Tytuł projektu: System Zarzadzania Dziekanatem

Repozytorium: https://github.com/mazureks/DeansOfficeManagement.git

Autor: Sebastian Mazurek

System Zarządzania Dziekanatem to aplikacja internetowa wykonana **w architekturze MVC** oparta na platformie .NET 8, stworzona z myślą o wsparciu podstawowych czynności administracyjnych dziekanatu. System umożliwia rozliczanie wyników w nauce studentów w formie punkcji ECTS. Punkty ECTS (European Credit Transfer and Accumulation System) to system punktowy stosowany w europejskim szkolnictwie wyższym, który ułatwia porównywanie osiągnięć akademickich i przenoszenie ich między różnymi uczelniami oraz krajami. Został wprowadzony w ramach procesu bolońskiego, aby umożliwić większą mobilność studentów i lepsze uznawanie kwalifikacji w Europie. System stanowi bazę wyjściową do rozbudowy pełnej funkcjonalności charakterystycznej dla biura Dziekana wyższej uczelni.

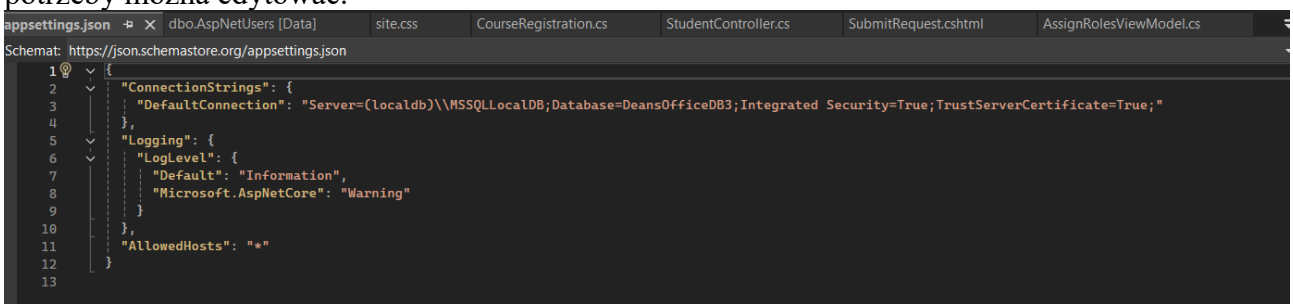
1.**Użyte technologie:**

Backend:	.NET 8 (ASP.NET MVC)
Baza danych:	SQL Server (skonfigurowany za pomocą Entity Framework Core w podejściu Code-First)
Uwierzytelnianie:	Uwierzytelnianie oparte na ciasteczkach z ASP.NET Identity i haszowaniem haseł
Frontend:	Strony Razor oraz Bootstrap.

2. Uruchomienie lokalne projektu

Aplikację można uruchomić w środowisku .NET 8 SDK. Mając prawidłowo zainstalowane w/w środowisko z repozytorium GitHub System Zarządzania Dziekanatu pobieramy pliki projektu, kopiujemy je na lokalny dysk. Po skopiowaniu plików otwieramy rozwiązanie **DeansOfficeManagement.sln**.

W pliku appsettings.json znajduje się konfiguracja połączenia do bazy danych, którą w razie potrzeby można edytować.

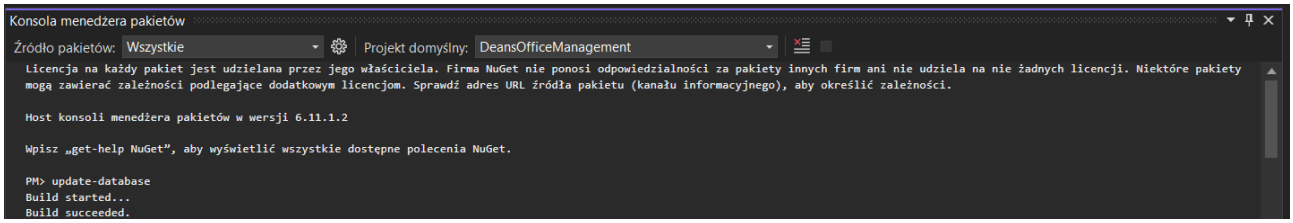


Przed pierwszym uruchomieniem projektu należy wykonać aktualizację bazy danych:

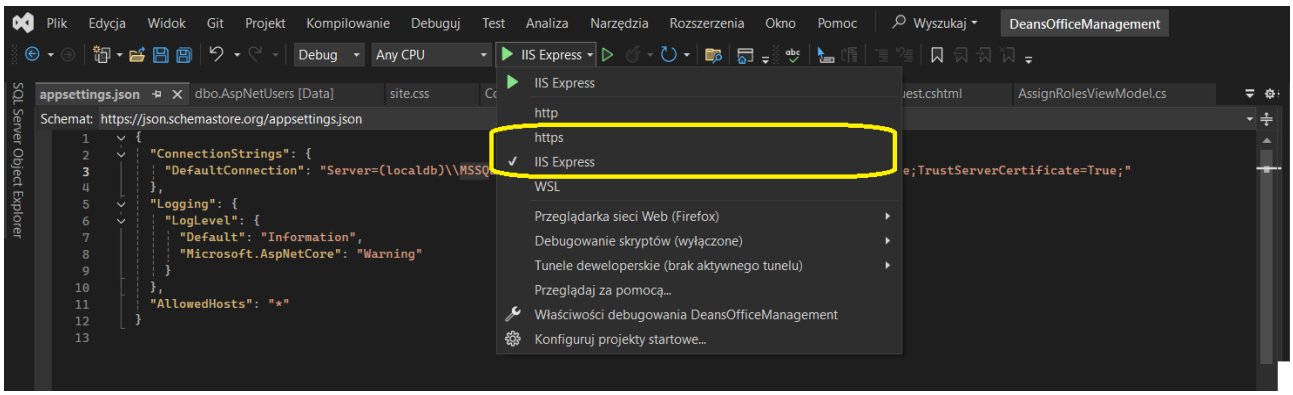
Przejdź do
Narzędzia > **Menedżer pakietów NuGet** > **Konsola Menedżera pakietów**

Uruchom poniższą komendę, aby utworzyć bazę danych i zastosować migracje:

update-database



Uruchom projekt:
Naciśnij F5 lub wybierz Uruchom, aby rozpocząć projekt na lokalnym serwerze.
Uwaga: Możliwe, że przy uruchamianiu projektu zajdzie konieczność wykonania zmian w menu wyboru profilu uruchamiania w Visual Studio pomiędzy IIS a https



3. Struktura projektu
Controllers:

- Zawiera kontrolery, które obsługują logikę aplikacji.
- Kontrolery odbierają żądania od użytkownika (np. za pośrednictwem przeglądarki), przetwarzają dane (korzystając z modeli) i zwracają odpowiednią odpowiedź (np. widok).

