Politechnika Łódzka

Wydział Elektrotechniki, Elektroniki, Informatyki i Automatyki

Instytut Informatyki Stosowanej

PRACA DYPLOMOWA INŻYNIERSKA

„Internetowy system do zarządzania procesem recenzowania prac dyplomowych”

„Online system for management of the reviewing process of diploma theses”

Dawid Sowała

Numer albumu: 202295

Opiekun pracy:

Dr Joanna Sekulska-Nalewajko

Łódź, 2019

Spis treści

Streszczenie

Celem niniejszej pracy dyplomowej jest zaprojektowanie oraz wdrożenie aplikacji internetowej do zarządzania procesem recenzowania prac dyplomowych oraz podyplomowych. Projekt składa się ze strony WWW zaprojektowanej zarówno dla studentów jak i pracowników oraz relacyjnej bazy danych przechowującej odpowiednie dane.

Strona internetowa zawiera panel zaprojektowany dla administratora, w którym może on dodawać konta użytkowników oraz modyfikować ich ustawienia np. resetować hasło. Sam proces recenzji zaczyna się od wysłania poprzez serwis WWW zgłoszenia, w którym użytkownik wpisać musi tytuł, streszczenie oraz swój mail w celach identyfikacyjnych, zawrzeć także musi mail swojego opiekuna i ew. recenzenta, jeśli takowy jest potrzebny dla typu pracy dyplomowej. Zgłoszenie zostaje odebrane przez recenzenta, a ten przechodzi do oceny samej pracy, by po jej ocenieniu udostępnić studentowi swoją opinię.

W celu stworzenia systemu użyty został popularny framework do aplikacji internetowych ASP.NET Core wykorzystujący platformę .NET Core w języku C#. Do obsługi CSS i HTML użyto frameworku Bootstrap oraz biblioteki jQuery. Do baz wykorzystano MSSQL a do obsługi jej z poziomu aplikacji - Entity Framework Core. Strona w większej części napisana została w programie Visual Studio 2017 z małą pomocą Notepad++.

Praca podzielona została na 5 rozdziałów, które poruszają następujące tematy: Rozdział pierwszy odpowiada za wstęp, w którym przedstawione są aplikacje internetowe oraz ogólny opis systemu, technologii i narzędzi. Rozdział 2 opisuje szczegółowo narzędzie i technologie. Rozdział 3 przedstawia opis aplikacji oraz oprowadza czytelnika po jej funkcjach. Rozdział 4 i ostatni mówi o rozwiązaniach jakie zostały użyte w aplikacji. Rozdział 5 podsumowuje całą aplikację i przedstawia koncepcje dalszego rozwoju.

Słowa kluczowe: aplikacja internetowa, baza danych.

Abstract

The purpose of this thesis was to design and implement web application for managing the process of reviewing thesis and post-graduate work. Project consists of a website, that was designed for both students and employees, and relational database to store relevant data.

The website has a panel designed for the administrator, in which he can add users and modify their setting e.g reset the password. The review process begins with sending an application via the website in which the user must enter the tittle, summary and his or her e-mail for identification purposes, and also include e-mail address of his/her mentor and/or reviewer if it is needed for the type of thesis. Submissions are received by reviewer and he proceeds to the evaluation of the work itself, to provide the student with their opinion after its completion.

In order to create the system, the popular ASP.Net Core web application framework using .NET Core platform in C# was used. The Bootstrap framework and jQuery library were used to support CSS and HTML. MSSQL was used for databases and Entity Framework Core for operating it from the application level. The site was mostly written in Visual Studio 2017 with the small help from Notepad++.

The thesis was divided into 5 chapters, in which the following topics were covered:

Chapter 1 is responsible for the introduction, which present internet applications as well as a brief description of the system, technology and tools. Chapter 2 describes the tool and technologies in detail. Chapter 3 presents the application description and guides the reader through its functions. Chapter 4 - talks about the implementation that were used in application. Chapter 5 summarizes the whole application and presents ideas for further development.

Keywords: web application, database.

1. Wstęp

1.1 Cel

Prace dyplomowe to ważna część życia nie tylko akademickiego, ale także zawodowego. Uczniowie, studenci czy doktoranci stoją przed zadaniem samodzielnego napisania pracy kwalifikacyjnej. Każda praca przed przedstawieniem jej komisji musi zostać oceniona. Ukończeniu pracy towarzyszy jej recenzja, napisana zazwyczaj przez dwie osoby: opiekuna/promotora oraz recenzenta. Zabieg ten ma na celu pomóc nie tylko jednostce naukowej, by ta nie oceniała prac nie spełniających wymagań, ale także autorowi samej pracy, by nie podchodził do obrony z brakami. Proces recenzji nie jest obecnie zdigitalizowany w zbyt dużym stopniu. Po odpowiedniej analizie, można dojść do paru wniosków. Wykonując swoją pracę w czasach, gdzie część rzeczy jest na papierze a część skomputeryzowana, recenzenci zaczynając mieć problem nie tylko ze ocenianiem obecnych prac, ale i z przeszukiwaniem archiwalnych recenzji. Ujednolicenie obecnego systemu poprzez zwiększenie nacisku na digitalizacje, znacząco usprawniło by ten proces. Kreowana była więc wizja użycia popularnych w Internecie stron WWW. Strony internetowe w dzisiejszych czasach cechują się wygodą oraz uniwersalnością. Tworzone są one w taki sposób, aby niezależnie od wybranego przez użytkownika środowiska, czy to w postaci systemu operacyjnego, czy smartfonu - możliwe było jej bezproblemowe używanie. Głównym zadaniem dla programisty jest stworzenie strony internetowej z własną logiką i danymi, zwaną też aplikacją internetową. Stworzenie takiej aplikacji nie tylko polepszyło by obecny ekosystem, ale umożliwiło by skorzystanie z dobrodziejstw Internetu. Na etapach koncepcyjnych kreowała się wizja portalu informującego mailowo o zakończeniu recenzji. Pozwalającego na stworzenie archiwum prac, w których po indeksach studentów można było by znaleźć recenzje nawet sprzed paru lat. W obliczu tych zalet, padła decyzja o stworzeniu takiego właśnie systemu. Za cel postawiono więc tytułowe stworzenie internetowe systemu do zarządzania procesem recenzowania prac.

