WYŻSZA SZKOŁA
INFORMATYKI i ZARZĄDZANIA
z siedzibą w Rzeszowie

KOLEGIUM INFORMATYKI STOSOWANEJ

System do zarządzania strukturą medyczną

Klaudia Jońca

Numer albumu: 69566

Spis treści

1. Opis projektu

Aplikacja do zarzadzania strukturą medyczną. Dzięki niemu użytkownicy, tacy jak lekarze, pacjenci i administratorzy medyczni, mają dostęp do różnorodnych funkcjonalności, takich jak wyświetlanie listy chorób leczonych, szczegółowe informacje o chorobach, tworzenie i edycja wpisów medycznych, opis badania przeprowadzonego przez lekarza, zarządzanie powiązaniami między lekarzami, a placówkami medycznymi, monitorowanie terminów wizyt oraz zarządzanie danymi pacjentów i placówek medycznych.

Projekt jest skonfigurowany do wykorzystania technologii ASP.NET MVC Core, Entity Framework Core i modułu Identity do uwierzytelniania i zarządzania tożsamościami. Zawiera również narzędzia ułatwiające rozwój aplikacji, takie jak generowanie kodu i kompilacja widoków w czasie rzeczywistym.

Identity to framework do uwierzytelniania i autoryzacji, który jest wbudowany w platformę ASP.NET. Identity zapewnia zestaw funkcji do zarządzania użytkownikami, logowaniem, rejestracją, uprawnieniami i innymi związanymi zabezpieczeniami w aplikacjach internetowych.

Framework Identity opiera się na strukturze użytkowników i ról. Umożliwia tworzenie i zarządzanie kontami użytkowników, w tym uwierzytelnianie poprzez różne metody, takie jak hasło, tokeny bezpieczeństwa lub zewnętrznych dostawców uwierzytelniania (np. Google lub Facebook). Można tworzyć nowe konta użytkowników, zezwalać na logowanie przy użyciu nazwy użytkownika lub adresu e-mail, resetować hasła, zarządzać danymi użytkownika i rolami oraz implementować dodatkowe zabezpieczenia, takie jak blokowanie konta po kilkukrotnych nieudanych próbach logowania.

Identity oferuje również mechanizm autoryzacji, który umożliwia kontrolę dostępu do różnych części aplikacji na podstawie ról lub uprawnień użytkownika. Można definiować niestandardowe role i polityki autoryzacji oraz stosować dekoratory i atrybuty autoryzacyjne, aby zabezpieczyć kontrolery, akcje i widoki przed nieuprawnionym dostępem. Framework Identity jest elastyczny i dostosowalny do indywidualnych wymagań aplikacji. Można go rozszerzać poprzez dodawanie niestandardowych pól użytkownika, integrację z innymi usługami uwierzytelniania (np. Azure AD) oraz dostosowywanie szablonów widoków i logiki uwierzytelniania.

Odwołania do pakietów, które są zależnościami projektu. Oto krótki opis każdego z zainstalowanych pakietów:

- Microsoft.AspNetCore.Diagnostics.EntityFrameworkCore: Pakiet zawierający narzędzia diagnostyczne dla aplikacji ASP.NET Core, które korzystają z Entity Framework Core.
- Microsoft.AspNetCore.Identity.EntityFrameworkCore: Pakiet zawierający narzędzia i funkcje do obsługi uwierzytelniania i zarządzania tożsamościami w aplikacjach ASP.NET Core przy użyciu Entity Framework Core.
- Microsoft.AspNetCore.Identity.UI: Pakiet zawierający zestaw interfejsów użytkownika (UI) i widoków, które są gotowe do użycia w celu obsługi funkcji uwierzytelniania i zarządzania tożsamościami.
- Microsoft.AspNetCore.Mvc.Razor.RuntimeCompilation: Pakiet zawierający narzędzia do kompilacji widoków Razor w czasie rzeczywistym podczas rozwoju aplikacji.

