrane do realizacji przedstawionego w pracy projektu inżynierskiego. W ostatnim trzecim rozdziale pokazany został proces realizacji projektu, aż do jego wdrożenia.

1. Analiza założeń i wymagań projektowych

1.1. Wprowadzenie do problematyki mediów społecznościowych

W dzisiejszych czasach większość ludzi uwielbia podróżowanie, ale niestety często spotyka się z różnego rodzaju problemami.

Czynnikami powodującymi obawy ludzi przed danego rodzaju podróżowaniem są:

- Po pierwsze, obawy o swoje zdrowie i zdrowie swoich bliskich.
- Po drugie, konflikty zbrojne między państwami wraz z konfliktami wewnętrznymi - powodują brak poczucia bezpieczeństwa wśród turystów.
- Po trzecie, często problem w planowaniu podróży jest braku aktualnych informacji.

Najczęstszym problemem jest trudność w podjęciu decyzji przez osobę planującą wyjazd co do oczekiwań związanych z podróżą. Powoduje to stratę czasu na przeglądanie zbyt dużej liczby ofert i analizowanie aktualnych informacji, tj.:

- Organizacja pobytu: dostępne pola biwakowe oraz kempingi umożliwiające nocleg, cena za dobę oraz dostępne usługi fakultatywne.
- Aktualny cennik za usługi turystyczne.
- Dostęp oraz lokalizacja sklepów spożywczych.
- Brak informacji o wyjątkowych zakątkach, w których można podziwiać piękno natury.

Poprzez wyżej wymienione argumenty, większość osób nie potrafi się zdecydować na konkretną ofertę i właśnie dlatego powstała idea o stworzeniu dedykowanego portalu społecznościowego. Dodatkowo warto zauważyć, iż nie istnieje żadna spełniająca takie wymagania aplikacja webowa, umożliwiającą dzielenie się własnym doświadczeniem, unikalnymi miejscami oraz aktualnymi informacjami o podróżach z innymi użytkownikami.

Latem ubiegłego roku zrealizowałem kurs, który dotyczył jednej z technologii, jaką wykorzystałem w moim projekcie. W trakcie uczenia się tej technologii wpadłem na pomysł o stworzeniu mediów społecznościowych na temat turystyki.

Z mojego doświadczenia podróżnika, wynika, że często trudno zaplanować ciekawy plan podróży. Po pierwsze, miałem problem w znalezieniu miejsca, w którym można zorganizować pobyt. Po drugie, spotykałem się z brakiem aktualnych informacji o tym, co znajduje się w wybranym miejscu, co powodowało problem z planowaniem spędzenia czasu. Po trzecie, w danym rodzaju turystyki ciężko znaleźć informacje o dostępności sklepów spożywczych oraz punktów informacji dla turysty znajdujących się na wybranym obszarze.

Moim zdaniem aplikacja, którą stworzyłem, będzie pomocna miłośnikom podróży w wygodnym znajdowaniu nowych miejsc, które zaspokajają podstawowe potrzeby, ale także miejsc, które oferują niezapomniane widoki i unikalne doświadczenia, zapierające dech w piersiach.

1.2. Cel projektu

Stworzenie aplikacji noszącej nazwę Travel App.

Celem tego projektu jest ułatwienie znalezienia atrakcyjnych miejsc do podróży, jasnych informacji o danym miejscu oraz możliwość do dzielenia się własnym doświadczeniem wraz z udostępnianiem zdjęć z miejsc, w których użytkownicy spędzali czas. Posty udostępnione na platformie wspomagają innym użytkownikom w poznawaniu nowych lokalizacji.

Umieszczone w aplikacji posty, będą zaprojektowane tak, aby wyrazić jasną i krótką informację, pomocną przy planowaniu podróży. Podczas tworzenia postu, użytkownik powinien określić lokalizację danego miejsca, zaznaczyć, czy korzystał z dodatkowych usług, na przykład - dojazd do wyznaczonego miejsca, wraz z podaną ceną za osobę oraz ile czasu na to poświęcił. Również, użytkownik ma formularz, który pozwala na szczegółowy opis miejsca wraz z różnego rodzaju uwagami podróżującego. Szczegółowy opis doświadczenia podróżującego pozwala na otrzymaniu ekskluzywnych informacji niedostępnych na innych platformach internetowych.

Travel App, pozwala na komunikację w aplikacji pomiędzy użytkownikami za pomocą bezpośrednich wiadomości, co to daje możliwość na uzyskanie dokładnej informacji przydatnej podczas podróży oraz stwarza wspierającą społeczność dla miłośników turystyki przygodowej.

1.3. Wymagania funkcjonalne

Funkcjonalność portali społecznościowych dla podróżujących:

- **Rejestracja** – Użytkownik może założyć konto w tylko w przypadku, gdy nazwa użytkownika jest unikalna, czyli nie istnieje w bazie danych. Posiadanie konta odblokowuje szereg funkcjonalności: możliwość udostępniania postów, wysyłanie prywatnych wiadomości do uczestników sieci społecznościowej itd. W celu założenia konta w aplikacji użytkownik powinien udać się do strony „Register”, gdzie użytkownik wypełnia formularz z następującymi polami:
 - „Gender”
 - „Username”
 - „Known As”

- „Date of birth”
- „City”
- „Country”
- „Password”
- „Confirm Password”

Po wypełnieniu formularza użytkownik klika przycisk „Register”, znajdujący się w lewym dolnym rogu strony. Po stworzeniu profilu system przekierowuje użytkownika na główną stronę, która odpowiada za wyświetlenie listy postów.

W przypadku, gdy formularz zostanie wypełniony nieprawidłowo, po kliknięciu na przycisk „Register” użytkownik otrzymuje stosowny komunikat.

- **Logowanie** – Użytkownik za pomocą wcześniej stworzonego konta posiada możliwość zalogowania się. Może to zrobić poprzez formularz na specjalnej stronie „Login”, bądź poprzez formularz w górnym navbarze. Po zalogowaniu aplikacja przekieruje użytkownika na główną stronę „Offers” i wyświetli komunikat „User logged in successfully”.

W formularzu logowania, w polu „Password” znajduje się ikonka „Oka” które pozwala na przeglądanie wpisanego hasła.

Jeżeli użytkownik poda błędną nazwę użytkownika lub nieprawidłowe hasło i kliknie przycisk „Login”, otrzyma komunikat o treści „Failed to login”.

- **Edytowanie profilu** – Właściciel profilu posiada możliwość edytowania danych swojego profilu. Edycja profilu obejmuje edycję miejsca urodzenia, zakładkę zainteresowań, dodawanie lub usuwanie zdjęcia profilowego.

Po zalogowaniu się do aplikacji użytkownik w celu edycji swojego profilu powinien przejść na własny profil, gdzie w lewym dolnym rogu w panelu pod opisem znajduje się przycisk „Edit profile”. Kliknięcie tego przycisku przekierowuje użytkownika na stronę „Edit”, gdzie może dodawać zdjęcia za pomocą przycisku „Add Photos”, oznaczyć jedno z nich jako główne, co spowoduje ustawienie tego zdjęcia jako awatar profilu, a także użytkownik może usuwać wybrane przez siebie zdjęcia.

Podczas edycji profilu użytkownik ma również możliwość edycji dodatkowych formularzy:

- „Description”, będące polem opisu swojego profilu
- „Interests”, przedstawiające pasje i zainteresowania użytkownika
- „Location details”: „City” i „Country” z informacjami o swoim miejscu zamieszkania

Wejście na formularz edycji profilu może nastąpić w dowolnym momencie korzystania z aplikacji. W celu zatwierdzenia zmian należy kliknąć przycisk „Save change”.

- **Tworzenie Postu** – Użytkownik może podzielić się swoim doświadczeniem o danej lokalizacji w formie postu. Post zawiera pola „Place name”, lokalizację („The country of this place” i „The region of this place”, w których odbyła się podróż), „Currency”, czyli walucie używanej w danym kraju oraz zdjęcie z odwiedzonego miejsca. Post może zostać dodany poprzez pasek nawigacyjny w zakładce „Share post”, bądź w zakładce swojego profilu, poprzez przycisk z ikoną „+” pod pytaniem „Want to share your journey?”. Kliknięcie tego przycisku przeniesie użytkownika na stronę „Add post”.

Post musi posiadać minimalnie jedno zdjęcie z miejsca podróży oraz musi posiadać wypełnione wszystkie wymagane pola:

- „Did you use local transport to get to your trip?”
- „Entrance was paid?”
- „Accommodation?”
- „Grocery stores nearby?”
- „Guide services?”

Podczas tworzenia postu użytkownik również powinien udzielić odpowiedzi na następujące pytania poprzez zaznaczenie checkboxów: „Low price”, „High price”, „Approximately, the journey took”, „Choose type of accommodation”. Dodatkowo twórca postu może dodać opis, który pomoże innym użytkownikom uzyskać więcej informacji na temat tej podróży. W celu publikacji postu należy kliknąć przycisk „Share post”, znajdujący się na dole, po lewej stronie. Po stworzeniu publikacji system przekieruje użytkownika na stronę profilu oraz wyświetli komunikat o udanym utworzeniu postu.

Formularz również posiada walidację, w przypadku pominięcia danego pola wyskoczy stosowny komunikat.

- **Edycja postów** – Użytkownik posiada możliwość edycji własnego postu. W celu jego edycji, użytkownik powinien przejść na własny profil i na wybranym poście nacisnąć ikonę edycji. Po naciśnięciu przycisku system przekieruje autora na stronę „Edit post”, gdzie za pomocą pól formularza użytkownik dokonuje edycji.

W celu zatwierdzenia zmian należy kliknąć przycisk „Edit post”, znajdujący się na dole, po lewej stronie. Jeżeli edycja się powiedzie system przekieruje użytkownika na stronę profilu i wyświetli komunikat o treści „Post updated successfully”.

W przypadku pominięcia danego pola pojawią się stosowne komunikaty.

- **Usuwanie postów** – Użytkownik posiada możliwość usuwania własnych postów. W celu usunięcia publikacji należy przejść na zakładkę własnego profilu i kliknąć przycisk

usuwania przy wybranym poście. Po naciśnięciu przycisku system usunie post i wyświetli komunikat o treści „Post deleted successfully”.

- **Polubienie postu** – Każdy użytkownik może polubić post udostępniony przez innych użytkowników, z wyjątkiem, że posty nie mogą zostać polubione przez ich autorów.

Post można polubić poprzez kliknięcie ikony serca znajdującej się przy nim. Można tego dokonać w zakładce „Offers”, w szczegółach danego postu bądź na profilu danego użytkownika, gdzie wyświetla się lista jego postów.

Polubienie danego postu powoduje zapisanie go w zakładce „Lists”, będącą archiwum polubionych przez użytkownika publikacji.

- **Wysyłanie wiadomości do innych użytkowników w czasie rzeczywistym** – Użytkownicy mogą komunikować się między sobą za pomocą prywatnych wiadomości w czasie rzeczywistym w celu uzyskania informacji na temat danej podróży.

W celu rozpoczęcia konwersacji użytkownik przechodzi do danego profilu innego użytkownika. Następnie po lewej stronie znajduje się przycisk „Send a message”. Po jego kliknięciu użytkownik zostaje przekierowany do zakładki „Messages”, gdzie może nawiązać komunikację z wybraną osobą. Istnieje również możliwość rozpoczęcia konwersacji po odwiedzaniu profilu danego członka, gdzie w centralnej części strony znajduje się zakładka „Messages”.

Strona „Messages”, znajdująca się w pasku nawigacyjnym, służy do zarządzania powiadomieniami oraz wiadomościami użytkowników. Po przejściu na tą stronę użytkownik uzyskuje dostęp do trzech zakładek: „Unread”, „Inbox” oraz „Outbox”. W zakładce „Unread” znajdują się wiadomości, które nie zostały jeszcze przeczytane przez użytkownika. Wiadomość tego typu posiada status „unread” zarówno dla odbiorcy na stronie „Messages”, jak i dla nadawcy w czacie, który ją wysłał. Wiadomość również jest wyświetlana jako „unread” podczas przeglądania prywatnej konwersacji między użytkownikami.

W zakładce „Inbox” znajdują się wiadomości przychodzące do użytkownika. Mogą one posiadać jeden z dwóch statusów: „unread” lub „read”. Status ten jest widoczny zarówno dla odbiorcy na stronie „Messages” jak i dla nadawcy, który może sprawdzić status wiadomości na czacie.

W trzeciej zakładce o nazwie „Outbox” znajdują się wiadomości wysłane przez użytkownika do innych członków. Podobnie jak wiadomości przychodzące, wiadomości wysłane mogą posiadać jeden z dwóch statusów: „unread” lub „read”. Status wiadomości

jest widoczny zarówno dla odbiorcy na stronie „Messages”, jak i dla nadawcy, który może sprawdzić jej status na czacie.

- **Rola Administratora** – Użytkownik z rolą administratora posiada dwie dodatkowe funkcje w porównaniu do „zwykłego” użytkownika: dostęp do panelu administratora oraz dostęp do strony „Errors” w pasku nawigacyjnym.

Administrator ma również możliwość zarządzania rolami innych użytkowników. Aby tego dokonać przechodzi do zakładki „Admin”, gdzie znajduje się lista wszystkich użytkowników sieci społecznościowej wraz z możliwością zarządzania rolami każdego z nich.

Dzięki zakładce „Errors” administrator lub programista może monitorować i analizować problemy występujące w systemie, co znacząco usprawnia proces projektowania oraz rozwiązywania błędów.

Dodatkowo administrator posiada możliwość usuwania postów poprzez kliknięcie na przycisk znajdujący się przy poście.

- **Rola Moderadora** – Moderator, podobnie jak administrator posiada możliwość usuwania postów użytkowników poprzez przycisk znajdujący się przy danym poście.

1.4. Wymagania pozafunkcjonalne

1. Aplikacja powinna być wydajna i skalowalna.
2. Aplikacja powinna być kompatybilna z nowoczesnymi przeglądarkami.
3. Aplikacja powinna być dostępna 24 godziny na dobę.
4. Aplikacja powinna posiadać intuicyjny interfejs graficzny.
5. Aplikacja powinna zapewniać bezpieczeństwo danych użytkowników.

1.5. Wizja realizacji projektu

1.5.1. Makiety

Przed przystąpieniem do tworzenia aplikacji zdecydowałem się stworzyć makiety przedstawiające wygląd poszczególnych podstron aplikacji. W celu stworzenia mockupów wykorzystałem narzędzie Figma. Na poniższych rysunkach przedstawiłem część z nich.

Rys. 1 Makiet: Logowania i Rejestrowania użytkownika
źródło: Opracowanie własne

Rys. 2 Makiet: Oferty (Strona główna) i element dropdown
źródło: Opracowanie własne

Pierwsze dwie makiety przedstawiają wygląd strony logowania i rejestracji, kolejne dwie wygląd strony z ofertami podróży oraz wygląd rozsuwalnego menu w prawym, górnym rogu ekranu.

W ostatecznej wersji aplikacji niektóre strony zmieniły swoją ostateczną formę względem początkowych założeń z makiet.