1.2 Zarys ogólny aplikacji

Aplikacja internetowa która konceptualnie nosi nazwę „ThesisReview” ma proste założenie. Odwiedzający stronę studenci będą mogli wysłać formularz dotyczący recenzji ich pracy dyplomowej. Zarejestrowani przez administratora pracownicy, będą mogli dostawać takie zgłoszenia od studentów. Wysyłany formularz posiada informacje takie jak tytuł pracy, wstęp/streszczenie, plik z pracą w formacie pdf oraz podstawowe dane do identyfikacji studenta przez adres mail stworzony na uczelni, jego imię i nazwisko oraz jednostka naukowa w której studiuje. Oprócz tego zawrzeć trzeba również mail opiekuna i ewentualnego recenzenta, zależnie od typu pracy. Na podaną przez wysyłającego pocztę, zostanie przysłany link dający możliwość przejrzenia recenzji. Ten adres url będzie mu potrzebny do dostania się do widoku recenzji. Sam student nie ma możliwości zarejestrowania się, aby nie tworzyć niepotrzebnych kont tylko do jednego tymczasowego celu. Same konta, należeć będą wyłącznie do pracowników uczelni i ich podstawowym celem będzie możliwość oceny formularza poprzez napisanie odpowiedzi na pytania dotyczących pracy, zgodnych z jej typem. Recenzent po ocenieniu pracy uniemożliwia sobie jej dalszą edycję. Po zakończeniu edycji przez niego oraz ewentualnego drugiego recenzenta, student na podany mail dostaje powiadomienie o zakończeniu procesu recenzji wraz z oceną końcową. Korzystając z linku może on sprawdzić wystawioną przez opinię. To podsumowuje działanie aplikacji w relacji recenzent - student, spełniając tym samym podstawowe założenia.

1.3 Istniejące rozwiązania

Po odpowiedniej analizie poszczególnych uczelni w Polsce, można zauważyć dwa główne sposoby na recenzje prac dyplomowych. Pierwszym jest częściowy brak zdigitalizowania tego procesu. Student dostarcza swoją pracę poprzez udanie się do opiekuna zazwyczaj z pracą wgraną na nośnik w postaci przenośnego dysku lub płytki CD, można odnotować przypadki wydrukowania prac, ale jest to raczej marginalnie zjawisko. Student nie dostaje żadnej informacji zwrotnej odnośnie statusu recenzji i musi się on po nią udać do opiekuna. Drugim sposobem jest załatwianie tych spraw poprzez pocztę elektroniczną. Problem leży bardziej w kwestii archiwizowania recenzji. Te zostają zapisane na dyskach twardych opiekunów, co oczywiście naraża je na usunięcie przy awarii lub są drukowane po czym stawiane na półkach, co zapełnia miejsce, którego w środowisku akademickim zawsze brakuje. Najważniejszym jednak jest fakt, że brakuje

rozwiązania w postaci strony internetowej czy aplikacji mobilnej. To samo w sobie utrudniło pracę, gdyż brak dostępnych rozwiązań uniemożliwia zweryfikowanie przydatnych oraz nieprzydatnych dla takiego systemu funkcji. Brakuje opinii w chociażby w Internecie, które mogłyby pomóc zaprojektować odpowiednio taką aplikację.

1.4 Ogólny opis narzędzi

Proces tworzenia aplikacji to rzecz często bardzo indywidualna. Programista może przebierać nie tylko w ogromnej liczbie narzędzi do pisania, ale przede wszystkim w różnych językach i bibliotekach. Wybór odpowiednich technologii, więc jest bardzo ważny. Do stworzenia aplikacji użyto ASP.NET Core. Napisany jest on w języku C# na platformie .NET Core. System potrzebował też bazy danych, a odpowiednim kandydatem do zarządzania danymi był MSSQL znany też jako Microsoft SQL Server. Do obsługi HTML, CSS oraz JavaScript wykorzystano popularny zestaw narzędzi Bootstrap i towarzyszącej mu biblioteki jQuery. Jako zintegrowane środowisko programistyczne zwane też jako IDE użyto Visual Studio 2017 w wersji community. Służył on również jako edytor tekstowy, wraz z małą pomocą Notepad++. Tak przedstawia się lista głównych narzędzi. Opis bibliotek z których owe narzędzia korzystały, zostaną opisane szczegółowo w dalszej części.

1.5 Plan pracy dyplomowej

W celu łatwiejszej nawigacji po pracy, została ona odpowiednio podzielona. Finalnie praca składa się z 5 następujących rozdziałów:

Rozdział 1: Wstęp

Rozdział 2: Technologie i narzędzia

Rozdział 3: Aplikacja

Rozdział 4: Rozwiązania

Rozdział 5: Podsumowanie

W Rozdziale 1 skupiono się bardziej na ogółach, aby można było mieć bardziej powierzchowny obraz aplikacji i tego czego ona dotyczy.