- Microsoft.EntityFrameworkCore.SqlServer: Pakiet zawierający dostawcę bazy danych dla Entity Framework Core, który obsługuje platformę SQL Server.
- Microsoft.EntityFrameworkCore.Tools: Pakiet zawierający narzędzia, które wspomagają pracę z Entity Framework Core, takie jak migracje bazy danych i generowanie kodu na podstawie modelu danych.
- Microsoft.VisualStudio.Web.CodeGeneration.Design: Pakiet zawierający narzędzia projektowe do generowania kodu, takie jak scaffolding (budowanie struktury projektu na podstawie modelu) i automatyczne generowanie kontrolerów i widoków.

2. Wymagania funkcjonalne i niefunkcjonalne

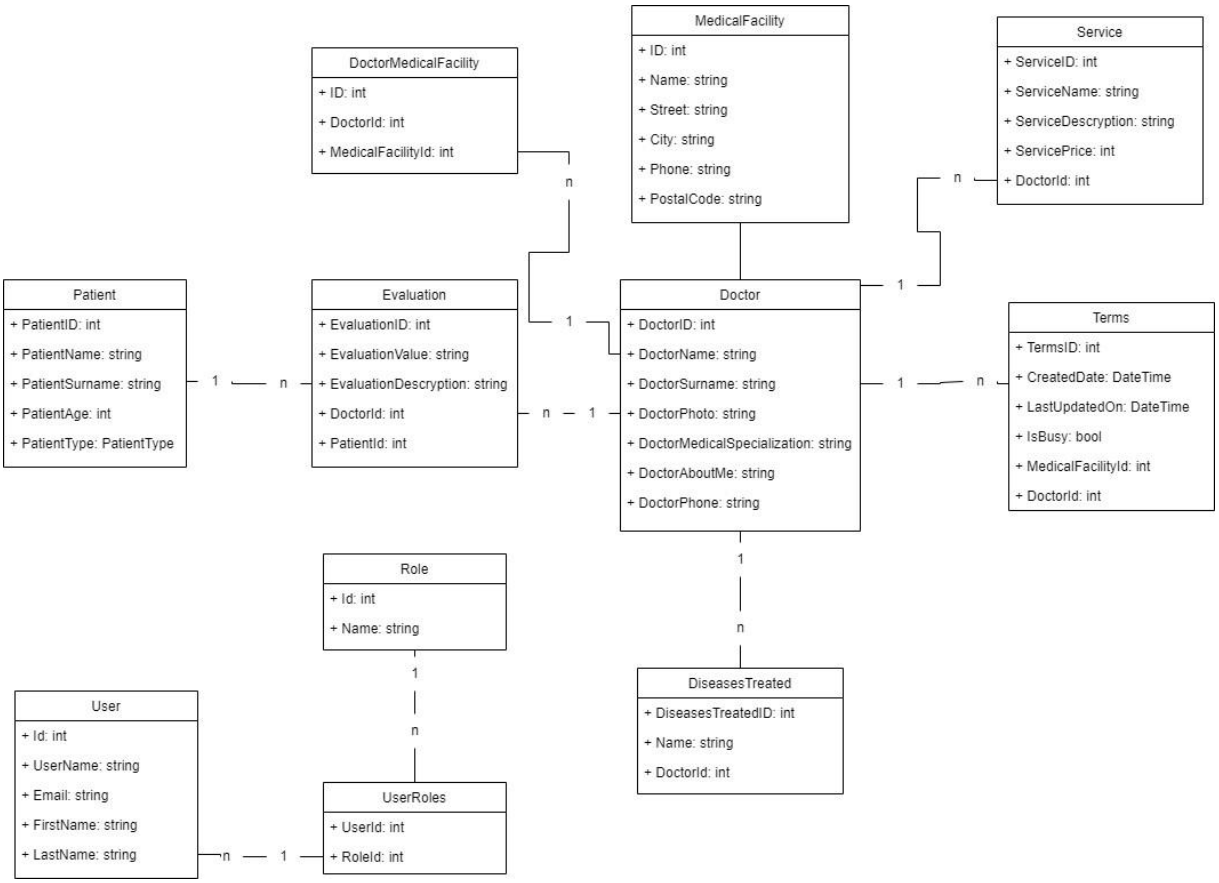
Wymagania funkcjonalne:

