

UNIwersytet Zielonogórski

Wydział Informatyki, Elektrotechniki i Automatyki

Praca dyplomowa

Kierunek: Informatyka

PORTAL WWW DO PRZEPROWADZANIA
TESTÓW ZALICZENIOWYCH

Przemysław Cukierski

Promotor:
dr inż. Robert Szulim

Pracę akceptuję:

.....

(data i podpis promotora)

Zielona Góra, luty 2022

Streszczenie

Głównym zagadnieniem poruszonym w ramach zrealizowanej pracy dyplomowej jest budowa i działanie autorskiego systemu służącego do tworzenia i rozwiązywania testów zaliczeniowych w formie online. Praca zawiera informacje na temat utworzonej aplikacji, również w kontekście obecnie stosowanych rozwiązań tego typu. Omówiona zostaje implementacja wraz z przykładami kodu napisanego z użyciem języka C oraz technologii .NET.

Spis treści

| | |
|--|-----------|
| 1. Wstęp | 1 |
| 1.1. Wprowadzenie i przegląd literatury | 1 |
| 1.2. Cel i zakres pracy | 2 |
| 1.3. Struktura pracy | 2 |
| 2. Stos technologiczny | 5 |
| 2.1. Technologia .NET | 5 |
| 2.2. Platforma Blazor | 6 |
| 2.3. Wdrożenie aplikacji - Microsoft Azure | 7 |
| 3. Struktura aplikacji | 9 |
| 3.1. Struktura projektu ASP .NET | 9 |
| 3.2. Budowa i struktura bazy danych MS SQL | 10 |
| 4. Obsługa kont użytkowników | 13 |
| 4.1. Uwierzytelnianie i autentykacja w ASP .NET Core z użyciem Blazor | 13 |
| 4.2. Nadawanie ról użytkownikom za pomocą konta administratora | 15 |
| 4.3. Zmiana hasła oraz usuwanie konta użytkownika | 16 |
| 4.4. Przypisanie informacji o personaliach do użytkownika | 16 |
| 5. Proces przeprowadzenia testu zaliczeniowego | 19 |
| 5.1. Utworzenie grupy studenckiej przez użytkownika o roli wykładowcy | 19 |
| 5.2. Utworzenie formularza testowego wraz z pytaniami zamkniętymi | 20 |
| 5.3. Rozwiązanie utworzonego testu przez użytkownika o roli studenta | 22 |
| 5.4. Przeprowadzenie oceny rozwiązania testu przez użytkownika o roli wykładowcy | 27 |
| 5.5. Wyświetlenie formularza z oceną przyznaną za rozwiązanie egzaminu | 31 |
| 6. Dyskusja rezultatów i wnioski końcowe | 34 |

Spis rysunków

| | |
|---|----|
| 2.1. Widok App Service utworzonego w ramach wdrożenia aplikacji | 7 |
| 2.2. Szczegóły dotyczące wykorzystania usługi Azure SQL Database w ramach pracy dyplomowej | 8 |
| 3.1. Schemat struktury biblioteki klas platformy .NET Core | 9 |
| 3.2. Struktura projektu w eksploratorze rozwiązań Visual Studio | 10 |
| 3.3. Diagram przedstawiający tabele w bazie danych odnoszące się do procesu autentykacji | 11 |
| 3.4. Diagram przedstawiający tabele w bazie danych odnoszące się do funkcjonalności aplikacji | 11 |
| 4.1. Wybór typu uwierzytelniania przy tworzeniu aplikacji platformy Blazor | 13 |
| 4.2. Ekran startowy aplikacji | 14 |
| 4.3. Formularz rejestracji nowego użytkownika | 14 |
| 4.4. Widok panelu administracyjnego | 15 |
| 4.5. Widok panelu ustawień dostępnego dla użytkownika | 16 |
| 4.6. Przypisanie danych personalnych do użytkownika w roli studenta . . | 17 |
| 5.1. Dodanie grupy studenckiej - formularz | 19 |
| 5.2. Weryfikacja poprawności dodania grupy studenckiej w bazie danych . | 20 |
| 5.3. Widok umożliwiający wyszukanie egzaminu przeznaczonego do rozwiązania | 22 |
| 5.4. Formularz umożliwiający rozwiązanie egzaminu | 23 |
| 5.5. Weryfikacja poprawności dodania rekordu odwzierciedlającego rozwiązanie egzaminu w bazie danych | 26 |
| 5.6. Widok egzaminu przeznaczonego do oceny przez wykładowcę | 27 |
| 5.7. Widok egzaminu przeznaczonego do oceny przez wykładowcę - zliczanie zdobytych punktów oraz wybranie oceny do wystawienia | 28 |
| 5.8. Widok ocenionego egzaminu | 31 |

Rozdział 1

Wstęp

1.1. Wprowadzenie i przegląd literatury

1.1.1. Wprowadzenie

Niniejsza praca naukowa koncentruje się na temacie, jakim jest przeprowadzanie egzaminów zaliczeniowych w formie zdalnej za pośrednictwem Internetu. W ramach pracy została zrealizowana aplikacja webowa, która umożliwia tworzenie, rozwiązywanie oraz ocenianie egzaminów przeprowadzonych w takiej właśnie formie.

Temat ten zdaje się być ważny, szczególnie w kontekście trwającej pandemii choroby COVID-19, która przyczyniła się do wzrostu zapotrzebowania na rozwiązania umożliwiające naukę zdalną i zdalne sprawdzanie postępów w nauce, jakich dokonują uczniowie oraz studenci uczelni wyższych.

Zrealizowana aplikacja, poza wartością wynikającą z przydatności w obecnych warunkach, z jakimi ludzkość mierzy się w kontekście pandemii, jest także przykładem możliwości i zastosowania jednej z najnowszych technologii firmy Microsoft - mowa o technologii Blazor, której premiera miała miejsce w 2020 roku, a o której w szczegółach mowa będzie w późniejszej części niniejszej pracy.

1.1.2. Przegląd literatury

Rynek rozwiązań dotyczących przeprowadzania testów w formie online został zdominowany przez największych graczy takich jak Google czy Microsoft, które udostępniają swoje zaawansowane kreatory, umożliwiające tworzenie tego typu egzaminów na dużą skalę, cechujące się bardzo wysoką stabilnością i niezawodnością. Umożliwiają one dodawanie pytań zarówno jednokrotnego jak i wielokrotnego wyboru, pytań otwartych, a także pytań innych typów.

Ze względu na ograniczone zasoby czasowe oraz finansowe, oczywistym jest, że tworzona przez autora aplikacja w założeniach miała być raczej zbliżona do aplikacji mniej zaawansowanych, a przez to mniej popularnych wśród użytkowników. W tym miejscu autor chciałby krótko opisać kilka takich rozwiązań.

- **Online Quiz Creator** - aplikacja, która pomaga tworzyć ciekawe quizy z zaawansowanymi funkcjami, opcjami stylizacji i statystykami. Kreator quizów online integruje kilka elementów grywalizacji, funkcji społecznościowych i zasad uczenia się. Cena takiego rozwiązania zaczyna się od 50 dolarów miesięcznie, jednak dostępna jest darmowa wersja próbna.
- **QuestionWriter HTML5** - QuestionWriter to narzędzie do quizów na komputery stacjonarne z bogatymi opcjami dostosowywania. Można w nim tworzyć testy od podstaw lub skorzystać z jednego z 20 gotowych szablonów quizów. Narzędzie obsługuje format HTML5 gotowy na urządzenia mobilne, lecz design quizu nie jest responsywny, więc rozwiązywanie testów na urządzeniach mobilnych nie jest tak łatwe. Cena programu bez ograniczeń wynosi 495 dolarów.
- **QuizStar** - QuizStar to bezpłatna usługa online dla nauczycieli, która umożliwia tworzenie quizów dla uczniów, zapraszanie ich na zajęcia online oraz przeglądanie wyników testów w przeglądarce. Mimo, że testy przygotowuje się w przeglądarce, lepiej wypełniać je na komputerach PC, ponieważ witryna nie ma wersji mobilnej. [1]

1.2. Cel i zakres pracy

Celem pracy jest opracowanie portalu WWW umożliwiającego zdalne przeprowadzanie testów z użyciem technologii .NET Core oraz innych technologii Microsoftu. Zakres pracy obejmuje następujące zagadnienia:

- opracowanie założeń funkcjonalnych systemu
- przygotowanie struktury bazy danych
- opracowanie oprogramowania portalu WWW
- przygotowanie portalu do pracy na urządzeniach mobilnych
- testowanie i wdrożenie

1.3. Struktura pracy

Niniejsza praca dyplomowa została podzielona na 5 rozdziałów, na które składa się łącznie 17 podrozdziałów. Poniżej zostały one krótko scharakteryzowane, w celu przybliżenia tematyki, jaka będzie podejmowana w ramach pracy.

- **Wprowadzenie i przegląd literatury** - informacje na temat tego, co stanowi główne zagadnienie poruszane w pracy dyplomowej oraz powód wyboru jej tematyki.
- **Cel i zakres pracy** - informacje na temat celu pracy dyplomowej oraz zakresu podjętych w celu jej realizacji prac.

- **Struktura pracy** - charakterystyka podrozdziałów pracy dyplomowej.
- **Technologia .NET** - charakterystyka technologii .NET oraz uszczegółowienie korzyści płynących z jej zastosowania w kontekście aplikacji tworzonej w ramach niniejszej pracy.
- **Platforma Blazor** - charakterystyka najważniejszych właściwości platformy Blazor, która została zastosowana w ramach tworzonej aplikacji.
- **Wdrożenie aplikacji - Microsoft Azure** - krótkie przedstawienie technologii chmurowej Microsoftu oraz uzasadnienie wyboru tego właśnie rozwiązania do wdrożenia projektu tworzonego w ramach pracy.
- **Struktura projektu ASP .NET** - opis struktury aplikacji ASP .NET, charakterystyka modułów składających się na całość projektu.
- **Budowa i struktura bazy danych MS SQL** - przedstawienie korzyści płynących z wyboru bazy danych Microsoft SQL Server, przedstawienie jej struktury za pomocą odpowiednich diagramów.
- **Uwierzytelnianie i autentykacja w ASP .NET Core z użyciem Blazor** - charakterystyka technologii .NET oraz wymienione zostały korzyści płynące z jej zastosowania w kontekście aplikacji tworzone w ramach niniejszej pracy.
- **Nadawanie ról użytkownikom za pomocą konta administratora** - przedstawienie działania konta administratora, które pozwala nadawać role innym użytkownikom aplikacji, a także zmieniać ich hasła.
- **Zmiana hasła oraz usuwanie konta użytkownika** - przedstawienie możliwości, jakie daje panel ustawień dostępny dla użytkownika, czyli umożliwienie zmiany hasła do konta, pobranie kopii danych zalogowanego użytkownika bądź trwałe usunięcie konta.
- **Przypisanie informacji o personaliach do użytkownika** - zaprezentowanie implementacji funkcjonalności przypisania imienia i nazwiska (a także grupy studenckiej, w przypadku użytkownika o roli studenta) do danego użytkownika.
- **Utworzenie grupy studenckiej przez użytkownika o roli wykładowcy** - scharakteryzowanie procesu dodania nowej grupy studenckiej oraz zaprezentowanie odpowiedniej zmiany w bazie danych.
- **Utworzenie formularza testowego wraz z pytaniami zamkniętymi** - opis funkcjonalności utworzenia nowego egzaminu przez wykładowcę, wraz z uwzględnieniem wykonanej walidacji danych oraz struktury komponentu Blazor odpowiedzialnego za tę funkcjonalność.

- **Rozwiązanie utworzonego testu przez użytkownika o roli studenta** - przedstawienie implementacji funkcjonalności rozwiązania utworzonego egzaminu wraz ze szczegółowym opisem zastosowanych w tym celu rozwiązań i algorytmów.
- **Przeprowadzenie oceny rozwiązania testu przez użytkownika o roli wykładowcy** - zaprezentowanie procesu oceniania rozwiązane egzaminu, wraz ze szczegółowym opisem implementacji
- **Wyświetlenie formularza z oceną przyznaną za rozwiązanie egzaminu** - prezentacja ostatniego elementu procesu, który stanowi główną funkcjonalność aplikacji, czyli przeprowadzania egzaminu w formie zdalnej, mianowicie wyświetlenia przyznanej przez wykładowcę oceny. Wraz z opisem implementacji funkcjonalności w podrozdziale został umieszczony odpowiedni fragment kodu aplikacji.