Rozdział 2 to opis technologii oraz użytych narzędzi. Szczegółowe przedstawienie tego na czym wszystko zostało napisane, wraz z powodami ich użycia ponad innymi.

Rozdział 3 skupi się na bardziej szczegółowym opisie aplikacji, co dokładnie jest w niej możliwe. Część tą potraktować można jako dokumentacje użytkownika, w której skupiono się na stronie wizualnej i pokierowaniu po wszystkich funkcjach.

Rozdział 4 ma na celu ukazanie programistycznej strony aplikacji. Pokazane zostaną tu fragmenty kodu oraz ich wyczerpujące deskrypcje, która pomogą w zrozumieniu meandrów działania poszczególnych elementów.

Rozdział 5 podsumowuje pracę, przedstawiając możliwości dalszego rozwoju.

2. Technologie i narzędzia

Dobry wybór technologii oraz narzędzi ma ogromne znaczenie w procesie kreowania aplikacji. W projekcie wykorzystano dużo narzędzi od Microsoftu. Narzędzie tego dewelopera cechują się dużą kompatybilnością, dzięki czemu można było w większym stopniu skupić się na używaniu narzędzi a nie na ich implementacji.

2.1 Technologie

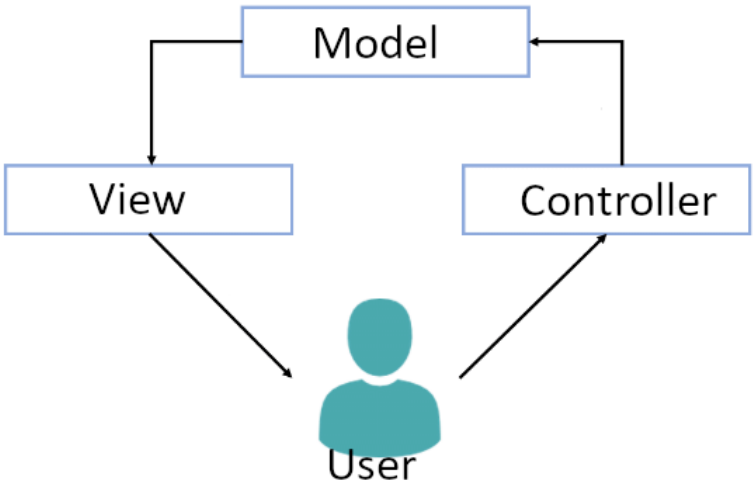
2.1.1 ASP.NET Core

Jednym z najważniejszych wyborów przy pisaniu aplikacji webowej jest dobór odpowiednich frameworków. Liczba mnoga nie jest tutaj użyta bez powodu, zazwyczaj dobiera się dwie technologie, jedna zwana też back-endem odpowiada za to, czego użytkownik nie zobaczy. Część funkcjonalna aplikacji zawiera w sobie obsługę logiki strony. Drugim typem frameworka jest tzw. frontend. Ten z kolei odpowiada za obsługę strony bardziej od strony użytkownika. Tutaj wysyłamy komunikaty do backendu, gdzie obsługiwane są już bardziej programowo.

To krótkie objaśnienie ma pokazać jak to zazwyczaj wygląda. ASP.NET Core odchodzi od tego schematu. Ma nieco inną strukturę, gdyż zawiera w sobie zarówno front jak i back end. Napisana jest ona przez inżynierów z Microsoft oraz społeczność. Jej pierwsza wersja wydana została 27 czerwca 2016 roku. W aplikacji użyta jest wersja 2.1, głównie ze względu na jej domyślnie wsparcie w używanym środowisku programistycznym Visual Studio 2017. Wspiera on pakiety Nuget, pozwalające mu na rozszerzenie swoich możliwości o dodatkowe funkcje, stworzone zarówno przez firmy jak i pojedynczych programistów. Jedną z bardziej używanych funkcji w projekcie jest wzorzec MVC (Model View Controller) oraz Razor Page, czyli oparty na stronach model programowania. ASP.NET Core posiada wbudowaną obsługę użytkowników i ich ról dzięki temu nie trzeba tego robić ręcznie.

2.1.2 ASP.NET Core MVC

MVC to popularny wzorzec projektowy dla aplikacji internetowych. Rozwijając skrót otrzymujemy: Model, czyli modele, view oznaczającego widoki oraz controller – kontroler (Rys. 1). Główną cechą jest komunikowanie się tych 3 składników. Kontroler odpowiada za odbieranie komunikatów od użytkownika i obsługiwanie ich, wykorzystując do tego model a następnie używając danych z modelu, pokazuje użytkownikowi odpowiedni widok.



Rys. 1 Model MVC

Tworzenie oddzielnych składników od obsłużenia różnych zadań lepiej organizuje strukturę aplikacji. Występuje tutaj ułatwione kodowanie, debugowanie oraz testowanie. ASP.NET Core MVC dodaje do tego wszystkie parę przydanych elementów, które szerzej zostaną opisane w segmencie dotyczącym użytych rozwiązań.

2.1.3 .NET Core

ASP.NET Core obsługuje kilka platform programistycznych, w tej pracy inżynierskiej zdecydowano się na wykorzystanie .NET Core w wersji 2.1 oraz język C szarp (C#).

2.1.4 MSSQL

Bazy danych są niezwykle istotne w aplikacja skupiających się na przechowywaniu informacji. Spośród rożnych rodzaj baz, w projekcie użyto bazy relacyjnej. Do języka zapytań docelowo miał być więc użyty SQL lub jego pochodna. No podstawie tych wymagań do obsługi baz danych idealnie pasowało więc rozwiązania Microsoftu. MSSQL zwany też Microsoft SQL Server. Jest to system zarządzania relacyjną bazą danych, który do zapytań wykorzystuje język Transact-SQL będącego swoistym rozszerzeniem do SQL. Jest on wspierany przez używany przeze mnie IDE Visual Studio 2017.