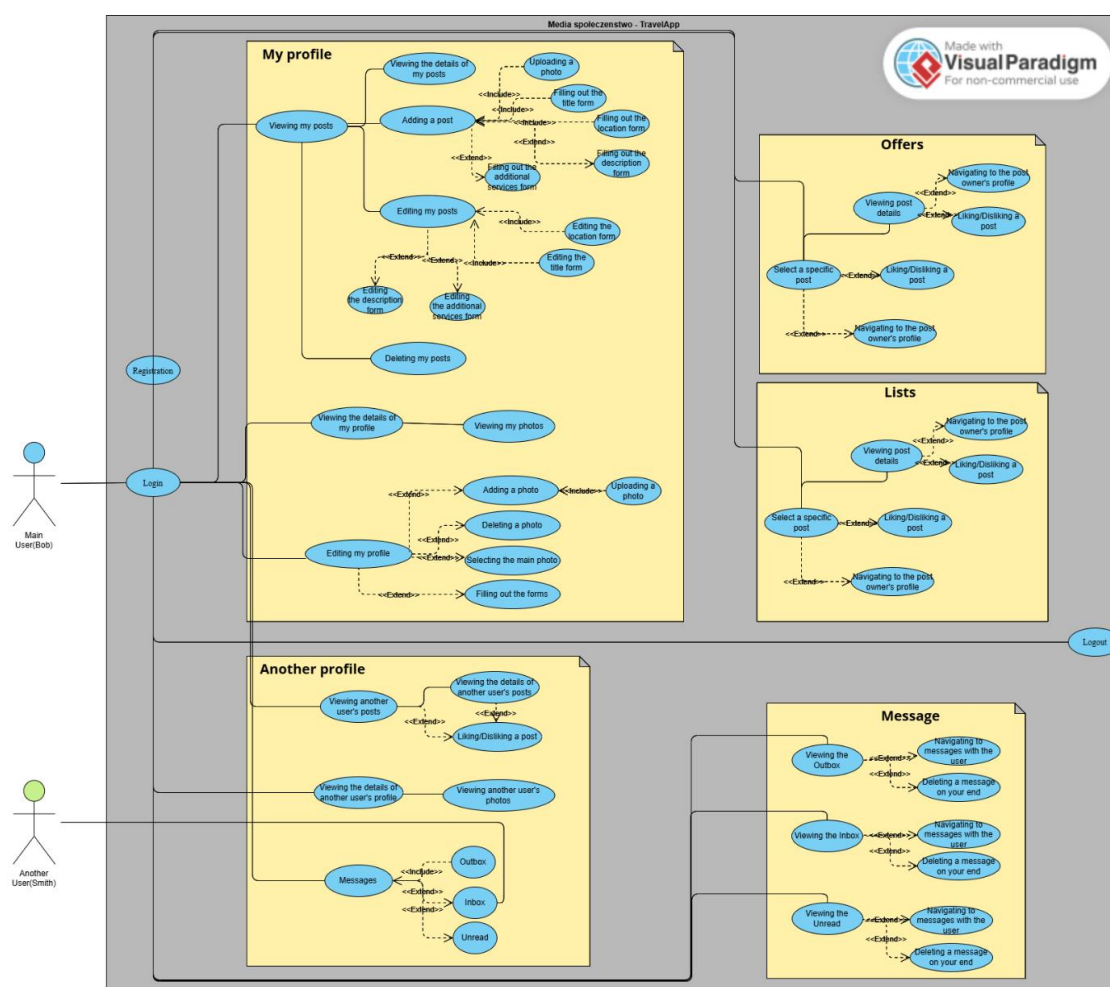
1.6. Struktura baz danych

1.6.1. Diagram przypadków użycia / Use Case Diagram

Diagram przypadków użycia¹ znajdujący się na rysunku poniżej przedstawia wszystkie główne scenariusze interakcji aktorów z aplikacją. Był on pomocny w procesie projektowania, zapewniając wizualne zrozumienie typów ról, funkcjonalności oraz interakcji między aktorami, a przypadkami użycia.

¹ Śmiałek M., Rybiński K., „Inżynieria oprogramowania w praktyce od wymagań do kodu z językiem UML”, Helion, Gliwice 2017, s. 116

Do stworzenia diagramu przypadków użycia wykorzystałem edytor Visual Paradigm Online. W projekcie posłużyłem się językiem UML², który jest standardowym językiem modelowania.



Rys. 3 Diagram przypadków użycia: scenariusz dla użytkownika
źródło: Opracowanie własne

Powyższy diagram przedstawia w jaki sposób użytkownicy mogą korzystać z funkcji mediów społecznościowych. Główny użytkownik o przykładowym imieniu Bob, który już istnieje w systemie posiada możliwość zalogowania się, co odblokowuje wiele dalszych funkcjonalności aplikacji:

- **Zarządzanie swoim profilem**

Bob może przejść na stronę własnego profilu, gdzie ma możliwość przeglądania swoich postów oraz wyświetlania szczegółowych informacji swojego profilu. Może tworzyć nowe posty, edytować istniejące, a także usuwać je, edytować swój profil, dodawać zdjęcia oraz wyznaczyć jedno z dostępnych zdjęć jako awatar.

² Śmiałek M., Rybiński K., „Inżynieria oprogramowania w praktyce od wymagań do kodu z językiem UML”, Helion, Gliwice 2017, s. 77

- **Interakcja z innymi użytkownikami**

Bob posiada możliwość wejścia na stronę innego użytkownika, gdzie może przeglądać jego posty oraz szczegółowe informacje o nich oraz polubić je. Może również zapoznać się z informacjami o danym użytkowniku, znajdującymi się na jego profilu, w tym opisem i zainteresowaniami, a także rozpocząć prywatną konwersację z danym użytkownikiem.

- **Przegląd publikacji w zakładce Offers**

Bob może przeglądać publikacje w zakładce „Offers”, polubić je lub przechodzić z tego miejsca na profile autorów postów.

- **Zarządzanie polubionymi publikacjami w sekcji Lists**

Polubione posty można przeglądać w sekcji Lists. Bob ma możliwość przeglądania zapisanych publikacji, wyświetlania szczegółowych informacji o wybranym poście oraz usuwania go z listy zapisanych.

- **Zarządzanie wiadomościami**

W zakładce wiadomości Bob może przeglądać swoje konwersacje, w tym nieprzeczytane oraz wysłane do innych użytkowników wiadomości. Bob ma także możliwość usunięcia wiadomości, ale tylko tych wysłanych ze swojego konta.

- **Wylogowanie z aplikacji**

Bob posiada możliwość wylogowania się z aplikacji.

1.6.2. Diagram Klas / Class Diagram

Diagram klas jest jednym z najważniejszych elementów przed przystąpieniem do tworzenia projektu. Na diagramie znajdują się zaprojektowane klasy, ich pola, relacje między nimi oraz funkcjonalności, co poprawia zrozumienie projektu jeszcze przed jego realizacją.

1.6.3. Diagram relaciji encji / Entity–Relationship Diagram

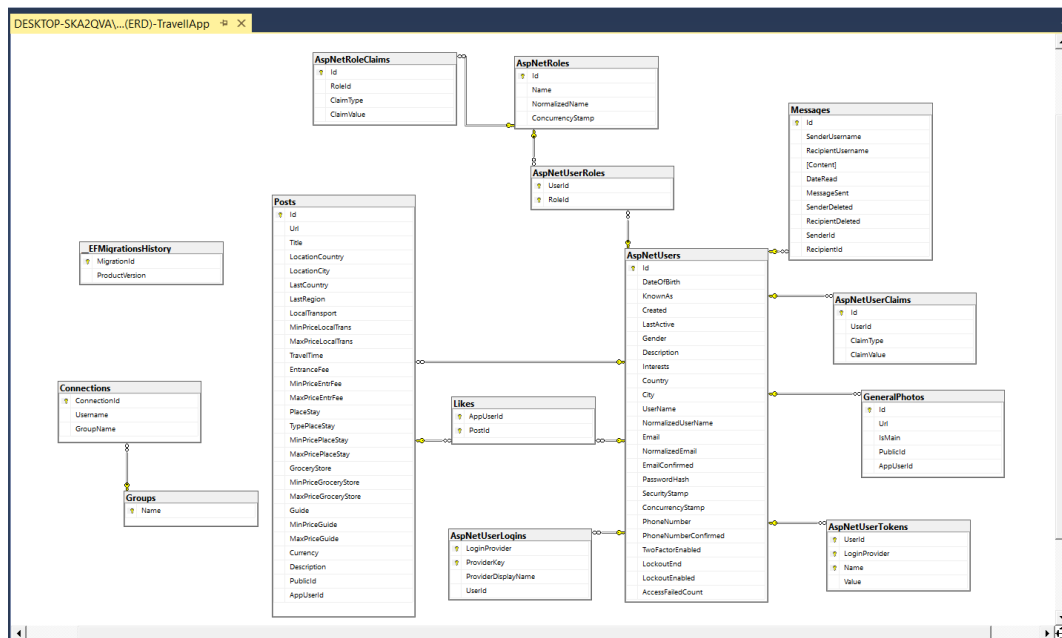
The diagram illustrates the following entities and their attributes:

- AspNetRoleClaims**: Id, RoleId, ClaimType, ClaimValue
- AspNetRoles**: Id, Name, NormalizedName, ConcurrencyStamp
- AspNetUsers**: Id, AccessFailedCount, City, Country, ConcurrencyStamp, Created, DateOfBirth, Email, EmailConfirmed, Gender, Interests, KnownAs, LastActive, LockoutEnabled, LockoutEnd, LocalizedEmail, NormalizedEmail, NormalizedUserName, NormalizedHash, PhoneNumber, ProfileNumberConfirmed, SecurityStamp, TwoFactorEnabled, UserName
- AspNetUserRoles**: UserId, RoleId
- AspNetUserLogins**: LoginProvider, ProviderKey, ProviderDisplayName, UserId
- Posts**: Id, AppUsers, Currency, Description, EntranceFee, GroceryStore, Guide, LastRegion, LastCountry, LocalTransport, LocationCity, LocationCountry, MasPriceEntranceFee, MasPriceGroceryStore, MasPriceGuide, MasPriceLocalTrans, MasPricePlaceStay, MinPriceEntranceFee, MinPriceGroceryStore, MinPriceGuide, MinPriceLocalTrans, MinPricePlaceStay, TravelTime, TypePlaceStay, Uri
- Likes**: AppUserId, PostId
- Messages**: Id, Content, Detailed, MessageSent, RecipientDeleted, RecipientId, RecipientUsername, RecipientDeleted, SenderDeleted, SenderId, SenderUsername
- AspNetUserClaims**: Id, UserId, ClaimType, ClaimValue
- GeneralPhotos**: Id, AppUserId, IsMain, IsPublic, Uri
- AspNetUserToken**: UserId, LoginProvider, Name, Value
- Connections**: ConnectionId, Username, GroupName
- Groups**: Name
- __EFMigrationsHistory**: MigrationId, ProductVersion

Relationships are indicated by lines connecting the entities. For example, **AspNetUsers** is connected to **AspNetRoles** via **AspNetUserRoles**, and **AspNetUsers** is connected to **AspNetUserClaims** via **AspNetUserClaims**. **Posts** is connected to **AspNetUsers** via **AppUsers** and **AppUserId**. **Likes** is connected to **AspNetUsers** via **AppUserId** and to **Posts** via **PostId**. **Messages** is connected to **AspNetUsers** via **RecipientId** and **SenderId**. **AspNetUserToken** is connected to **AspNetUsers** via **UserId**. **Connections** is connected to **Groups** via **GroupName**. **__EFMigrationsHistory** is connected to **MigrationId**.

16

Po zaimplementowaniu wszystkich wymagań funkcjonalnych projektu zmieniłem bazę danych na Microsoft SQL Server. Jest ona bezpieczniejsza w porównaniu do SQL. Do wygenerowania diagramu relacji encji wykorzystałem zintegrowane środowisko programistyczne Microsoft SQL Server Management Studio.



Rys. 6 Diagram relacji encji – MSSQL
źródło: Opracowanie własne

1.7. Technologie i narzędzia

Do realizacji projektu wykorzystano następujące technologie:

Frontend:

Framework:

- Angular - v. 17.3.10 (v. 17)

Środowisko wspierające:

- Środowisko uruchomienia: Node - v. 20.10.0 (v. 20)
- Package Manager: NPM – 10.8.3 (v.10)

Biblioteka:

- font-awesome
- mkcert
- ngx-bootstrap --component dropdowns
- ngx-toastr
- ng-gallery
- ngx-spinner

- ng2-file-upload
- ngx-timeago

Programming language, biblioteka oraz style:

- JavaScript (TypeScript)
- CSS (SCSS)
- HTML

Backend:

Framework:

- .Net Core – v 8.0.0
- ASP .Net Core Web API – v 8.0.0

Listę pakietów NuGet:

- SignalR – v 8.0.0
- AutoMapper – v 13.0.1
- CloudinaryDotNet – v 1.26.2
- Microsoft.AspNetCore.Authentication.JwtBearer – v 8.0.8
- Microsoft.AspNetCore.Identity.EntityFrameworkCore – v 8.0.8
- Microsoft.EntityFrameworkCore.Design – v 8.0.8
- Microsoft.EntityFrameworkCore.Sqlite – v 8.0.8
- Microsoft.EntityFrameworkCore.SqlServer – v 8.0.8
- Swashbuckle.AspNetCore – v 6.4.0
- System.IdentityModel.Tokens.Jwt – v 8.1.1

Programming language:

- C#

Architektura projektu i podejścia projektowe:

- Monolit – jedna aplikacja obsługująca frontend i backend.
- Single Page Application (SPA) – frontend w Angular.
- Clean Architecture – modularne podejście z warstwami.
- RESTful API – backend w ASP.NET Core Web API.
- SignalR – wykorzysta computer communications protocol WebSocket.
Komunikacja między Serverem a Clientem.

Database:

Database (SQL):

- SQLite (SQL)

- Microsoft SQL Server (SQL)

Protocol:

- HTTPS

Hostowanie:

- Cluodinary

Systemy kontroli wersji i hosting:

- Git
- GitHub

Narzędzia do komunikacji i testowania API:

- Postman

Integrated Development Environment (IDE):

- Visual Studio
- Visual Studio Code
- DBeaver
- Microsoft SQL Server Management Studio

Browsers:

- Google Chrome
- Microsoft Edge

Narzędzia do design:

- Figma

Narzędzia DevOps:

- Jira

Web-Aplikacji (Pomocne narzędzia do programowania):

- json2csharp - <https://json2csharp.com/>
- jwt.io - <https://jwt.io/>
- json-generator - <https://json-generator.com/>
- transform.tools - <https://transform.tools/json-to-typescript>
- fontawesome.com - <https://fontawesome.com/v4/>
- VisualParadigmOnline-<https://online.visual-paradigm.com/diagrams/features/uml-tool/>
- Pexels - <https://www.pexels.com/>

2. Analiza technologii i narzędzi wykorzystanych w projekcie

2.1. Opis Technologii:

2.1.1. Framework Angular

Angular³ to framework frontendowy do tworzenia aplikacji SPA (ang. Single Page Application). Umożliwia on automatyczne renderowanie treści na stronie bez konieczności odświeżania jej. Do jego poprawnego funkcjonowania wymagane jest środowisko uruchomieniowe Node.js⁴, działającego na silniku V8. Jest ono używane zarówno do tworzenia części backendowych, jak i frontendowych aplikacji w technologiach, takich jak Express, React, Sass, Less czy też omawiany Angular. Oferuje ono zestaw narzędzi npm, który został wykorzystany do instalacji niezbędnych, zewnętrznych bibliotek oraz w zarządzaniu zależnościami.

Zalety i wady frameworka Angular

Zalety:

- Podział struktury pojedynczego komponentu na część widoku, logiki i stylów (HTML, CSS, TypeScript). Dzięki temu podziałowi prościej utrzymać porządek w kodzie oraz zapewnić szybkość poruszania się po nim.
- Duża liczba zewnętrznych bibliotek, dzięki dużej społeczności tej technologii
- Obsługa poleceń w CLI⁵ (ang. Command Line Interface), która zapewnia wygodę podczas programowania.
- Zerowy koszt korzystania z technologii

Wady:

- Stroma krzywa uczenia się

Porównanie Angular z React

Alternatywną technologią do Angulara jest biblioteka React. W tabeli nr 1 przedstawiono porównanie obu technologii.

³ <https://v18.angular.dev/overview> na dzień 31 stycznia 2025 roku

⁴ <https://nodejs.org/docs/latest/api/documentation.html> na dzień 31 stycznia 2025 roku

⁵ <https://v18.angular.dev/tools/cli> na dzień 31 stycznia 2025 roku

	Angular	React
Rodzaj technologii	Framework	Biblioteka
Krzywa uczenia się	Stroma	Łagodna
Komponenty i ich formaty	Podział na trzy części: widok, logikę oraz style (HTML, TypeScript, CSS)	Widok i logika są umieszczone w jednym pliku JSX ⁶ , bądź TSX (w zależności od rodzaju używanej technologii JavaScript, bądź TypeScript). Style (CSS) są umieszczone w drugim pliku.
Funkcjonalność i Biblioteki	Większa liczba wbudowanych rozwiązań.	Mniejsza liczba oferowanych, wbudowanych funkcjonalności, co prowadzi do konieczności instalacji zewnętrznych bibliotek. Przykładami funkcjonalności, które są wbudowane w Angularze, a które należy zainstalować z zewnętrznych bibliotek w React'cie są routing i stan.

*Tabela 1. Porównanie Angular'a i React'a
źródło: Opracowanie własne*

Na podstawie powyższej przedstawionych zalet zdecydowałem się na wykorzystanie technologii Angular do stworzenia części frontendowej mojej aplikacji.

Przykładowy komponent

Na poniższym rysunku znajduje się zrzut ekranu komponentu odpowiedzialnego za wyświetlanie wszystkich postów w mojej aplikacji.

⁶ Morgan J., "How To Code in React.js", DigitalOcean, New York 2021, s 49



Rys. 7 Przykładowy rzut ekranu Komponentu TypeScript - Ten Komponent odp. za wyświetlenia publikacji
 źródło: Opracowanie własne

Komponent⁷ jest elementem, oferującym tworzenie widoku w aplikacji oraz wielokrotne używanie go w wielu miejscach w aplikacji. Jak wcześniej wspomniałem, w Angularze jest on podzielony na części widoku HTML, logiki TypeScript oraz styli CSS. Użycie HTML'a oraz TypeScripta pozwala na definiowanie wyświetlanych w aplikacji treści. Automatycznie przekazywane wartości w części TypeScriptowej za pomocą strumieni są natychmiastowo aktualizowane na stronie, bez konieczności odświeżania jej. Część CSS z kolei służy do definiowania wyglądu aplikacji.