Rozdział 2

Stos technologiczny

2.1. Technologia .NET

2.1.1. Czym jest platforma .NET?

Platforma .NET składa się z zestawu klas i maszyny wirtualnej zwanej CLR (Common Language Runtime - środowisko uruchomieniowe wspólnego języka). CLR jest środowiskiem wykonawczym dla kodu zapisanego koncepcyjnie w języku pośrednim, podobnie jak w przypadku kodu bajtowego języka Java. Common Language Runtime zapewnia usługi takie jak zarządzanie pamięcią, obsługę wyjątków, dbanie o bezpieczeństwo aplikacji, wersjonowanie, debugowanie i profilowanie kodu. a w sumi Ważnym aspektem platformy programistycznej .NET jest możliwość mieszania różnych języków programowania w ramach tego samego projektu - szczególnie widoczne będzie to w kontekście frameworka Blazor, który umożliwia tworzenie kodu frontend-owego w języku HTML wraz z kodem backend-owym pisanym w języku programowania C# w ramach jednego pliku.

2.1.2. Korzyści płynące z zastosowania platformy .NET Core

Odnosząc się do starszej technologii Microsoftu służącej do budowania aplikacji internetowych, czyli .NET Framework można wyróżnić korzyści płynące ze stosowania nowszej technologii tego samego typu, mianowicie platformy .NET Core Framework. Przykłady takich korzyści są następujące:

- platforma .NET Core Framework i powiązane z nią narzędzia oparte są na otwartym kodzie źródłowym.
- platforma .NET Core Framework może być wdrażana razem z aplikacją, natomiast pełna platforma .NET Framework może być jedynie instalowana na komputerze docelowym i wspólnie wykorzystywana przez wszystkie aplikacje, co jest źródłem problemów z wersjonowaniem.
- platforma .NET Core Framework może być wykorzystywana do tworzenia aplikacji międzyplatformowych, które mogą też działać w systemach operacyjnych Linux i Mac.

2.2. Platforma Blazor

2.2.1. WebAssembly

Szybki rozwój Internetu spowodował konieczność ciągłego podnoszenia wydajności aplikacji webowych, a co za tym idzie, podnoszenia wydajności technologii, w jakich są one tworzone. Odpowiadając na tą potrzebę, w marcu 2017 roku zaprezentowany został język WebAssembly (WASM) — drugi po JavaScript język programowania w historii (i jednocześnie pierwszy język binarny) obsługiwany przez przeglądarki [2].

Istnieje także możliwość tworzenia rozwiązań, w których aplikacja działająca w ramach WebAssembly będzie korzystać z bibliotek JavaScript i na odwrót. Dzięki temu jesteśmy w stanie wykorzystać to wszystko, czego używamy podczas tworzenia aplikacji w JavaScript. Dzięki WebAssembly można także przenieść technologie wykorzystywane do tej pory po stronie backendowej do tworzenia frontendu, tak jak to jest w przypadku .NET i C# w Blazorze.

2.2.2. Blazor

Blazor to jeden z frameworków, które powstały na bazie WebAssembly, dzięki któremu możemy tworzyć aplikacje SPA w .NET, C oraz Razor, która następnie wykona się po stronie przeglądarki. Framework ten umożliwia tworzenie interfejsów API przy użyciu języka C# (zamiast języka JavaScript) oraz udostępnianie logiki aplikacji utworzonej z użyciem technologii .NET. Platforma Blazor może funkcjonować w dwóch trybach [3] [4]:

- Client-Side - jest to tryb oparty na WebAssembly, gdzie całość wykonuje się po stronie przeglądarki.
- Server-Side - kod wykonuje się po stronie serwera, a wszystkie zdarzenia z przeglądarki oraz wygenerowany HTML są przesyłane do przeglądarki dzięki Signal R.

Wspomniany SignalR jest z kolei frameworkiem dla ASP.NET ułatwiającym pisanie aplikacji czasu rzeczywistego, w których nie chcemy przeładowywać całej strony, lecz tylko jej poszczególne fragmenty. Jest to usprawniona wersja choćby takiej technologii jak AJAX, który umożliwiał tylko jednostronną komunikację między przeglądarką a serwerem. SignalR ze względu na to, że jest oparty o Web Socket API umożliwia pełną komunikację między przeglądarką a serwerem. Możliwe jest zatem dzięki Web Socket, że klient wysyła w tle wiadomość do serwera jak również serwer do przeglądarki (co we wcześniejszych technologiach nie było możliwe). [5]

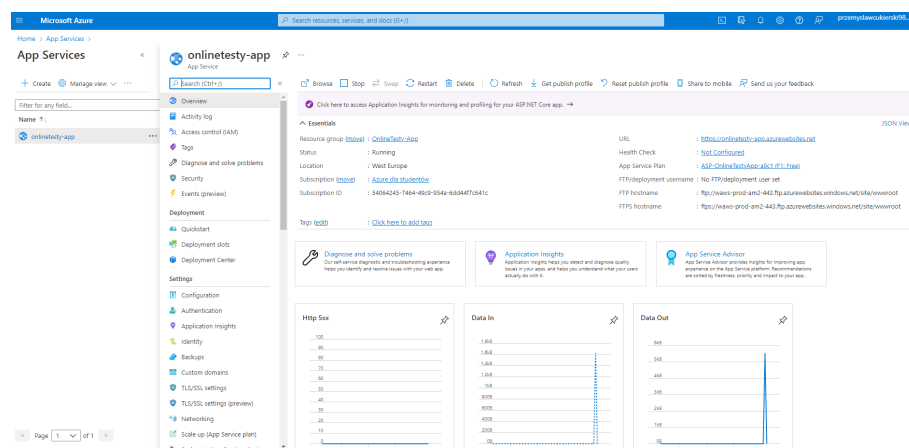
Użycie technologii Blazor w utworzonej przez autora aplikacji nie tylko wpłynęło na poprawę jej wydajności (w porównaniu z potencjalną wydajnością przy użyciu jednej ze starszych technologii) lecz także pozwoliło na utworzenie warstwy frontendowej aplikacji bez konieczności nauki dodatkowych frameworków (Angular, VueJS).

2.3. Wdrożenie aplikacji - Microsoft Azure

Microsoft Azure to platforma chmurowa udostępniająca mechanizmy pozwalające przetwarzać i składować dane. Platforma ta zbudowana jest z grupy trzech technologii zapewniających specjalizowany zestaw możliwości dla programistów. Są to: [6]

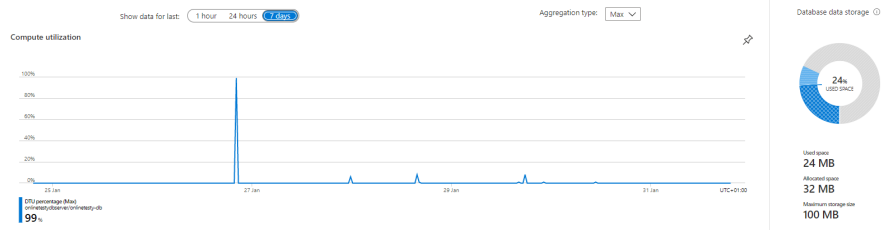
- Windows Azure, zapewniające przestrzeń do składowania danych i środowisko uruchomieniowe.
- SQL Azure, które zapewnia możliwość używania relacyjnej bazy danych zaimplementowanej do uruchamiania w chmurze.
- Windows Azure Platform AppFabric, który jest komponentem dostarczającym dodatkowe funkcjonalności w formie usług.

W celu wdrożenia utworzonej przez autora aplikacji został założony nowy App Service. Jest to usługa, która umożliwia tworzenie i hostowanie aplikacji internetowych utworzonych w wybranym języku programowania bez zarządzania infrastrukturą. Umożliwia ona skonfigurowanie ciągłej integracji, zapewnia wysoką dostępność oraz bezpieczeństwo. Na ilustracji poniżej widoczny jest panel App Service utworzony przez autora aplikacji.



Rys. 2.1. Widok App Service utworzonego w ramach wdrożenia aplikacji

Aby móc przechowywać dane w chmurze autor zastosował rozwiązanie Azure SQL Database, które jest w pełni zintegrowane z Microsoft SQL Server. Koszt bazy danych, która umożliwia przechowywanie maksymalnie 100 MB danych wynosi około 5 euro miesięcznie. Poniżej przedstawiono wykres obrazujący zapytania wysłane do bazy danych na przestrzeni ostatnich 7 dni oraz procentowe zużycie dostępnych zasobów.



Rys. 2.2. Szczegóły dotyczące wykorzystania usługi Azure SQL Database w ramach pracy dyplomowej

Dlaczego do wdrożenia aplikacji autor wybrał rozwiązanie Microsoftu?

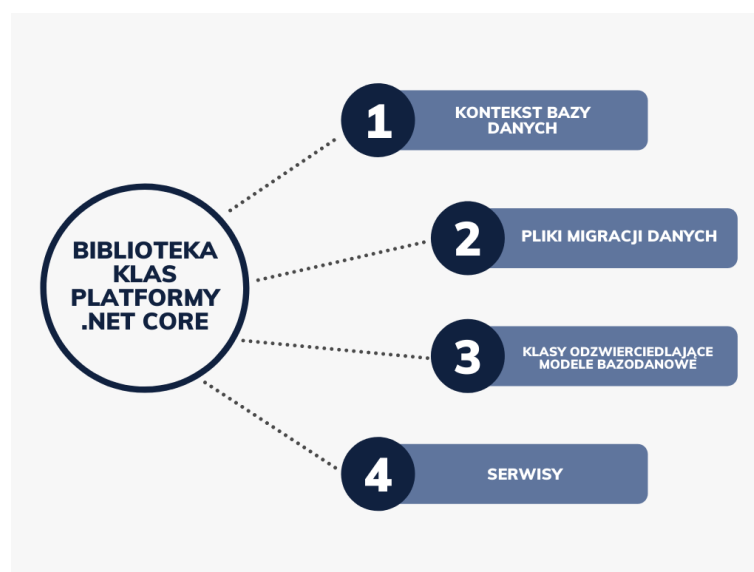
- stabilność działania na poziomie 99,95 procent, dzięki czemu użytkownicy mają pewność nieprzerwanego dostępu do usług, maszyn i aplikacji.
- zapewnienie ciągłego monitoringu i całodobowej pomocy technicznej
- bardzo dobry stosunek ceny do wydajności i jakości oferowanych usług

Rozdział 3

Struktura aplikacji

3.1. Struktura projektu ASP .NET

Utworzona aplikacja składa się z dwóch modułów. Pierwszym z nich jest biblioteka klas platformy .NET Core, która zawiera elementy takie jak kontekst bazy danych, pliki dotyczące migracji danych związane z technologią Entity Framework, klasy odzwierciedlające modele bazodanowe oraz serwisy. Jest to moduł, który w dużej mierze odpowiada za backend-ową część aplikacji.



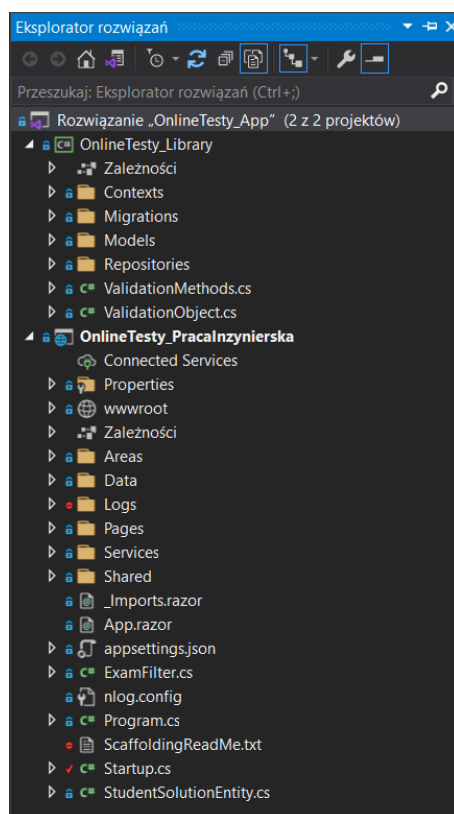
Rys. 3.1. Schemat struktury biblioteki klas platformy .NET Core

Drugim z modułów jest aplikacja internetowa ASP .NET Core oparta na frameworku Blazor, który został opisany w jednym z poprzednich rozdziałów. Zawartość poszczególnych folderów można opisać w następujący sposób.