1. Kontroler **`HomeController`** zarządza stroną główną aplikacji, obsługując przekierowanie zalogowanych użytkowników do odpowiednich paneli na podstawie ich roli (Student, Lecturer, Admin) oraz wyświetlanie domyślnego widoku dla niezalogowanych użytkowników. Dodatkowo obsługuje stronę prywatności i stronę błędów, przekazując odpowiednie dane do widoków

2. Kontroler **`AccountController`** odpowiada za zarządzanie procesami uwierzytelniania i autoryzacji w aplikacji. Obsługuje rejestrację nowych użytkowników (`Register`), logowanie (`Login`) oraz wylogowanie (`Logout`). Dodatkowo przypisuje rolę "Admin" pierwszemu zarejestrowanemu użytkownikowi, a dla pozostałych użytkowników pozostawia role do przypisania przez administratora. Kontroler korzysta z usług `UserManager`, `SignInManager` i `RoleManager` w celu zarządzania użytkownikami, ich hasłami, sesjami oraz rolami.

3. Kontroler **`AdminController`** odpowiada za zarządzanie systemem przez administratora (dizekana), obsługując zarządzanie użytkownikami (wyświetlanie, dodawanie, edycję studentów i przypisywanie ról), kursami (tworzenie, edycję, przypisywanie wykładowców i przegląd kursów), rejestracjami studentów na kursy (zatwierdzanie i odrzucanie rejestracji) oraz wnioskami studentów (przegląd, zatwierdzanie, odrzucanie). Wykorzystuje `UserManager`, `RoleManager` i `ApplicationDbContext` do realizacji tych zadań i jest przeznaczony wyłącznie dla użytkowników z rolą "ADMIN"(dziekan).

4. Kontroler **`LecturerController`** odpowiada za zarządzanie funkcjonalnościami dla wykładowców, takimi jak wyświetlanie przypisanych kursów, wprowadzanie ocen studentów oraz zarządzanie listą kursów. Obsługuje wyświetlanie kursów przypisanych do wykładowcy, wprowadzanie lub aktualizację ocen studentów w ramach kursu, a także wyświetlanie listy kursów wykładowcy. Wykorzystuje `UserManager`, `RoleManager` i `ApplicationDbContext` do operacji na użytkownikach, rolach oraz danych aplikacji, i jest dostępny tylko dla użytkowników z rolą "LECTURER".

5. Kontroler **`StudentController`** zarządza funkcjonalnościami dostępnymi dla studentów, takimi jak wyświetlanie profilu, rejestracja na kursy, przegląd ocen oraz składanie i przeglądanie wniosków. Obsługuje operacje, takie jak zapisywanie studenta na kurs, wyświetlanie ocen przypisanych do kursów, przeglądanie osobistych informacji użytkownika oraz składanie wniosków do dziekanatu. Korzysta z `ApplicationDbContext` do obsługi bazy danych i `UserManager` do zarządzania użytkownikami, a dostęp do tego kontrolera mają tylko użytkownicy z rolą "STUDENT".

Modele

Każdy model reprezentuje tabelę bazy danych w SQL Server. Modele reprezentują dane i logikę biznesową aplikacji. Służą do definiowania struktur danych, które są przechowywane w bazie danych i przetwarzane w aplikacji.

Modele zastosowane w moim projekcie:

1. `AdminPanel`

- Rola: Reprezentuje zadania administratora systemu.
- Działanie: Zawiera informacje o zadaniach do wykonania przez administratora, takie jak nazwa zadania i termin realizacji.

2. `ApplicationUser`

- Rola: Reprezentuje użytkowników systemu (studentów, wykładowców, administratorów).
- Działanie: Rozszerza wbudowany model `IdentityUser`, dodając właściwości takie jak `FirstName`, `LastName`, `Role`, oraz `StudentNumber`. Obsługuje relacje z kursami i wnioskami przypisanymi do użytkownika.

3. `Course`

- Rola: Reprezentuje kurs akademicki w systemie.
- Działanie: Przechowuje informacje o kursie, takie jak nazwa, liczba punktów ECTS, oraz identyfikator prowadzącego (wykładowcy). Umożliwia nawigację do rejestracji kursów i przypisanych ocen.

4. `CourseRegistration`

- Rola: Reprezentuje rejestrację studenta na kurs.
- Działanie: Przechowuje informacje o statusie rejestracji (`W toku`, `Zatwierdzony`), powiązaniach między studentem (`StudentId`) a kursem (`CourseId`).

5. `Grade`

- Rola: Reprezentuje ocenę studenta z kursu.
- Działanie: Zawiera szczegóły oceny, takie jak wartość punktowa (`Score`) i opcjonalna litera oceny (`GradeLetter`). Powiązana z tabelami studentów i kursów.

6. `Request`

- **Rola:** Reprezentuje wniosek złożony przez studenta do administracji.
- **Działanie:** Przechowuje informacje o typie wniosku (`Urlop`, `Zmiana kursu`), jego opisie, dacie złożenia i statusie (`Oczekujące`, `Zatwierdzone`). Powiązany z tabelą użytkowników (`StudentId`).

Ponadto podkatalog `ViewModels` zawiera 8 modeli, które są używane wyłącznie do przesyłania danych między kontrolerami a widokami, dostosowując dane do specyficznych potrzeb interfejsu użytkownika, co ułatwia organizację i utrzymanie kodu.

