

Collegium Witelona Uczelnia Państwowa



Wydział Nauk Technicznych i Ekonomicznych

Kierunek: INFORMATYKA

specjalność: Projektowanie aplikacji mobilnych i internetowych

Sprawozdanie

Platforma Internetowa "AMORA" w. 1.0.0

Skład zespołu projektowego:

Roman Rusinek - nr albumu: 43257
Mateusz Tęcza - nr albumu: 43263
Jakub Trznadel - nr albumu: 42765

Projekt programistyczny

prowadzony pod kierunkiem
mgr inż. Krzysztof Rewak

Legnica 2024

Streszczenie

Projekt randkowej platformy internetowej, kojarzącej ze sobą osoby przeciwnej płci. Wersja 1.0.0 Projektu "Amora" obsługuje języki: angielski i polski.

Abstract

A project for an online dating platform that matches people of the opposite sex with each other. Version 1.0.0 of Project "Amora" supports English and Polish languages.

Spis treści

1	Wstęp	1
1.1	Podział prac w zespole	1
2	Opis funkcjonalny systemu	2
3	Opis technologiczny	8
3.1	Microsoft ASP.Net CORE wer. 8.0 - zestawienie modułów wykorzystanych w projekcie	10
3.2	Zarządzanie zależnościami	11
3.3	Mapowanie	12
3.4	Technologie do budowy warstwy interfejsu użytkownika - frontend . .	12
3.5	Cache	12
4	Realizacja wymagań projektowych	13
4.1	Wymagania projektowe i stopień realizacji	14
5	Instrukcja lokalnego i zdalnego uruchomienia	17
5.1	Instrukcja lokalnego uruchomienia systemu	17
6	Rozmieszczenie dokumentacji i plików projektu na git-hubie	19
7	Wnioski projektowe	20

Rozdział 1

Wstęp

Zadaniem projektu programistycznego "Amora" jest zbudowanie platformy internetowej, kojarzącej osoby przeciwnej płci w pary. Projekt jest realizowany zgodnie z filozofią MVP - Minimum Viable Product, czyli staramy się zaprojektować system o minimalnej użyteczności ale o cechach, które pozwolą już go testować, oceniać przez zainteresowanego tym produktem użytkownika. Po zebraniu opinii od użytkowników, można projekt rozbudowywać rozszerzając istniejące funkcjonalności albo poprzez dodawanie całkiem innych.

1.1 Podział prac w zespole

Podział prac w zespole:

- Jakub Trznadel - frontend (HTML, framework CSS)
- Mateusz Tęcza - MVC
- Roman Rusinek - opracowanie dokumentacji i bazy danych

Rozdział 2

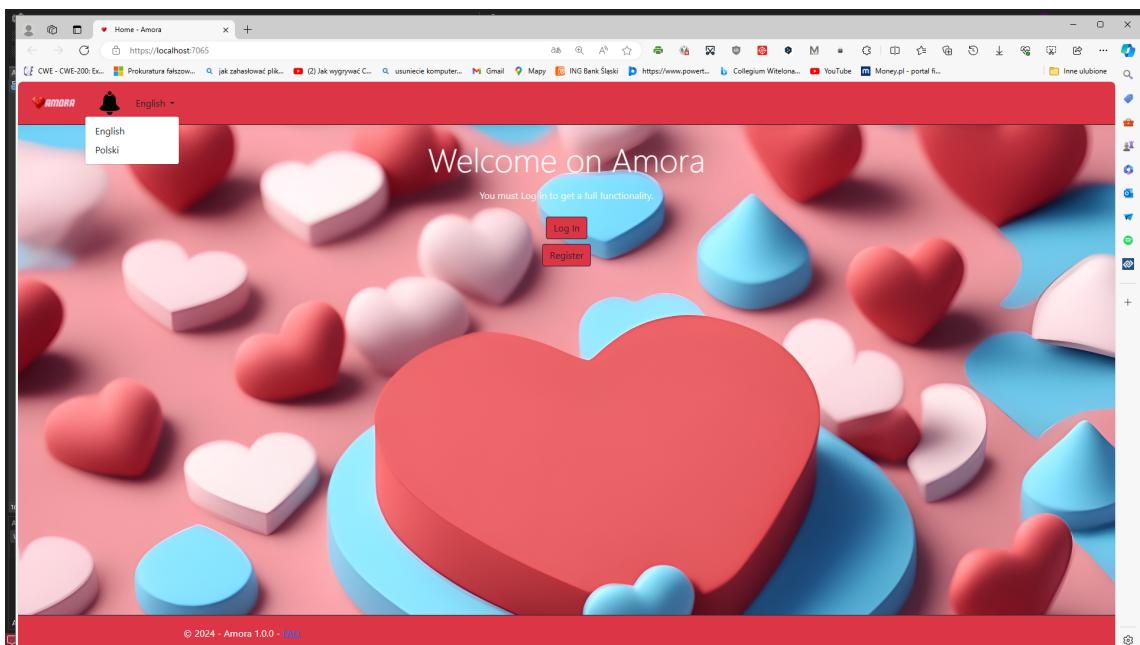
Opis funkcjonalny systemu

Cel: opracowanie i wdrożenie aplikacji randkowej "AMORA" ułatwiającej kojarzenie w pary.