- System medyczny powinien umożliwiać wyświetlanie listy chorób leczonych oraz dostarczać szczegółowych informacji na temat każdej choroby. Użytkownicy powinni mieć możliwość tworzenia nowych wpisów dotyczących choroby, edycji istniejących wpisów oraz usuwania wpisów w celu skutecznego zarządzania historią medyczną lekarzy.
- System powinien umożliwiać wyświetlanie listy powiązań między lekarzami a placówkami medycznymi, a także dostarczać szczegółowych informacji o każdym powiązaniu. Użytkownicy, tak jak administratorzy medyczni, powinni mieć możliwość tworzenia nowych powiązań, edycji istniejących powiązań oraz usuwania powiązań w celu skutecznego zarządzania dostępem lekarzy do konkretnych placówek medycznych.
- System powinien umożliwiać wyświetlanie listy lekarzy oraz dostarczać szczegółowych informacji na temat każdego lekarza. Użytkownicy, tak jak administratorzy medyczni, powinni mieć możliwość tworzenia nowych rekordów lekarzy, edycji istniejących danych lekarzy oraz usuwania lekarzy z systemu.
- System powinien umożliwiać wyświetlenie listy przeprowadzonych badań z opisem dla każdego pacjenta. Użytkownicy, tacy jak lekarze, powinni mieć możliwość tworzenia nowych wpisów dotyczących przeprowadzonych badań, ich edycji oraz usuwania.
- System powinien umożliwiać wyświetlanie listy placówek medycznych oraz dostarczać szczegółowych informacji na temat każdej placówki. Administratorzy medyczni powinni mieć możliwość tworzenia nowych placówek medycznych, edycji istniejących danych placówek oraz usuwania placówek z systemu.
- System medyczny powinien umożliwiać wyświetlanie listy pacjentów oraz dostarczać szczegółowych informacji na temat każdego pacjenta. Administratorzy medyczni powinni mieć możliwość tworzenia nowych rekordów pacjentów, edycji istniejących danych pacjentów oraz usuwania pacjentów z systemu. To pozwoli na skuteczne zarządzanie informacjami medycznymi pacjentów, historią wizyt i wynikami badań.
- System powinien umożliwiać wyświetlanie listy ról w systemie, takich jak lekarz, administrator medyczny itp. Administratorzy medyczni powinni mieć możliwość tworzenia nowych ról, usuwania istniejących ról oraz aktualizowania uprawnień przypisanych do poszczególnych ról. To umożliwi efektywne zarządzanie dostępem i uprawnieniami użytkowników w systemie medycznym.
- System powinien umożliwiać wyświetlanie listy dostępnych usług medycznych oraz dostarczać szczegółowych informacji na ich temat. Administratorzy medyczni powinni mieć możliwość tworzenia nowych usług, edycji istniejących usług oraz usuwania usług z systemu. To pozwoli na skuteczne zarządzanie oferowanymi usługami medycznymi i ich parametrami.
- System powinien umożliwiać wyświetlanie listy dostępnych terminów wizyt, badań i innych procedur medycznych. Użytkownicy, tacy jak personel medyczny, powinni mieć możliwość tworzenia nowych terminów, edycji istniejących terminów oraz usuwania terminów. To pozwoli na efektywne planowanie i organizację spotkań medycznych.

Wymagania niefunkcjonalne:

- System powinien zapewnić odpowiednie zabezpieczenia widoków, tak aby użytkownicy mieli dostęp tylko do tych funkcjonalności i informacji, które są zgodne z ich rolą w systemie. Zabezpieczenia powinny być skonfigurowane w taki sposób, aby zapewnić poufność danych oraz ochronę prywatności pacjentów.
- Interfejs użytkownika systemu medycznego powinien być intuicyjny i przyjazny dla użytkownika. Powinien umożliwiać łatwą nawigację po aplikacji, intuicyjne wprowadzanie danych oraz zapewniać czytelne prezentowanie informacji. Dbałość o ergonomiczny interfejs przyczyni się do wygodnego i efektywnego korzystania z systemu przez personel medyczny.
- Komunikaty systemowe, takie jak informacje o sukcesie operacji lub komunikaty błędów, powinny być czytelne, zrozumiałe i klarowne dla użytkowników. Dzięki temu będą w stanie łatwo zrozumieć, co się dzieje w systemie i jakie działania należy podjąć w przypadku wystąpienia problemów.
- System medyczny powinien zapewniać mechanizm uwierzytelniania użytkowników, aby potwierdzić ich tożsamość przed udostępnieniem dostępu do aplikacji. Proces uwierzytelniania może obejmować logowanie za pomocą unikalnego identyfikatora użytkownika i hasła.
- System powinien zapewnić weryfikację tożsamości i autoryzację użytkowników przy użyciu odpowiednich mechanizmów, takich jak Identity Framework. To umożliwi kontrolę dostępu do różnych funkcjonalności i danych w systemie na podstawie przypisanych uprawnień i roli użytkownika.