- Areas - folder, w którym znajdują się katalogi oraz pliki związane z widokami autentykacji i uwierzytelniania użytkownika. Jest to folder tworzony automatycznie, podczas generowania projektu za pomocą narzędzia Visual Studio przy uwzględnieniu opcji autoryzacji za pomocą pojedynczych kont użytkowników.

- Data - folder zawierający kontekst bazy danych zawierającej informacje o kontaktach użytkowników
- Logs - folder przeznaczony do przechowywania logów aplikacji ASP .NET Core
- Pages - folder, w którym znajdują się komponenty Blazor, odzwierciedlające widoki poszczególnych stron, po których użytkownik może nawigować.
- Shared - folder, w którym znajdują się komponenty współdzielone przez strony, takie jak menu oraz plik MainLayout.cshtml, który określa ogólny wygląd aplikacji.

Moduł ten zawiera również klasę Startup.cs, która jest głównym plikiem konfiguracyjnym aplikacji webowej. W tym pliku definiujemy wszystkie najważniejsze zależności, takie jak odwołania do plików definiujących kontekst bazy danych jak również do klas serwisów. W tym pliku określamy także z jakich middleware korzysta tworzona aplikacja - jest to definiowane wewnątrz metody Configure.



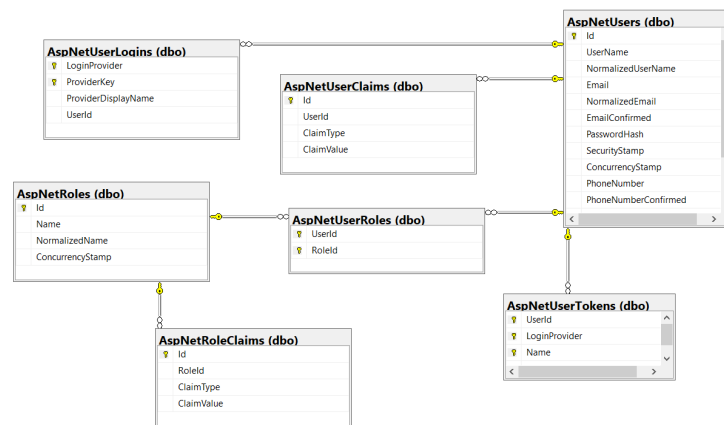
Rys. 3.2. Struktura projektu w eksploratorze rozwiązań Visual Studio

3.2. Budowa i struktura bazy danych MS SQL

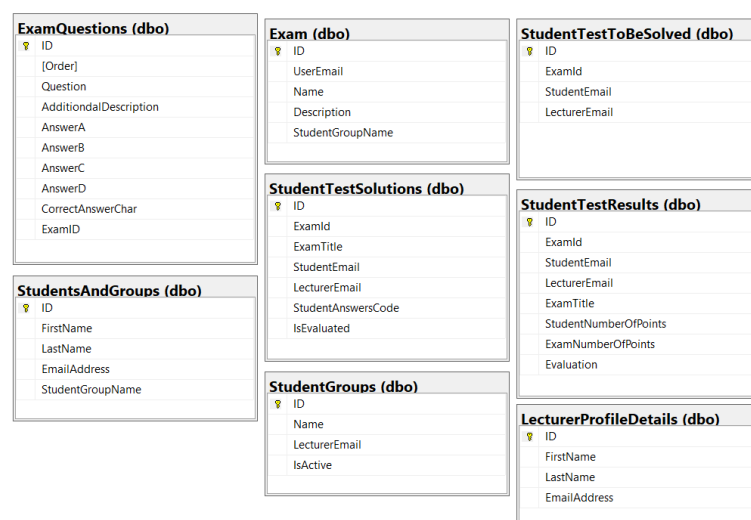
Do zarządzania bazą danych w ramach utworzonej aplikacji wybrany został system bazodanowy Microsoft SQL Server (MS SQL) w wersji 15. Jest to platforma bazodanowa typu klient-serwer. Wybrana została ona przez autora ze względu na wysoką wydajność, niezawodność oraz skalowalność [1].

Strukturę bazy danych można podzielić na dwie części. Pierwsza z nich zawiera tabele wykorzystywane w procesie autentykacji. Główną tabelą na tym diagramie jest tabela z użytkownikami, z nią powiązane są pozostałe tabele. Każdy użytkownik posiada przypisaną rolę, zatem musi istnieć tabela zawierająca zdefiniowane role, jak również tabela, która mówi o powiązaniach między danym użytkownikiem a rolą. Dodatkowo elementem bazy danych są tabele zawierające claimy (pary klucz-wartość), które są istotne w procesie autoryzacji.

Pozostałe tabele odnoszą się do konkretnych funkcjonalności, które zostały zaimplementowane w ramach utworzonego rozwiązania. Znajdujące się na poniższym diagramie tabele są odzwierciedleniem egzaminów oraz pytań egzaminacyjnych jednokrotnego wyboru, grup studenckich, rozwiązań nadesłanych przez studentów jak również rezultatów otrzymanych za te właśnie rozwiązania. Istotnymi elementami tej struktury są również tabele zawierające szczegółowe informacje na temat użytkowników - osobne tabele przechowują informacje o studentach oraz wykładowcach.



Rys. 3.3. Diagram przedstawiający tabele w bazie danych odnoszące się do procesu autentykacji



Rys. 3.4. Diagram przedstawiający tabele w bazie danych odnoszące się do funkcjonalności aplikacji

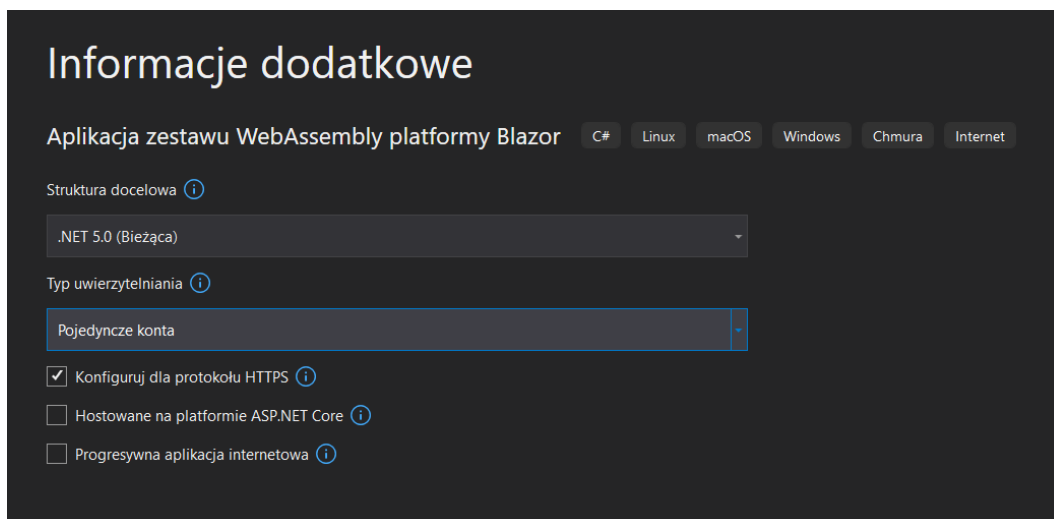
Rozdział 4

Obsługa kont użytkowników

4.1. Uwierzytelnianie i autentykacja w ASP .NET Core z użyciem Blazor

W celu zaimplementowania uwierzytelniania kont użytkowników w tworzonej aplikacji, przy tworzeniu projektu WebAssembly platformy Blazor została przez autora wybrana jako typ uwierzytelniania opcja "Pojedyncze konta".

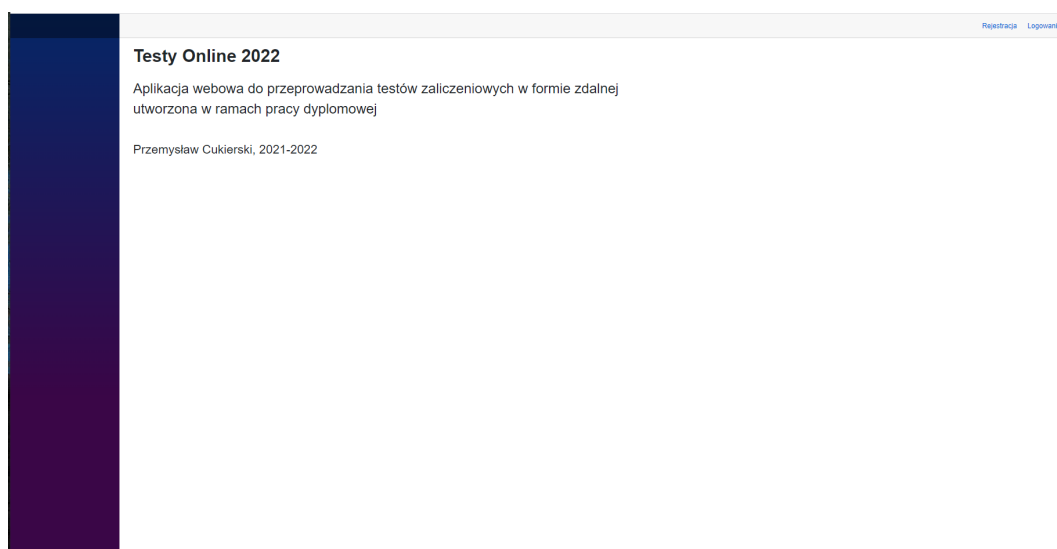
Wybór tej opcji powoduje automatyczne wygenerowanie odpowiedniego szkieletu aplikacji oraz modeli bazodanowych, które odpowiadają wszystkim zasobom, jakie potrzebne są w procesie autoryzacji oraz autentykacji.



Rys. 4.1. Wybór typu uwierzytelniania przy tworzeniu aplikacji platformy Blazor

Domyślnie generowane są komponenty Razor odpowiedzialne za proces rejestracji i logowania użytkownika. Istnieje możliwość dodania innych komponentów, odpowiedzialnych za istotne funkcjonalności dotyczące procesu uwierzytelniania.

Uwierzytelnianie użytkowników to funkcjonalność od której warto rozpocząć opis działania aplikacji. Do formularza rejestracji lub logowania użytkownika użytkownik ma możliwość przejścia w pierwszej kolejności po uruchomieniu aplikacji w przeglądarce. Ekran startowy prezentuje się jak na ilustracji poniżej.



Rys. 4.2. Ekran startowy aplikacji

W górnym panelu ekranu startowego znajdują się przyciski służące do utworzenia nowego konta użytkownika oraz zalogowania się przez użytkownika już wcześniej utworzonego. W celu wyświetlenia formularza rejestracji należy wybrać opcję "Rejestracja". Wyświetlony zostanie formularz, w którym należy podać adres e-mail użytkownika oraz hasło. Należy również wprowadzić ponownie to samo hasło, celem potwierdzenia, że nie doszło do pomyłki przy jego wpisywaniu.

Formularz jest odpowiednio walidowany - adres e-mail musi posiadać odpowiedni format. Hasło użytkownika nie może być krótsze niż 8 znaków, musi zawierać przynajmniej jedną liczbę oraz znak specjalny.

The image shows the registration form titled 'Rejestracja nowego użytkownika' with the subtitle 'Zarejestruj się!'. It contains three input fields: 'Adres e-mail', 'Hasło', and 'Potwierdź hasło'. Below the fields is a blue button labeled 'Zarejestruj'. The top of the form has the text 'Testy Online 2022' and links for 'Rejestracja' and 'Logowanie'.