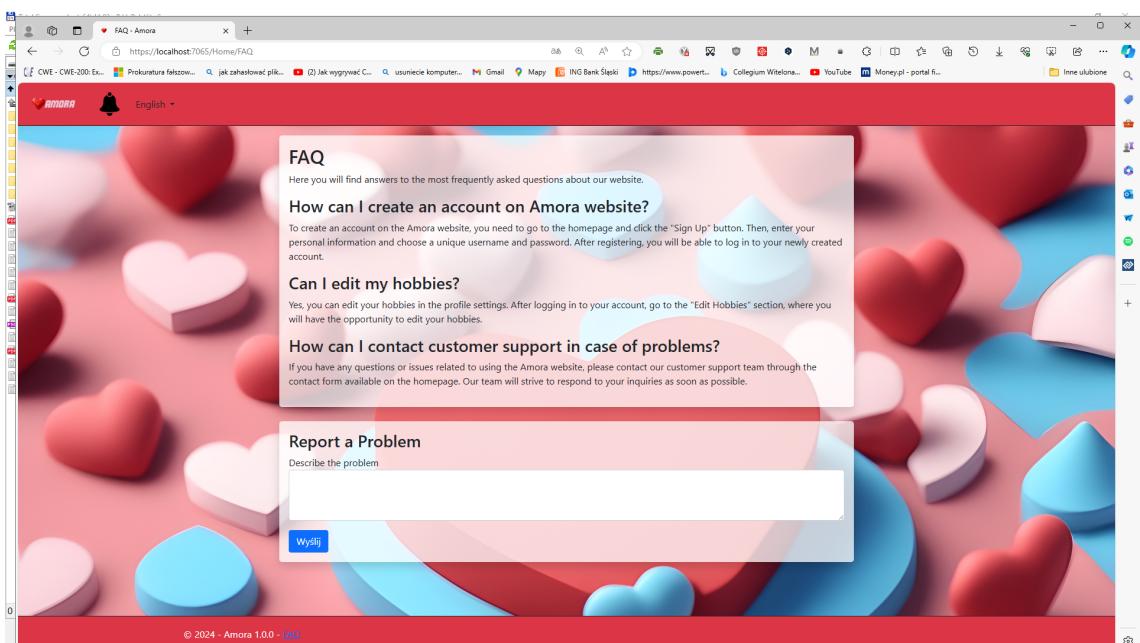
Cel: opracowanie i wdrożenie aplikacji randkowej "AMORA" ułatwiającej kojarzenie w pary.

Użytkownicy platformy mogą:

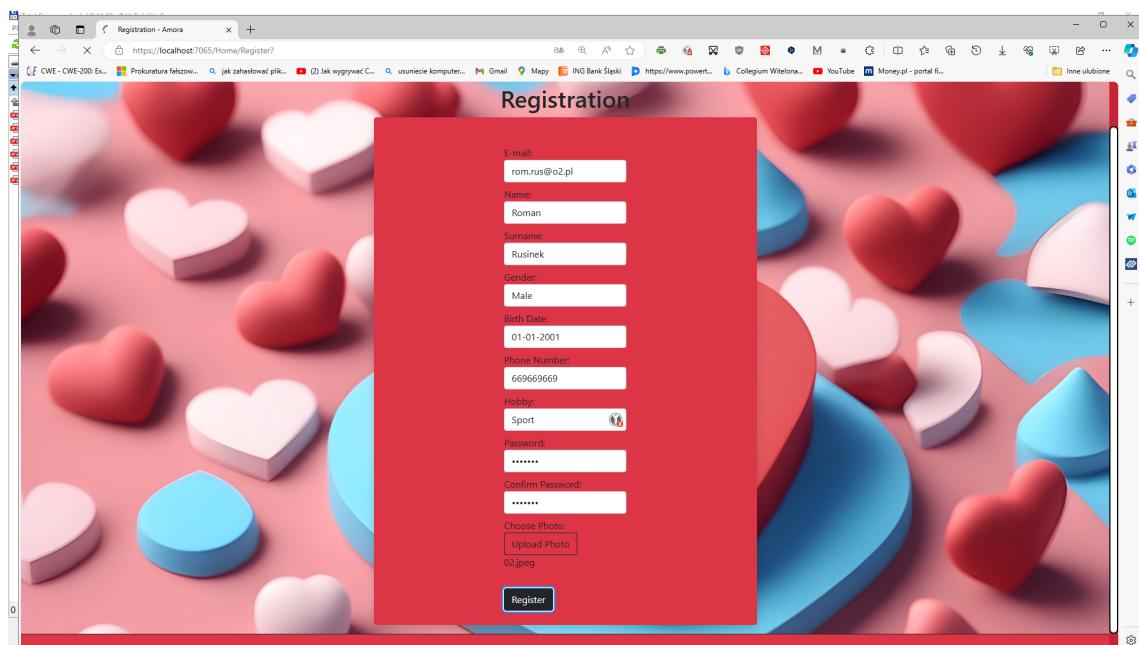
- wybrać język platformy: angielski albo polski
- zarejestrować się
- utworzyć profil
- szukać dopasowań (wyborów potencjalnego partnera) - "match'y"z innymi użytkownikami platformy
- przeglądać poprzednie dopasowania - "match'e" w zakładce "Previous Matches"
- odnaleźć numer telefonu swojej pary i poprzez niego nawiązać kontakt. Do pomocy w rozmowie mogą wykorzystać kawały, które znajdują się w tej samej zakładce.
- zapoznać się z odpowiedziami na najczęściej spotykane problemy, zapytania - w sekcji FAQ(Frequently Asked Questions). Sekcja ta jest dostępna również dla niezarejestrowanych użytkowników.