3. Diagram klas



Rys. 1. Diagram klas systemu. Źródło: Opracowanie własne.

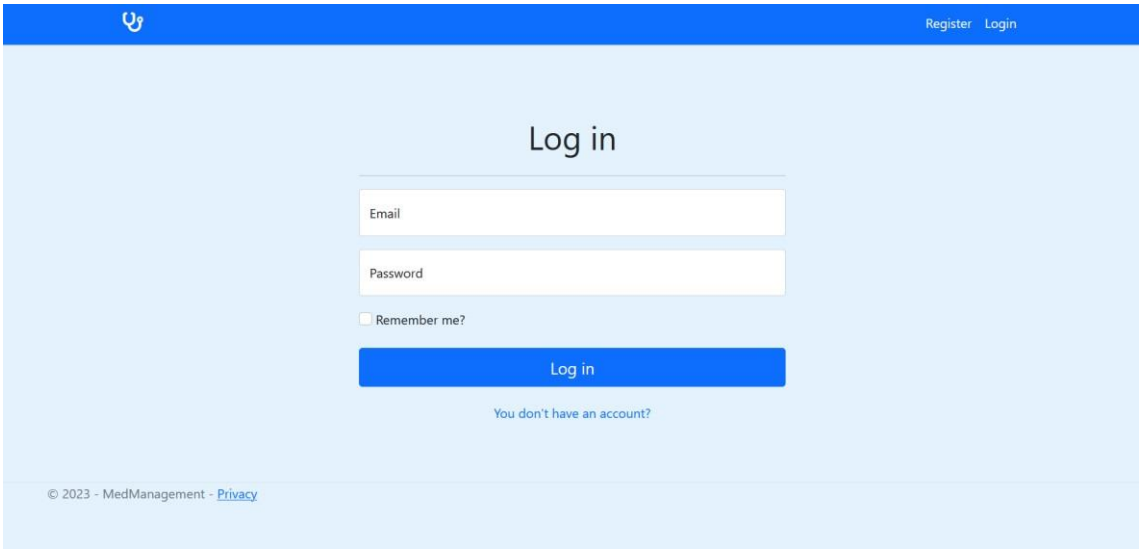
Diagram klas to graficzna reprezentacja struktury bazy danych. Relacje są reprezentowane jako linie łączące tabele. W diagramie znajdują się tabele: Pacjent, Badanie, Lekarz, Placówka Medyczna, Usługa, Termin, Choroba, Użytkownik, Rola.

Relacje wynikające z klas:

- Jeden pacjent może wystawić wiele ocen
- Wiele różnych badań jest przypisanych do jednego lekarza
- Jeden lekarz może mieć wiele usług
- Jeden lekarz może mieć wiele terminów
- Jeden lekarz może mieć wiele leczonych chorób (spis chorób jakich leczenia podejmuje się lekarz)
- Wielu lekarzy może być w wielu różnych placówkach medycznych

4. GUI Aplikacji

Aplikacja ze względów bezpieczeństwa w pierwszej kolejności wymusza rejestrację, bądź logowanie do zawartości systemu.