2.1.2. Język programowania TypeScript

TypeScript⁸ to język programowania rozszerzający możliwości języka JavaScript. Jest to typowany język programowania wysokiego poziomu o otwartym kodzie źródłowym. W odróżnieniu od JavaScript oferuje typowanie statyczne, tworzenie typów złożonych, czy chociażby modyfikatory dostępu do klas.

Zalety i wady TypeScript

Zalety:

- Typowanie statyczne

⁷ <https://v18.angular.dev/guide/components>, <https://v18.angular.dev/essentials/components> na dzień 31 stycznia 2025 roku

⁸ Cherny B., „Programming TypeScript Making Your JavaScript Applications Scale”, O'Reilly, Sebastopol 2019, s 6

- Zgłaszanie błędów przez kompilator jeszcze przed kompilacją kodu
- Przez fakt, iż TypeScript jest rozszerzeniem języka JavaScript, kod z JavaScript może zostać użyty w TypeScript
- Duża społeczność, dzięki czemu istnieje duża liczba materiałów pomocniczych do tego języka

Wady:

- Krzywa uczenia się jest bardziej stroma w stosunku do JavaScript

Porównanie TypeScript z JavaScript⁹

	TypeScript	JavaScript
Typowani	Statyczne	Dynamiczne
Wykrywanie błędów	Ze względu na typowanie statyczne kompilator zgłasza błędy przed uruchomieniem	Ze względu na typowanie dynamiczne kompilator informuje o błędach po uruchomieniu
Ekosystem bibliotek	Szeroki zakres wsparcia dla różnych bibliotek	Szeroki zakres wsparcia dla różnych bibliotek
Krzywa uczenia się	Stroma	Łagodna
Interakcja z Angulariem	TypeScript jest oficjalnym językiem programowania w Angularze	Istnieje możliwość korzystania z języka JavaScript w Angularze poprzez nadanie wartości <code>true</code> właściwości <code>allowJs</code> w pliku konfiguracyjnym <code>tsconfig.json</code> .

*Tabela 2 Porównanie TypeScript'a i JavaScript'a
źródło: Opracowanie własne*

Ze względu na wyżej wymienione zalety zdecydowałem się na wykorzystanie języka TypeScript w mojej aplikacji, jednak głównym czynnikiem był fakt, iż jest to oficjalny język programowania frameworka Angular.

⁹ Cherny B., „Programming TypeScript Making Your JavaScript Applications Scale”, O'Reilly, Sebastopol 2019, s 8-10

Przykładowy interfejs w języku programowania TypeScript

Na poniższym rysunku znajduje się zrzut ekranu interfejsu postu w mojej aplikacji. Przedstawiony kod opisuje strukturę postu, zawierającego właściwość Id o typie numer oraz właściwości: Url, Title, LocationCountry, LocationCity, LastCountry i LastRegion o typie string.

A screenshot of a code editor window with a dark background and light-colored text. The code defines a TypeScript interface named 'Post'. The interface has eight properties: 'id' of type 'number', and 'url', 'title', 'locationCountry', 'locationCity', 'lastCountry', and 'lastRegion' all of type 'string'. The code is as follows:

```
1 export interface Post {  
2   id: number;  
3   url: string;  
4   title: string;  
5   locationCountry: string;  
6   locationCity: string;  
7   lastCountry: string;  
8   lastRegion: string;  
}
```

Rys. 8 Przykładowy rzut ekranu języka programowania TypeScript - Ten kod odp. za definicję interfejsu Post'a
źródło: Opracowanie własne

Interfejs¹⁰ jest elementem języka TypeScript, służy do definiowania zestawu właściwości i metod, które mają zostać użyte w danej klasie. Implementacja interfejsu wymaga użycia pól i metod, które się w nim znajdują.

2.1.3. Język znaczników HTML

HTML¹¹ to język znaczników, służący do tworzenia szkieletu stron internetowych. Dzięki użyciu DOM istnieje możliwość manipulowania poszczególnym elementem strony za pomocą języka programowania JavaScript. W celu zdefiniowania wyglądu strony wykorzystywany jest język znaczników CSS.

DOM¹² (ang. Document Object Model) jest drzewiastą strukturą strony HTML. Dzięki tej hierarchicznej strukturze istnieje możliwość manipulowania elementami.

Zalety i wady HTML

Zalety:

- Łagodna krzywa uczenia się

¹⁰ Cherny B., „Programming TypeScript Making Your JavaScript Applications Scale”, O'Reilly, Sebastopol 2019, s 91-95

¹¹ Althoff C., „Programista samouk, Profesjonalny przewodnik do samodzielnej nauki kodowania”, Helion, Gliwice 2017, s 200 - 201

¹² Harris A., „HTML5 and CSS3 All-in-One For Dummies”, John Wiley & Sons, New Jersey 2014, s 375

- Duża popularność. Dzięki dużej popularności tego języka istnieje dużo materiałów pomocniczych i szczegółowych dokumentacji.
- HTML jest obsługiwany przez większość nowoczesnych przeglądarek.

Wady:

- Brak możliwości definiowania wyglądu strony, ani tworzenia złożonych funkcjonalności. Język ten skupia się jedynie na tworzeniu szkieletu strony.
- Brak mechanizmów zabezpieczających stronę

Warto również wspomnieć o idei semantyki. HTML5 oferuje korzystanie ze znaczników semantycznych, które odzwierciedlają ich przeznaczenie w kodzie. Przykładowo znacznik `<nav>` odzwierciedla miejsce na stronie, przeznaczone do umieszczenia nawigacji. Innymi aspektami, które warto przytoczyć jest rozbudowanie języka HTML w Angularze o używanie dyrektyw w znacznikach, dzięki którym HTML może przekazywać, bądź odbierać dane z części odpowiedzialnej za logikę. Z kolei wyświetlanie elementów zdefiniowanych w języku TypeScript następuje w klamrach `{{ }}`.

HTML stał się standardem w tworzeniu stron internetowych, a użycie innych tego typu języków do tworzenia szkieletu strony jest praktycznie niespotykane. Z tego powodu nie myślałem nad użyciem alternatywnego języka w mojej aplikacji.

2.1.4. Język arkusza stylów Sass (SCSS)

Sass (Scss)¹³ to język skryptowy preprocesora, który rozszerza możliwości języka CSS (Cascading Style Sheet). Oferuje on przechowywanie korzystanie w kodzie ze zmiennych, zagnieżdżanie reguł, tworzenie metod, operatorów warunkowych oraz pętli.

Typy składni Sass¹⁴:

- **Sass** – w porównaniu do CSS, w SaSS kod pisany jest bez użycia nawiasów klamrowych. Zamiast nich stosowane są spacje i znaki nowej linii.
- **SCSS** – w SCSS kod jest pisany przy użyciu nawiasów klamrowych, jak w CSS, a także przy użyciu zagnieżdżonych selektorów wewnątrz siebie.

¹³ <https://sass-lang.com/documentation/> na dzień 31 stycznia 2025 roku

¹⁴ <https://sass-lang.com/documentation/style-rules/> na dzień 31 stycznia 2025 roku

SASS	SCSS	CSS
<pre> nav ul margin: 0 padding: 2px list-style: square li display: inline-block a display: block padding: 5px 10px text-decoration: underline dotted green </pre>	<pre> nav { ul { margin: 0; padding: 2px; list-style: square; } li { display: inline-block; } a { display: block; padding: 5px 10px; text-decoration: underline dotted green; } } </pre>	<pre> nav ul { margin: 0; padding: 2px; list-style: square; } nav li { display: inline-block; } nav a { display: block; padding: 5px 10px; text-decoration: underline dotted green; } </pre>

Rys. 9 Przykładowy rzut ekranu porównania składni SASS, SCSS, CSS
źródło: <https://highload.tech/scss/>

Zalety i wady Sass (SCSS)

Zalety:

- Możliwość przechowywania wartości za pomocą zmiennych
- Możliwość korzystania z operatorów warunkowych oraz pętli
- Poprawiona czytelność kodu dzięki zagnieżdżeniom

Wady:

- Dodatkowa kompilacja. Kod SCSS jest następnie kompilowany do CSS.
- Konieczność instalacji zewnętrznych zależności

Porównania CSS, SASS i SCSS

Alternatywnym językiem dla Sass jest CSS. W tej części porównam podstawową wersję preprocesora Sass, zmodernizowaną wersję - SCSS oraz język arkuszy stylów CSS.

	CSS	SASS	SCSS
Format pliku	.css	.sass	.scss
Struktura składni	Reguły są pisane w klamrowych nawiasach „{ }”	Korzysta ze znaków spacji i symboli nowej linii. Wprowadza dodatkowe możliwości (pętle, warunki, zmienne itd.)	Reguły pisane w klamrowych nawiasach mogą być zagnieżdżone. Wprowadza dodatkowe możliwości (pętle, warunki, zmienne itd.)
Kompilacja	Brak dodatkowej kompilacji	Potrzebuje dodatkowej kompilacji do CSS	Potrzebuje dodatkowej kompilacji do CSS

*Tabela 3 Porównanie CSS'a, SASS'a i SCSS'a
źródło: Opracowanie własne*

Ze względu na wymienione wyżej zalety zdecydowałem się użyć języka arkuszy stylów SCSS w mojej aplikacji. Jest on również kompatybilny z popularnymi frameworkami, w tym z Angularem.

2.1.5. Framework ASP.NET Core

ASP.NET Core¹⁵ to framework backendowy typu open source, co oznacza, że można z niego korzystać bezpłatnie. Jest to technologia wieloplatformowa, obsługująca różne systemy operacyjne: Windows, Linux i macOS.

Technologia ta służy do implementacji logiki po stronie serwera, umożliwiając tworzenie punktów końcowych REST API oraz implementacje funkcjonalności pod daną logikę biznesową, jak przykładowo uwierzytelnianie, autoryzację, czy przetwarzanie danych. ASP.NET Core łączy się z bazą danych przy pomocy mapera ORM Entity Framework.

Zalety i wady ASP.NET Core

Zalety:

- Możliwość instalacji bogatej oferty dodatkowych pakietów przy pomocy menedżera pakietów NuGet.
- ASP.NET Core opiera się na licencji open source, co oznacza, że korzystanie z tej technologii jest bezpłatne.

¹⁵ Hans-Petter Halvorsen, „Web Programming ASP.NET Core”, 2021, s 17

- Duża popularność ASP.NET sprawia, że istnieje duża liczba materiałów pomocniczych i szczegółowych dokumentacji
- Platforma ta jest międzyplatformowa, można z niej korzystać na różnych systemach operacyjnych, jak Windows, Linux i macOS, nie ogranicza się tylko do jednego systemu.
- Obsługa popularnych zintegrowanych środowisk programistycznych IDE, jak: Visual Studio, Visual Studio Code, Atom, Emacs czy JetBrains.

Wady:

- Stroma krzywa uczenia się
- Konieczność umiejętności posługiwania się językiem C#

Porównanie ASP.NET ze Spring

Alternatywną technologią tworzenia serwera aplikacji jest framework Spring.

	ASP .Net Core	Spring ¹⁶
Języki programowania	C#, F# i VB.Net	Java, Kotlin i Apache Groovy
Zewnętrzne biblioteki	Szeroki zakres wsparcia dla bibliotek zewnętrznych	Szeroki zakres wsparcia dla bibliotek zewnętrznych
Multi-platformowość	Obsługuje multi-platformowość	Obsługuje multi-platformowość
Integracja z IDE	Obsługuje różne środowiska zintegrowanego rozwoju IDE, najbardziej dostosowana do Visual Studio.	Obsługuje różne środowiska zintegrowanego rozwoju IDE, głównie: IntelliJ IDEA, Eclipse, VS Code.

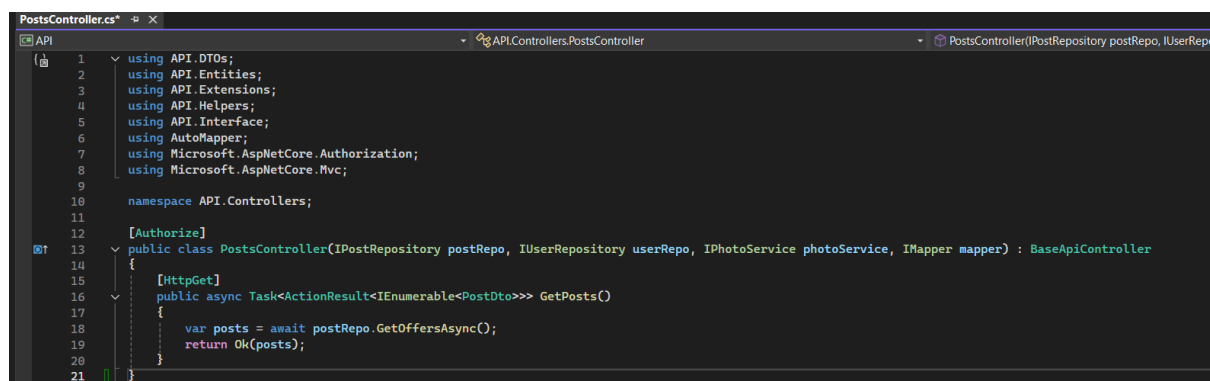
*Tabela 4 Porównanie ASP.Net Core`a i Spring`a
źródło: Opracowanie własne*

Ze względu na przedstawione wyżej zalety zdecydowałem się wybrać ASP.NET jako backend mojej aplikacji. Jednym z głównych powodów były również moje wcześniejsze doświadczenie z tą platformą oraz z językiem C#.

¹⁶ <https://docs.spring.io/spring-framework/reference/index.html> na dzień 31 stycznia 2025 roku

Przykładowy kontroler ASP.NET Core

Na poniższym rysunku znajduje się zrzut ekranu przykładowego kontrolera dla postów w mojej aplikacji, odpowiedzialnego za przetwarzanie żądań HTTP. W kontrolerze tym znajduje się jedna metoda GET, zwracająca listę wszystkich postów z bazy danych.



```
1 using API.DTOs;
2 using API.Entities;
3 using API.Extensions;
4 using API.Helpers;
5 using API.Interface;
6 using AutoMapper;
7 using Microsoft.AspNetCore.Authorization;
8 using Microsoft.AspNetCore.Mvc;
9
10 namespace API.Controllers;
11
12 [Authorize]
13 public class PostsController(IPostRepository postRepo, IUserRepository userRepo, IPhotoService photoService, IMapper mapper) : BaseApiController
14 {
15     [HttpGet]
16     public async Task<ActionResult<IEnumerable<PostDto>>> GetPosts()
17     {
18         var posts = await postRepo.GetOffersAsync();
19         return Ok(posts);
20     }
21 }
```

Rys. 10 Przykładowy rzut ekranu Kontrolera ASP .Net Core- Ten Kontroler odp. za wszystkie metody akcji związane z postami
źródło: Opracowanie własne

Kontroler¹⁷ to klasa, która dziedziczy po głównym kontrolerze ControllerBase. Zadaniem kontrolera jest przyjmowanie żądań HTTP, przetwarzanie danych oraz wysyłanie odpowiedzi do klienta. Za realizację tych operacji odpowiadają metody Action, które są odzwierciedlają rodzaje zapytań HTTP (na przykład: Get, Post, Put, Delete, itd.).

Klasa¹⁸ to jeden z podstawowych paradygmatów programowania obiektowego (OOP), który służy do tworzenia obiektów. Klasa może posiadać pola, metody oraz konstruktor. Składowe te w praktyce opisują dany obiekt.

2.1.6. Język programowania C#

C#¹⁹ to obiektowy język programowania wysokiego poziomu. Programowanie obiektowe (OOP), to metodologia składająca się z czterech paradygmatów: hermetyzacji, dziedziczenia, polimorfizmu i abstrakcji. Zwiększają one elastyczność kodu, zapewniają wysoką wydajność i gwarantują bezpieczeństwo typów, co ułatwia tworzenie projektów w tym języku programowania. Język programowania wysokiego poziomu – oznacza, że język

¹⁷ Tsetsekas H., „Building real life web apps with Angular 14 and ASP.NET Core 6”, 2022, s 56

¹⁸ Svetlin Nakov & Co., „Fundamentals of computer programming with C#”, Telerik Software Academy, 2013, s 503 - 506

¹⁹ Svetlin Nakov & Co., „Fundamentals of computer programming with C#”, Telerik Software Academy, 2013, s 25

programowania C# charakteryzuje się przejrzystą składnią kodu, co ułatwia jego zrozumienie w trakcie tworzenia.

Przy pomocy języka C# w środowisku .NET można tworzyć aplikacje desktopowe, mobilne i internetowe, korzystając z różnego rodzaju frameworków (przykładowo Windows Presentation Foundation, .Net MAUI, ASP .Net).

.NET²⁰ to platforma programistyczna umożliwiająca tworzenie i uruchamianie programów w różnych językach programowania (C#, F# i VB.NET). .Net jest technologią wieloplatformową, co oznacza, że można jej używać w różnych systemach operacyjnych: Windows, macOS, Linux.