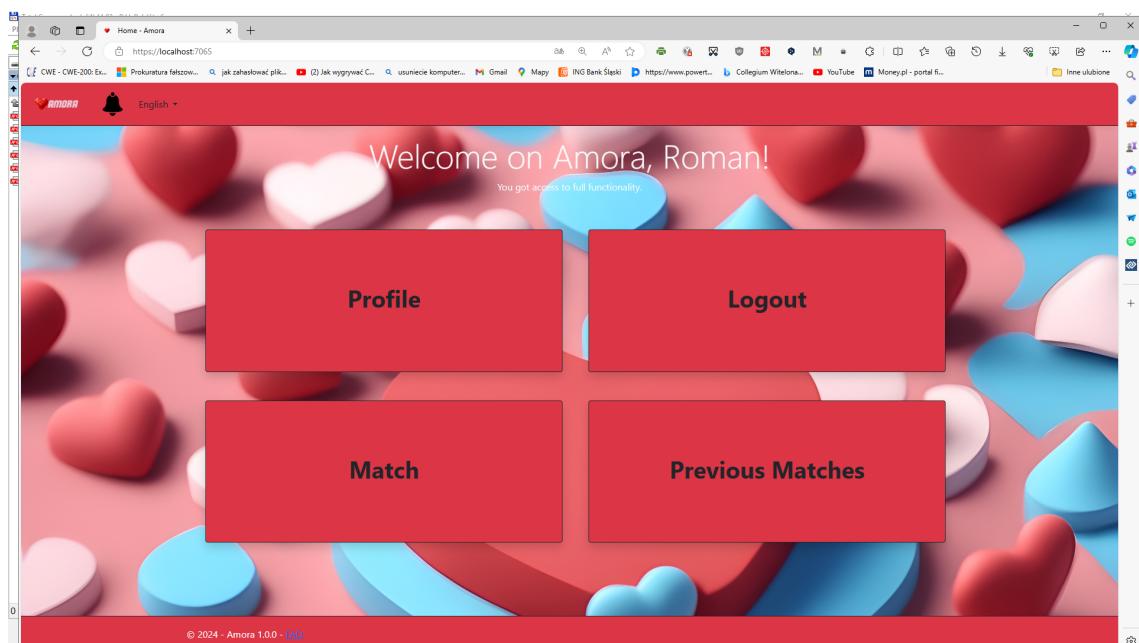
Rysunek 2.1
Strona początkowa - wybór języka



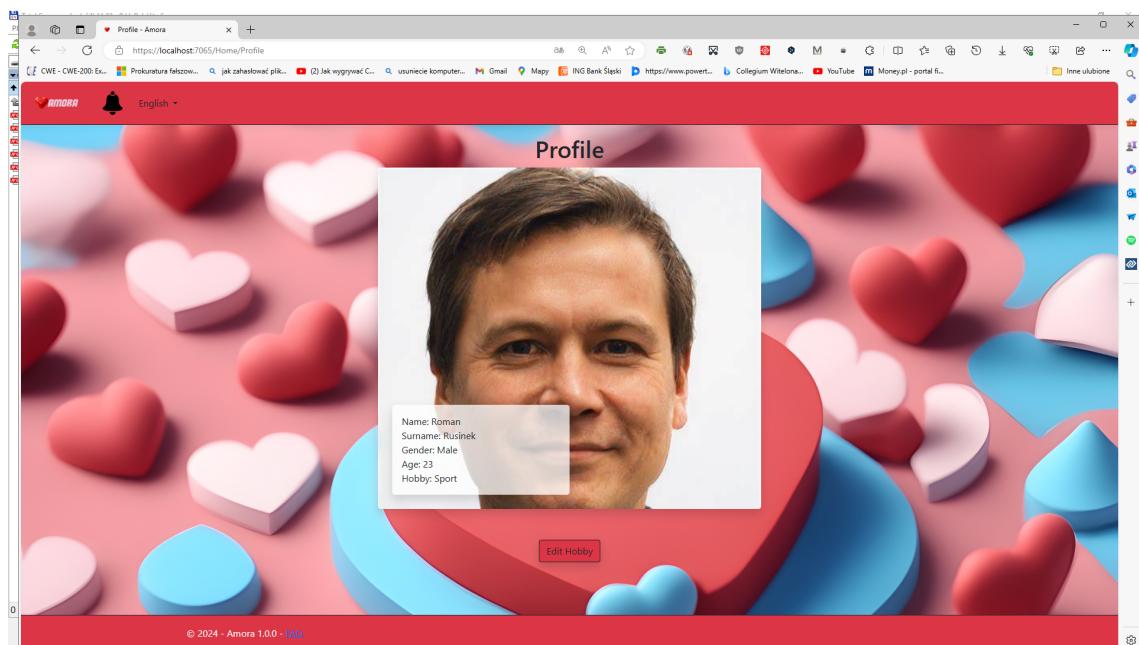
Rysunek 2.2
Strona początkowa - Przegląd FAQ