Rys. 4.3. Formularz rejestracji nowego użytkownika

Walidacja danych odbywa się z użyciem wyrażeń regularnych. Takie wyrażenia mogą być użyte dzięki specjalnej adnotacji, która jest podawana przed deklaracją zmiennej w klasie modelu. Dodatkowo wymagalność pola można zadeklarować za pomocą adnotacji "Required", co również zostało zastosowane, jak widać w poniższym fragmencie kodu źródłowego aplikacji.

```

1 public class InputModel
2 {
3     [Required(ErrorMessage = "Adres e-mail jest polem wymagany")]
4     [EmailAddress(ErrorMessage = "Niepoprawny format adresu e-mail")]
5     [Display(Name = "Email")]
6     public string Email { get; set; }
7
8     [Required(ErrorMessage = "Hasło jest polem wymagany")]
9     [RegularExpression(@"^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)
10    (?=.*[!@#$%^&*])[A-Za-z\d@#$!%*?&]{8,}$",
11     ErrorMessage = "Hasło musi zawierać minimum 8 znaków,
12     jedna duża i mała litera, cyfry oraz znak specjalny")]
13     [DataType(DataType.Password)]
14     [Display(Name = "Password")]
15     public string Password { get; set; }
16
17     [DataType(DataType.Password)]
18     [Display(Name = "Confirm password")]
19     [Compare("Password", ErrorMessage = "Wprowadzone hasła
20     różnią się od siebie")]
21     public string ConfirmPassword { get; set; }
22 }

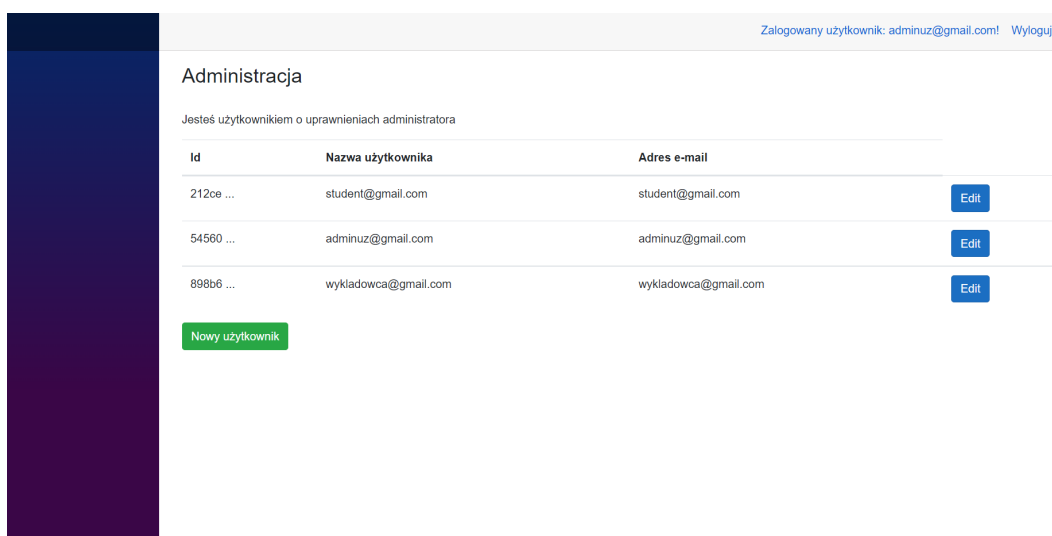
```

Listing 4.1. Klasa modelu rejestracji nowego użytkownika

Podobny mechanizm został zastosowany w przypadku operacji logowania użytkownika do systemu. Oba pola są w tym przypadku wymagane, ponadto adres e-mail musi posiadać odpowiedni format.

4.2. Nadawanie ról użytkownikom za pomocą konta administratora

Funkcjonowanie opisywanej aplikacji opiera się o wykorzystanie dwóch podstawowych ról użytkowników: roli wykładowcy oraz studenta. Do nadawania tych ról użytkownikom potrzebna jest jednak jeszcze jedna rola, mianowicie rola administratora. W zaprojektowanym systemie to administrator za pomocą swojego specjalnego uprawnienia przypisuje użytkownikom odpowiednie role. Po zalogowaniu na swoje konto administratorowi ukazuje się następujący panel.



Rys. 4.4. Widok panelu administracyjnego

Po wybraniu przycisku edycji administratorowi ukazuje się okno modalne, w którym istnieje możliwość zmiany hasła danego użytkownika oraz jego przypisania do danej roli.

4.3. Zmiana hasła oraz usuwanie konta użytkownika

Zmiana hasła użytkownika jest możliwa nie tylko z poziomu konta administratora, lecz także z poziomu konta danego użytkownika. Po zalogowaniu się na swoje konto należy kliknąć w widoczny w prawym górnym rogu dowolnego widoku napis "Zalogowany użytkownik". Po wybraniu tej opcji użytkownikowi ukaże się następujący widok.

Testy Online 2022 Konfiguracja konta użytkownika Wyloguj się

Manage your account
Change your account settings

[Twoje dane](#)

[Zmień hasło](#)

[Pobierz kopię danych / usuń konto](#)

Zmień hasło użytkownika

Aktualne hasło

Nowe hasło

Potwierdź nowe hasło

[Zatwierdź zmiany](#)

© 2021 - OnlineTesty_Pracalniczyńska - [Privacy](#)

Rys. 4.5. Widok panelu ustawień dostępnego dla użytkownika

Z poziomu wyświetlonego widoku można skorzystać z następujących opcji:

- Twoje dane - wyświetla adres e-mail zalogowanego użytkownika
- Zmiana hasła - umożliwia zmianę hasła konta zalogowanego użytkownika
- Pobierz kopię danych/usuń konto - umożliwia pobranie kopii danych zalogowanego użytkownika w formacie JSON lub trwałe usunięcie konta użytkownika

4.4. Przypisanie informacji o personaliach do użytkownika

Pierwszą rzeczą, jaką powinien zrobić użytkownik po pierwszym zalogowaniu w aplikacji powinno być przypisanie do swojego konta podstawowych informacji, takich jak imię i nazwisko oraz (w przypadku studenta) przynależność do grupy studenckiej. W tym celu zarówno w widoku dostępnym dla wykładowcy jak i w widoku dostępnym dla studenta należy wybrać zakładkę "Ustawienia konta".

Po wybraniu tej zakładki wyświetlony zostanie formularz, w którym zarówno dla wykładowcy jak i studenta znajdować się będą pola edytowalne umożliwiające podanie imienia i nazwiska użytkownika. Dodatkowo dla studenta dostępna będzie lista rozwijana zawierająca grupy studenckie zarejestrowane w systemie.

Rys. 4.6. Przypisanie danych personalnych do użytkownika w roli studenta

Dane personalne dotyczące użytkowników są przechowywane w bazie danych, odpowiednio w tabelach StudentAndGroups (dla studenta) oraz LecturerProfileDetails (dla wykładowcy). W całej aplikacji, aby zachować odpowiednią architekturę kodu zastosowane zostały klasy serwisów. Serwisy są klasami pośredniczącymi między klasami DAO - odzwierciedlającymi tabele w bazie danych - a widokami.

Na listingu poniżej przedstawiono przykład serwisu odpowiadającego za operacje wykonywane z użyciem tabeli LecturerProfileDetails. W celu zachowania dobrych praktyk każdy serwis implementuje odpowiadający mu interfejs.

```

1  using Microsoft.AspNetCore.Components.Authorization;
2  using Microsoft.AspNetCore.Http;
3  using OnlineTesty_Library.Contexts;
4  using OnlineTesty_Library.Models;
5  using System;
6  using System.Collections.Generic;
7  using System.Linq;
8  using System.Security.Claims;
9  using System.Text;
10 using System.Threading.Tasks;
11
12 namespace OnlineTesty_Library.Repositories
13 {
14     public class LecturerProfileDetailsRepositories : BaseRepositoryEF,
15         ILecturerProfileDetailsRepositories
16     {
17         private readonly AuthenticationStateProvider _authenticationStateProvider;
18         public LecturerProfileDetailsRepositories(IUnitOfWork unitOfWork,
19             AuthenticationStateProvider authenticationStateProvider) : base(unitOfWork)
20         {
21             _authenticationStateProvider = authenticationStateProvider;
22         }
23
24         public Guid AssignProfileDetailsToLecturer(LecturerProfileDetails model)
25         {
26             string userEmail = _authenticationStateProvider.GetAuthenticationStateAsync().
27                 Result.User.Identity.Name;
28
29             this.GetDbSet<LecturerProfileDetails>().Add(model);

```

```

30         this.UnitOfWork.SaveChanges();
31         return model.ID;
32     }
33
34     public IEnumerable<LecturerProfileDetails> FindAll()
35     {
36         return this.GetDbSet<LecturerProfileDetails>();
37     }
38
39     public LecturerProfileDetails Read(string emailAddress)
40     {
41         return this.GetDbSet<LecturerProfileDetails>()
42             .Where(e => e.EmailAddress == emailAddress).FirstOrDefault();
43     }
44
45     public LecturerProfileDetails ReadByLastName(string lastName)
46     {
47         return this.GetDbSet<LecturerProfileDetails>()
48             .Where(e => e.LastName == lastName).FirstOrDefault();
49     }
50
51     public bool WhetherEmailIsInTheDb(LecturerProfileDetails model)
52     {
53         string userEmail = _authenticationStateProvider.GetAuthenticationStateAsync().
54             Result.User.Identity.Name;
55         var allRecords = FindAll();
56         bool emailIsInDatabase = false;
57
58         model.EmailAddress = userEmail;
59
60         foreach (var item in allRecords)
61         {
62             if (item.EmailAddress.Equals(model.EmailAddress))
63             {
64                 emailIsInDatabase = true;
65             }
66         }
67
68         if (emailIsInDatabase)
69         {
70             return true;
71         }
72
73         else
74         {
75             return false;
76         }
77     }
78 }
79
80 }
81
82 public interface ILecturerProfileDetailsRepositories
83 {
84     Guid AssignProfileDetailsToLecturer(LecturerProfileDetails model);
85     LecturerProfileDetails Read(string emailAddress);
86     LecturerProfileDetails ReadByLastName(string lastName);
87     IEnumerable<LecturerProfileDetails> FindAll();
88     bool WhetherEmailIsInTheDb(LecturerProfileDetails model);
89 }
90 }

```

Listing 4.2. Klasa serwisu LecturerProfileDetailsRepositories

Przedstawiona powyżej klasa serwisu zawiera metody odpowiadające za możliwość znalezienia danych personalnych wszystkich wykładowców, odczytanie danych wykładowcy przyjmując za parametr jego nazwiska bądź adres e-mail a także na sprawdzenie czy wykładowca o danym nazwisku nie został już przypisany wcześniej dane personalne.