Zalety i wady języka programowania C#

Zalety:

- Obsługa programowania obiektowego
- Automatyczne zarządzanie pamięcią za pomocą Garbage Collection²¹
- Łagodna krzywa uczenia się dla osób znających język Java, ze względu na zbliżoną składnię tych języków
- C# jest oprogramowaniem typu open source
- Duża popularność tego języka sprawiła, że istnieją liczne dokumentacje i materiały pomocnicze

Wady:

- W porównaniu do innych języków programowania (przykładowo swojego poprzednika - C++) C# jest wolniejszy pod względem kompilacji kodu
- Stroma krzywa uczenia się

²⁰ Svetlin Nakov & Co., „Fundamentals of computer programming with C#”, Telerik Software Academy, 2013, s 26

²¹ Svetlin Nakov & Co., „Fundamentals of computer programming with C#”, Telerik Software Academy, 2013, s 84

Porównanie C# z Java

Alternatywnym językiem programowania dla C# jest Java.

	C#	Java ²²
Środowiska uruchomienia	CLR	JRE
Programowanie	Obiektywnie	Obiektowe
Wydajność	Szybszy od Java	Wolniejszy od C#
Ekosystem bibliotek	Szeroki zakres wsparcia dla różnych bibliotek	Szeroki zakres wsparcia dla różnych bibliotek
Integracja z IDE	Obsługuje różne środowiska IDE, jednak najbardziej jest dostosowany pod Visual Studio	Obsługuje różne środowiska IDE, jednak najbardziej jest dostosowany pod IntelliJ IDEA

*Tabela 5 Porównanie C#`a i Java`a
źródło: Opracowanie własne*

Postanowiłem wybrać język C# ze względu na fakt, iż jest to najpopularniejszy język dla środowiska .NET, a także ze względu na moje wcześniejsze doświadczenie z tym językiem programowania.

2.2. System Kontroli Wersji

2.2.1. Git

Git²³ jest systemem kontroli wersji, umożliwiającym programistom prostsze zarządzanie projektem, historią zmian czy też cofanie ich. Git oferuje te funkcje za pomocą tak zwanych gałęzi (ang. branches). Programista może tworzyć nowe gałęzie, łączyć je ze sobą, usuwać je lub przełączać się między nimi.

2.2.2. GitHub

GitHub²⁴ to platforma, umożliwiająca programistom przechowywanie kodu w repozytorium, które jest zarządzane przez Git'a. Dzięki temu rozwiązaniu zespół programistów

²² <https://docs.oracle.com/javase/8/docs/technotes/guides/language/> na dzień 31 stycznia 2025 roku

²³ Tsitoara M., „Git I GitHub Kontrola wersji, zarządzanie projektami i zasady pracy zespołowej”, Helion, Gliwice 2022, s. 23

²⁴ Tsitoara M., „Git I GitHub Kontrola wersji, zarządzanie projektami i zasady pracy zespołowej”, Helion, Gliwice 2022, s. 89-90

może w jednym momencie pracować nad projektem przy pomocy gałęzi, a także można je traktować jako archiwum historii zmian.

Zalety i wady GitHub

Zalety

- Intuicyjny interfejs
- Zarządzanie repozytoriami
- Łagodna krzywa uczenia się
- Duża popularność
- Integracja z innymi aplikacjami, przykładowo Docker, Jira i Travis CI

Wady

- Brak zaawansowanych funkcji analizy kodu CI / CD

Porównanie GitHub z GitLab

Alternatywną usługą do GitHuba jest GitLab.

	GitHub	GitLab
Graficzny interface	Prosty interfejs graficzny	Bardziej zaawansowany interfejs graficzny, co jest uzasadnione bogatszą funkcjonalnością
Zarządzanie repozytoriami	Tworzenie lub usuwanie gałęzi, łączenie gałęzi ze sobą oraz przypisywanie tagów do commit'ów	Tworzenie lub usuwanie gałęzi, łączenie gałęzi ze sobą oraz automatyczne testowanie i recenzowanie
Zarządzanie dostępem do repozytorium	Podstawowa funkcjonalność nadawania praw odczytu lub edycji do repozytorium dla innych użytkowników	Zaawansowana funkcjonalność tworzenia ról i przypisywanie ich użytkownikom
Importowanie lub eksportowanie projektów na inne platformy	Brak	Istnieje możliwość eksportowania lub importowania projektów na inne platformy, jak na przykład GitHub

*Tabela 6 Porównanie GitHub'a i GitLab'a
źródło: Opracowanie własne*

Ze względu na wyżej wymienione zalety oraz moje wcześniejsze doświadczenia zdecydowałem się skorzystać z usług GitHuba w mojej aplikacji. Dodatkowo mój wybór jest uzasadniony faktem, iż jest to mały projekt.

2.3. Bazy danych

2.3.1. SQLite

SQLite²⁵ to lekka w zamyśle i bez serwerowa, relacyjna baza danych. W odróżnieniu od innych systemów zarządzania relacyjnymi bazami danych, takich jak Microsoft SQL Server, PostgreSQL czy MySQL, SQLite nie posiada wbudowanego serwera. Oznacza to, że proces uruchamiania aplikacji nie wymaga łączenia się z serwerem w celu korzystania z bazy danych. To lekkie rozwiązanie pozwala w prosty i szybki sposób tworzyć aplikację na jej wczesnym poziomie rozwoju. Nie oznacza to jednak, że w procesie tworzenia aplikacji programista jest pozbawiony możliwości korzystania z system zarządzania relacyjną bazą danych (skrót z ang. RDBMS) – można to zrobić za pomocą zewnętrznej biblioteki SQLite. Omawiana baza danych obsługuje język zapytań SQL (SQL-92).

2.3.2. Microsoft SQL Server

Microsoft SQL Server²⁶ to system zarządzania relacyjnymi bazami danych (skrót z ang. RDBMS). W odróżnieniu od SQLite, MSSQL posiada serwer bazodanowy, który umożliwia zarządzanie bazą danych i jej danymi. RDBMS posiada architekturę klient-serwer, co wymaga podłączenia aplikacji do serwera. MSSQL zapewnia również większe bezpieczeństwo danych poprzez szyfrowanie oraz jest obsługiwana przez język zapytań T-SQL (Transact-SQL).

Podczas procesu realizacji projektu w celu szybkiego zarządzania bazą danych i zaoszczędzenia czasu korzystałem z SQLite, a po zakończeniu tworzenia aplikacji zmieniłem bazę danych na MSSQL, aby baza danych korzystała z serwera.

Zalety i wady Microsoft SQL Server

Zalety

- Intuicyjny interfejs graficzny
- Zwiększone bezpieczeństwo, dzięki szyfrowaniu danych
- Bardziej zaawansowany język T-SQL w porównaniu do SQL

²⁵ <https://www.sqlite.org/about.html> na dzień 31 stycznia 2025 roku

²⁶ Dewson R., „SQL Server wstęp dla programista Wydanie IV”, Helion, Gliwice 2016, s 18-19

Wady

- Konieczność ponoszenia kosztów w wersjach komercyjnych
- Konieczność posiadania wydajnych podzespołów w celu utrzymania dobrej jakości obsługi serwera

Zalety i wady SQLite

Zalety

- Prostota użycia i integracji z aplikacją przez brak serwera

Wady

- Brak serwera
- Brak mechanizmów szyfrujących dane, podnoszące bezpieczeństwo danych. W przypadku chęci zadbania o bezpieczeństwo konieczność instalacji zewnętrznych bibliotek, jak na przykład SQLCipher

Porównanie SQLite z MSSQL

	SQLite	MSSQL
Typ modelu bazy danych	Relacyjna baza danych	Relacyjna baza danych
Serwer	Brak	Istnieje możliwość implementacji kilku serwerów na różnych systemach operacyjnych: Linux i Microsoft
Wspierane języki programowania	Wspiera 28 języków programowania (przykładowo C#, Java, JavaScript, PHP, Python, C++, C, Object-C itd.)	Wspiera 11 języków programowania (przykładowo C#, Java, JavaScript, PHP, Python, C++ itd.)
Język zapytań	SQL	T-SQL
Interakcja z bazą danych	Bezpośrednia praca z plikiem w formacie .db	Narzędzie klienckie Microsoft SQL Server Management Studio

*Tabela 7 Porównanie SQLite`a i MSSQL`a
źródło: Opracowanie własne*

2.4. Bezpieczeństwo

W projekcie zdecydowałem się użyć różne mechanizmy i technologie podnoszące bezpieczeństwo aplikacji. Poniżej opisuje te technologie, z których zdecydowałem się skorzystać.

HTTPS - Protokół transferu hipertekstu, który rozszerza protokół HTTP. Zapewnia ochronę przesyłanych danych, szyfrując je za pomocą protokołu SSL/TLS pomiędzy przeglądarką użytkownika a serwerem.

Porównanie HTTPS i HTTP

	HTTPS	HTTP
Szyfrowanie	Szyfruje dane za pomocą protokołu SSL/TLS	Brak
Bezpieczeństwo	Połączenie jest zabezpieczone	Połączenie nie jest zabezpieczone
Wydajność	Wolniejsze ładowanie strony internetowej	Szybsze ładowanie strony internetowej

*Tabela 8 Porównanie HTTPS`a i HTTP`a
źródło: Opracowanie własne*

Token JWT²⁷ (ang. JSON Web-Token) – Token wykorzystywany przy wysyłaniu zapytań na serwer w autoryzacji ich. Dzięki temu rozwiązaniu zapytania są chronione przed nieautoryzowanymi użytkownikami. Token potwierdza tożsamość użytkownika, a system weryfikuje ważność tokena przy każdym żądaniu. W przypadku, gdy tokenowi wygaśnie ważność, użytkownik nie uzyskuje dostępu do żądanych informacji. Za pomocą zapisywania tokena do local storage zrealizowałem również funkcjonalność nie wylogowywania użytkownika po zamknięciu aplikacji, po ponownym włączeniu jej pozostaje on zalogowany.

CORS²⁸ (ang. Cross-Origin Resource Sharing) - Mechanizm umożliwiający bezpieczną komunikację między domenami, chroniący użytkowników przed potencjalnymi intruzami, którzy mogliby próbować uzyskać informacje o użytkownikach lub w jakikolwiek sposób manipulować ich danymi.

²⁷ https://en.wikipedia.org/wiki/JSON_Web_Token na dzień 31 stycznia 2025 roku

²⁸ https://en.wikipedia.org/wiki/Cross-origin_resource_sharing na dzień 31 stycznia 2025 roku

ASP.NET Core Identity²⁹ - to system zarządzania encjami użytkowników w środowisku ASP.NET, odpowiedzialny za identyfikację, uwierzytelnianie i autoryzację użytkowników. Jest to pakiet udostępniający gotową funkcjonalność zarządzania rolami, rejestracją, logowaniem i wylogowaniem użytkownika. Oferuje listę gotowych encji powiązanych z różnymi aspektami tabeli użytkownika.

Walidacja formularzy³⁰ polega na weryfikacji pól w formularzu przed wysłaniem zapytania na serwer. Jednym ze sposobów zabezpieczenia się przed atakami typu Cross-Site Scripting (XSS) jest właściwa obsługa danych wejściowych.

2.5. Testowanie aplikacji

2.5.1. Testy jednostkowe i integracyjne

Ze względu na fakt, iż w momencie pisania pracy inżynierskiej proces tworzenia aplikacji nie został jeszcze ukończony, nie tworzyłem testów jednostkowych ani integracyjnych. Podczas dodawania nowych funkcjonalności do aplikacji kod często ulegał zmianie, co w przypadku istnienia testów mogło prowadzić do nieuniknionego aktualizowania ich. Mój wybór jest uzasadniony chęcią zaoszczędzenia czasu i przeniesieniem procesu pisania testów na końcowy etap tworzenia aplikacji. Zdaję jednak sobie sprawę z fundamentalnej konieczności tworzenia testów i z faktu, iż jest to nieodłączny proces inżynierii oprogramowania.

2.5.2. Postman

Podczas procesu realizacji projektu testowałem poprawność API oraz obsługę endpointów po stronie serwera. W tym celu użyłem narzędzia Postman³¹, które pozwala wysyłać żądania na serwer bez konieczności posiadania części frontendowej. Narzędzie to pozwala na tworzenie różnych rodzajów zapytań (GET, POST, PUT, DELETE, itd.) oraz weryfikowanie i analizowanie otrzymanych poprawności odpowiedzi w kilku formatach (JSON, XML).

²⁹ Hans-Petter Halvorsen, „Web Programming ASP.NET Core”, 2021, s 199

³⁰ <https://cyrekdigital.com/pl/baza-wiedzy/cross-site-scripting/> na dzień 31 stycznia 2025 roku

³¹ <https://learning.postman.com/docs/introduction/overview/> na dzień 31 stycznia 2025 roku

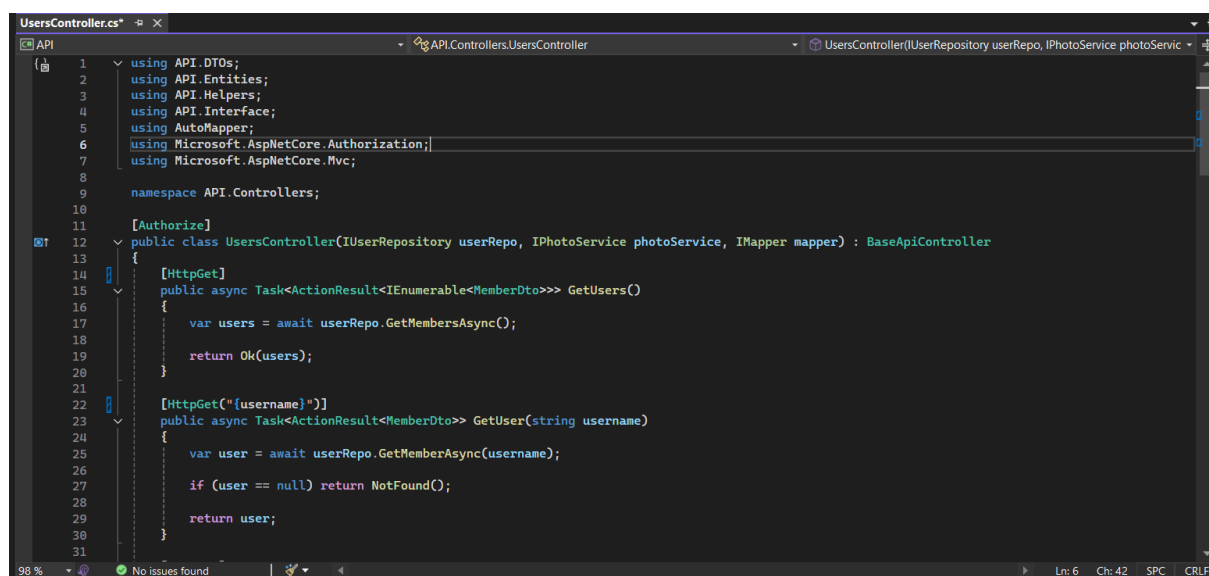
2.5.3. Development Tool

Weryfikowania poprawności otrzymywanych odpowiedzi z serwera dokonywałem również korzystając z narzędzia Development Tool, wbudowanego w przeglądarki internetowe. Korzystałem z niego jednak już w momencie wysyłania żądań poprzez gotowy interfejs graficzny aplikacji, a nie jak w przypadku Postmana bez konieczności posiadania front-endu. Analizowanie odpowiedzi w tym narzędziu jest możliwe w zakładce „Network” w oknie „Preview”.

3. Dokumentacja procesu i implementacji

3.1. Opis procesu realizacji projektu

Do stworzenia projektu części serwerowej wykorzystałem zintegrowane środowisko programistyczne Visual Studio 2022. W pierwszej kolejności stworzyłem projekt ASP.NET Core API, jako serwer backendowy mojej aplikacji i zacząłem od stworzenia wszystkich klas encji ze wcześniej zaprojektowanej struktury bazy danych oraz klas kontekstu. W celu korzystania z niezbędnych funkcjonalności zainstalowałem również wymagane pakiety z narzędzia NuGet i wykonałem pierwszą migrację do bazy danych SQLite. Następnie zacząłem pisać punkty końcowe pierwszego kontrolera „UserController”, odpowiadającego przede wszystkim za logowanie i rejestrację użytkowników. W ten sposób dokonałem pierwszego commita. Każdą kolejną gałąź nazywałem odpowiednio „EW-numer kolejnej gałęzi” i każda odzwierciedlała inne wymaganie funkcjonalne. Warto również wspomnieć, iż aplikacja została w całości stworzona przeze mnie samodzielnie.