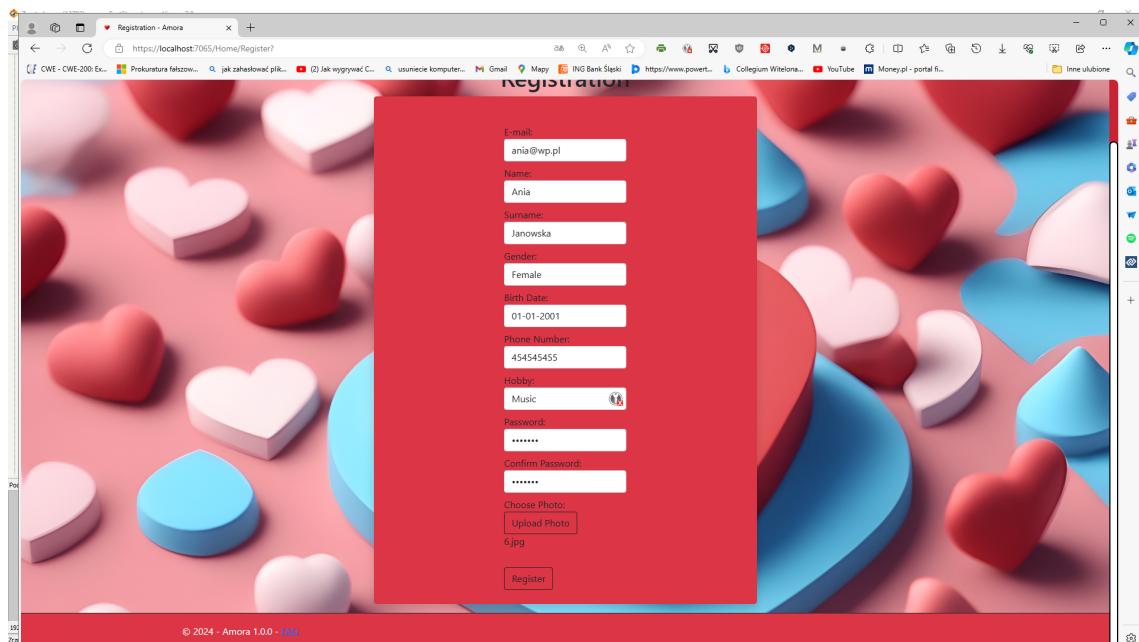
Rysunek 2.3
Rejestracja



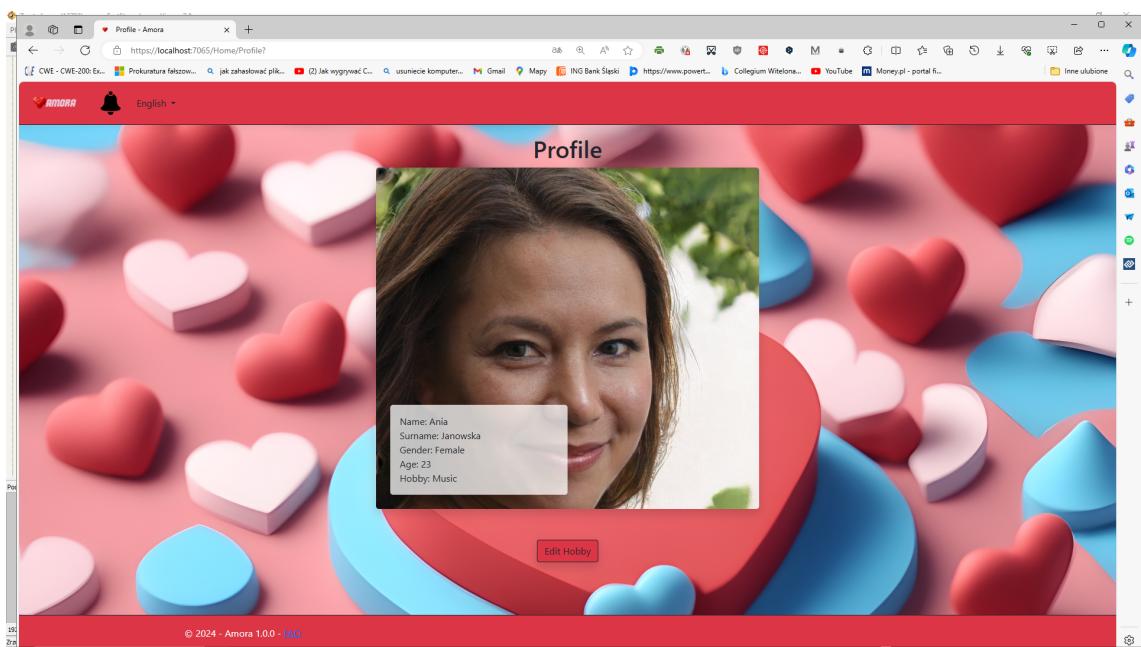
Rysunek 2.4
Opcje użytkownika



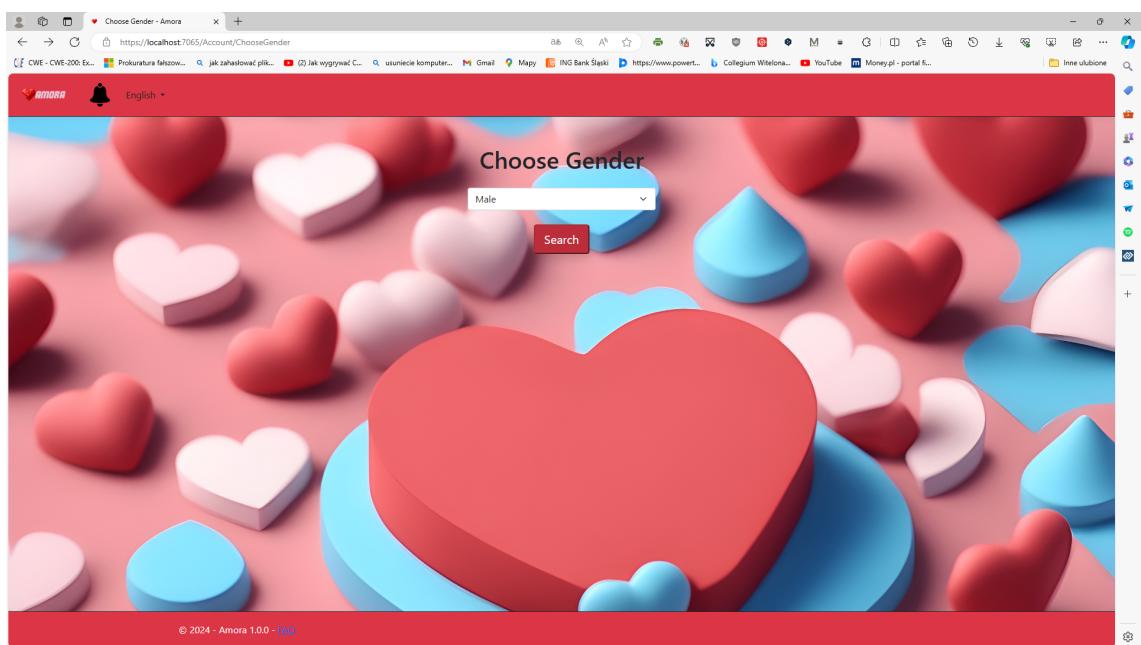
Rysunek 2.5
Zdjęcie profilowe użytkownika Roman