2.1.5 Entity Framework Core

Baza danych musi ona zostać uzupełniona o dane. Dodawanie ręczne rekordów do zbioru to pierwsze co przychodzi na myśl, lecz wstawianie ich manualnie nie tylko jest żmudne, ale i narażone na błędy ludzkie. Aplikacja powinna wstawiać rekordy do bazy automatycznie, a do ułatwienia tego zadania można wspomóc się frameworkiem Entity Framework Core. Odpowiada on za mapowanie obiektowo-relacyjne, ponieważ pomimo tego, że baza jest relacyjna, obiekty w kodzie aplikacji są obiektami, co wynika z obiektowości języka C#. Narzędzie to pomaga łączyć te dwie struktury.

2.1.6 Bootstrap

Jedną z reguł interfejsu jest, aby osoba odwiedzająca stronę powinna na podstawie samego designu mieć odpowiednie pojęcie co do czego służy. Aplikacja internetowa po stronie klienta powinna mieć więc odpowiedni wygląd, który w jasny i prosty sposób przekaże użytkownikowi cel poszczególnych elementów. W tym celu używa się HTML oraz CSS, te jednak znane są ze swoich problemów z użytkowaniem ich. Dlatego też do wyboru mamy rożne narzędzie wspomagające budowę strony, a jednym z popularniejszych jest Bootstrap. Framework ten wspomaga programistę w zaprojektowaniu odpowiedniego wzornictwa strony.

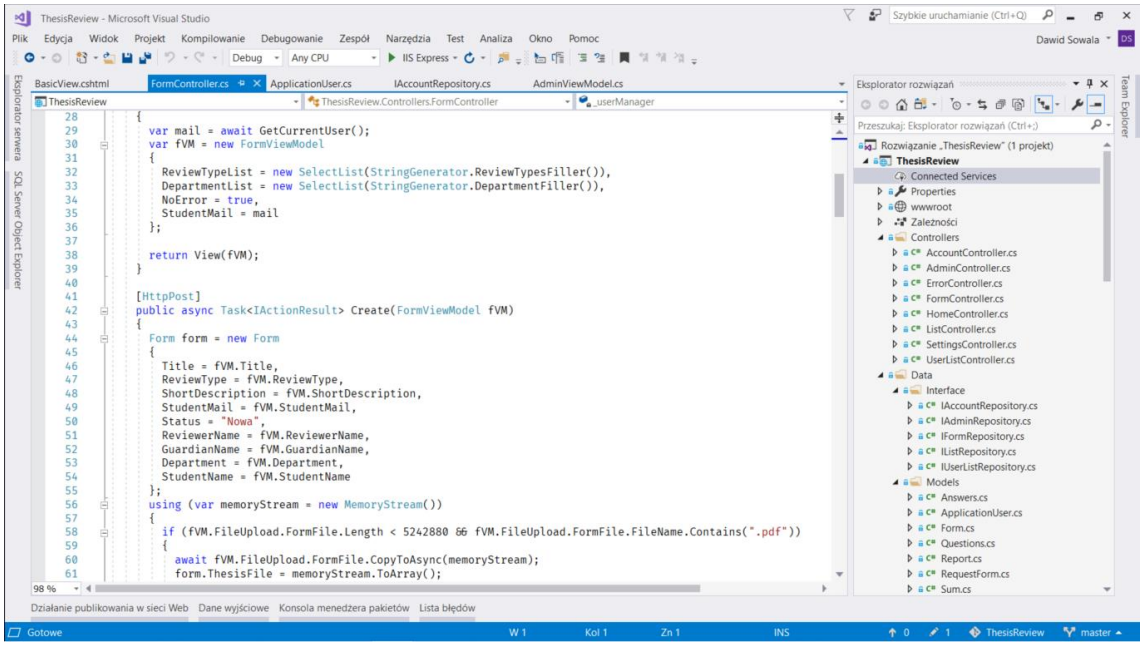
2.1.7 MailKit

W aplikacji internetowej, szczególnie takiej która obsługuje użytkowników, potrzebne jest czasem wysłanie maila. Aplikacja w tej pracy inżynierskiej identyfikuje pracowników poprzez ich służbowy, adres mailowy przez co obsługa maili jest niezwykle istotna. W tym celu użyto stosunkowo prostej biblioteki obsługującą pocztę elektroniczną zwaną MailKit. Pozwala wysyłać cyfrowe listy po wpisaniu danych do maila.