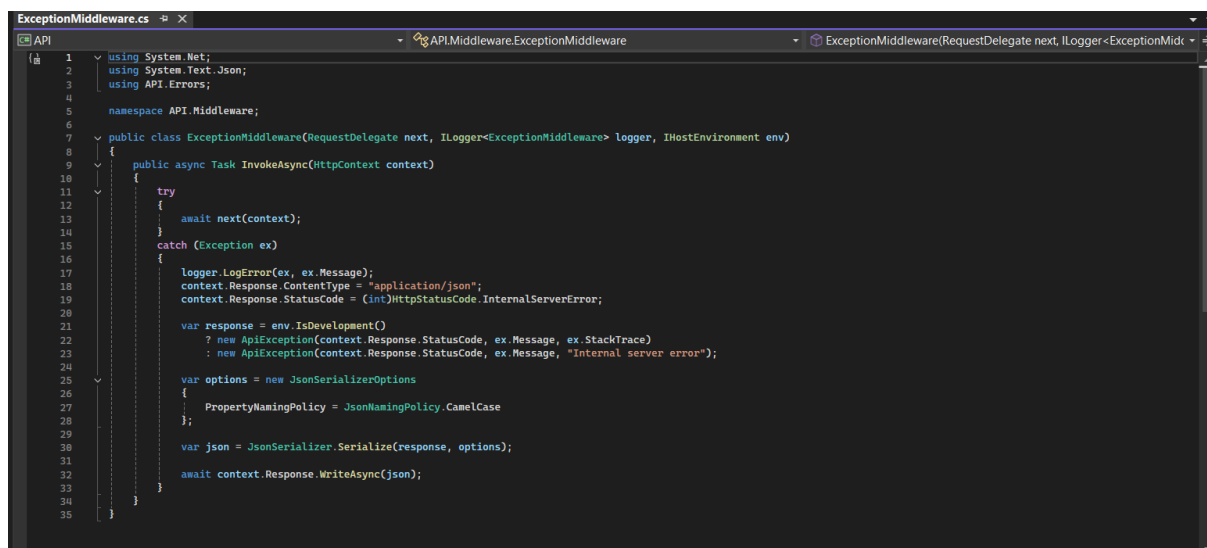


Rys. 11 Przykładowy rzut ekranu Kontrolera ASP .Net Core – „UserController”
źródło: Opracowanie własne

Następnie postanowiłem zająć się frontendową częścią aplikacji i stworzyłem projekt w frameworku Angular. Po stronie backendowej dodałem więc obsługę CORS w celu zwiększenia bezpieczeństwa aplikacji oraz zaimplementowałem pierwszą funkcjonalność aplikacji – uwierzytelnianie. Stworzyłem odpowiednie strony z formularzami logowania i rejestracji oraz napisałem odpowiednie endpointy po stronie backendu. Do obsługi logowania wykorzystałem token JWT, będący popularnym i bezpiecznym standardem. Dzięki temu rozwiązaniu również, zalogowany użytkownik po zamknięciu aplikacji i uruchomieniu jej na

nowo dalej pozostanie zalogowany, bez konieczności ponownego logowania. Na tym etapie dodałem również panel nawigacji w górnej części strony.

W kolejnym kroku dodałem strony „Lists”, „Messages”, „Offers-List” oraz „Offer-Detail”, których funkcjonalności wraz z rozwojem aplikacji, z czasem implementowałem. Na tym etapie dodałem również obsługę błędów po stronie Angulara za pomocą interceptora, przechwytyjącego odpowiedzi z serwera. Błędy są wypisywane dla użytkownika w formie wyskakującego okienka przy pomocy biblioteki Toastr. Do testowania błędów na frontendzie stworzyłem specjalną stronę, której widoczność w późniejszej części aplikacji zablokowałem dla standardowych użytkowników i może na nią wejść jedynie użytkownik z rolą administratora. Również obsłużyłem błędy po stronie backendu na poziomie wykonywania Middleware.



```
1 using System.Net;
2 using System.Text.Json;
3 using API.Errors;
4
5 namespace API.Middleware;
6
7 public class ExceptionMiddleware(RequestDelegate next, ILogger<ExceptionMiddleware> logger, IHostEnvironment env)
8 {
9     public async Task InvokeAsync(HttpContext context)
10     {
11         try
12         {
13             await next(context);
14         }
15         catch (Exception ex)
16         {
17             logger.LogError(ex, ex.Message);
18             context.Response.ContentType = "application/json";
19             context.Response.StatusCode = (int)HttpStatusCode.InternalServerError;
20
21             var response = env.IsDevelopment()
22                 ? new ApiException(context.Response.StatusCode, ex.Message, ex.StackTrace)
23                 : new ApiException(context.Response.StatusCode, ex.Message, "Internal server error");
24
25             var options = new JsonSerializerOptions
26             {
27                 PropertyNamingPolicy = JsonNamingPolicy.CamelCase
28             };
29
30             var json = JsonSerializer.Serialize(response, options);
31
32             await context.Response.WriteAsync(json);
33         }
34     }
35 }
```

Rys. 12 Przykładowy rzut ekranu ASP .Net Core – „ExceptionMiddleware”
źródło: Opracowanie własne

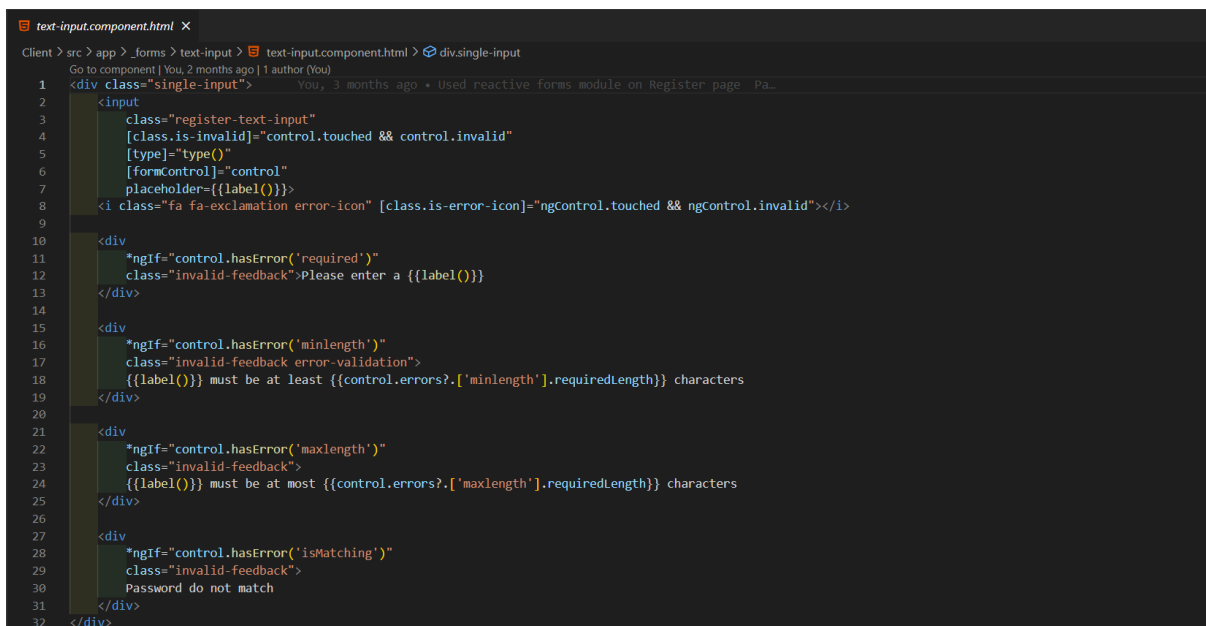
Ze względu na charakter mojej aplikacji na etapie realizacji potrzebowałem fałszywych danych, na których mogłem wykonywać przykładowe operacje. W tym celu skorzystałem z zewnętrznego API do wygenerowania fałszywych kont użytkowników oraz postów i wykorzystałem je w bazie danych mojej aplikacji.

Posiadając fałszywe dane, dokończyłem wygląd strony „Offers”, która wyświetla oferty podróży. Pojedyncza „oferta” jest postem wstawionym przez użytkownika. Ze względu na dużą liczbę wyświetlanych postów w jednym momencie zaimplementowałem technologię lazy-loading.

W następnym kroku dodałem funkcjonalność edycji własnego profilu użytkownika. Przy okazji tworzenia tej funkcjonalności postanowiłem dodać możliwość przesyłania własnych

zdjęć profilowych. W tym celu skorzystałem z zewnętrznej platformy Cloudinary do przechowywania zdjęć w chmurze. Posiadając możliwość przesyłania zdjęć dokończyłem funkcjonalność tworzenia własnych postów i przy okazji edycji oraz usuwania ich.

Na tym etapie postanowiłem wrócić do funkcjonalności logowania i rejestracji oraz nadać tym formularzom walidacji pól. W tym celu zmieniłem również postać użytych formularzy, które były oparte na szablonach na formularze reaktywne. Następnie zrealizowałem możliwość polubienia postów innych użytkowników.



```
1 <div class="single-input">
2   <input
3     class="register-text-input"
4     [class.is-invalid]="control.touched && control.invalid"
5     [type]="type()"
6     [formControl]="control"
7     placeholder="{{label()}}">
8   <i class="fa fa-exclamation error-icon" [class.is-error-icon]="ngControl.touched && ngControl.invalid"></i>
9
10  <div
11    *ngIf="control.hasError('required')"
12    class="invalid-feedback">Please enter a {{label()}}
13  </div>
14
15  <div
16    *ngIf="control.hasError('minlength')"
17    class="invalid-feedback error-validation">
18    {{label()}} must be at least {{control.errors?.['minlength'].requiredLength}} characters
19  </div>
20
21  <div
22    *ngIf="control.hasError('maxlength')"
23    class="invalid-feedback">
24    {{label()}} must be at most {{control.errors?.['maxlength'].requiredLength}} characters
25  </div>
26
27  <div
28    *ngIf="control.hasError('ismatching')"
29    class="invalid-feedback">
30    Password do not match
31  </div>
32 </div>
```

Rys. 13 Przykładowy rzut ekranu Angular – „text-input.component.html”
źródło: Opracowanie własne

Kolejną funkcjonalnością, którą zdecydowałem się zrealizować było prowadzenie konwersacji za pomocą czatu z innymi użytkownikami w czasie rzeczywistym. W tym celu skorzystałem z biblioteki SignalR.

Następnie do bazy danych dodałem konto administratora, które jest zapisywane do bazy danych przy pierwszym uruchomieniu serwera o nazwie „admin” z hasłem „Admin2024”. Na tym etapie zrealizowałem rozróżnianie ról użytkowników przez backend oraz możliwość zarządzania rolami w panelu administratora.

Posiadając wszystkie wymagania funkcjonalne spełnione skonfigurowałem protokół HTTPS dla strony oraz przeprowadziłem testy wydajnościowe i testy skalowalności K6. Ostatnią czynnością jaką wykonałem była zmiana bazy danych z SQLite na MSSQL.

3.2. Problemy napotkane w trakcie realizacji projektu

Podczas realizacji projektu napotkałem na szereg problemów, które postanowiłem opisać w osobnym podrozdziale. Podczas implementacji operacji CRUD dla postów po stronie klienta napotkałem na błąd związany z brakiem aktualizacji postów. Po wcześniejszym stworzeniu postu, bądź jego edycji lub usunięciu, nie zostawał on dodany, bądź zaktualizowany na stronie „Offers”, lecz po odświeżeniu strony lista wyświetlała się poprawnie. Rozwiązaniem tego problemu okazało się być dodanie metody update do sygnału po stronie Angulara. Metoda ta aktualizuje stan sygnału bez konieczności ponownego odświeżania strony.

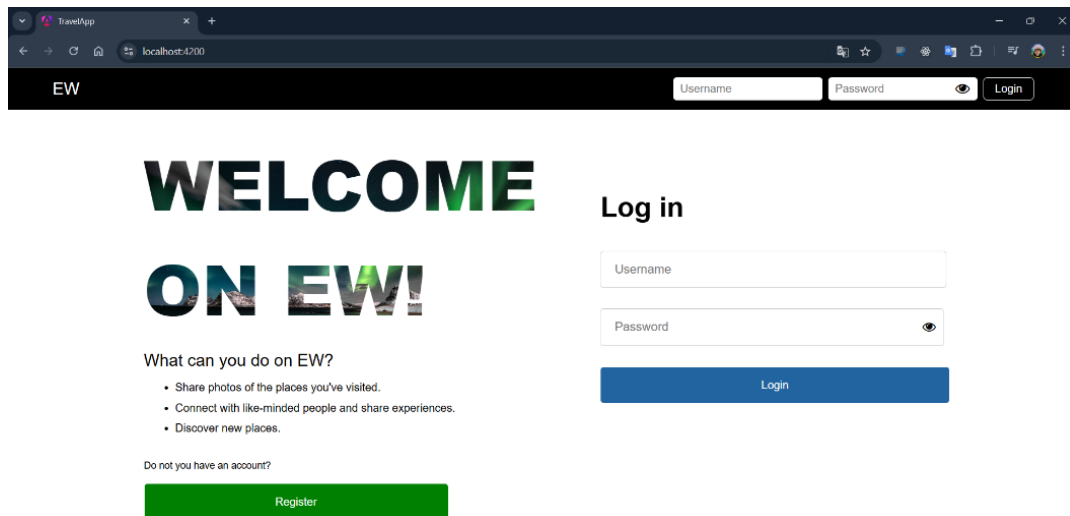
Kolejny problem, który mi się przytrafił polegał na skonfigurowaniu protokołu HTTPS. Nie potrafiłem bowiem znaleźć jednoznacznych instrukcji instalacji tej biblioteki. Jednym z rozwiązań okazało się być wykorzystanie pakietu Chocolatey, na który ja się zdecydowałem.

Po zmianie bazy danych z SQLite na MSSQL zauważyłem, że niektóre operacje usuwania postów były zakończone niepowodzeniem. Okazało się, iż problem leżał w skonfigurowanych regułach usuwania, które były ustawione na NO ACTION. Sprawiało to, że próba usunięcia rekordu, który był powiązany z innymi rekordami innej tabeli była blokowana. Rozwiązaniem tego problemu było ustawienie reguł usuwania bazy danych na kaskadowe. Dzięki temu przy próbie usunięcia rekordów są one usuwane razem z powiązanymi rekordami.

Oprócz problemów związanych z danymi funkcjonalnościami miałem także różne dylematy. Zastanawiałem się nad interfejsem graficznym strony postów. Próbowałem wielu wariantów, lecz wybór ostatecznej wersji, która byłaby najbardziej satysfakcjonująca i intuicyjna dla użytkownika nie była oczywista. Po analizie wielu interfejsów innych stron internetowych zdecydowałem się na obecną wersję.

Innym dylematem był wybór formy przechowywania zdjęć. Zdecydowałem się na platformę Cloudinary, która oferuje przechowywanie zdjęć w chmurze. W tym rozwiązaniu w bazie danych przechowywana jest jedynie adnotacja do danego zdjęcia, a nie samo zdjęcie, co znacznie pozwala zminimalizować wielkość zasobów przechowywanych na serwerze. Wybór tej konkretnej platformy był również podyktowany wysoką wydajnością, Cloudinary jest w stanie w szybko zwrócić interesujące nas zdjęcia, jak również w krótkim czasie obsługuje przesyłanie zdjęć. Zapewnia również bezpieczeństwo, dostęp do dysku w chmurze przechowującego zdjęcia jest dostępny jedynie dla posiadaczy odpowiedniego klucza dostępu. Cloudinary posiada także intuicyjny interfejs graficzny oraz oferuje liczne funkcje jak generowanie miniatur, znaków wodnych, bądź wersji responsywnych.

3.3. Przedstawienie gotowej aplikacji

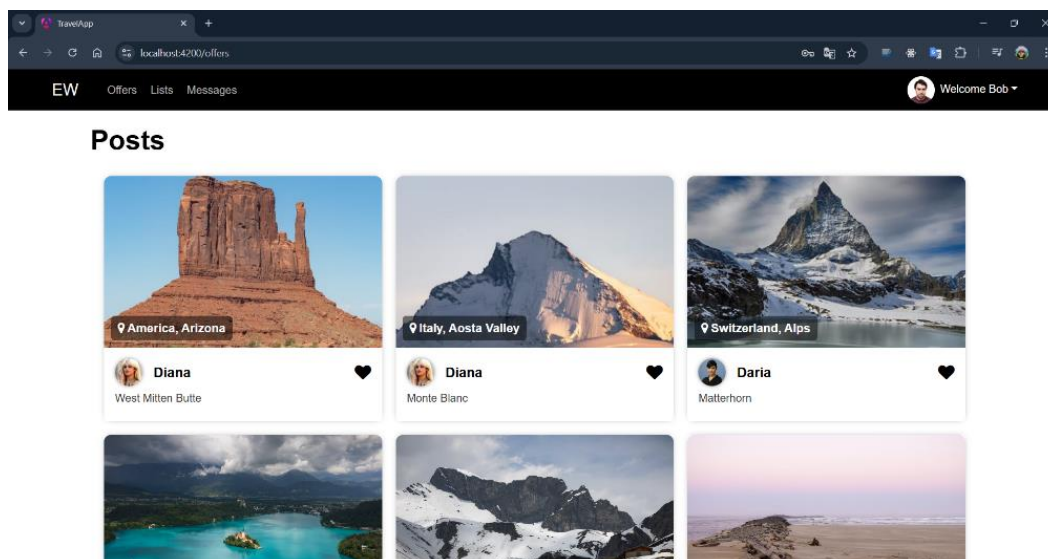


Rys. 14 Strona logowania
źródło: opracowanie własne

W tej sekcji zaprezentuję poszczególne strony, podstrony i okna końcowej wersji aplikacji. Zostały one stworzone na podstawie makiet, które przedstawiłem we wcześniejszej części pracy. Na poniższych rysunkach znajdują się zrzuty ekranu zrealizowanej aplikacji wraz z krótkim opisem pod nimi.