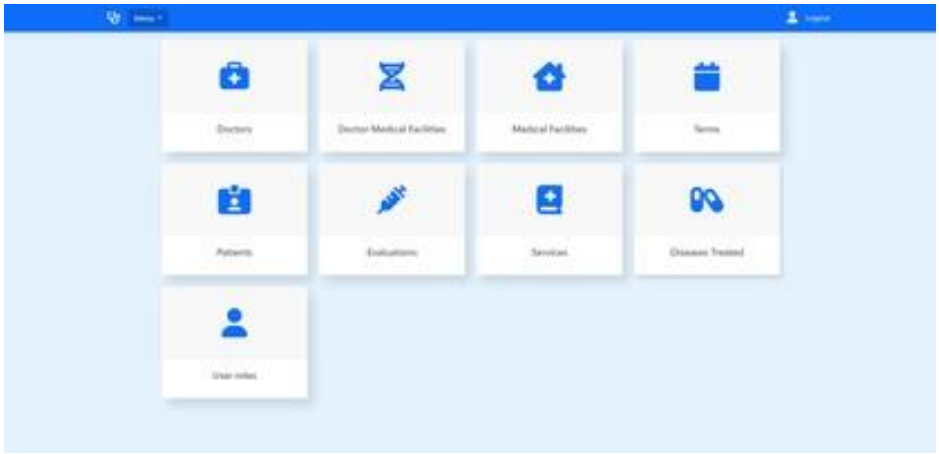


Rys. 2. Widok logowania do systemu. Źródło: Opracowanie własne.

Aplikacja dostarcza możliwość przeglądania zawartości serwisu w zależności od roli użytkownika:

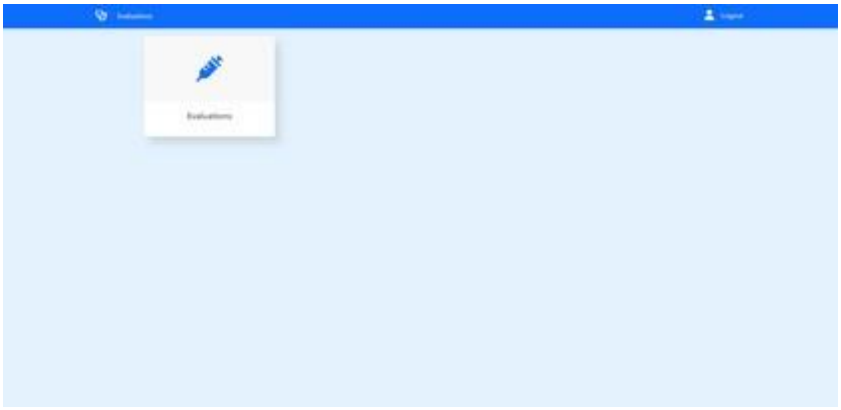
1) Administrator – pełne prawa do wszystkich elementów systemu
2) Doktor – możliwość dodawania badań
3) Recepcjonista/Recepcjonistka – możliwość zapisów terminów oraz pacjentów

Widok panelu dla roli nr 1:



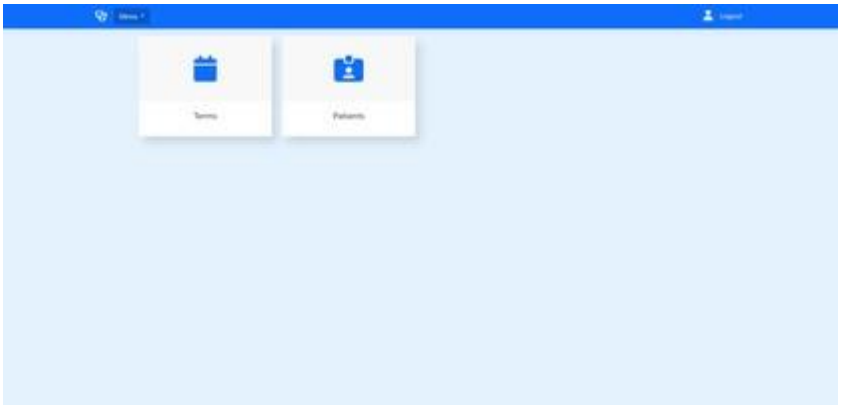
Rys. 3. Widok systemu użytkownika o roli: Administrator. Źródło: Opracowanie własne.

Widok panelu dla roli nr 2:



Rys. 4. Widok systemu użytkownika o roli: Doktor. Źródło: Opracowanie własne.

Widok panelu dla roli nr 3:



Rys. 5. Widok systemu użytkownika o roli: Recepcjonista/Recepcjonistka. Źródło: Opracowanie własne.