2.2 Narzędzia programistyczne

2.2.1 Visual Studio 2017

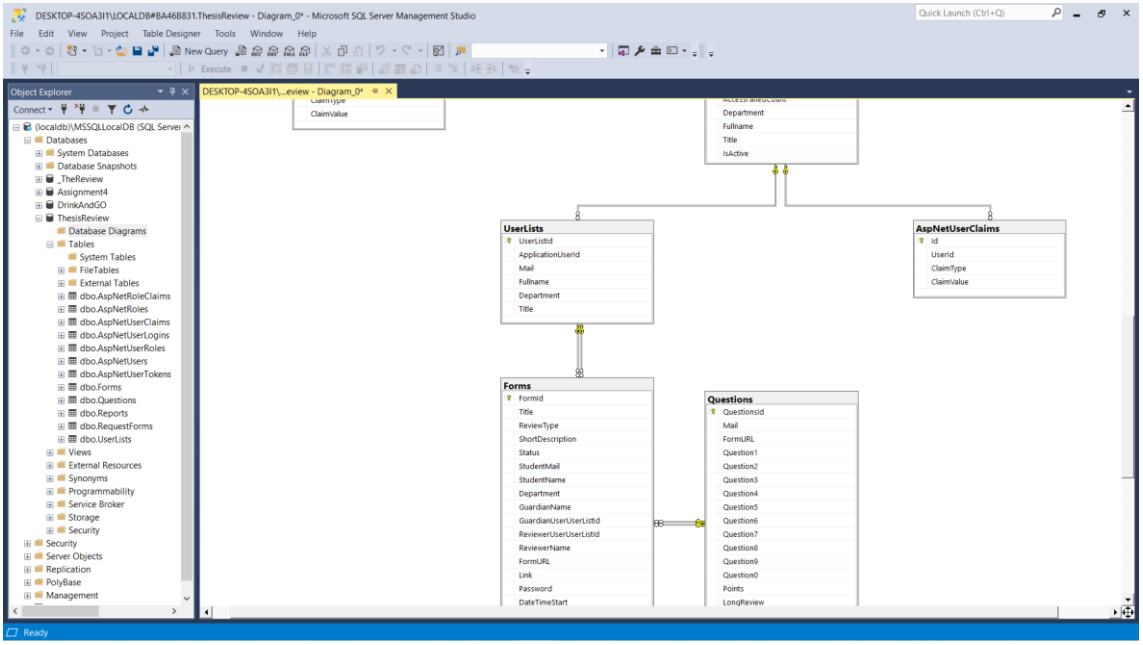
Kontynuując wcześniej wspominany trend używania narzędzi Microsoftu, do pisania kodu użyto popularne IDE - zintegrowane środowisko programistyczne Visual Studio 2017 w darmowej wersji Community (Rys. 2). Program ten to swoisty kombajn, w którym stworzyć można nie tylko aplikacje internetową, ale i oprogramowanie konsolowe czy takie z graficznym interfejsem. Visual Studio wspiera domyślnie wszystkie wyżej wymienione frameworki jak ASP.NET Core czy MSSQL z Entity Framework Core, dzięki czemu nie problemy z kompatybilnością pomiędzy tymi komponentami zmniejszone są do minimum. Oprogramowanie jest własnościowe w przeciwieństwie do użytych technologii, jednakże dostępna jest edycja Community za darmo dla celów niekomercyjnych z myślą o zastosowaniach akademickich.



Rys. 2 Visual Studio 2017 Community Edition

2.2.2 SQL Server Management Studio

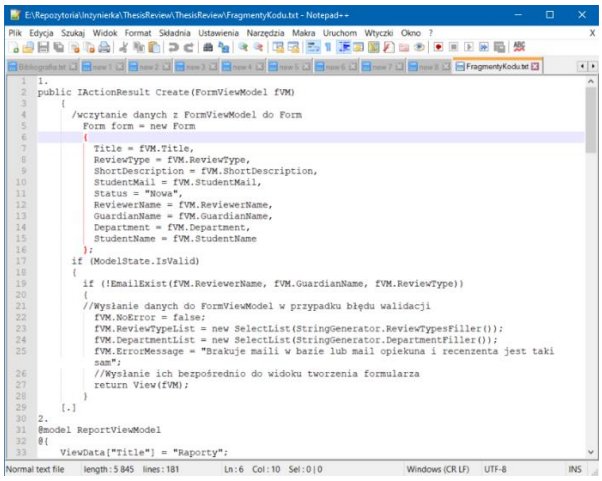
Do obsłużenia baz danych oprócz IDE, posłużono się również zintegrowanym środowiskiem do zarządzania MSSQL - SQL Server Managment Studio (Rys. 3). W programie tym można było przejrzeć tablice oraz rekordy bazy danych i co najważniejsze stworzyć jej schemat w oparciu o istniejące tablice. Oprogramowanie często służyło do testowania zmiennych, kiedy nie było jeszcze możliwości wpisania ich do bazy lub chwilowa zmiana wartości w celu przejrzenie jej wpływu na działanie aplikacji.



Rys. 3 Microsoft SQL Server Management Studio

2.2.3 Notepad++

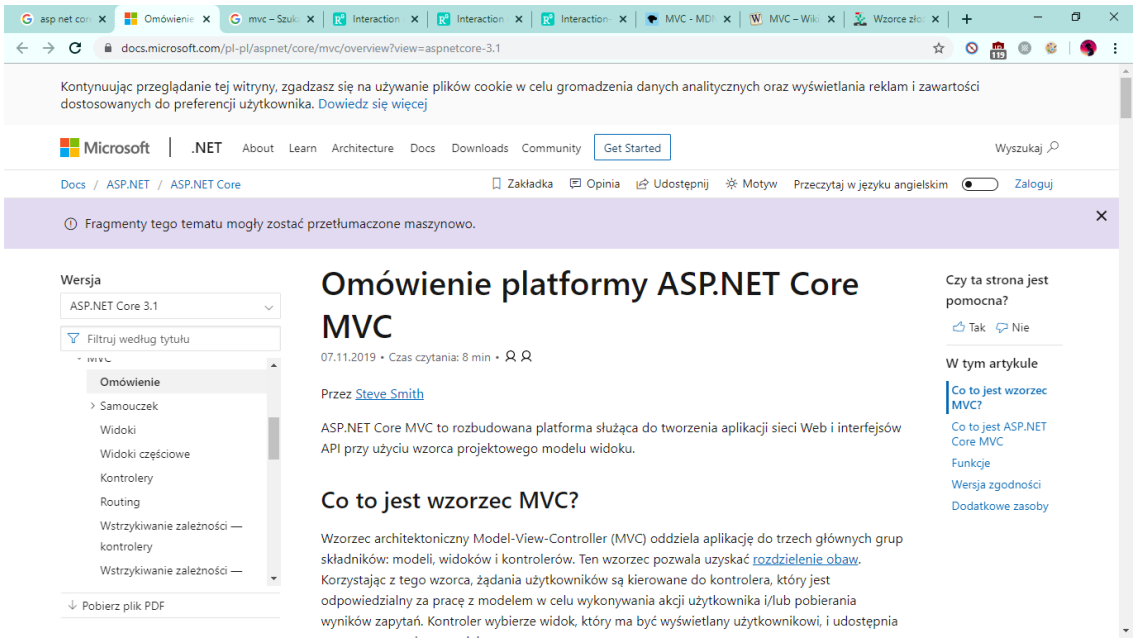
Otwarto-źródłowy notatnik (Rys. 4), który w pracy inżynierskiej miał za zadanie wspierać otwieranie mniejszych plików często poza projektowych. Pozwala on na wczytywanie formatów odnośnie wykorzystywanego języka programowania na podstawie rozszerzenia, co czyniło go idealnym dopełnieniem Visual Studio.