Na powyższym rysunku znajduje się zrzut ekranu strony odpowiedzialnej za logowanie użytkownika z wyświetlonymi sloganami, które definiują cele mojej aplikacji:

- Udostępnianie zdjęć odwiedzonych miejsc
- Łączenie się z ludźmi o podobnych poglądach i dzielenie się doświadczeniami
- Odkrywanie nowych miejsc



Rys. 15 Strona ofert (Strona główna aplikacji)
źródło: opracowanie własne

Strona ofert, będącej również stroną główną zalogowanego użytkownika, na której wyświetlane są posty wszystkich użytkowników.

3.4. Opis testów skalowalności oraz wydajności zrealizowanego projektu

Wydajność (ang. Performance)

Celem testu wydajnościowego jest diagnostyka wydajności aplikacji internetowej oraz analiza prędkości i stabilności przetwarzania żądań. Wysoka jakość wydajności jest kluczowym aspektem w sieciach społecznościowych, takich jak moja aplikacja.

Scenariusz testowania:

W ramach testu wydajności przeprowadzono symulację za pomocą programu k6 potencjalnego scenariusza z udziałem trzystu jednocześnie korzystających z aplikacji użytkowników w ciągu sześćdziesięciu sekund. Użytkownicy wysyłali żądania HTTP na serwer, w celu pobrania listy wszystkich postów.

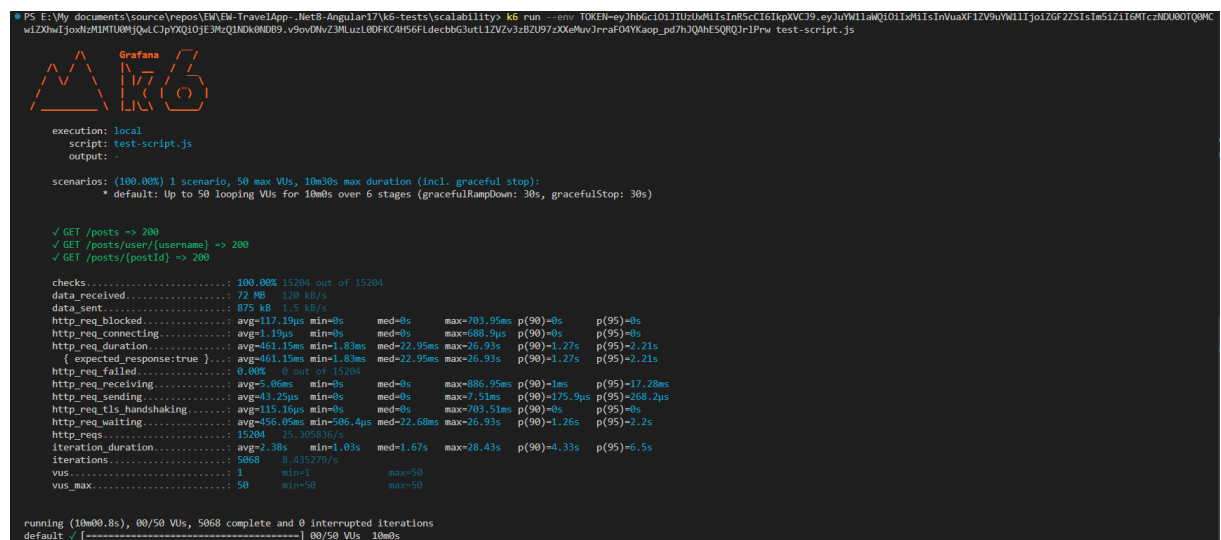
Scenariusz testowania:

W ramach testu skalowalności przeprowadzono następujący scenariusz:

- Test rozpoczyna się od minimalnego obciążenia wynoszącego pięciu użytkowników na minutę.
- Liczba wirtualnych użytkowników stopniowo się zwiększa o 10, 20, 30 i 50 w odstępach dwu minutowych.
- Pod koniec liczba użytkowników stopniowo maleje do zera w ciągu jednej minuty.

Podczas testu użytkownicy wysyłają żądania z wykorzystaniem metody GET w celu:

- pobrania wszystkich postów
- pobrania wszystkich postów konkretnego użytkownika
- pobrania konkretnego postu



Rys. 17 Wynik obliczeń testu skalowalności k6
źródło: opracowanie własne

Wynik testu:

Liczba żądań wyniosła 15 204 w ciągu dziesięciu minut. Wszystkie zapytania zakończyły się statusem 200 (OK), co wskazuje na poprawne przetwarzanie żądań. Test zakończył się sukcesem, a wyniki zostały ocenione jako bardzo dobre.

Czas odpowiedzi:

- Średni czas odpowiedzi: 6,37 ms
- Maksymalny czas odpowiedzi: 260,36 ms.

Liczba zapytań wyniosła 15 204 w ciągu dziesięciu minut, przy pięćdziesięciu użytkownikach wykonujących trzy rodzaje zapytań. Odpowiedzi były uzyskiwane średnio w czasie czterystu sześćdziesięciu jeden milisekund. Gdy tylko laptop osiągnął akceptowalny poziom obciążenia, wyniki potwierdziły, że serwer jest w stanie przetwarzać zapytania z wysoką prędkością, spełniającą współczesne wymagania dotyczące wydajności.

Kompatybilność (ang. Compatibility)

Aplikacja jest kompatybilna z nowoczesnymi przeglądarkami:

- Google Chrome: wersja 95+
- Mozilla Firefox: wersja 90+
- Microsoft Edge: wersja 90+
- Opera: wersja 80+

Obsługiwane są również połączenia internetowe za pośrednictwem sieci Wi-Fi. System działa stabilnie przy średniej prędkości dziesięciu megabitów na sekundę, co zapewnia komfortowe korzystanie z sieci społecznościowej. Dodatkowo aplikacja obsługuje sieci komórkowe, takie jak 4G i 3G, zapewniając stabilną pracę przy prędkości pięciu megabitów na sekundę.

Zakończenie

Celem mojej pracy inżynierskiej było stworzenie aplikacji zaspokajającej potrzeby osób zajmujących się ogólnie pojętą problematyką turystyki i pragnącym przeżyć niezapomniane przygody podczas planowanego wyjazdu. Zrealizowany w ramach pracy projekt inżynierski spełnia zróżnicowane wymagania użytkowników.

Dzięki pracy nad tym projektem rozwinąłem swoje umiejętności w zakresie programowania w języku C# oraz TypeScript, a także udoskonaliłem wiedzę w korzystaniu z frameworków, takich jak ASP.NET Core Web API i Angular.

Podczas realizacji projektu napotkałem kilka problemów, które udało się sukcesywnie rozwiązać. Najważniejsze z nich to:

- **Brak automatycznej aktualizacji danych po stronie klienta** – błąd ten naprawiłem, wykorzystując wbudowaną metodę „update” w sygnale w pliku service.
- **Problem migracji danych z SQLite na MSSQL** – dotyczył on zasad usuwania w niektórych encjach w klasie kontekstu. Rozwiązałem go, zmieniając regułę usuwania na NO ACTION, zgodną z wymogami Microsoft SQL Server.

Praca nad tym projektem pomogła mi stać się bardziej kompetentnym programistą w dziedzinie projektowania i implementacji aplikacji internetowych. W przyszłości pragnę rozwinąć aplikację, pracując nad graficznym interfejsem systemu, dodając wsparcie wielojęzyczne oraz urozmaicając funkcjonalności mediów społecznościowych.

Bibliografia

1. Śmiałek M., Rybiński K., „Inżynieria oprogramowania w praktyce od wymagań do kodu z językiem UML”, Helion, Gliwice 2017
2. <https://v18.angular.dev/overview> na dzień 31 stycznia 2025 roku
3. <https://nodejs.org/docs/latest/api/documentation.html> na dzień 31 stycznia 2025 roku
4. <https://v18.angular.dev/tools/cli> na dzień 31 stycznia 2025 roku
5. Morgan J., „How To Code in React.js”, DigitalOcean, New York 2021
6. <https://v18.angular.dev/guide/components> na dzień 31 stycznia 2025 roku
7. <https://v18.angular.dev/essentials/components> na dzień 31 stycznia 2025 roku
8. Cherny B., „Programming TypeScript Making Your JavaScript Applications Scale”, O’Reilly, Sebastopol 2019
9. Althoff C., „Programista samouk, Profesjonalny przewodnik do samodzielnej nauki kodowania”, Helion, Gliwice 2017
10. Harris A., „HTML5 and CSS3 All-in-One For Dummies”, John Wiley & Sons, New Jersey 2014
11. <https://sass-lang.com/documentation/> na dzień 31 stycznia 2025 roku
12. <https://sass-lang.com/documentation/style-rules/> na dzień 31 stycznia 2025 roku
13. Hans-Petter Halvorsen, „Web Programming ASP.NET Core”, 2021
14. <https://docs.spring.io/spring-framework/reference/index.html> na dzień 31 stycznia 2025 roku
15. Tsetsekas H., „Building real life web apps with Angular 14 and ASP.NET Core 6”, 2022
16. Svetlin Nakov & Co., „Fundamentals of computer programming with C#”, Telerik Software Academy, 2013
17. <https://docs.oracle.com/javase/8/docs/technotes/guides/language/> na dzień 31 stycznia 2025 roku
18. Tsitoara M., „Git I GitHub Kontrola wersji, zarządzanie projektami i zasady pracy zespołowej”, Helion, Gliwice 2022
19. <https://www.sqlite.org/about.html> na dzień 31 stycznia 2025 roku
20. Dewson R., „SQL Server wstęp dla programista Wydanie IV”, Helion, Gliwice 2016
21. https://en.wikipedia.org/wiki/JSON_Web_Token na dzień 31 stycznia 2025 roku
22. https://en.wikipedia.org/wiki/Cross-origin_resource_sharing na dzień 31 stycznia 2025 roku
23. <https://cyrekdigital.com/pl/baza-wiedzy/cross-site-scripting/> na dzień 31 stycznia 2025 roku
24. <https://learning.postman.com/docs/introduction/overview/> na dzień 31 stycznia 2025 roku

Spis ilustracji

Rys. 1 Makiet: Logowania i Rejestrowania użytkownika źródło: Opracowanie własne ..	13
Rys. 2 Makiet: Oferty (Strona główna) i element dropdown źródło: Opracowanie własne	13
Rys. 3 Diagram przypadków użycia: scenariusz dla użytkownika źródło: Opracowanie własne.....	14
Rys. 4 Diagram klasów: web-aplikacji TravelApp źródło: Opracowanie własne.....	16
Rys. 5 Diagram relacji encji - SQLite	16
Rys. 6 Diagram relacji encji – MSSQL źródło: Opracowanie własne	17
Rys. 7 Przykładowy rzut ekranu Komponentu TypeScript - Ten Komponent odp. za wyświetlenia publikacji.....	22
Rys. 8 Przykładowy rzut ekranu języka programowania TypeScript - Ten kod odp. za definicje interfejsu Post'a.....	24
Rys. 9 Przykładowy rzut ekranu porównania składni SASS, SCSS, CSS źródło: https://highload.tech/scss/	26
Rys. 10 Przykładowy rzut ekranu Kontrolera ASP .Net Core- Ten Kontroler odp. za wszystkie metody akcji związane z postami	29
Rys. 11 Przykładowy rzut ekranu Kontrolera ASP .Net Core – „UserController”	38
Rys. 12 Przykładowy rzut ekranu ASP .Net Core – „ExceptionMiddleware”	39
Rys. 13 Przykładowy rzut ekranu Angular – „text-input.component.html”	40
Rys. 14 <i>Strona logowania</i>	42
Rys. 15 <i>Strona ofert (Strona główna aplikacji)</i>	43
Rys. 16 Wynik obliczeń testu wydajnościowego k6 źródło: opracowanie własne	44
Rys. 17 Wynik obliczeń testu skalowalności k6 źródło: opracowanie własne.....	45
Rys. 18 Przykład poprawnie wykonanych poleceń dodawania certyfikatu SSL.....	53
Rys. 19 Przykład poprawnie podmienionej nazwy serwera lokalnego	53
Rys. 20 Przykładowa nowo utworzona baza danych w programie Microsoft Server	54
Rys. 21 Uruchomienie serwera backendowego.....	54
Rys. 22 Uruchomienie serwera frontendowego.....	55
Rys. 23 Wydanie: Strona „Lists” – strona, która odpowiada za polubienia postów	56
Rys. 24 Wydanie: Strona „Message” – która odpowiada za powiadomienia „Unread, Inbox, Outbox”.....	56
Rys. 25 Wydanie: przykład Outbox.....	57

Rys. 26 Wydanie: Strona „Share Post ” – odpowiada za dodawania postów źródło: opracowanie własne	57
Rys. 27 Wydanie: Strona „Profile/Posts” – odpowiada za wyświetlenia postów własnego profilu	57
Rys. 28 Wydanie: Strona „Profile/About” – odpowiada za wyświetleniu dokładnej informacji o użytkowniku (Jego zdjęcia, i krótką informacją o nim)	58
Rys. 29 Wydanie: Strona „Edit Profile” – odpowiada za redagowania profilu (zdjęć oraz informacje o nim)	58
Rys. 30 Wydanie: Strona „Edit post” – odpowiada za redagowania postu	59
Rys. 31 Wydanie: Strona „Member/[user’s] posts”	59
Rys. 32 Wydanie: Strona „Member/About [user]”	60
Rys. 33 Wydanie: Strona „Member/Messages”	60
Rys. 34 Wydanie: Strona „Admin” – dostęp dla role Admin i Moderator	60

Spis tabel

Tabela 1. Porównanie Angular'a i React'a źródło: Opracowanie własne	21
Tabela 2 Porównanie TypeScript'a i JavaScript'a.....	23
Tabela 3 Porównanie CSS'a, SASS'a i SCSS'a.....	27
Tabela 4 Porównanie ASP.Net Core'a i Spring'a	28
Tabela 5 Porównanie C#`a i Java`a	31
Tabela 6 Porównanie GitHub`a i GitLab`a.....	32
Tabela 7 Porównanie SQLite`a i MSSQL`a	34
Tabela 8 Porównanie HTTPS`a i HTTP`a.....	35

Aneks

Instrukcja instalacji aplikacji lokalnie

1. Aby móc uruchomić aplikację lokalnie, należy najpierw sklonować repozytorium z GitHuba, korzystając z poniższego linka.

<https://github.com/RuslanPidhainyi/EW-TravelApp-.Net8-Angular17.git>

2. W celu uruchomienia serwera ASP.NET Core należy zainstalować pakiety .NET 8 SDK, znajdujące się pod poniższym linkiem.

<https://dotnet.microsoft.com/en-us/download/dotnet/8.0>

3. Aby dodać certyfikat HTTPS aplikacji do listy zaufanych certyfikatów systemowych należy w terminalu przejść do katalogu API i wykonać poniższe polecenia:

„dotnet dev-certs https --clean ”

“ dotnet dev-certs https --trust ”

4. Do poprawnego uruchomienia aplikacji należy będzie dokonać migracji. Będzie można tego dokonać poprzez narzędzie Entity Framework Core Command-line Tools, które można zainstalować poniższym poleceniem w terminalu.