1. `AssignRolesViewModel`

- Rola: Używany do przypisywania ról użytkownikom przez administratora.
- Działanie: Przechowuje informacje o użytkownikach i ich rolach, takie jak ID użytkownika, wybrane role, dostępne role oraz lista użytkowników.

2. `ErrorViewModel`

- Rola: Wyświetla szczegóły błędów w aplikacji.
- Działanie: Przechowuje identyfikator żądania (`RequestId`) i logicznie wskazuje, czy pokazać ID błędu w widoku.

3 `GradeInputViewModel`

- Rola: Używany do wprowadzania ocen przez wykładowców.
- Działanie: Przechowuje informacje o kursie (`CourseId`, `CourseName`) oraz listę studentów i ich ocen do przypisania.

4. `LecturerDashboardViewModel`

- Rola: Reprezentuje dane dla panelu wykładowcy.
- Działanie: Przechowuje listę kursów przypisanych do zalogowanego wykładowcy.

5. `LoginViewModel`

- Rola: Używany podczas logowania użytkownika.
- Działanie: Przechowuje dane logowania, takie jak e-mail, hasło oraz opcję zapamiętania użytkownika (`RememberMe`).

6. `RegisterViewModel`

- Rola: Używany podczas rejestracji nowych użytkowników.
- Działanie: Zawiera informacje o nowym użytkowniku, takie jak e-mail, imię, nazwisko, hasło i potwierdzenie hasła. Weryfikuje poprawność danych (np. zgodność haseł).

7. `StudentGradeViewModel`

- Rola: Wyświetla szczegóły ocen studenta.
- Działanie: Przechowuje dane o kursie (`CourseName`), wartości oceny (`GradeValue`) oraz dacie jej przypisania.

8. `SubmitRequestViewModel`

- Rola: Używany do składania wniosków przez studentów.
- Działanie: Zawiera informacje o typie wniosku (`RequestType`) oraz jego opisie. Ogranicza długość opisu do 500 znaków.

Widoki (Views)

Przechowują interfejs użytkownika aplikacji. Widoki są generowane na podstawie danych przekazanych przez kontrolery. Pliki widoków zazwyczaj mają rozszerzenie `.cshtml` i zawierają kod HTML z możliwością wstawiania kodu C#.

Ze względów porządkowych pliki widoków pogrupowane zostały w 6 katalogach odpowiadających pewnej grupie logicznych czynności:
Account – logowanie i rejsetracja
Admin – interfejs administratora (dziekana)
Home – interfejs niezalogowanego użytkownika.
Lecturer – interfejs wykładowcy
Student – interfejs studenta
Shared – elementy wspólne interfejsów (oprawy graficzne, nagłówek, stopka).

Migrations:

Zawiera pliki migracji do inicjalizacji i aktualizacji schematu bazy danych

Folder Data zawiera konfiguracje dostępu do bazy danych oraz logikę wypełniania danych startowych.

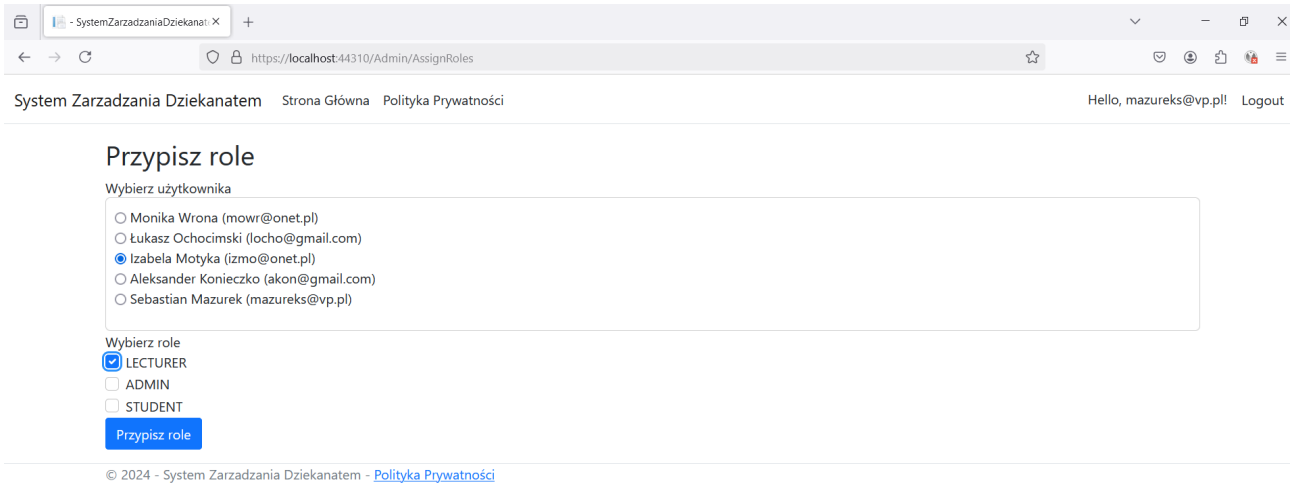
Plik	Rola
ApplicationDbContext	Centralny kontekst bazy danych: połączenia, konfiguracja tabel, relacje.
ModelBuilderExtensions	Wypełnianie bazy danych przykładowymi danymi startowymi. W wersji 'produkcyjnej' funkcjonalność została wyłączona.