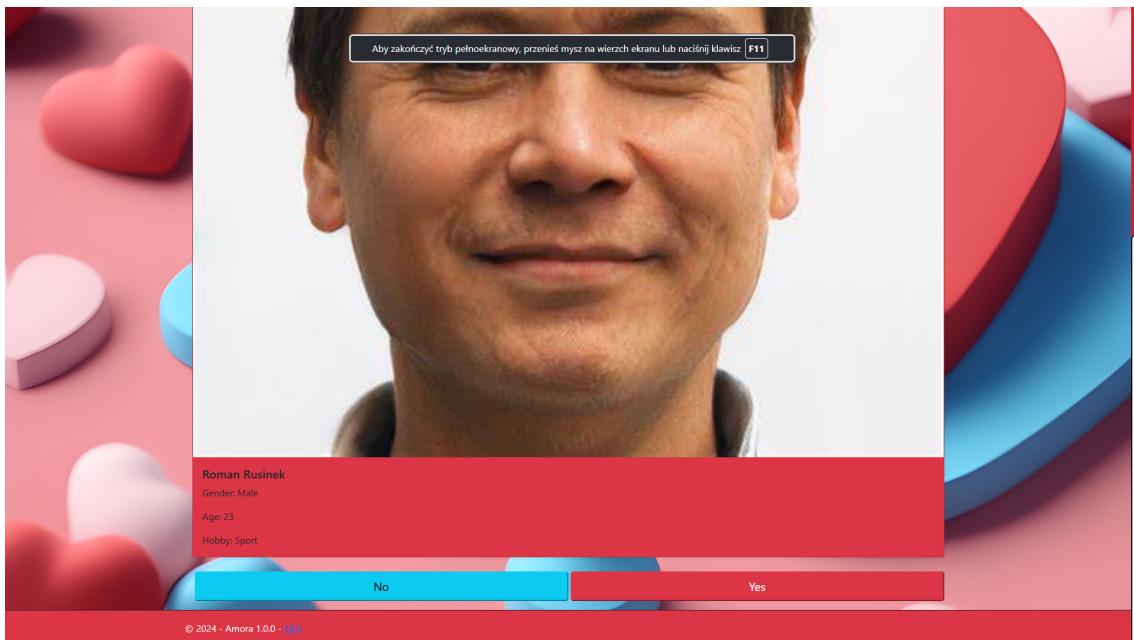
Rysunek 2.6
Rejestracja użytkownika Ania



Rysunek 2.7
Profil użytkownika Ania

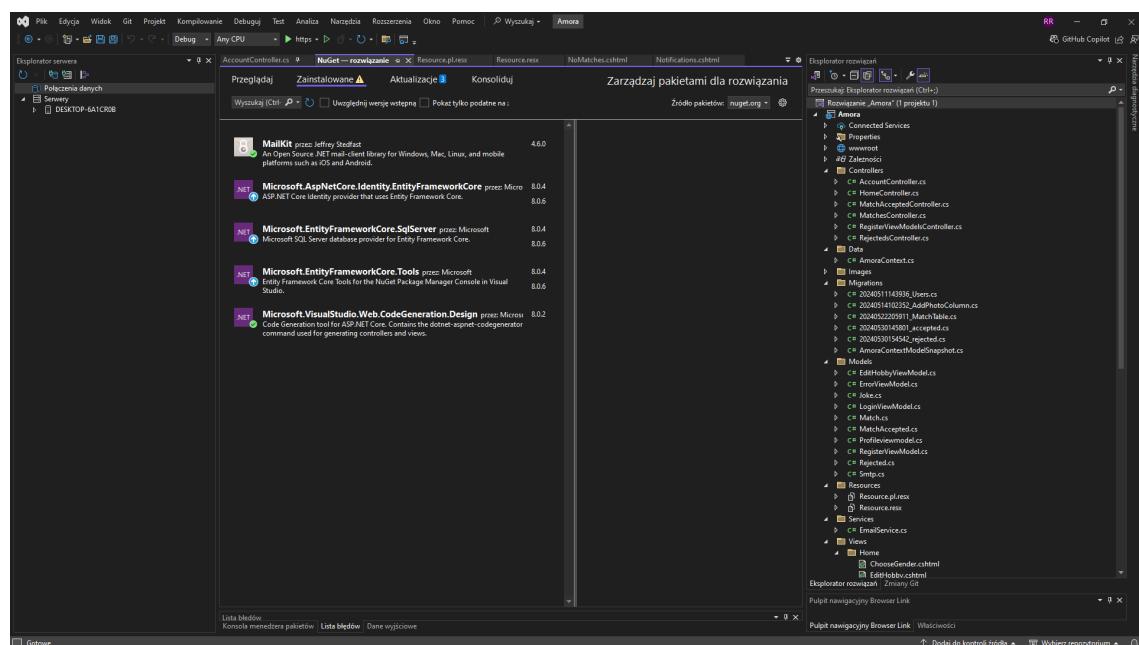


Rysunek 2.8
Wyszukiwanie przez Anię partnerów



Rysunek 2.9

Wynik wyszukiwania Ani-znaleziono jedną propozycję znajomości z użytkownikiem Romanem