„dotnet tool install --global dotnet-ef ”

5. Wymagana jest instalacja Node.js, aby móc korzystać z narzędzia npm.

<https://nodejs.org/en/download>

6. Gdy korzystanie z narzędzia npm jest już możliwe, należy zainstalować Angulara za pomocą polecenia:

„npm install -g @angular/cli ”

W przypadku wystąpienia błędów, w celu pozbycia się ich można spróbować wykonać poniższe polecenie:

„Set-ExecutionPolicy -Scope Process -ExecutionPolicy Bypass ”

I spróbować ponownie:

„npm install ”

7. Kolejnym krokiem będzie dodanie certyfikatu SSL. W wierszu poleceń (otworzonym jako administrator), w katalogu frontendowej części aplikacji należy wykonać wymienianej niżej polecenia:

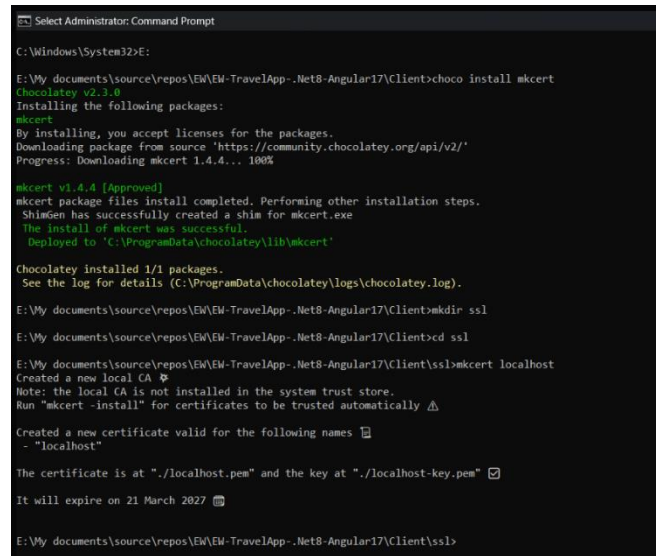
„choco install mkcert ”

„mkdir ssl ”

„cd ssl ”

„mkcert localhost”

Przykład poprawnie wykonanych poleceń dodawania certyfikatu SSL:



```
Select Administrator: Command Prompt
C:\Windows\System32>E:
E:\My documents\source\repos\EW\TravelApp-.Net8-Angular17\Client>choco install mkcert
Chocolatey v2.3.0
Installing the following packages:
mkcert
By installing, you accept licenses for the packages.
Downloading package from source 'https://community.chocolatey.org/api/v2/'
Progress: Downloading mkcert 1.4.4... 100%

mkcert v1.4.4 [Approved]
mkcert package files install completed. Performing other installation steps.
ShimGen has successfully created a shim for mkcert.exe
The install of mkcert was successful.
Deployed to 'C:\ProgramData\chocolatey\lib\mkcert'

Chocolatey installed 1/1 packages.
See the log for details (C:\ProgramData\chocolatey\logs\chocolatey.log).
E:\My documents\source\repos\EW\TravelApp-.Net8-Angular17\Client>mkdir ssl
E:\My documents\source\repos\EW\TravelApp-.Net8-Angular17\Client>cd ssl
E:\My documents\source\repos\EW\TravelApp-.Net8-Angular17\Client\ssl>mkcert localhost
Created a new local CA
Note: the local CA is not installed in the system trust store.
Run "mkcert -install" for certificates to be trusted automatically
Created a new certificate valid for the following names
- "localhost"
The certificate is at ".\localhost.pem" and the key at ".\localhost-key.pem"
It will expire on 21 March 2027
E:\My documents\source\repos\EW\TravelApp-.Net8-Angular17\Client\ssl>
```

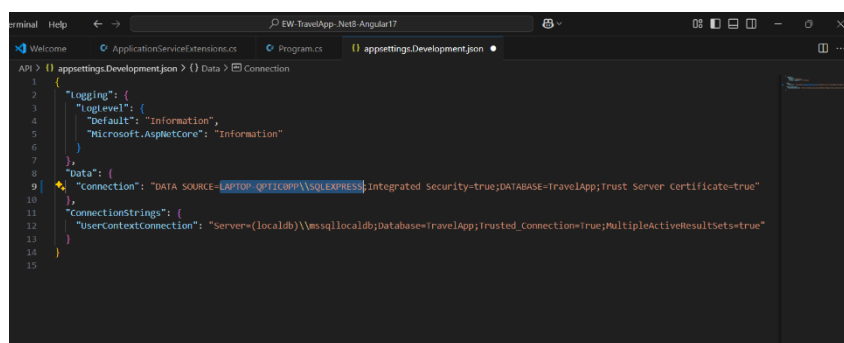
Rys. 18 Przykład poprawnie wykonanych poleceń dodawania certyfikatu SSL
źródło: opracowanie własne

8. Znajdując się w wierszu poleceń należy również zainstalować mkcert poniższą komendą:

„mkcert -install”

Po poprawnie przeprowadzonej instalacji uruchomić ponownie komputer.

9. W celu skonfigurowania bazy danych MSSQL, należy podać nazwę lokalnego serwera w pliku „appsettings.Development.json”. Na poniższym rysunku znajduje się zrzut ekranu przykładowej podmienionej nazwy serwera.



```
terminal  Help  EW-TravelApp-.Net8-Angular17
Welcome  ApplicationServiceExtension.cs  Program.cs  appsettings.Development.json
API > | appsettings.Development.json > | Data > | Connection
1
2
3  "Logging": {
4    "LogLevel": {
5      "Default": "Information",
6      "Microsoft.AspNetCore": "Information"
7    }
8  },
9  "Data": {
10   "Connection": "DATA SOURCE=LAPTOP-Q7T1C8PP\\SQLSERVER;Integrated Security=true;DATABASE=TravelApp;Trust Server Certificate=true"
11 },
12 "ConnectionStrings": {
13   "UserContextConnection": "Server=(localdb)\\mssqllocaldb;Database=TravelApp;Trusted_Connection=true;MultipleActiveResultsets=true"
14 }
15 }
```

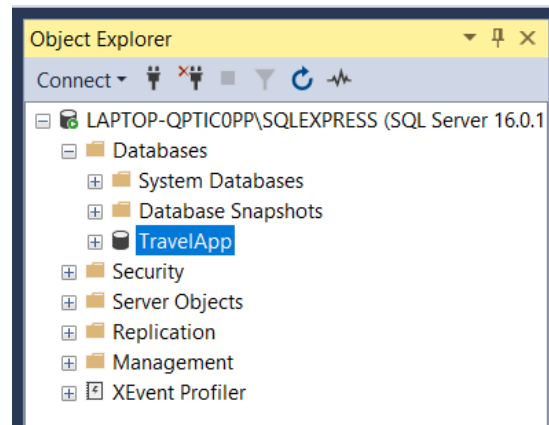
Rys. 19 Przykład poprawnie podmienionej nazwy serwera lokalnego
źródło: opracowanie własne

10. Stworzenie migracji można dokonać poprzez polecenie:

„dotnet ef migrations add InitialCreate --output-dir Data/Migrations”

Jednak dopiero polecenie „dotnet ef database update” utworzy bazę danych.

Na poniższym rysunku znajduje się zrzut ekranu z przykładowo nowo utworzoną bazą danych.



Rys. 20 Przykładowa nowo utworzona baza danych w programie Microsoft Server
źródło: opracowanie własne

11. Po wykonaniu poprzednich kroków aplikacja jest gotowa do uruchomienia

- Po stronie backendu: „dotnet run”
- Po stronie frontendu: „ng serve”

Na poniższych rysunkach znajdują się zrzuty ekranu uruchomienia serwerów:

frontendowego i backendowego.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TEST RESULTS  TERMINAL  PORTS  GITLENS  QUERY RESULTS (PREVIEW)  NUGET  COMMENTS

PS E:\My documents\source\repos\EW\EW-TravelApp-.Net8-Angular17> cd api
PS E:\My documents\source\repos\EW\EW-TravelApp-.Net8-Angular17\api> dotnet run
Using launch settings from E:\My documents\source\repos\EW\EW-TravelApp-.Net8-Angular17\api\Properties\launchSettings.json...
Building...
E:\My documents\source\repos\EW\EW-TravelApp-.Net8-Angular17\api\API.csproj Restore (5.3)
info: Microsoft.EntityFrameworkCore.Database.Command[20101]
      Executed DbCommand (20ms) [Parameters=[], CommandType='Text', CommandTimeout='30']
      SELECT 1
info: Microsoft.EntityFrameworkCore.Database.Command[20101]
      Executed DbCommand (16ms) [Parameters=[], CommandType='Text', CommandTimeout='30']
      SELECT OBJECT_ID(N'[_EFMigrationsHistory]');
info: Microsoft.EntityFrameworkCore.Database.Command[20101]
      Executed DbCommand (1ms) [Parameters=[], CommandType='Text', CommandTimeout='30']
      SELECT 1
info: Microsoft.EntityFrameworkCore.Database.Command[20101]
      Executed DbCommand (1ms) [Parameters=[], CommandType='Text', CommandTimeout='30']
      SELECT OBJECT_ID(N'[_EFMigrationsHistory]');
ate.
info: Microsoft.EntityFrameworkCore.Database.Command[20101]
      Executed DbCommand (3ms) [Parameters=[], CommandType='Text', CommandTimeout='30']
      DELETE FROM [Connections]
info: Microsoft.EntityFrameworkCore.Database.Command[20101]
      Executed DbCommand (3ms) [Parameters=[], CommandType='Text', CommandTimeout='30']
      SELECT CASE
        WHEN EXISTS (
          SELECT 1
            FROM [AspNetUsers] AS [a]) THEN CAST(1 AS bit)
        ELSE CAST(0 AS bit)
      END
info: Microsoft.AspNetCore.DataProtection.KeyManagement.XmlKeyManager[63]
      User profile is available. Using 'C:\Users\Lenovo\AppData\Local\ASP.NET\DataProtection-Keys' as key repository and Windows DPAPI to encrypt keys at rest.
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://localhost:5000
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: https://localhost:5001
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
      Content root path: E:\My documents\source\repos\EW\EW-TravelApp-.Net8-Angular17\api
```

Rys. 21 Uruchomienie serwera backendowego
źródło: opracowanie własne

```

PROBLEMS OUTPUT DEBUG CONSOLE TEST RESULTS TERMINAL PORTS GITLENS QUERY RESULTS (PREVIEW) NUGET COMMENTS

Watch mode enabled. Watching for file changesPS E:\My documents\source\repos\EW\EW-TravelApp-.Net8-Angular17> cd client
PS E:\My documents\source\repos\EW\EW-TravelApp-.Net8-Angular17\client> ng serve
Initial chunk files | Names | Raw size
styles.css | styles | 608.04 kB |
main.js | main | 322.21 kB |
polyfills.js | polyfills | 88.09 kB |
| Initial total | 1018.34 kB |

Application bundle generation complete. [4.857 seconds]
| Initial total | 1018.34 kB |

Application bundle generation complete. [4.857 seconds]