Rozdział 5

Proces przeprowadzenia testu zaliczeniowego

5.1. Utworzenie grupy studenckiej przez użytkownika o roli wykładowcy

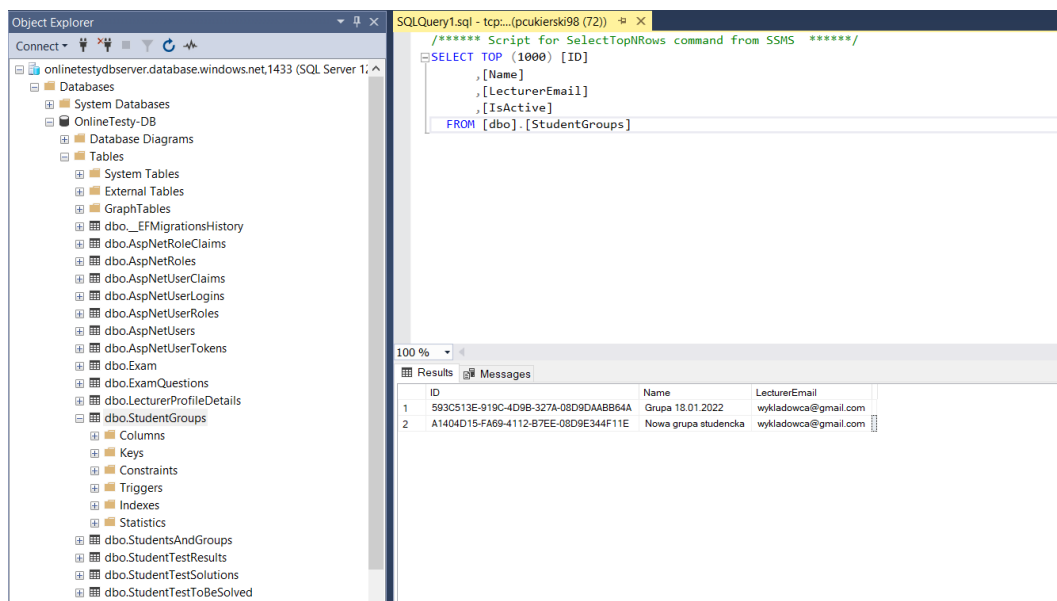
Grupy studenckie są dodawane przez użytkowników o roli wykładowcy. Każdy wykładowca może utworzyć wiele grup studenckich, do których następnie mogą przypisywać się studenci. W celu utworzenia grupy studenckiej wykładowca powinien wybrać zakładkę "Dodaj grupę studencką", która znajduje się w menu bocznym aplikacji.

Po wybraniu zakładki użytkownikowi ukazuje się formularz, w którym należy uzupełnić nazwę grupy studenckiej, a następnie zatwierdzić operację za pomocą przycisku "Dodaj grupę". Po poprawnym dodaniu grupy studenckiej powinien zostać wyświetlony komunikat jak na poniższej ilustracji. Następnie użytkownik zostaje przeniesiony do widoku, w którym widoczne są wszystkie dodane przez zalogowanego użytkownika grupy studenckie.

The screenshot shows the 'Dodaj grupę studencką' (Add student group) form. On the left is a dark sidebar menu with the following items: 'Strona główna' (Home), 'Przypisane egzaminy' (Assigned exams), 'Do sprawdzenia' (To be checked), 'Oceniłone egzaminy' (Graded exams), 'Utwórz nowy egzamin' (Create new exam), 'Dodaj grupę studencką' (Add student group - highlighted), 'Grupy studenckie' (Student groups), and 'Ustawienia konta' (Account settings). The main content area has a title bar with the text 'Dodaj grupę studencką'. Below the title bar is a notification box with the text: 'Komunikat ze strony onlinetesty-app.azurewebsites.net Grupa studencka została dodana pomyślnie' (Message from the onlinetesty-app.azurewebsites.net site: Student group was added successfully) and an 'OK' button. Below the notification box is a form field labeled 'Nazwa grupy' (Group name) with the placeholder text 'Nowa grupa studencka' (New student group). Below the form field is a blue button labeled 'Dodaj grupę' (Add group). In the top right corner, there is a user status bar showing 'Zalogowany użytkownik: wykładowca@gmail.com!' (Logged in user: wykładowca@gmail.com!) and a 'Wyloguj' (Logout) link.

Rys. 5.1. Dodanie grupy studenckiej - formularz

Po wykonaniu odpowiedniego zapytania SQL w bazie danych widać, że operacja dodania grupy studenckiej przebiegła poprawnie - odpowiedni rekord znalazł się w tabeli StudentGroups.



Rys. 5.2. Weryfikacja poprawności dodania grupy studenckiej w bazie danych

5.2. Utworzenie formularza testowego wraz z pytaniami zamkniętymi

Kolejnym krokiem procesu związanego z funkcjonalnością opisywanej aplikacji jest utworzenie nowego egzaminu przez wykładowcę, który zostanie przypisany do studentów należących do grupy, jaka zostanie wybrana na początku jego tworzenia. Aby rozpocząć tworzenie nowego testu, użytkownik o roli wykładowcy powinien wybrać zakładkę "Dodaj nowy egzamin". Po wybraniu tej zakładki zostanie wyświetlony formularz, w którym należy uzupełnić tytuł oraz opis egzaminu, jak również wybrać grupę studencką, jaka ma rozwiązać dany egzamin. Aby przejść do następnego kroku, jakim jest dodanie pytań do testu, należy wybrać przycisk "Dodaj egzamin".

Po poprawnej walidacji danych w formularzu dodawania nowego egzaminu następuje przejście do widoku umożliwiającego dodawanie pytań zamkniętych do testu. Każde pytanie musi zawierać treść, dodatkowy opis oraz cztery warianty odpowiedzi. Na końcu tego procesu wykładowca musi również wybrać poprawną odpowiedź do danego pytania. Po dodaniu pytania do testu wyświetlany jest odpowiedni komunikat oraz czyszczone są wszystkie pola w formularzu.

Na poniższym listingu przedstawiono komponent Blazor przedstawiający widok umożliwiający dodanie nowego egzaminu. Jak widać we fragmencie zamieszczonego kodu taki komponent dzieli się na dwie części. Pierwsza z nich odpowiada za wygląd danego widoku i jest utworzona z użyciem języka HTML. Druga część natomiast zawarta jest wewnątrz bloku @code i zawiera kod napisany w języku C#, odpowiedzialny za operacje po stronie back-endu.

Gdy wykładowca doda ostatnie pytanie do swojego egzaminu może przejść do zakładki "Przypisane egzaminy", aby zobaczyć utworzony przez siebie egzamin.

```

1  @page "/add-exam"
2  @using OnlineTesty_Library.Models
3  @inject IJSRuntime JsRuntime
4  @inject OnlineTesty_Library.Repositories.IExamRepositories ExamService
5  @inject OnlineTesty_Library.Repositories.IStudentGroupsRepositories StudentGroupsService
6  @inject NavigationManager NavigationManager
7  @inject Blazored.LocalStorage.ILocalStorageService sessionStorage
8
9  @attribute [Authorize(Roles = "Lecturer")]
10
11 <h3>Dodaj egzamin</h3>
12
13 <EditForm Model="@exam" OnValidSubmit="SubmitExam">
14     <DataAnnotationsValidator />
15     <div class="form-group">
16         <label for="name">Tytu egzaminu</label>
17         <InputText id="Name" @bind-Value="exam.Name" class="form-control" />
18         <ValidationMessage For="() => exam.Name" />
19     </div>
20     <div class="form-group">
21         <label for="name">Opis egzaminu</label>
22         <InputText id="Description" @bind-Value="exam.Description" class="form-control" />
23         <ValidationMessage For="() => exam.Description" />
24     </div>
25     <div class="form-group">
26         <label for="name">Grupa studencka</label><br />
27         <InputSelect id="student-group" @bind-Value="exam.StudentGroupName">
28             <option value=""></option>
29             @foreach (var item in StudentGroupsService.FindAllForLecturer())
30             {
31                 <option value="@item.Name">@item.Name</option>
32             }
33         </InputSelect>
34     </div>
35
36     <button type="submit" class="btn btn-primary">Dodaj egzamin</button>
37 </EditForm>
38
39 @code {
40     private Exam exam = new Exam();
41
42     protected void SubmitExam()
43     {
44         if(exam.StudentGroupName == null)
45         {
46             return;
47         }
48
49         bool nameIsEmpty = string.IsNullOrEmpty(exam.Name);
50         bool descriptionIsEmpty = string.IsNullOrEmpty(exam.Description);
51
52         var result = ExamService.Create(exam);
53
54         sessionStorage.SetItemAsync("idExam", exam.ID);
55         NavigationManager.NavigateTo("add-question");
56     }
57 }

```

Listing 5.1. Komponent Blazor - formularz dodawania nowego egzaminu

5.3. Rozwiązanie utworzonego testu przez użytkownika o roli studenta

W poprzednim podrozdziale został utworzony nowy egzamin, który automatycznie staje się dostępny dla wszystkich użytkowników, którzy mają przypisaną rolę studenta oraz są członkami odpowiedniej grupy studenckiej. W celu prezentacji funkcjonalności rozwiązywania testu został utworzony nowy użytkownik spełniający podane wyżej wymagania.

Po zalogowaniu się na konto tego użytkownika wybrana została zakładka "Rozwiąż egzamin". Wybór tej zakładki powoduje ukazanie się widoku, który umożliwia wyszukanie egzaminu przeznaczonego do rozwiązywania po skorzystaniu z przynajmniej jednego z dostępnych filtrów.

Rys. 5.3. Widok umożliwiający wyszukanie egzaminu przeznaczonego do rozwiązywania

Dzięki zastosowaniu trójwarstwowej architektury aplikacji, budowa komponentów Blazor jest prosta i lekka - można zobrazować to na przykładzie omawianego właśnie widoku filtra egzaminów do rozwiązywania. Cała logika odpowiedzialna za wyszukanie odpowiednich egzaminów w bazie danych jest ukryta w odpowiednich metodach znajdujących się w serwisie, do którego odwołuje się dany komponent.

```

1 @page "/exams-to-write"
2 @using OnlineTesty_Library.Models
3
4 @inject OnlineTesty_Library.Repositories.IExamRepositories ExamsService
5 @inject Blazored.LocalStorage.ILocalStorageService sessionStorage
6 @inject NavigationManager NavigationManager
7
8 @attribute [Authorize(Roles = "Student")]
9
10 <h3>Egzaminy do rozwiązania</h3>
11 <h5>Aby wyświetlić egzaminy przeznaczone do rozwiązania,
12 uzupełnij przynajmniej jeden z filtrów</h5>
13 <br />
14
15 <style>
16 (...)
17 </style>
18
19 <EditForm Model="@filter">

```

```

20     <div class="form-group">
21         <label for="Question">Nazwa egzaminu</label>
22         <InputText id="NameFilter" @bind-Value="filter.NameFilter" class="form-control" />
23     </div>
24     <div class="form-group">
25         <label for="Question">Nazwisko wykładowcy</label>
26         <InputText id="GroupFilter" @bind-Value="filter.LecturerFilter"
27             class="form-control" />
28     </div>
29 </EditForm>
30
31 <input class="btnCreateExamEnding_style" type="button" name="btnCreateExamEnding"
32 value="Filtruj" @onclick="eventArgs => { CreateFilter(); }">
33 <br />
34 <br />
35
36 <EditForm Model="@exam">
37
38     <ul class="list">
39         @foreach (var exam in ExamsService.FindAssignedExamsForStudent
40             (nameFilter, lecturerFilter))
41         {
42             <li class="group-record"><a href="/student-exam-to-be-solved/@exam.ID">
43                 @exam.Name</a></li>
44         }
45     </ul>
46 </EditForm>
47
48 @code {
49     private Exam exam = new Exam();
50     private ExamFilter filter = new ExamFilter();
51     private string nameFilter;
52     private string lecturerFilter;
53
54     protected void CreateFilter()
55     {
56         nameFilter = filter.NameFilter;
57         lecturerFilter = filter.LecturerFilter;
58     }
59 }

```

Listing 5.2. Komponent Blazor - wyszukiwanie egzaminów gotowych do rozwiązania

Rys. 5.4. Formularz umożliwiający rozwiązanie egzaminu

Po wyszukaniu odpowiedniego egzaminu możemy przejść do jego rozwiązania, klikając w jego nazwę. Zostanie otwarty formularz zawierający pytania należące do danego egzaminu.

Po zaznaczeniu przez studenta poprawnych według niego odpowiedzi może on przesłać swoje rozwiązanie do wykładowcy za pomocą przycisku "Prześlij rozwiązanie". Odpowiedzi studenta zostaną przesłane, a aplikacja powróci do widoku filtrowania egzaminów gotowych do rozwiązania.

Implementacja funkcjonalności rozwiązywania egzaminów została przedstawiona na poniższym listingu. Utworzony został formularz, który korzystając z pętli `foreach()` generuje tyle elementów zawierających pytania egzaminacyjne, ile jest pytań przypisanych do egzaminu, odczytanych za pomocą metody `ReadAllQuestions` z serwisu `ExamQuestionService`.