Rys. 4 Notepad++

2.2.4 Google Chrome

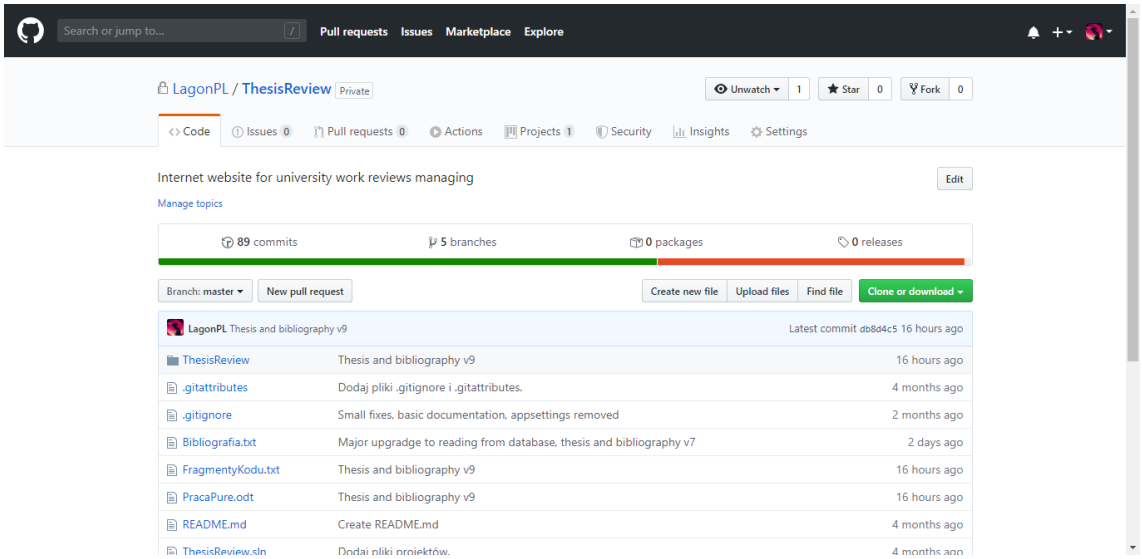
Strony internetowe wyświetlane są w przeglądarkach internetowych. Odpowiednie silniki renderujące, interpretują kod źródłowy strony, pokazując jego graficzną wersję. Przeglądarki wbudowane mają narzędzia deweloperskie dla programistów, dzięki którym mogą oni debugować swoją aplikacje. W projekcie do tego celu służył Google Chrome (Rys. 5).



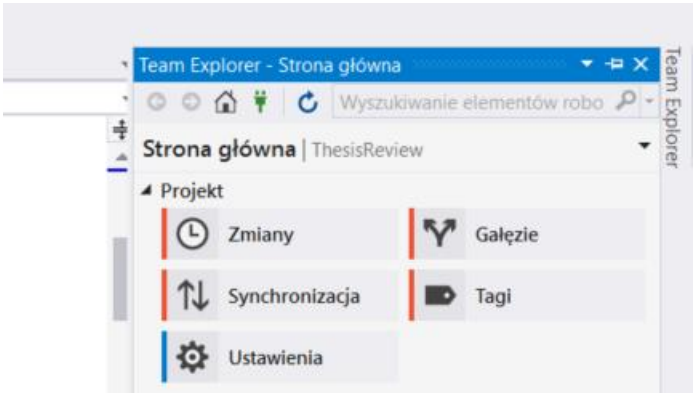
Rys. 5 Przeglądarka Google Chrome

2.2.5 Git oraz GitHub

„Git to nowoczesny, rozproszony system kontroli wersji”[1]. „Głównym zadaniem systemu kontroli wersji jest ułatwienie przeprowadzania operacji synchronizacji plików projektu …” [2]. Git w pracy inżynierskiej umożliwił tworzenie gałęzi, które wdrażały nowe funkcjonalności w poszczególnych etapach rozwoju projektu. Zapisywane były poszczególny zmiany zwane też commitami, dzięki czemu przy błędach lub dodaniu czegoś, co ostatecznie nie było potrzebne, można było posiłkować się kontrolą wersji. Projekt przechowywany jest w popularnym hostingowym serwisie GitHub (Rys. 6). Do obsługi gita służył zarówno zawarta w aplikację Git konsola jak i pomocne narzędzia w Visual Studio 2017 (Rys. 7), który wspiera kontrole wersji.