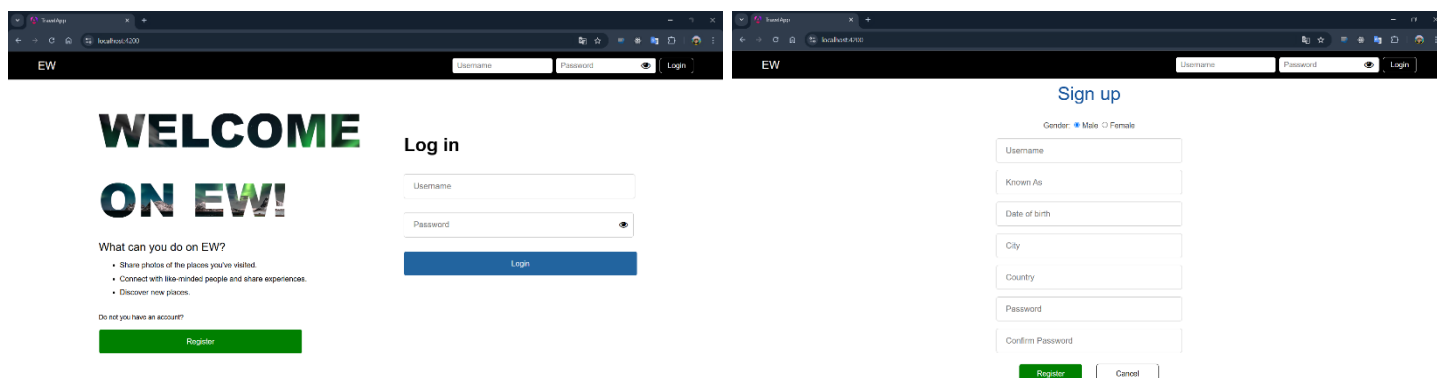
Watch mode enabled. Watching for file changes...
→ Local: https://localhost:4200/
→ press h + enter to show help

```

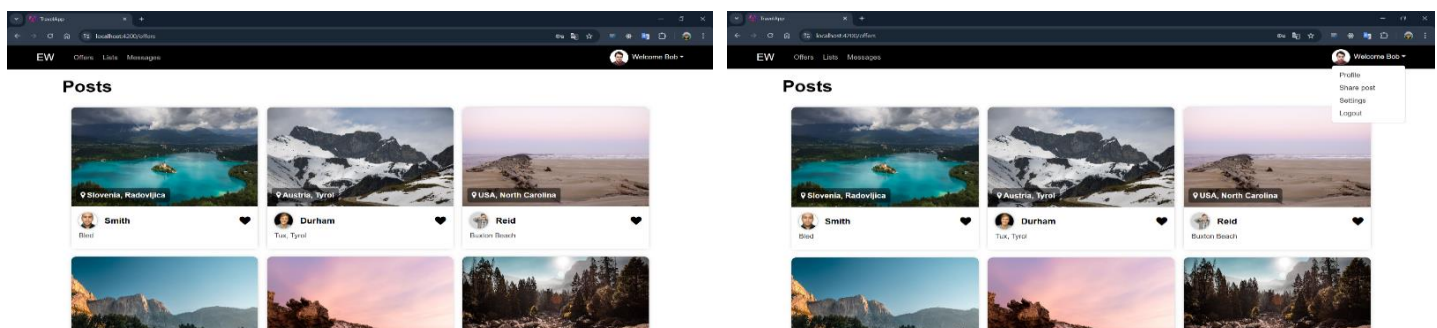
Rys. 22 Uruchomienie serwera frontendowego
źródło: opracowanie własne

12. Po uruchomieniu serwerów należy w oknie przeglądarki wejść na URL <https://localhost:4200/>, po czym zostanie uruchomiona aplikacja.

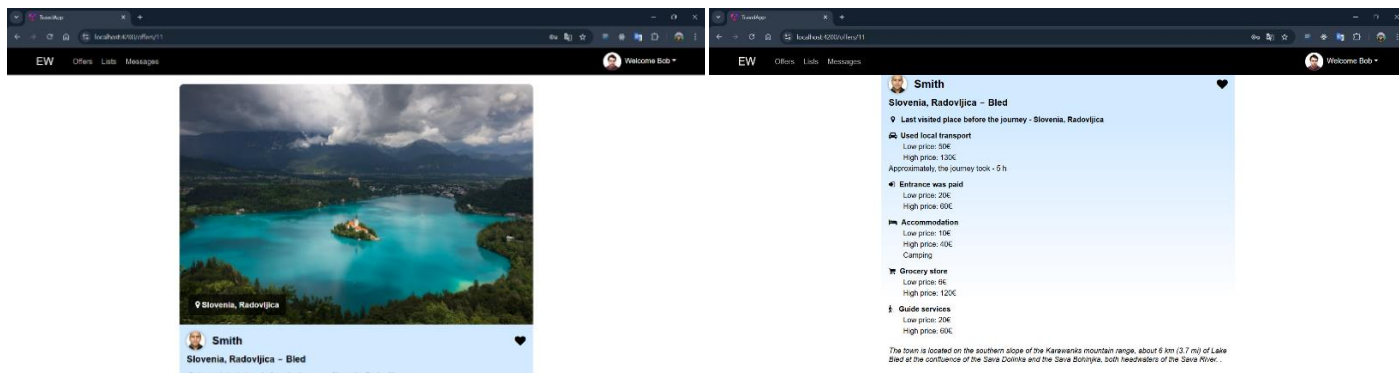
Screeny z web-aplikacji



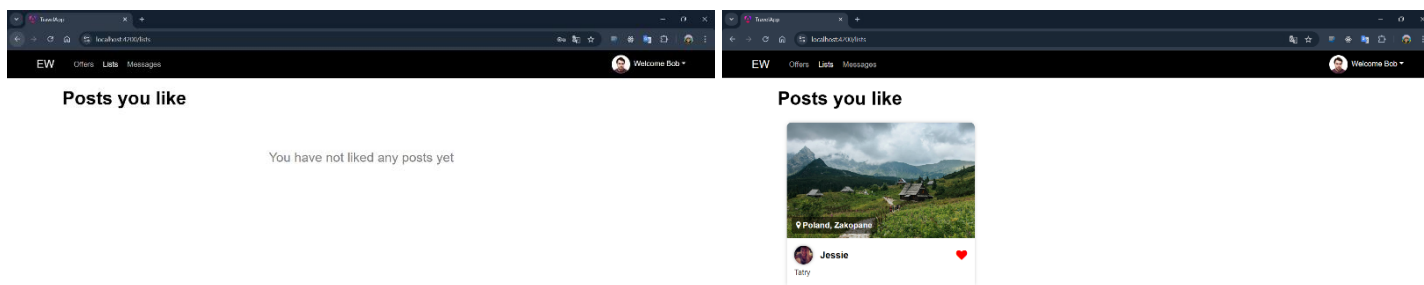
Rys. 18 Wydania: Logowania i Rejestrowania użytkownika
źródło: opracowanie własne



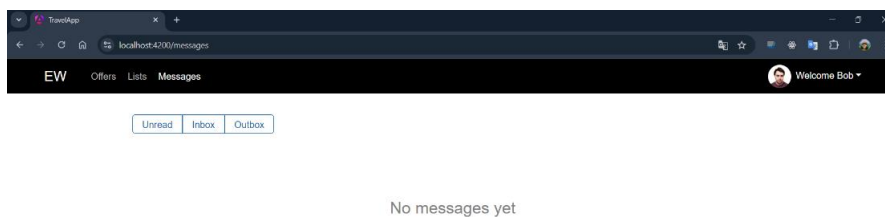
Rys. 19 Wydania: Strona „Offers” (Strona główna) i element dropdown
źródło: opracowanie własne



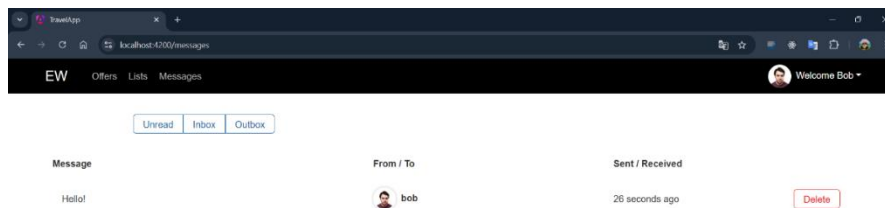
Rys. 20 Wydanie: Szczegóły postu
źródło: opracowanie własne



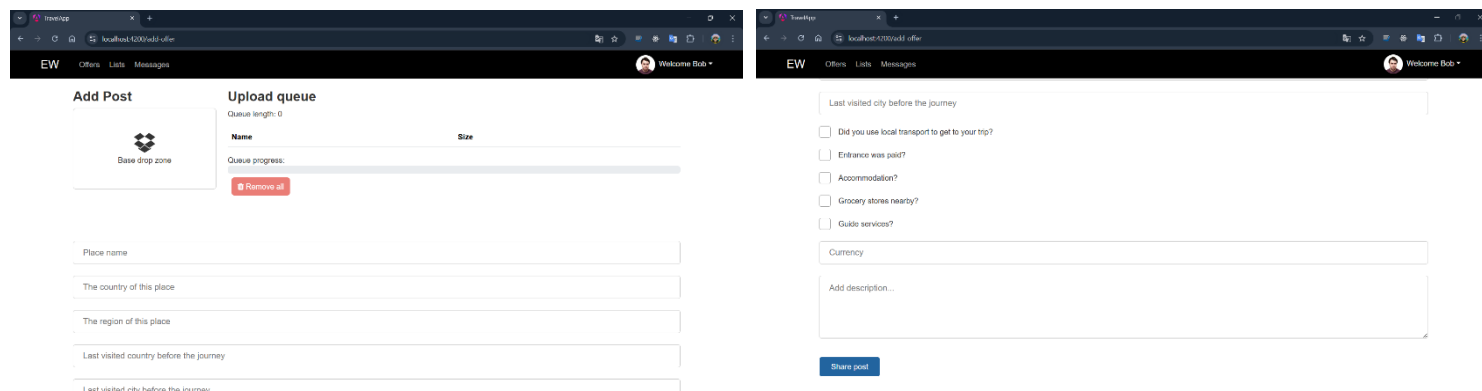
Rys. 23 Wydanie: Strona „Lists” – strona, która odpowiada za polubienia postów
źródło: opracowanie własne



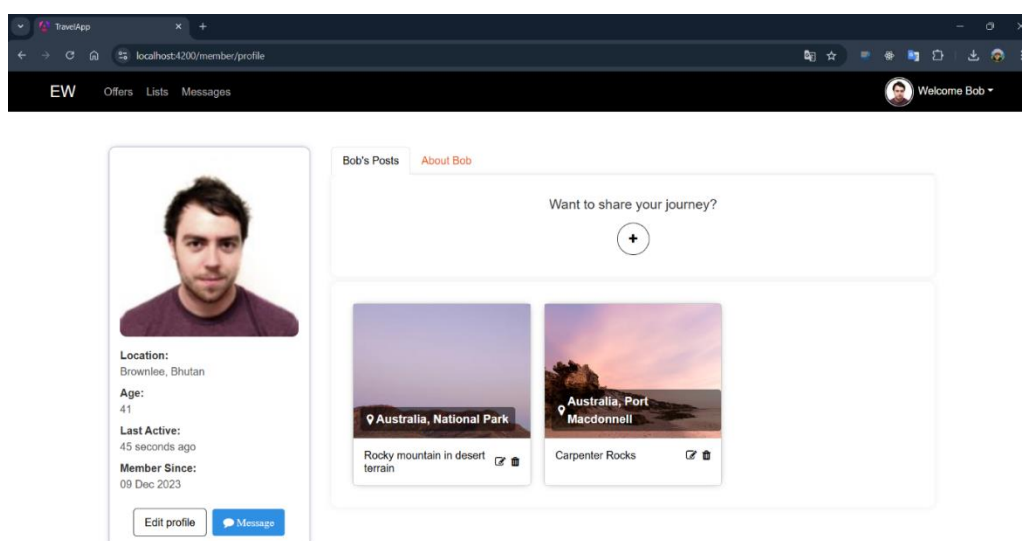
Rys. 24 Wydanie: Strona „Message” – która odpowiada za powiadomienia „Unread, Inbox, Outbox”
źródło: opracowanie własne



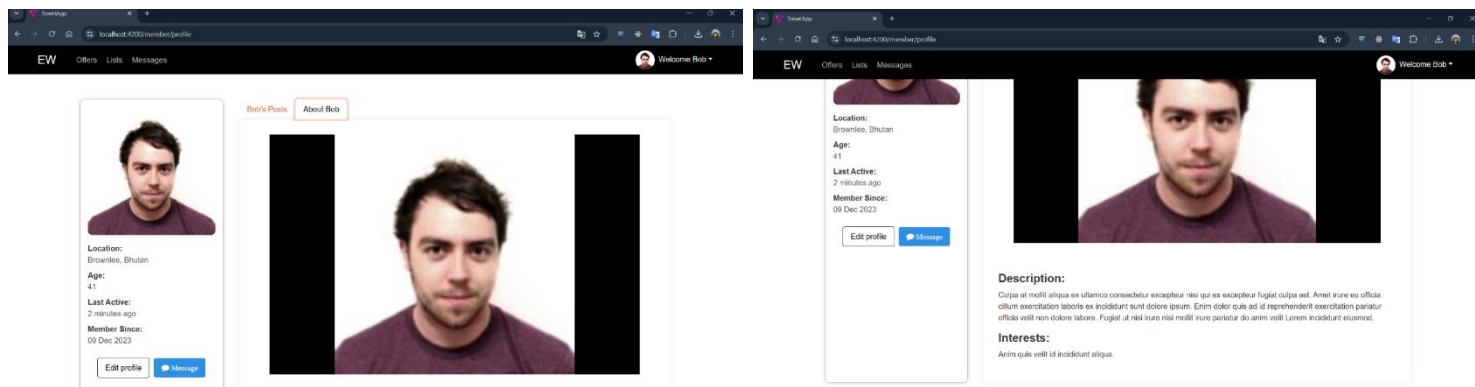
Rys. 25 Wydanie: przykład Outbox
źródło: opracowanie własne



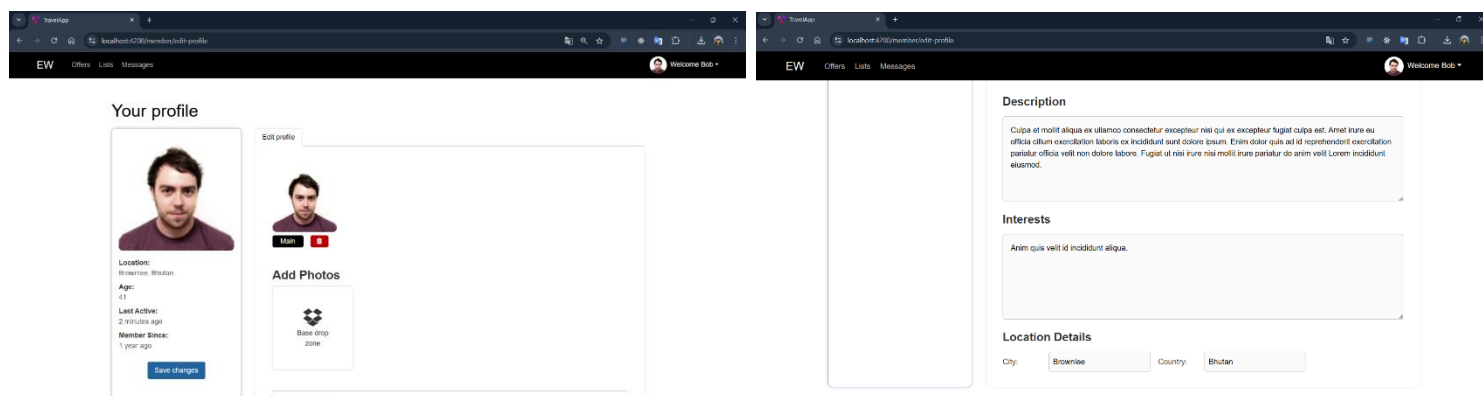
Rys. 26 Wydanie: Strona „Share Post” – odpowiada za dodawania postów
źródło: opracowanie własne



Rys. 27 Wydanie: Strona „Profile/Posts” – odpowiada za wyświetlenia postów własnego profilu
źródło: opracowanie własne

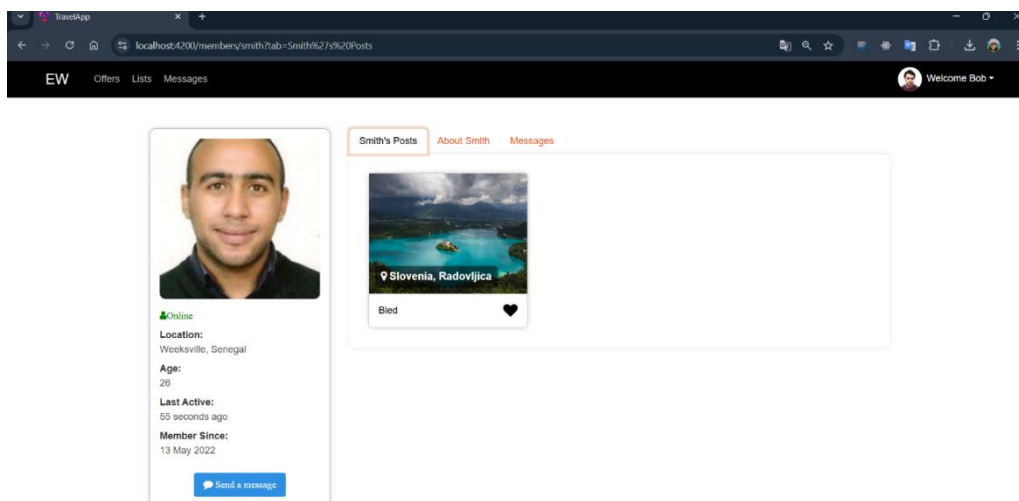


Rys. 28 Wydanie: Strona „Profile/About” – odpowiada za wyświetleniu dokładnej informacji o użytkowniku (Jego zdjęcia, i krótką informacją o nim)
źródło: opracowanie własne

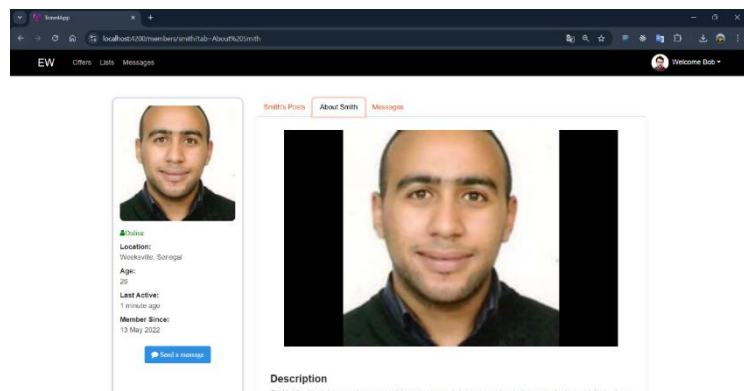


Rys. 29 Wydanie: Strona „Edit Profile” – odpowiada za redagowania profilu (zdjęć oraz informacje o nim)
źródło: opracowanie własne

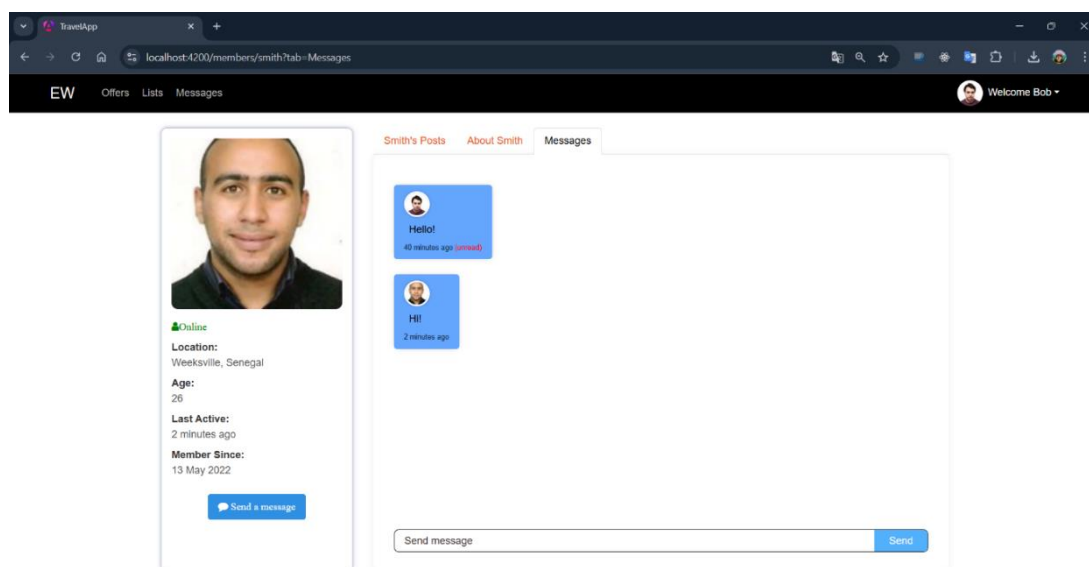
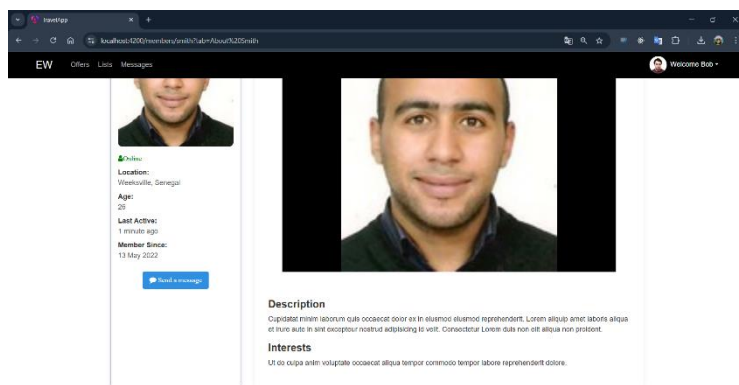
Rys. 30 Wydanie: Strona „Edit post” – odpowiada za redagowania postu
źródło: opracowanie własne



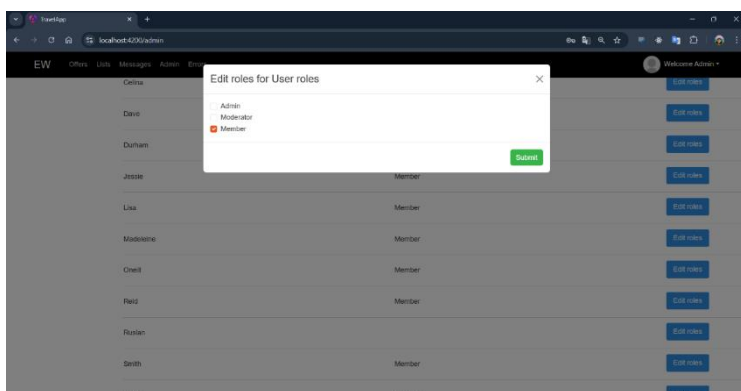
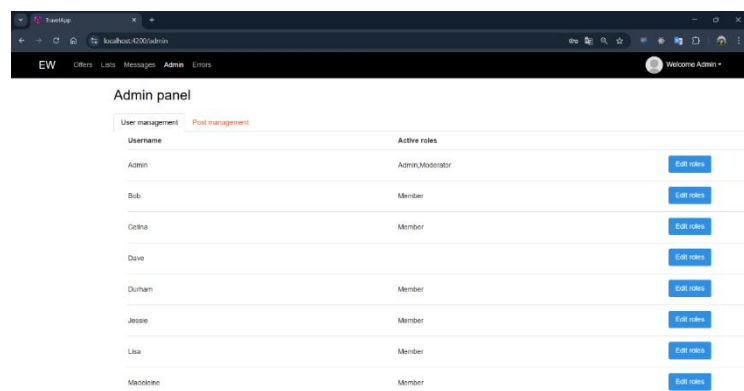
Rys. 31 Wydanie: Strona „Member/[user's] posts”
źródło: opracowanie własne



Rys. 32 Wydanie: Strona „Member/About [user]”
źródło: opracowanie własne



Rys. 33 Wydanie: Strona „Member/Messages”
źródło: opracowanie własne



Rys. 34 Wydanie: Strona „Admin” – dostęp dla role Admin i Moderator
źródło: opracowanie własne