Rysunek 2.10
Zależności projektu

Rozdział 3

Opis technologiczny

	MailKit przez: Jeffrey Stedfast An Open Source .NET mail-client library for Windows, Mac, Linux, and mobile platforms such as iOS and Android.	4.6.0
	Microsoft.AspNetCore.Identity.EntityFrameworkCore przez: Microsoft ASP.NET Core Identity provider that uses Entity Framework Core.	8.0.4 8.0.6
	Microsoft.EntityFrameworkCore.SqlServer przez: Microsoft Microsoft SQL Server database provider for Entity Framework Core.	8.0.4 8.0.6
	Microsoft.EntityFrameworkCore.Tools przez: Microsoft Entity Framework Core Tools for the NuGet Package Manager Console in Visual Studio.	8.0.4 8.0.6
	Microsoft.VisualStudio.Web.CodeGeneration.Design przez: Microsoft Code Generation tool for ASP.NET Core. Contains the dotnet-aspnet-codegenerator command used for generating controllers and views.	8.0.2

Rysunek 3.1

Wymagane zależności w VSCommunity 2002

Projekt zrealizowany w oparciu o:

- relacyjną bazę danych msSQL
- framework Microsoft.ASP.NetCore 8.0.0.0 zawierający pakiety:
MicrosoftASPNetCore.Identify.EntityFrameworkCore ver. 8.0.4
Microsoft.EntityFrameworkCore.Tools 8.0.4
- framework do bazy - Microsoft.Entity.FrameworkCore.SQLServer 8.0.4
- framework CSS -bootstrap ver. 5.1.0
- biblioteka kliencka do poczty - MailKit 4.6.0

Wymagany system operacyjny w procesie projektowym, to Windows 10 lub 11 z zainstalowanym Microsoft ASP.NetCORE w wersji 8.0

Środowisko programistyczne oparliśmy na Visual Studio Community 2022 z skonfigurowanym menadżerem zależności NuGet. VSC pozwala obsługiwać język C#, pozwala

na integrację z frameworkami i bazą danych, oferuje wszechstronne wsparcie dla języka C# poprzez rozszerzenie C# Dev Kit. Pozwala na efektywną pracę z projektami .NET, w tym ASP.NET Core, Blazor czy aplikacjami konsolowymi. Aby zintegrować VS Code z frameworkiem, należy zainstalować odpowiednie rozszerzenia i skonfigurować środowisko pod kątem wymagań projektu.

3.1 Microsoft ASP.Net CORE ver. 8.0 - zestawienie modułów wykorzystanych w projekcie

System.Runtime 8.0.0.0

Microsoft.Extensions.Options 8.0.0.0

Microsoft.Extensions.Options.ConfigurationExtensions 8.0.0.0

Microsoft.AspNetCore.HttpOverrides 8.0.0.0

Microsoft.Extensions.Configuration.Abstractions 8.0.0.0

Microsoft.AspNetCore.Routing 8.0.0.0

Microsoft.Extensions.DependencyInjection 8.0.0.0 **wstrzykiwanie zależności**

Microsoft.Extensions.DependencyInjection.Abstractions 8.0.0.0

Microsoft.AspNetCore.Http 8.0.0.0

Microsoft.AspNetCore.Http.Abstractions 8.0.0.0

Microsoft.AspNetCore.Serve.KestrelCore 8.0.0.0 **domyślny serwer HTTP**

Microsoft.AspNetCore.HostFiltering 8.0.0.0

Microsoft.Extensions.Configuration 8.0.0.0

Microsoft.Extensions.Hosting 8.0.0.0

Microsoft.System.ComponentModel 8.0.0.0

Microsoft.Extensions.Features 8.0.0.0

Microsoft.Extensions.Configuration.CommandLine 8.0.0.0

Microsoft.AspNetCore.Hosting 8.0.0.0

Microsoft.Extensions.Hosting.Abstractions 8.0.0.0

Microsoft.AspNetCore.Hosting.Abstractions 8.0.0.0

Microsoft.AspNetCore.Hosting.Server.Abstractions 8.0.0.0

Microsoft.Extensions.Configuration.EnvironmentVariables 8.0.0.0
Microsoft.Extensions.Configuration.Json 8.0.0.0
Microsoft.Extensions.Configuration.UserSecrets 8.0.0.0

Microsoft.AspNetCore.Authentication.Abstractions 8.0.0.0
Microsoft.AspNetCore.Authentication 8.0.0.0
Microsoft.AspNetCore.Authorization 8.0.0.0
Microsoft.AspNetCore.Authorization.Policy 8.0.0.0

Microsoft.Extensions.Logging 8.0.0.0
Microsoft.Extensions.Logging.Abstractions 8.0.0.0
Microsoft.Extensions.Logging.Configuration 8.0.0.0
Microsoft.Extensions.Logging.Console 8.0.0.0
Microsoft.Extensions.Logging.Debug 8.0.0.0
Microsoft.Extensions.Logging.EventSource 8.0.0.0