Przy inicjalizacji widoku pobierane są pytania egzaminacyjne na podstawie parametru typu GUID, który jest unikalnym numerem danego egzaminu. Tworzony jest obiekt solucji (rozwiązania) egzaminu. Sprawdzane jest również to, czy dany egzamin został już wcześniej rozwiązany - jeżeli tak, następuje przejście do strony, która informuje o takim fakcie.

Przy zaznaczeniu odpowiedzi przy danym pytaniu wybór studenta jest zapisywany w słowniku zawierający pary klucz-wartość, gdzie kluczem jest numer zadania, a wartością wybrana przez użytkownika odpowiedź. W przypadku zmiany odpowiedzi, poprzednia wartość dla danego klucza jest usuwana i zastępowana jest nową.

Po zatwierdzeniu rozwiązania na podstawie wspomnianego słownika do zmiennej typu `string` przypisywane są kolejno: numer zadania oraz kod odpowiedzi. Taka zmienna jest zapisywana w bazie danych i na jej podstawie odczytywane są kody odpowiedzi w ramach innych widoków.

```

1
2 inject OnlineTesty_Library.Repositories.IStudentTestToBeSolvedRepositories
3 StudentTestToBeSolvedService
4 [inject NavigationManager NavigationManager
5
6 [Attribute (Authorize(Roles = "Student"))]
7
8 <h3>Egzamin: @exam.Name</h3>
9 <h5>Opis egzaminu: @exam.Description</h5>
10 <br />
11
12 <style>(...)</style>
13
14 <EditForm Model="@examQuestions" OnValidSubmit="SubmitStudentResult">
15     <div>
16
17         @foreach (var item in ExamQuestionService.ReadAllQuestionsByExam
18             (Guid.Parse(GuidParam)))
19         {
20             string answerAId = item.Order + "A";
21             string answerBId = item.Order + "B";
22             string answerCId = item.Order + "C";
23             string answerDId = item.Order + "D";
24
25             <ul class="list">
26                 <li class="exam-question-record"><b>@item.Question</b></li>
27                 <li class="exam-question-record">
28                     A: @item.AnswerA
29                     <input asp-for="AnswerAIsChecked" type="checkbox" name="answerA"
30                         @onchange="eventArgs => { CheckChanged(answerAId, item.AnswerA); }">
31                 </li>
32                 <li class="exam-question-record">
33                     B: @item.AnswerB
34                     <input asp-for="AnswerBIsChecked" type="checkbox" name="answerB"
35                         @onchange="eventArgs => { CheckChanged(answerBId, item.AnswerB); }">

```

```

36         </li>
37         <li class="exam-question-record">
38             C: @item.AnswerC
39             <input asp-for="AnswerCIsChecked" type="checkbox" name="answerC"
40                 @onchange="eventArgs => { CheckChanged(answerCId, item.AnswerC); }">
41         </li>
42         <li class="exam-question-record">
43             D: @item.AnswerD
44             <input asp-for="AnswerDIsChecked" type="checkbox" name="answerD"
45                 @onchange="eventArgs => { CheckChanged(answerDId, item.AnswerD); }">
46         </li>
47     </ul>
48     <br /><br />
49 }
50
51     <button type="submit" class="btn btn-primary">Przeslij rozwiazanie</button>
52 </div>
53 </EditForm>
54
55 @code {
56     private Exam exam = new Exam();
57     private StudentTestSolution testSolution = new StudentTestSolution();
58     private StudentTestToBeSolved studentTestToBeSolved = new StudentTestToBeSolved();
59     private IEnumerable<ExamQuestion> examQuestions;
60     private IEnumerable<StudentTestToBeSolved> studentTestsToBeSolved;
61     private Dictionary<string, string> studentAnswers = new Dictionary<string, string>();
62
63     public bool AnswerAIsChecked { get; set; }
64     public bool AnswerBIsChecked { get; set; }
65     public bool AnswerCIsChecked { get; set; }
66     public bool AnswerDIsChecked { get; set; }
67
68     [Parameter]
69     public string GuidParam { get; set; }
70
71     protected override void OnInitialized()
72     {
73         exam = ExamService.Read(Guid.Parse(GuidParam));
74         examQuestions = ExamQuestionService.ReadAllQuestionsByExam(Guid.Parse(GuidParam));
75
76         testSolution.ExamId = Guid.Parse(GuidParam.ToUpper());
77         testSolution.ExamTitle = exam.Name;
78         testSolution.LecturerEmail = exam.UserEmail;
79
80         studentTestsToBeSolved = StudentTestToBeSolvedService.FindSolutionsForStudent();
81
82         foreach (var item in studentTestsToBeSolved)
83         {
84             if (testSolution.ExamId == item.ExamId)
85             {
86                 NavigationManager.NavigateTo("this-exam-is-resolved");
87             }
88         }
89     }
90
91     private EventCallback CheckChanged(string answerId, string answerDetails)
92     {
93         foreach (KeyValuePair<string, string> entry in studentAnswers)
94         {
95             if (answerId.StartsWith(entry.Key.Substring(0, 1)))
96             {
97                 studentAnswers.Remove(entry.Key);
98             }
99         }
100
101         studentAnswers.Add(answerId, answerDetails);
102
103         return EventCallback.Empty;
104     }
105
106     protected void SubmitStudentResult()
107     {
108         StringBuilder studentAnswersCode = new StringBuilder();

```

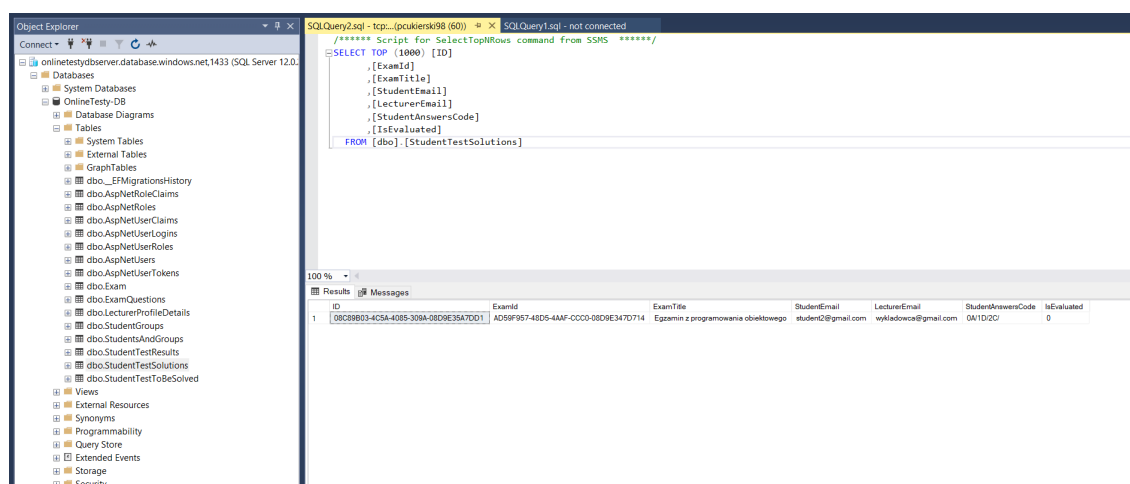
```

109
110     foreach (KeyValuePair<string, string> item in studentAnswers)
111     {
112         studentAnswersCode.Append(item.Key);
113         studentAnswersCode.Append("/");
114     }
115
116     studentTestToBeSolved.LecturerEmail = exam.UserEmail;
117     studentTestToBeSolved.ExamId = exam.ID;
118
119     testSolution.StudentAnswersCode = studentAnswersCode.ToString();
120     var result = StudentTestSolutionService.SaveExamSolution(testSolution);
121     var toBeSolved = StudentTestToBeSolvedService.
122     SaveInfoAboutSolution(studentTestToBeSolved);
123
124     GoToExamList();
125 }
126
127 protected void GoToExamList()
128 {
129     NavigationManager.NavigateTo("exams-to-write");
130 }
131 }

```

Listing 5.3. Komponent Blazor - wyszukiwanie egzaminów gotowych do rozwiązania

Rozwiązania egzaminów w bazie danych reprezentowane są poprzez rekordy w tabeli o nazwie StudentTestSolutions. Tabela ta zawiera: ID rozwiązywanego egzaminu oraz jego tytuł, adresy e-mail rozwiązującego oraz zlecającego egzamin (wykładowcy), a także kolumnę przechowującą kody odpowiedzi udzielonych przez studenta. W tej tabeli znajduje się także kolumna wskazująca, czy dane rozwiązanie zostało już ocenione przez wykładowcę.



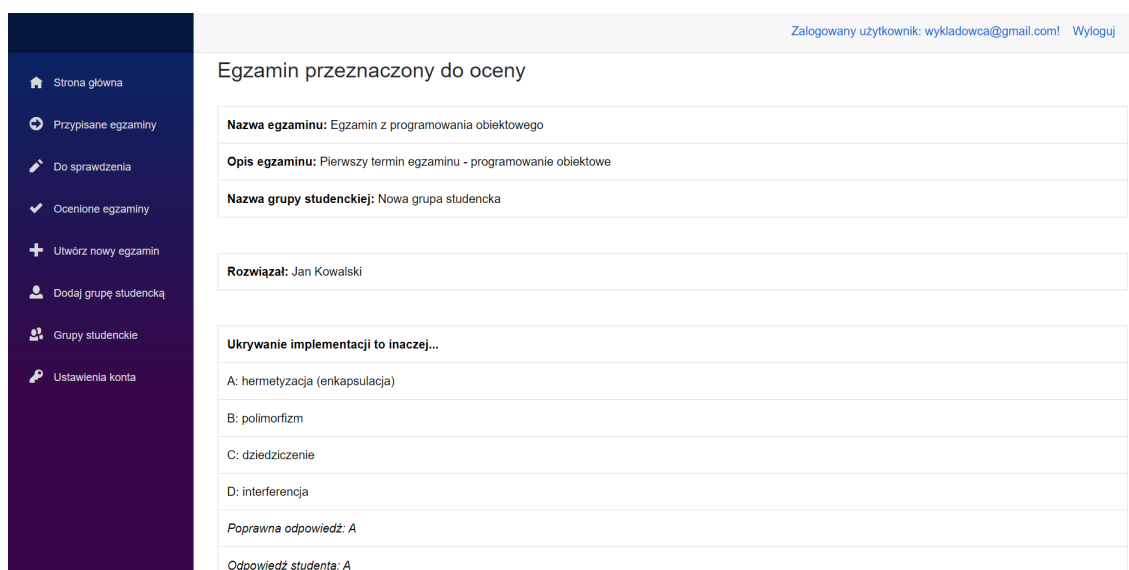
Rys. 5.5. Weryfikacja poprawności dodania rekordu odwzorowującego rozwiązanie egzaminu w bazie danych

5.4. Przeprowadzenie oceny rozwiązania testu przez użytkownika o roli wykładowcy

Przesłanie rozwiązania egzaminu przez studenta skutkuje możliwością jego oceny przez wykładowcę, który zlecał dany test odpowiedniej grupie studenckiej, a co za tym idzie - konkretnemu studentowi. Aby ocenić dane rozwiązanie, z poziomu konta o uprawnieniach wykładowcy należy w panelu bocznym wybrać opcję "Do sprawdzenia".

Następnie wykładowca ma możliwość filtrowania rozwiązań z uwzględnieniem nazwy egzaminu oraz nazwiska studenta, analogicznie jak wyglądało to w funkcjonalności wyszukiwania utworzonych egzaminów.

Po wybraniu odpowiedniego egzaminu użytkownikowi ukazuje się widok, w którym widoczny jest tytuł oraz opis egzaminu, nazwa grupy studenckiej, której przypisany był test oraz imię i nazwiska autora wyświetlonego rozwiązania. Poniżej sekcji zawierającej te informacje wyświetlane są pytania wraz z informacjami o poprawnej odpowiedzi oraz odpowiedzi udzielonej przez studenta.