Struktura użytkowników

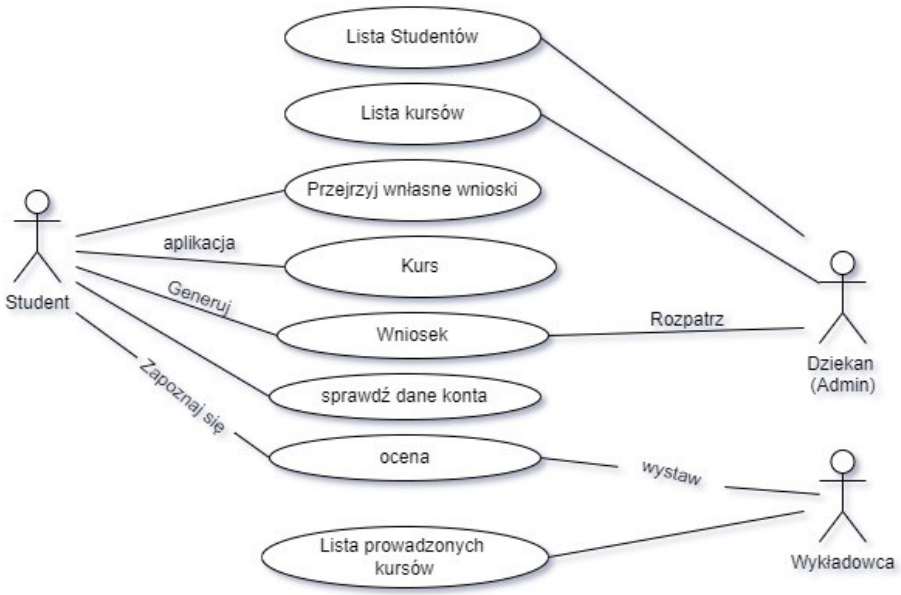
W Systemie Zarządzania Dziekanatu wykorzystałem mechanizm **ASP.NET Identity** do tworzenia kont użytkowników.

Użytkownicy w systemie mogą mieć przypisaną jedną rolę spośród możliwych: student, wykładowca, admin (dziekan**).**

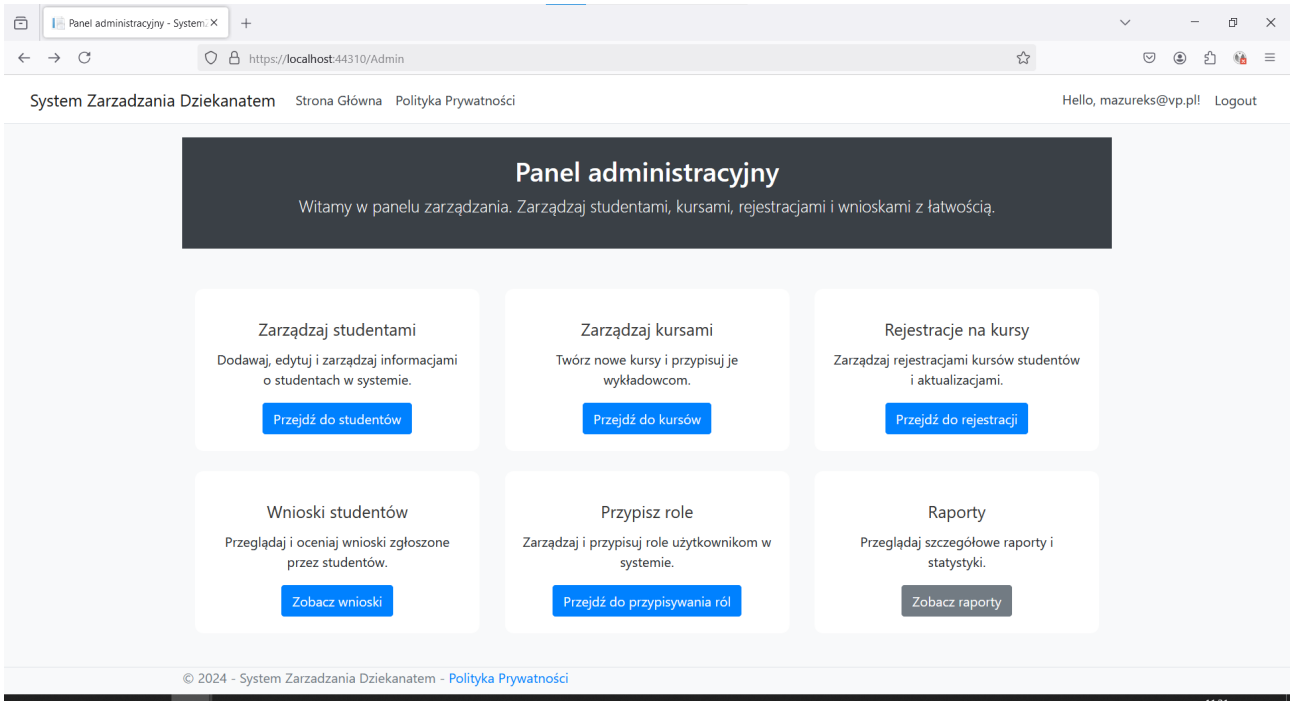
Pierwszy utworzony w ramach systemu użytkownik z automatu ma przypisaną rolę dziekana. Następni użytkownicy tworzeni są bez przypisanych ról. Rolę użytkownikom przypisuje lub zmienia dziekan w ramach modelu **AssignRolesViewModel.cs**



Możliwe aktywności poszczególnych aktorów przedstawia diagram przypadków użycia:

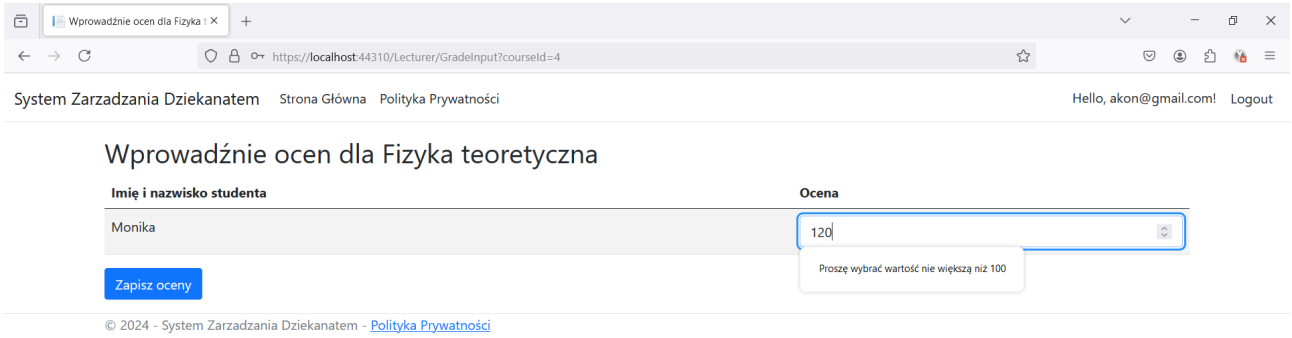


Użytkownik niezalogowany widzi stronę informacyjną systemu. Zalogowany użytkownik uzyskuje dostęp do funkcjonalności zgodnych z jego rolą w systemie:



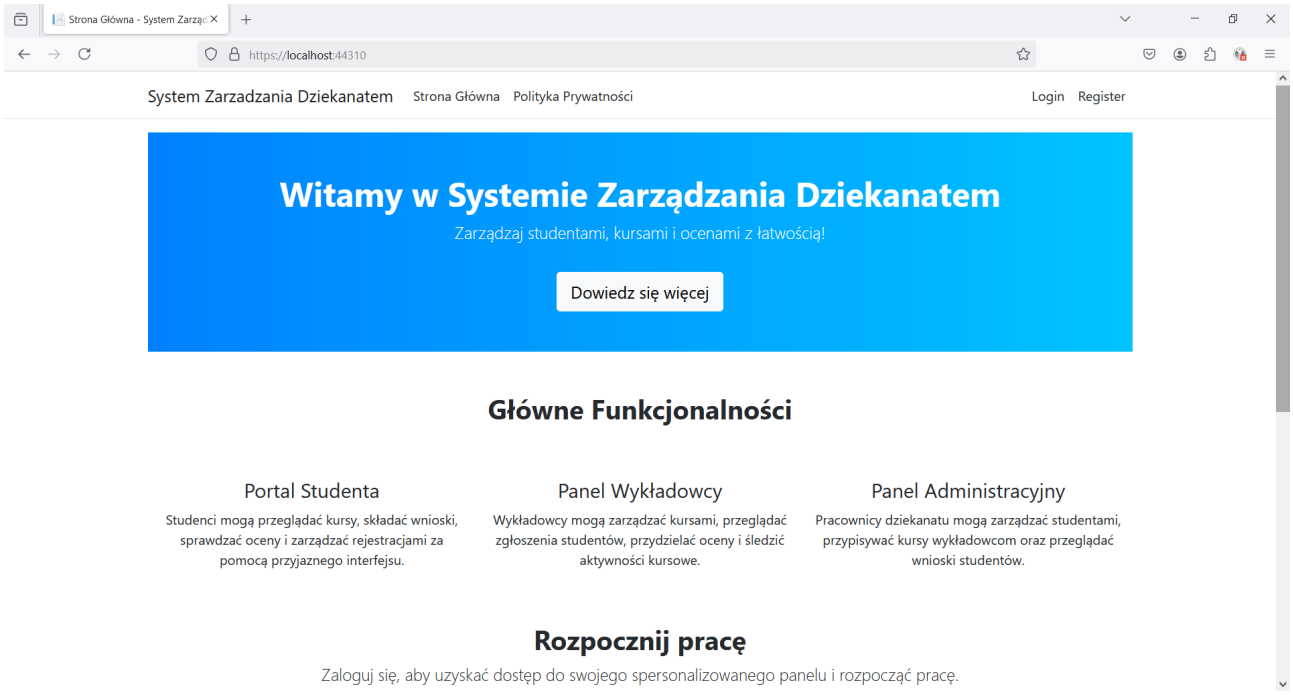
Walidacja

Dane wprowadzane przez użytkownika są walidowane pod względem zgodności merytorycznej.



Działanie systemu

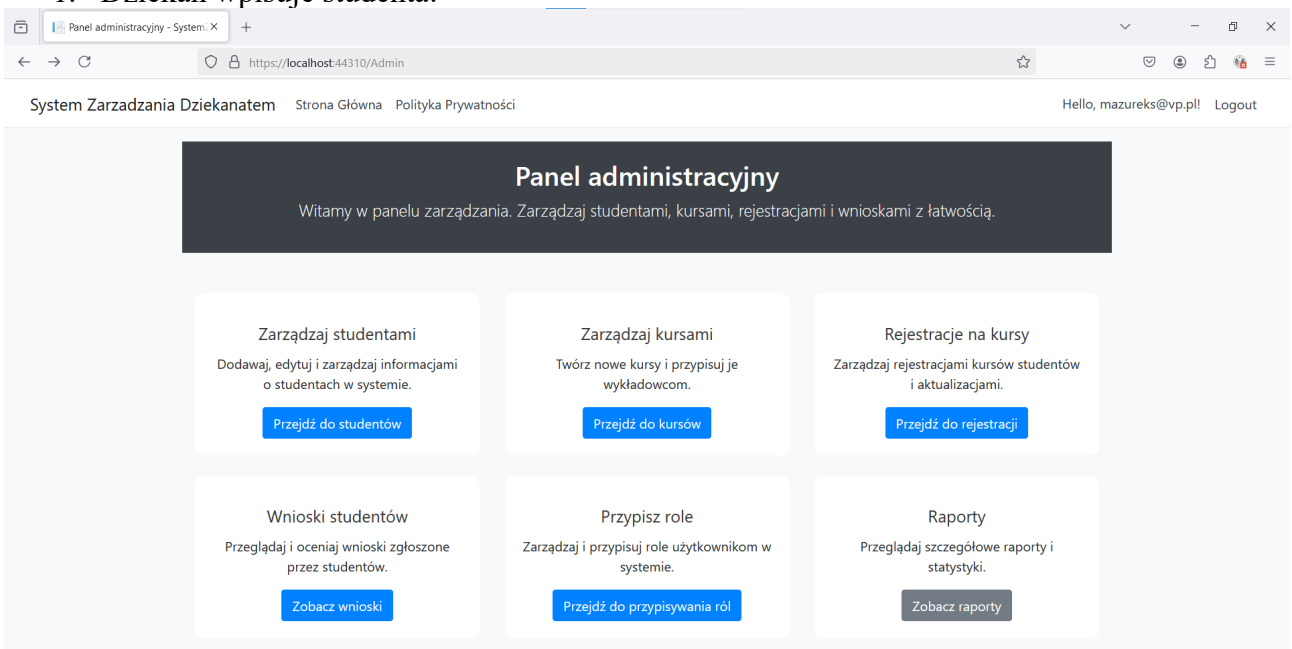
Widok niezalogowaego użytkownika:



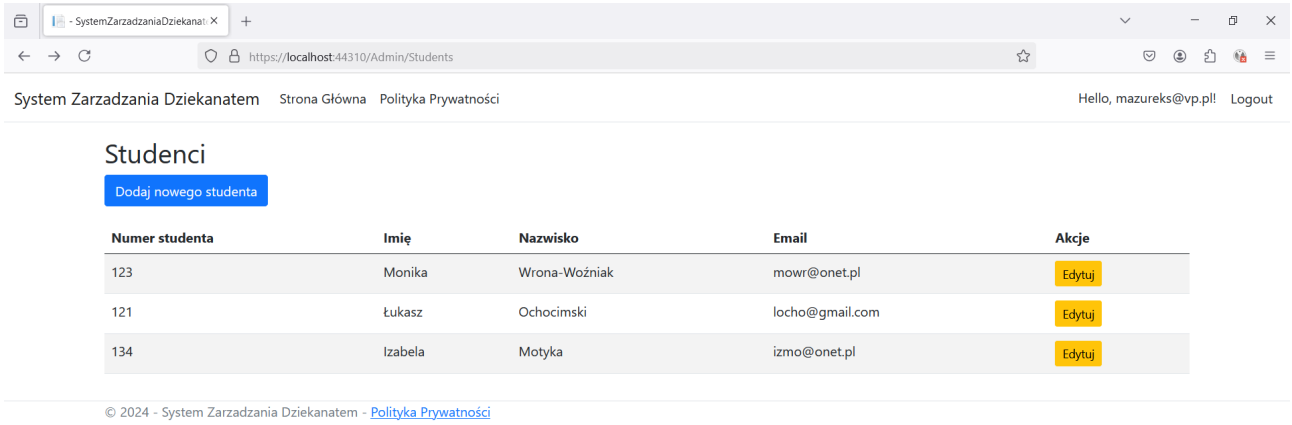
Aby można było zaprezentować funkcjonowanie systemu , oprócz dziekana, w systemie powinien być wpisany co najmniej jeden wykładowca i co najmniej jeden student. Zatem należy zadbać by ci użytkownicy byli zapisani.

Przykładowe scenariusze działania:

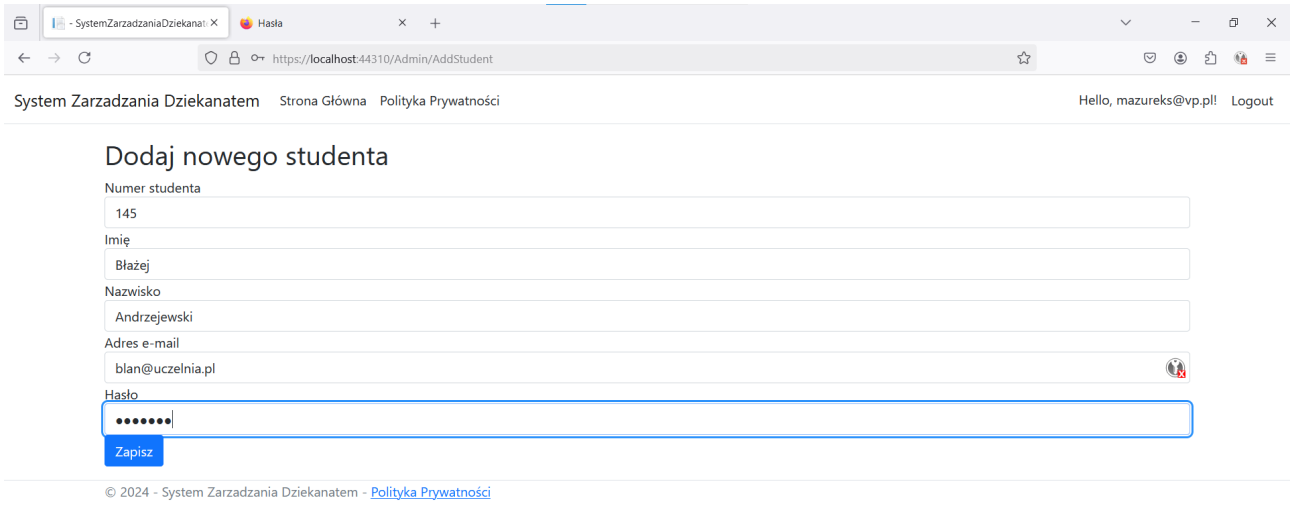
1. Dziekan wpisuje studenta:



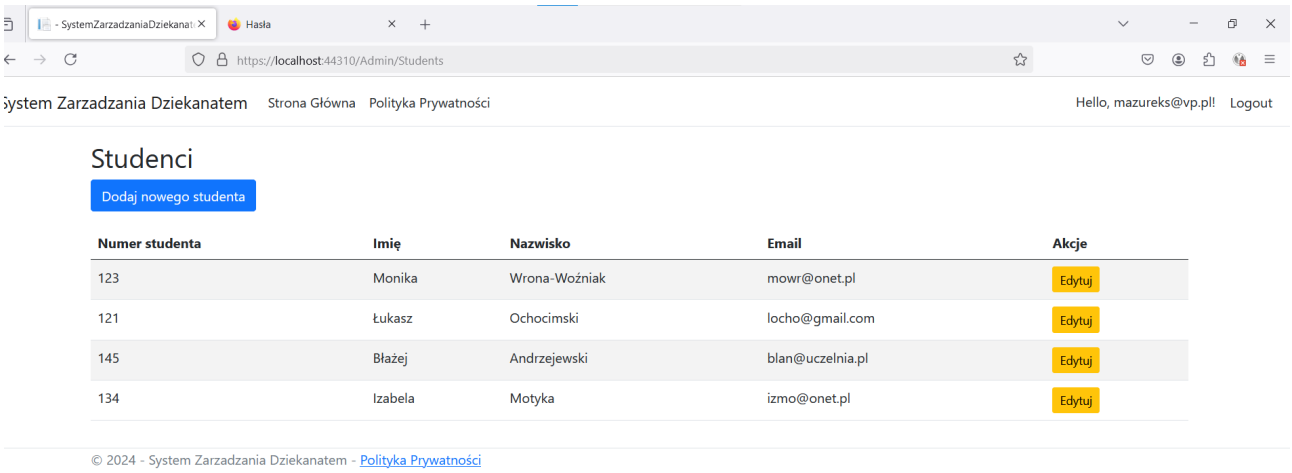
Dziekan na panelu administracyjnym klika „Przejdź do studentów”. Pojawia się widok podobny do tego poniżej:



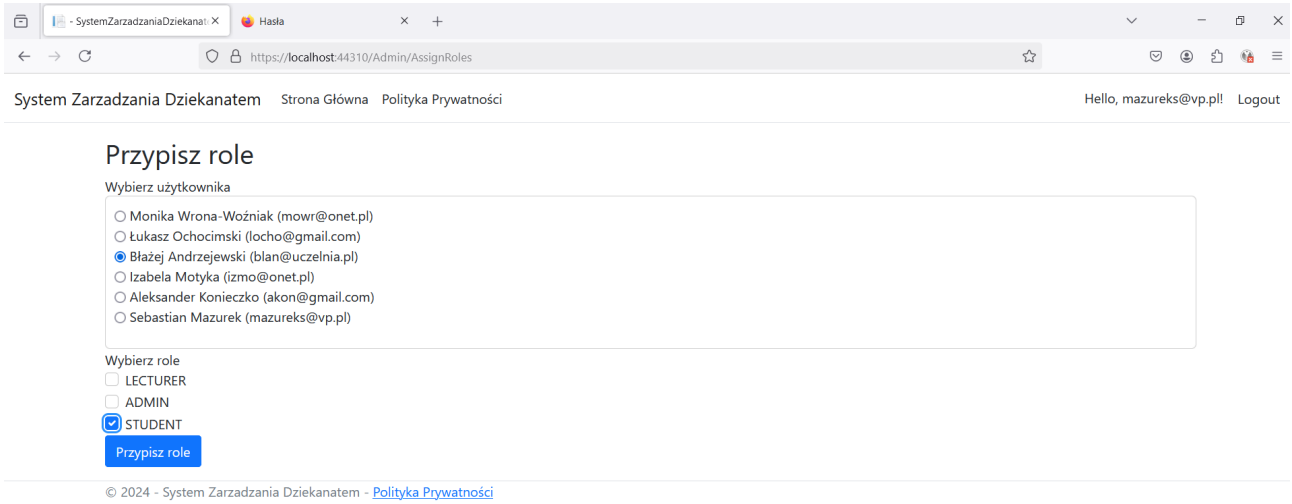
Dziekan klika „Dodaj nowego studenta”. Pojawia się widok jak poniżej:



Po uzupełnieniu danych dziekan klika zapisz. Widok powraca do wykazu studentów.

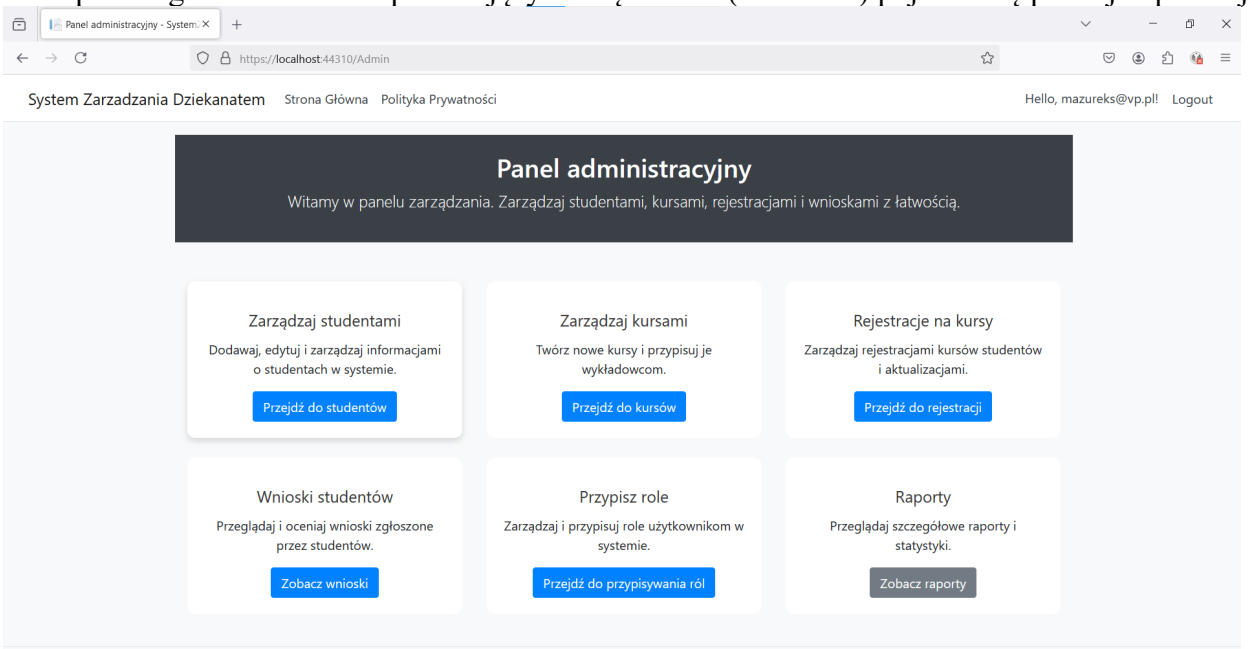


Studen jeszcze nie może korzystać z system, gdyż ma nie posiada przypisanej roli w systemie (w tym przypadku studenta). Przypisywanie ról użytkowniokmDziekan dokonuje w w panelu „Przypisz Rolę”. Czynność ta jest nieodzowna niezależnie od tego czy konto utworzył użytkownik metodą ASP.NET Identyty czy zostało utworzone prze dziekana (administratora).