Microsoft.AspNetCore.Diagnostics 8.0.0.0
Microsoft.Extensions.Diagnostics.Abstractions 8.0.0.0
System.Linq, Version=8.0.0.0

3.2 Zarządzanie zależnościami

Do zarządzania zależnościami - zarządcą pakietów Nuget dla .NET - wykorzystywaliśmy go do wygodnego instalowania pakietów .NET poprzez Visual Studio Code Community. Pakiety były pobierane z repozytorium Microsoftu lub z repozytorium nugeta. Wymagało to jednak wstępnej konfiguracji w VSC(Visual Studio Code) i oczywiście uzyskaniu dostępu do tych repozytoriów - poprzez zarejestrowanie się na stronach www.nuget.org i potem konfiguracji VSC aby "widział" te repozytoria podczas korzystania z manadżera pakietów NuGet

3.3 Mapowanie

Do mapowania obiektowo-relacyjnego (ORM) wykorzystujemy Entity Framework Core z ASP.Net CORE. Entity Framework umożliwia prace z danymi w bazie danych za pomocą obiektów i zapytań LINQ, co eliminuje potrzebę pisania bezpośrednich zapytań SQL.

3.4 Technologie do budowy warstwy interfejsu użytkownika - frontend

Java Script

HTML

framework CSS: Bootstrap ver. 5.1.0

3.5 Cache

System cache jest implementowany przy użyciu wbudowanego w ASP.NETCore mechanizmu pamięci podręcznej

Rozdział 4

Realizacja wymagań projektowych

4.1 Wymagania projektowe i stopień realizacji

Tabela 4.1: LISTA ZAGADNIEŃ KWALIFIKACYJNYCH

Lp.	Zagadnienie	Opis	
1	framework MVC	wykorzystanie frameworka na backenedzie	✓
2	framework CSS	wykorzystanie frameworka na frontenedzie	✓
3	baza danych	dołączenie do projektu bazy danych	✓
4	cache	dołączenie do projektu systemu cache	✓
5	dependency manager	dołączenie do projektu systemu zarządzania zależnościami	✓
6	HTML	szkielet aplikacji internetowej	✓
7	CSS	ostylowanie aplikacji internetowej	✓
8	JavaScript	interaktywniejszenie aplikacji internetowej	✓
9	routing	wykorzystany routing i tzw. pretty URLs	✓
10	ORM	wykorzystane mapowanie obiektoworelacyjne	✓
11	uwierzytelnianie	zaimplementowane mechanizmy nieuwierzytelnienia	✓
12	lokalizacja	możliwość przełączania języka aplikacji	✓
13	mailing	wysyłanie mejli z aplikacji	✓
14	formularze	przesyłanie danych do aplikacji przez formularze	✓
15	asynchroniczne interakcje	zaimplementowane asynchroniczne interakcje z serwerem	✓
16	konsumpcja API	wykorzystanie zewnętrznego API	✓
17	publikacja API	responsywny frontend	x
18	RWD	responsywny frontend	✓
19	logger	logowanie akcji w systemie	✓
20	deployment	wdrożenie aplikacji internetowej	x

Zrealizowane zagadnienia kwalifikacyjne wykonane według założeń z Tabela 4.1

1. Framework MVC: Wykorzystujemy framework ASP.NET Core MVC do budowy backendu aplikacji:
 - (Model) Model danych - opis struktur danych i powiązań pomiędzy nimi
 - (View) Interfejs, czyli to co widzi użytkownik
 - (Controller) Logika działania - powiązania między zdarzeniami zachodzącymi w systemie
2. Framework CSS: Do stylizacji interfejsu użytkownika używamy frameworka Bootstrap. Bootstrap to popularny framework front-endowy, który zapewnia zestaw narzędzi, komponentów i stylów CSS.
3. Baza danych: Projekt wykorzystuje Microsoft SQL jako bazę danych.
4. Cache: Wdrożono mechanizm pamięci podręcznej do optymalizowania aplikacji. System Cache jest wykorzystywany w naszym projekcie do zapisywania oraz wyświetlania danych z profilu użytkownika tak aby za każdym najciemniej profil projekt nie musiał na nowo pobierać danych z bazy danych.
5. Dependency manager: Do zarządzania zależnościami aplikacji używamy menedżera pakietów NuGet. NuGet jest menedżerem pakietów dla platformy .NET, który umożliwia łatwe dodawanie, usuwanie i aktualizowanie zależności bibliotek i narzędzi do projektów, co ułatwia zarządzanie zależnościami i zapewnia ponowne wykorzystanie kodu.
6. HTML: Szkielet aplikacji internetowej został zbudowany zgodnie z standardami HTML. HTML służy do strukturyzowania treści na stronach internetowych poprzez definiowanie różnych elementów, takich jak nagłówki, paragrafy, listy, obrazy, linki etc.
7. CSS: Wykorzystujemy arkusze stylów CSS do ostylowania aplikacji. W naszej aplikacji korzystaliśmy z CSS do m.in. dodawania zdjęcia w tle strony, oraz do dokładniejszej konfiguracji wyglądu strony.
8. JavaScript: W aplikacji użyto JavaScript do uinteraktywnienia interfejsu użytkownika. W naszym projekcie JavaScript używany jest do dodania wyskakującego okna z potwierdzeniem chęci wylogowania się oraz przy dodawaniu zdjęcia podczas rejestracji.