Rys. 6 Github



Rys. 7 Obsługa Git w Visual Studio

[1] Gajda W. (2013), Git. Rozproszony system kontroli wersji, Gliwice, Wydawnictwo Helion, ISBN 9788324673056 str. 13

[2] Gajda W. (2013), Git. Rozproszony system kontroli wersji, Gliwice, Wydawnictwo Helion, ISBN 9788324673056 str. 13

3. Aplikacja

Jednym z najważniejszych zadań dla programistów jest stworzenie zmyślnie połączonego systemu dla użytkownika. Interfejs i nawigacja po stronie powinny być intuicyjne i proste. Strona już samym wyglądem powinna zakomunikować swoje funkcje i możliwości. W tym punkcie skupiono się na poszczególnych podstronach aplikacji, opisując poszczególne pola.

W aplikacji występują 3 typy dostępu, pierwszym z nich jest niezalogowany użytkownik. W domyśle jest to widok zarezerwowany dla gości oraz studentów. W tym trybie liczba funkcji jest najbardziej ograniczona. Z racji tego, iż recenzować prace mogą tylko pracownicy uczelni, w domyśle bez odpowiednich uprawnień nie pozwala się gościom na wejście w główne opcje aplikacji. Drugim typem dostępu jest zwykły zalogowany użytkownik. Pracownik po dostaniu odpowiednich danych od administratora jest w stanie zalogować się i korzystać z większości funkcji serwisu. Jest on w stanie skorzystać z typowych ustawień dla stworzonego użytkownika jak zmiana hasła. Ostatnim typem dostępu jest bycie administratorem serwisu. Przy pierwszym kontrolowanym otwarciu strony, administrator zgodnie z zaleceniem programisty, dostaje dostęp do konta administratora. Z tego poziomu możemy tworzyć innych administratorów. Tutaj posiadamy pełnie praw i całkowity dostęp do strony. W tym rozdziale przybliżymy obsługę strony z poziomu gościa, użytkownika oraz administratora wraz z administratorem głównym.

3.1 Niezalogowany użytkownik

W tym segmencie skupiono się nie tylko na samych stronach, ale i na ich celach. Spora część wymienionych tu elementów, to nieodłączna część nowoczesnego wzornictwa, niezbędna do prawidłowego zaprojektowania strony internetowej. W przeciwieństwie do innych typów dostępów, ten konkretny jest dla każdego, więc odwiedzając aplikację, użytkownik będzie mógł używać każdego z komponentów w mniejszym lub większym stopniu. Niezależnie czy chce faktycznie skorzystać z głównej funkcji wysyłania zgłoszeń recenzji, czy tylko zobaczyć dostępne opcje.

3.1.1 Strona Główna

Po wejściu po raz pierwszy raz na stronę po wcześniejszym skonfigurowaniu jej przez admina, zobaczyć można stronę główną (Rys. 8). W języku angielskim nazywa się ją Home Page, co można przetłumaczyć jako strona domowa. Rzut oka na tę stronę, powinien dać użytkownikowi obraz tego, co będzie mógł tutaj zrobić. Odbiorca nie powinien być jednak przytłoczony, zwięzłe i treściwe przedstawienie jest bardziej istotne. Witryna główna to najczęściej odwiedzana strona na pojedynczego użytkownika, więc jej design dla twórców aplikacji internetowych jest niezwykle istotny.

W tym widoku można zauważyć parę elementów. Jednym z najbardziej rzucających się w oczy jest logo serwisu (1), przedstawiające herb Politechniki Łódzkiej wraz z polską nazwą strony. Kliknięcie na obrazek nie wywołuje żadnej akcji, głownie dlatego, że przenosił by w to samo miejsce. Pod nim znaleźć można komunikat z poleceniem wyboru czynności (2) oraz 2 przyciski. Te przyciski mogą zmieniać się w zależności od typu dostępu, ale dla gościa dostępna jest opcja logowania (3) oraz wysłania pracy (4). Niezależnie co klikniemy zostaniemy aplikacji przeniesie użytkownika na inną stroną, pierwszy do strony logowania, drugi do formularza zgłoszenia pracy.