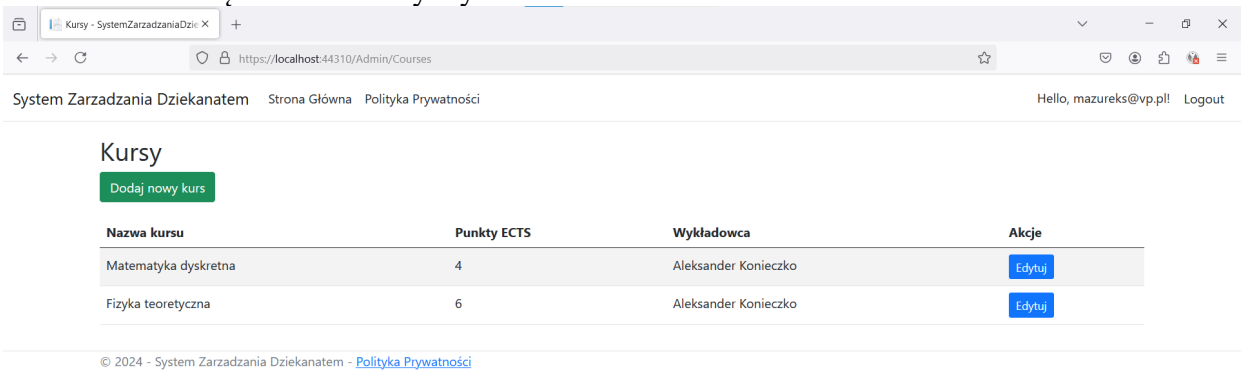


Dopiero po tej czynności użytkownik może się zalogować do systemu.

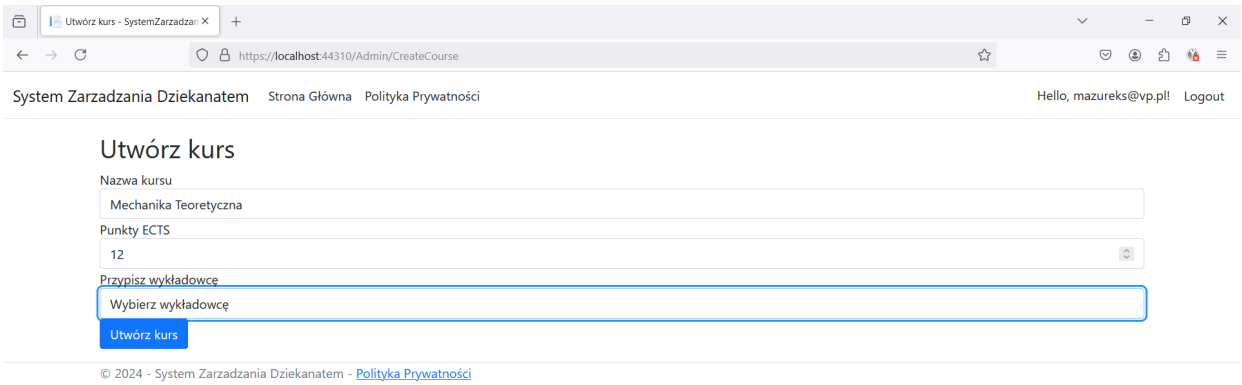
2. Dziekan tworzy nowy kurs:
   po zalogowaniu kontem posiadającym rolę Admin (Dziekana) pojawia się panel jak poniżej:



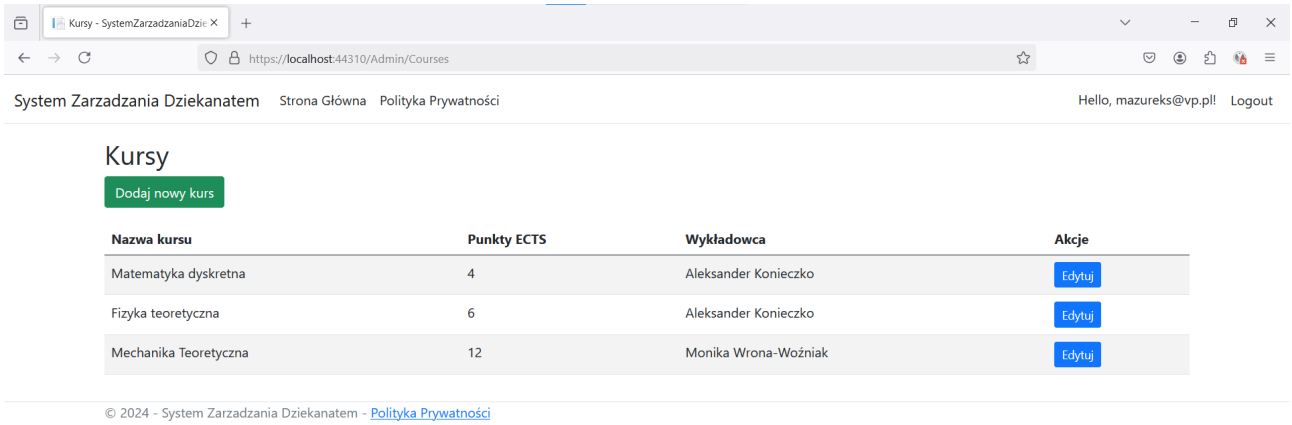
Dziekan klika przycisk „Przejdź do kursów”. Pojawia się nowy widok, ana którym widoczne są aktualne kursy wydziału / uczelni.



Dziekan klika przycisk „Dodaj nowy kurs”. Pojawi się formularz wprowadzenia nowego kursu. Należy wprowadić nazwę kursu, wagę kursu w systemie ECTS, oraz w menu rozwijanego wybrać przypisanego do kursu wykładowcę. Wykładowcy nie można wpisać „z ręki”. W chwili wprowadzania kursu wykładowca już musi być w systemie.



Po kliknięciu utwórz kurs, pojawia się zaktualizowany wykaz kursów:


Kursy - SystemZarzadzaniaDzie
https://localhost:44310/Admin/Courses
System Zarzadzania Dziekanatem
Strona Główna
Polityka Prywatności
Hello, mazureks@vp.pl!
Logout
Kursy
Dodaj nowy kurs
Nazwa kursu
Punkty ECTS
Wykładowca
Akcje
Matematyka dyskretna
4
Aleksander Konieczko
Edytuj
Fizyka teoretyczna
6
Aleksander Konieczko
Edytuj
Mechanika Teoretyczna
12
Monika Wrona-Woźniak
Edytuj
© 2024 - System Zarzadzania Dziekanatem - Polityka Prywatności