Zalogowany użytkownik: wyklawowca@gmail.com! Wyloguj

Egzamin przeznaczony do oceny

| |
|--|
| Nazwa egzaminu: Egzamin z programowania obiektowego |
| Opis egzaminu: Pierwszy termin egzaminu - programowanie obiektowe |
| Nazwa grupy studenckiej: Nowa grupa studencka |

Rozwiązał: Jan Kowalski

| |
|--|
| Ukrywanie implementacji to inaczej... |
| A: hermetyzacja (enkapsulacja) |
| B: polimorfizm |
| C: dziedziczenie |
| D: interferencja |
| Poprawna odpowiedź: A |
| Odpowiedź studenta: A |

Rys. 5.6. Widok egzaminu przeznaczonego do oceny przez wykładowcę

Ilość punktów zdobytych przez studenta jest zliczana automatycznie. Wynik punktowy wyświetlany jest na końcu formularza. Pod wynikiem znajduje się lista rozwijana, z której wykładowca może wybrać ocenę, jaką wystawia studentowi za dany egzamin. Aby zatwierdzić wynik po wybraniu oceny należy wybrać przycisk "Zatwierdź wynik". Wynik zostanie zapisany w bazie danych, a użytkownik będzie przeniesiony do widoku wyszukiwania rozwiązanych egzaminów.

Zalogowany użytkownik: wyklawowca@gmail.com! Wyloguj

Poprawna odpowiedź: D

Odpowiedź studenta: D

Która właściwość konstruktora domyślnego w C# jest fałszywa?

A: może mieć nazwę inną niż nazwa klasy

B: inicjuje wartości pól klasy zerem, false lub null w zależności od typu pól

C: ma taką samą nazwę jak nazwa klasy

D: posiada modyfikator dostępu public

Poprawna odpowiedź: A

Odpowiedź studenta: C

Wynik: 2 / 3

Ocena: dobry plus

Zatwierdź wynik

Rys. 5.7. Widok egzaminu przeznaczony do oceny przez wykładowcę - zliczanie zdobytych punktów oraz wybranie oceny do wystawienia

Na poniższym listingu została przedstawiona implementacja funkcjonalności oceniania rozwiązanych egzaminów. Funkcjonalność ta jest dostępna tylko dla użytkownika o roli wykładowcy, co jest możliwe dzięki zastosowaniu atrybutu `Authorize`, który jako parametr przyjmuje rolę, dla której ma być dostępny dany widok.

Informacje dotyczące wyświetlonego egzaminu są pobierane z wykorzystaniem unikalnego ID egzaminu znajdującego się w adresie URL, natomiast imię i nazwisko rozwiązującego jest otrzymywane z wykorzystaniem serwisu pobierającego dane z tabeli `StudentAndGroups`.

Dane dotyczące pytań pobierane są z użyciem pętli `foreach` oraz odpowiedniej metody z klasy serwisu `ExamQuestionService`, pobierającej z bazy danych wszystkie pytania dotyczące obecnie wyświetlanego egzaminu. Następnie w kodzie aplikacji znajduje się instrukcja warunkowa, która inkrementuje zmienną zawierającą liczbę zdobytych punktów, jeżeli odpowiedź na dane pytanie zgadza się z poprawną odpowiedzią.

Na końcu wynik wyliczany jest z użyciem wspomnianej zmiennej oraz drugiej zmiennej, która zawiera ilość pytań znajdujących się w egzaminie. Utworzona została także lista rozwijana zawierająca możliwe do przyznania oceny.

W części, która zawiera kod napisany w języku C# znajduje się algorytm odpowiedzialny za wydobycie z rozwiązania odpowiedzi udzielonych przez studenta. Sprawdzane jest również to, czy dany egzamin nie został już oceniony - jeżeli tak, nie ma możliwości jego ponownej oceny.

W metodzie, która zostaje uruchomiona po wybraniu przycisku zatwierdzenia oceny wykonywana jest walidacja, która uruchamia odpowiedni komunikat, jeżeli liczba punktów jest mniejsza od zera lub mniejsza od ilości pytań zawartych w egzaminie.

Jeżeli walidacje nie wykryją wspomnianych błędów następuje zapis oceny w bazie danych, za pomocą metody `SaveExamResult()` z klasy `StudentTestResultService`.

```

1
2 @page "/exam-to-be-assessed/{GuidIdParam}"
3 @using OnlineTesty_Library.Models
4 @inject OnlineTesty_Library.Repositories.IExamRepositories ExamService
5 @inject OnlineTesty_Library.Repositories.IStudentAndGroupRepositories
6 StudentAndGroupService
7 @inject OnlineTesty_Library.Repositories.IExamQuestionsRepositories
8 ExamQuestionService
9 @inject OnlineTesty_Library.Repositories.IStudentTestSolutionRepositories
10 StudentTestSolutionService
11 @inject OnlineTesty_Library.Repositories.IStudentTestResultRepositories
12 StudentTestResultService
13 @inject IJSRuntime JsRuntime
14 @inject NavigationManager NavigationManager
15
16 @attribute [Authorize(Roles = "Lecturer")]
17
18 <h3>Egzamin przeznaczony do oceny</h3>
19 <br />
20
21 <style>
22 (...)
23 </style>
24
25 <EditForm Model="@exam">
26     <div>
27         <ul class="list">
28             <li class="exam-record"><b>Nazwa egzaminu:</b> @exam.Name</li>
29             <li class="exam-record"><b>Opis egzaminu:</b> @exam.Description</li>
30             <li class="exam-record"><b>Nazwa grupy studenckiej:</b>
31                 @exam.StudentGroupName</li>
32         </ul>
33         <br /><br />
34     </div>
35 </EditForm>
36
37 <EditForm Model="@studentDetails">
38     <div>
39         <ul class="list">
40             <li class="exam-record"><b>Rozwiazal: </b>
41                 @studentDetails.FirstName @studentDetails.LastName</li>
42         </ul>
43         <br /><br />
44     </div>
45 </EditForm>
46
47 <EditForm Model="@examQuestions">
48     <div>
49         @foreach (var item in ExamQuestionService.ReadAllQuestionsByExam(ExamId))
50         {
51             <ul class="list">
52                 <li class="exam-question-record"><b>@item.Question</b></li>
53                 <li class="exam-question-record">A: @item.AnswerA</li>
54                 <li class="exam-question-record">B: @item.AnswerB</li>
55                 <li class="exam-question-record">C: @item.AnswerC</li>
56                 <li class="exam-question-record">D: @item.AnswerD</li>
57                 <li class="exam-question-record"><i>Poprawna odpowiedź :
58                     @item.CorrectAnswerChar</i></li>
59                 <li class="exam-question-record"><i>Odpowiedz studenta:
60                     @studentAnswerCodes.ElementAt(item.Order)</i></li>
61             </ul>
62
63             @if (item.CorrectAnswerChar.Equals(studentAnswerCodes.ElementAt
64                 (item.Order).ToString()) && questionsAreChecked == false)
65             {
66                 result.StudentNumberOfPoints++;
67             }
68
69             <br /><br />
70         }
71     }
72     @if (true)
73     {

```

```

74         questionsAreChecked = true;
75     }
76 </div>
77 </EditForm>
78
79 <EditForm Model="@result" OnValidSubmit="SubmitResult">
80     <div>
81         <ul class="list">
82             <li class="exam-record"><b>Wynik:</b> @result.StudentNumberOfPoints
83             / @questionsCount </li>
84         </ul>
85         <li class="exam-record">
86             <b>Ocena: </b>
87             <InputSelect id="student-group" @bind-Value="@result.Evaluation">
88                 <option value=""></option>
89                 <option value="niedostateczny">niedostateczny</option>
90                 (...)
91                 <option value="bardzo dobry">bardzo dobry</option>
92             </InputSelect>
93         </li>
94         <button type="submit" class="btn btn-primary">Zatwierdz wynik</button>
95         <br /><br />
96     </div>
97 </EditForm>
98
99 @code {
100     private Exam exam = new Exam();
101     private IEnumerable<ExamQuestion> examQuestions = Enumerable.Empty<ExamQuestion>();
102     private StudentAndGroup studentDetails = new StudentAndGroup();
103     private StudentTestSolution solution = new StudentTestSolution();
104     private StudentTestResult result = new StudentTestResult();
105     private List<char> studentAnswerCodes = new List<char>();
106
107     private string[] answersCodesPom = { };
108     private char separator = '/';
109     private int pointsSum = 0;
110     private int questionsCount = 0;
111     private bool questionsAreChecked = false;
112     private bool solutionExist;
113
114     [Parameter]
115     public string GuidParam { get; set; }
116     public Guid ExamId { get; set; }
117
118     protected override void OnInitialized()
119     {
120         solution = StudentTestSolutionService.GetSolution(Guid.Parse(GuidParam));
121         exam = ExamService.Read(solution.ExamId);
122         studentDetails = StudentAndGroupService.Read(solution.StudentEmail);
123         examQuestions = ExamQuestionService.ReadAllQuestionsByExam(solution.ExamId);
124
125         ExamId = solution.ExamId;
126         solutionExist = StudentTestResultService.IsThereResult(ExamId, solution.StudentEmail);
127
128         answersCodesPom = solution.StudentAnswersCode.Split(separator);
129
130         // wydobyć kod w odpowiedzi studenta
131         for (int i = 0; i < answersCodesPom.Length - 1; i++)
132         {
133             char answerChar = answersCodesPom[i].Last();
134             studentAnswerCodes.Add(answerChar);
135
136             questionsCount++;
137         }
138
139         if (solutionExist)
140         {
141             NavigationManager.NavigateTo("this-exam-is-evaluated");
142         }
143     }
144
145     async Task SubmitResult()
146     {

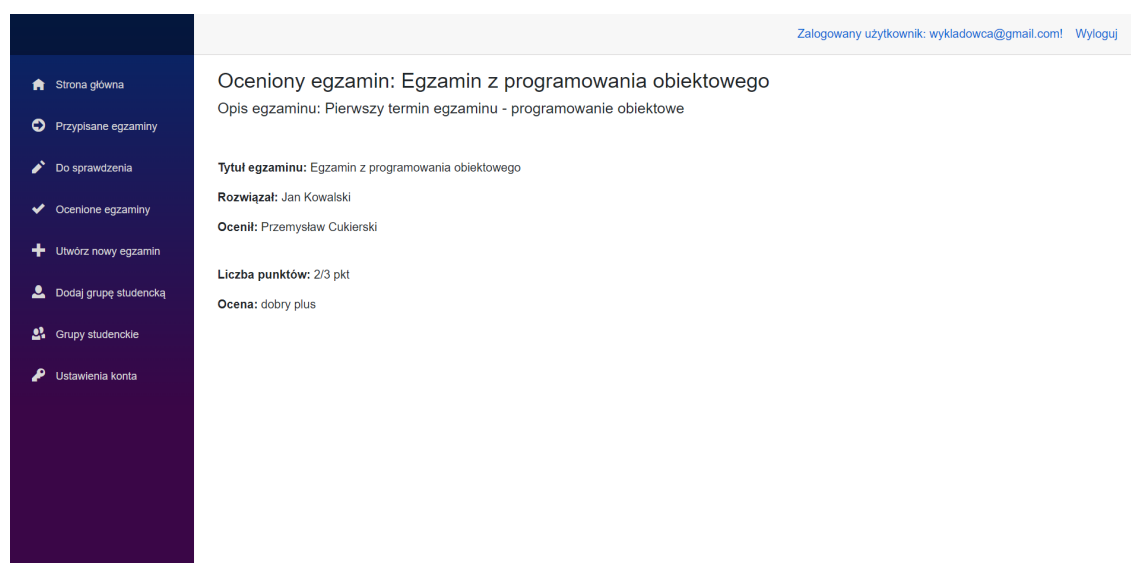
```

```
147         result.ExamNumberOfPoints = questionsCount;
148         result.ExamId = solution.ExamId;
149         result.StudentEmail = solution.StudentEmail;
150         result.LecturerEmail = solution.LecturerEmail;
151         result.ExamTitle = solution.ExamTitle;
152
153         solution.IsEvaluated = true;
154
155         if (result.StudentNumberOfPoints > result.ExamNumberOfPoints)
156         {
157             await JsRuntime.InvokeVoidAsync
158                 ("alert", "Liczba zdobytych punktów nie może być większa niż
159                 maksymalna ilość punktów do zdobycia");
160         }
161         else if (result.StudentNumberOfPoints < 0)
162         {
163             await JsRuntime.InvokeVoidAsync
164                 ("alert", "Liczba zdobytych punktów musi być większa lub równa zero");
165         }
166         else
167         {
168             StudentTestResultService.SaveExamResult(result);
169             NavigationManager.NavigateTo("solved-exams");
170         }
171     }
172 }
```

Listing 5.4. Implementacja funkcjonalności oceniania rozwiązanych egzaminów

5.5. Wyświetlenie formularza z oceną przyznaną za rozwiązanie egzaminu

Wyświetlenie widoku umożliwiającego zapoznanie się z oceną z danego egzaminu jest możliwe zarówno z poziomu konta wykładowcy, jak i konta studenta. Należy po wejściu w zakładkę "Ocenione egzaminy" z użyciem filtrów wyszukać dany egzamin oraz kliknąć na jego nazwę. W tym momencie użytkownikowi powinien ukazać się następujący widok.