9. Routing: Wykorzystujemy routing i tzw. pretty URLs dla estetycznych adresów URL.
10. ORM: Do mapowania obiektowo-relacyjnego używamy Entity Framework Core. Entity Framework umożliwia prace z danymi w bazie danych za pomocą obiektów i zapytań LINQ, co eliminuje potrzebę pisania bezpośrednich zapytań SQL.
11. Uwierzytelnianie: Uwierzytelnianie akcji w systemie polega na identyfikacji zalogowanego użytkownika przez system, co pozwala mu dostosować działania i udostępnić odpowiednie funkcjonalności zgodnie z jego uprawnieniami.
12. Lokalizacja: W banerze strony jest opcja wyboru języka, pomiędzy polskim a angielskim. Do tłumaczenia strony korzystaliśmy z plików o rozszerzeniu .resx
13. Mailing: Mailing to proces wysyłania wiadomości e-mail z aplikacji internetowej do określonych użytkowników lub grupy odbiorców. Używamy SMTP do wysyłania powiadomienia o utworzonym koncie. Mamy również możliwość wysyłania mejla ze zgłoszeniem.
14. Formularze: Formularze to interaktywne elementy interfejsu użytkownika, które umożliwiają użytkownikom przesyłanie danych do aplikacji poprzez wprowadzanie informacji w pola tekstowe, wybieranie opcji, czy przesyłanie plików. Korzystamy z formularzy poprzez rejestracje, logowanie, wybór płci, matchu, wysyłanie zgłoszenia oraz edycji hobby.
15. Asynchroniczne interakcje: Są procesami, w których zadania i odpowiedzi między klientem a serwerem odbywają się niezależnie, umożliwiając wykonywanie innych operacji w trakcie oczekiwania na odpowiedź. Używamy ich podczas wrzucania zdjęcia podczas rejestracji oraz podczas wysyłania e-maila.
16. Konsumpcja API: Konsumpcja API polega na pobieraniu danych lub wykonywaniu operacji z zewnętrznego interfejsu programistycznego. Korzystamy z Jokes API które generuje losowe żarty.
18. RWD: Responsywny front-end to taki, który automatycznie dostosowuje się do różnych rozmiarów i typów urządzeń, zapewniając optymalne doświadczenie użytkownika na każdym ekranie. Mamy go za prawie za darmo stosując Bootstrapa.

Rozdział 5

Instrukcja lokalnego i zdalnego uruchomienia

5.1 Instrukcja lokalnego uruchomienia systemu

1. Zainstaluj ASP.NET Core ver. 8.0
2. Zainstaluj Visual Studio Community 2022. Zarejestruj się na stronie www.nuget.org abyś mógł pobierać z repozytorium zależności i pakiety .NET poprzez menadżera pakietów NuGet w VSC. Skonfiguruj VSC do pobierania pakietów z repozytoriów NuGet-a i Microsoft-u.
3. Pobranie kodu źródłowego: Sklonuj repozytorium z Git Huba na swój lokalny komputer. Możesz to zrobić za pomocą polecenia git clone [adres repozytorium].Repozytorium projektu
4. Otwarcie projektu: Otwórz pobrany projekt uruchamiając plik Amora.sln w wybranym edytorze kodu,na przykład Visual Studio Code lub innym.
5. Instalacja zależności: Upewnij się, że zainstalowane są wszystkie zależności projektu. Możesz to zrobić za pomocą menadżera pakietów NuGet. W konsoli NuGet należy wpisać komendę update-database.
6. Uruchomienie aplikacji: Uruchom projekt poprzez ctrl+F5(Visual Studio) lub poprzez odpowiadającą opcje w innym edytorze.
7. Otwarcie w przeglądarce: Po zakończeniu procesu uruchamiania projekt będzie dostępny pod adresem <http://localhost:port>.

8. Testowanie funkcjonalności: Przetestuj różne funkcje aplikacji, takie jak logowanie, rejestracja, przeglądanie profili, itp., aby upewnić się, że aplikacja działa poprawnie.

Rozdział 6

Rozmieszczenie dokumentacji i plików projektu na git-hubie

Strona projektu "AMORA" na github-ie:

<https://github.com/jakubtrznadel/PipsiProject>

Pliki źródłowe "AMORA": w katalogu AMORA

Piki dokumentacji: w katalogu AMORADOCU

Rozdział 7

Wnioski projektowe

Korzystanie z .NET było fascynującym doświadczeniem, choć nie obyło się bez wyzwań, szczególnie podczas pierwszych kroków z Frameworkiem MVC. Metoda prób i błędów była naszą codziennością, a każde napotkane trudności stawały się szansą na naukę. Odkrywanie Frameworka CSS - Bootstrap-a stanowiło niezwykle interesująca część projektu, oferując narzędzia, których wcześniej nie mieliśmy okazji używać, co znaczco ułatwiło rozwój naszego projektu. Podsumowując, choć okres pracy był wymagający, zdobyte doświadczenie pozostanie z nami na dłucho, pozostawiając nam cenne lekcje na przyszłość.

Platforma ASP.NET Core MVC udostępnia funkcjonalności, które znacznie ułatwiają tworzenie internetowych interfejsów API i aplikacji internetowych. Jesteśmy zgodni w tym, że model MVC w ASP.NET Core wraz z mechanizmem wstrzykiwania zależności znacznie ułatwia projektowanie aplikacji internetowych i że od tej mamy inne spojrzenie na problematykę tworzenia portali internetowych. Nie zdążyliśmy jeszcze dokonać publikacji API. Mamy nadzieję, że i to doświadczenie nie ominie nas w dalszym etapie nauki.