Rys. 5.8. Widok ocenionego egzaminu

Widok ten zawiera informacje na temat tytułu egzaminu oraz nazwisk wykładowcy i studenta, a także liczby zdobytych punktów wraz z wystawioną przez wykładowcę oceną. Implementacja tej funkcjonalności w kodzie aplikacji wygląda w następujący sposób. Widok jest identyczny zarówno dla wykładowcy jak i studenta, więc atrybut dotyczący autoryzacji nie określa konkretnej roli użytkownika, jaka jest wymagana, lecz jedynie mówi o tym, że musi być to użytkownik zalogowany.

Przy inicjalizacji widoku z odpowiednich serwisów na podstawie parametru znajdującego się w adresie URL oraz adresów e-mail studenta i wykładowcy pobierane są odpowiednie wartości, które następnie wyświetlane są użytkownikowi.

```

1
2 @page "/single-evaluated-exam/{GuidId}"
3 @using OnlineTesty_Library.Models
4 @inject OnlineTesty_Library.Repositories.ILecturerProfileDetailsRepositories
5 LecturerProfileDetailsService
6 @inject OnlineTesty_Library.Repositories.IStudentAndGroupRepositories
7 StudentAndGroupService
8 @inject OnlineTesty_Library.Repositories.IStudentTestResultRepositories
9 StudentTestResultService
10 @inject OnlineTesty_Library.Repositories.IStudentTestToBeSolvedRepositories
11 StudentTestToBeSolvedService
12 @inject OnlineTesty_Library.Repositories.IExamRepositories ExamService
13 @inject NavigationManager NavigationManager
14
15 @attribute [Authorize]
16
17 <h3>Oceniony egzamin: @exam.Name</h3>
18 <h5>Opis egzaminu: @exam.Description</h5>
19 <br />
20
21 <style>
22     (...)
23 </style>
24
25 <br />
26
27 <p><b>Tytuł egzaminu: </b> @result.ExamTitle </p>
28
29 <p><b>Rozwiazal: </b> @studentAndGroup.FirstName @studentAndGroup.LastName</p>
30 <p><b>Ocenil: </b> @lecturerProfileDetails.FirstName @lecturerProfileDetails.LastName</p>
31 <br />
32 <p><b>Liczba punktow: </b>@result.StudentNumberOfPoints/@result.ExamNumberOfPoints pkt</p>
33 <p><b>Ocena: </b>@result.Evaluation</p>
34
35 @code {
36     private StudentTestResult result = new StudentTestResult();
37     private StudentAndGroup studentAndGroup = new StudentAndGroup();
38     private LecturerProfileDetails lecturerProfileDetails = new LecturerProfileDetails();
39     private Exam exam = new Exam();
40
41     [Parameter]
42     public string GuidParam { get; set; }
43
44     protected override void OnInitialized()
45     {
46         result = StudentTestResultService.Read(Guid.Parse(GuidParam));
47         lecturerProfileDetails = LecturerProfileDetailsService.Read(result.LecturerEmail);
48         studentAndGroup = StudentAndGroupService.Read(result.StudentEmail);
49
50         exam = ExamService.Read(result.ExamId);
51     }
52 }
```

Listing 5.5. Implementacja wyświetlania ocenionego egzaminu

```

1
2     public Guid SaveExamSolution(StudentTestSolution model)
3     {
4         string userEmail = _authenticationStateProvider.GetAuthenticationStateAsync().
5             Result.User.Identity.Name;
6
7         model.StudentEmail = userEmail;
8         this.GetDbSet<StudentTestSolution>().Add(model);
9         this.UnitOfWork.SaveChanges();
10        return model.ID;
11    }
12
13    public IEnumerable<StudentTestSolution> FindResolvedExamsForLecturer(string titleFilter,
14        string groupFilter, string studentFilter)
15    {
16        IQueryable<StudentTestSolution> resolvedExams = Enumerable.
17            Empty<StudentTestSolution>().AsQueryable();
18        StudentAndGroup studentAndGroup = new StudentAndGroup();
19        string userEmail = _authenticationStateProvider.GetAuthenticationStateAsync().
20            Result.User.Identity.Name;
21        string studentEmail = string.Empty;
22        string lecturerEmail = _authenticationStateProvider.GetAuthenticationStateAsync().
23            Result.User.Identity.Name;
24
25        if (titleFilter == null) titleFilter = string.Empty;
26        if (studentFilter == null) studentFilter = string.Empty;
27        if (groupFilter == null) groupFilter = string.Empty;
28
29        studentAndGroup = _studentAndGroupRepositories.ReadByLastName(studentFilter);
30
31        if (studentAndGroup != null)
32        {
33            studentEmail = studentAndGroup.EmailAddress;
34        }
35
36        ValidationObject validation = ValidationMethods.ValidationForExams
37            (titleFilter, studentFilter, groupFilter);
38
39        if ((validation.TitleFilterIsFilled) && (validation.StudentFilterIsFilled))
40        {
41            resolvedExams = this.GetDbSet<StudentTestSolution>().Where
42                (e => e.LecturerEmail == lecturerEmail)
43                .Where(e => e.ExamTitle.Contains(titleFilter))
44                .Where(e => e.StudentEmail == studentEmail);
45        }
46        else if ((validation.TitleFilterIsFilled) && (validation.StudentFilterIsNullOrEmpty))
47        {
48            resolvedExams = this.GetDbSet<StudentTestSolution>().
49                Where(e => e.LecturerEmail == lecturerEmail)
50                .Where(e => e.ExamTitle.Contains(titleFilter));
51        }
52        else if ((validation.TitleFilterIsNullOrEmpty) && (validation.StudentFilterIsFilled))
53        {
54            resolvedExams = this.GetDbSet<StudentTestSolution>().
55                Where(e => e.LecturerEmail == lecturerEmail)
56                .Where(e => e.StudentEmail == studentEmail);
57        }
58
59        return resolvedExams;
60    }
61
62    public StudentTestSolution GetSolution(Guid? ID)
63    {
64        return this.GetDbSet<StudentTestSolution>().
65            Where(e => e.ID == ID).FirstOrDefault();
66    }
67 }

```

Listing 5.6. Metody serwisu odpowiadające za operacje na tabeli z ocenami

Rozdział 6

Dyskusja rezultatów i wnioski końcowe

W odniesieniu do celu pracy sformułowanego na początku pracy dyplomowej, jakim było opracowanie portalu WWW umożliwiającego zdalne przeprowadzanie testów z użyciem technologii .NET Core można sformułować wniosek, że dany cel udało się zrealizować. Zostało utworzone oprogramowanie, które spełnia podstawowe warunki, jakie muszą zostać spełnione, aby można było uznać je za funkcjonalne. System tego typu powinien pozwalać na przeprowadzenie następującego procesu:

- utworzenie kont użytkowników dla wykładowcy i studenta wraz z obsługą walidacji wprowadzanych danych, między innymi uniemożliwieniem zarejestrowania dwóch kont za pomocą tego samego adresu e-mail
- zalogowanie się na konto wykładowcy i przypisanie danych personalnych
- utworzenie nowego egzaminu oraz przypisanie uprawnień do jego rozwiązania tylko wybranym przez siebie studentom. W ramach mojego rozwiązania jest to przypisanie do konkretnej grupy studentów.
- dodanie pytań do egzaminu, w przypadku omawianej aplikacji, tylko pytań jednokrotnego wyboru. System, który można byłoby uznać za rozwiązanie mające zastosowanie komercyjnie powinien pozwalać dodatkowo na dodawanie innego rodzaju pytań.
- zalogowanie się na konto studenta i przypisanie danych personalnych oraz nadanie przypisania do grupy studenckiej
- rozwiązanie przypisanego egzaminu
- ocenienie rozwiązanego egzaminu przez wykładowcę
- możliwość wyświetlenia otrzymanej oceny przez studenta

Omawiana w ramach pracy dyplomowej aplikacja spełnia wszystkie powyższe kryteria, zatem w pełni realizuje podstawową funkcjonalność tego typu rozwiązań.

Zrealizowane rozwiązanie nie jest jednak doskonałe. Jak każdy system informatyczny, tak również ten posiada swoje niedoskonałości i wymagałby poświęcenia jeszcze większej ilości czasu, aby doprowadzić go do stanu, który umożliwiłby zastosowanie tej aplikacji w celach komercyjnych, np. wdrożenie go jako system przeprowadzania egzaminów na uczelni takiej jak Uniwersytet Zielonogórski.

Z badawczego punktu widzenia istotne jest aby nazwać i wskazać problemy, które napotkać można podczas użytkowania aplikacji, aby posiadać punkt odniesienia w przypadku dalszych prac nad utworzonym w ramach mojej pracy dyplomowej oprogramowaniem. W ocenie autora są to między innymi:

- brak możliwości dodawania pytań innego typu niż pytania jednokrotnego wyboru. Aplikacja, która miałaby być wdrożona komercyjnie musiałaby posiadać również przynajmniej możliwość dodawania pytań wielokrotnego wyboru oraz pytań otwartych.
- brak możliwości przypisania studenta do więcej niż jednej grupy studenckiej. To poważny błąd popełniony w założeniach, jeszcze przed rozpoczęciem implementacji. Jeżeli student może być przypisany tylko do jednej grupy studenckiej, to w praktyce może rozwiązywać testy utworzone tylko przez jednego wykładowcę. Aby system był w pełni funkcjonalny, system musiałby zostać w pewnym zakresie przeprojektowany i zaimplementowany od nowa.
- brak zaimplementowanego rozwiązania dotyczącego odzyskiwania hasła. W utworzonej przeze mnie aplikacji jedyną możliwością w przypadku utracenia hasła przez użytkownika jest dokonanie zmiany hasła przez administratora systemu. Jest to rozwiązanie niedoskonałe i nie stosowane w komercyjnych aplikacjach webowych.

System utworzony w ramach niniejszej pracy dyplomowej nie będzie przez autora dalej rozwijany, jednak wskazanie powyższych niedoskonałości umożliwia rozwój aplikacji przez osoby trzecie.

Bibliografia

- [1] *Programy do tworzenia quizów: lista 7 najlepszych narzędzi.*
<https://www.ispring.pl/blog/programy-do-tworzenia-quizow>, 7 2021.
- [2] *WebAssembly. Moc ukryta w przeglądarce (i nie tylko).*
<https://geek.justjoin.it/webassembly-moc-ukryta-przegladarce>, 1 2019.
- [3] *Blazor – frontend w C.*
<https://plawgo.pl/2019/03/19/blazor-frontend-w-c/>, 7 2021.
- [4] *Wprowadzenie do ASP.NET Core Blazor.* <https://docs.microsoft.com/pl-pl/aspnet/core/blazor/?view=aspnetcore-6.0>, 7 2021.
- [5] *Introduction to ASP.NET Core SignalR.*
https://docs.microsoft.com/en-us/aspnet/core/signalr/introduction?WT.mc_id=dotnet-35129-website&view=aspnetcore-6.0, 10 2021.
- [6] *Microsoft Azure, czyli tworzenie, wdrażanie i zarządzanie aplikacjami.* <https://nav24.pl/microsoft-azure-czyli-tworzenie-wdrazanie-i-zarzadzanie-aplikacjami/>, 3 2018.