Rys. 8 Strona Główna dla niezalogowanego użytkownika

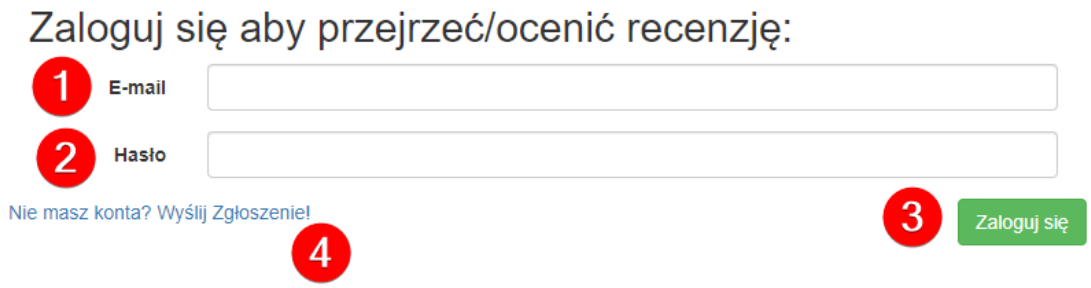
3.1.2 Pasek Nawigacyjny

Mówiąc o stronie, nie można nie wspomnieć o potencjalnie najważniejszym jej elemencie, jeśli chodzi o poruszanie się po niej. Pasek nawigacyjny (Rys. 9) jest nieodłączną częścią każdej prawidłowo zbudowanej strony internetowej. W teorii niepozorny element, ma jedną istotną cechę, jest widoczny na każdej podstronie witryny. Niezależnie od miejsca, na górze zawsze odnajdziemy pasek nawigacyjny. Jego głównym celem jest pomoc użytkownikowi w nawigacji. Za jego pomocą można przejść do prawie każdej strony, a jego zawartość zmienia się w zależności od typu dostępu. Nowe opcje jednak, nie powinny zmieniać dotychczasowego ułożenia linków. Domyślnie na pierwszej pozycji powinien być napis z nazwą aplikacji (1), który po kliknięciu przeniesie użytkownika do strony głównej. Kolejna opcja „Pracownicy" (2) przesyła do strony z zarejestrowanymi użytkownikami. Następne dwie opcje kopiują funkcjonalności ze strony głównej kolejno „Wyślij pracę" (3), która przesyła nas do formularza wysłania pracy do recenzji oraz ostatnia opcja dla gości - „Zaloguj się" (4), która przenosi do strony logowania. Jako że są to najważniejsze opcje dla odwiedzającego, powinien być do nich dostęp z każdej podstrony. Domyślnie logowanie nie jest możliwe bez danych logowania, które przydziela administrator.

Rys. 9 Pasek Nawigacyjny dla niezalogowanego użytkownika

3.1.3 Strona Logowania

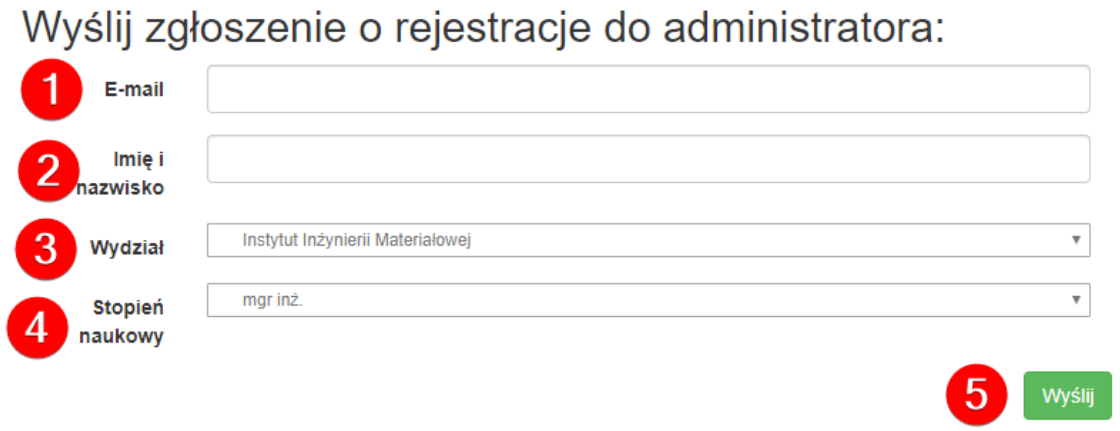
Obsługa modułu użytkownika jest niezwykle istotna dla aplikacji internetowych, które chcą w jakiś sposób ograniczać dostęp. Logowanie (Rys. 10) umożliwia rozpoznawania użytkowników, pozwalającą wyświetlić odpowiednie informacje z bazy danych. ASP.NET Core zapewnia obsługę identyfikacji użytkownika. W celu logowania wymagane są dane w postaci jakiegoś rodzaju identyfikatora, często w formie nazwy użytkownika lub maila, oraz hasło, które blokuje dostęp do prywatnych danych. Projekt wykorzystuje do identyfikacji adres mail (1) oraz hasło (2). Jeśli użytkownik posiada takie, może je tu wpisać po czym kliknąć „Zaloguj się" (3). Przy nieprawidłowych danych wyskoczy ogólny komunikat o błędzie. Aplikacja w zamyśle nie posiada opcji rejestracji, zamiast tego umożliwia ona na wysłanie zgłoszenia (4).



Rys. 10 Logowanie

3.1.4 Strona Zgłoszenia Użytkownika

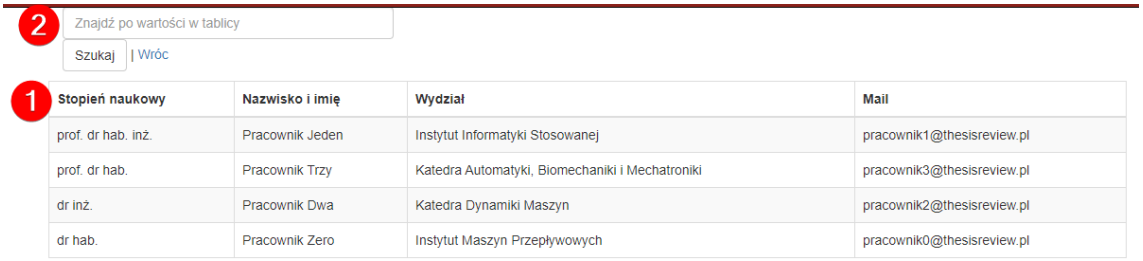
Zamysł projektu pozwala rejestrować użytkowników tylko administratorowi. Jeżeli jednak konta nie ma w systemie, pracownik uczelni może na powyższej stronie wysłać zgłoszenie (Rys. 11). Wpisując tu w odpowiednie pola swój mail (1), imię z nazwiskiem (2) oraz wydział (3) wraz ze stopniem naukowym (4) z listy. Kliknięcie w przycisk „Wyślij" (5), zweryfikuje dane, wyświetlając błąd przy błędzie walidacji. Prawidłowe zgłoszenie zostanie wysłane do administratorów serwisu, którzy to będą mogli zarejestrować brakującego użytkownika.



Rys. 11 Zgłaszanie braku użytkownika

3.1.5 Strona Pracownicy

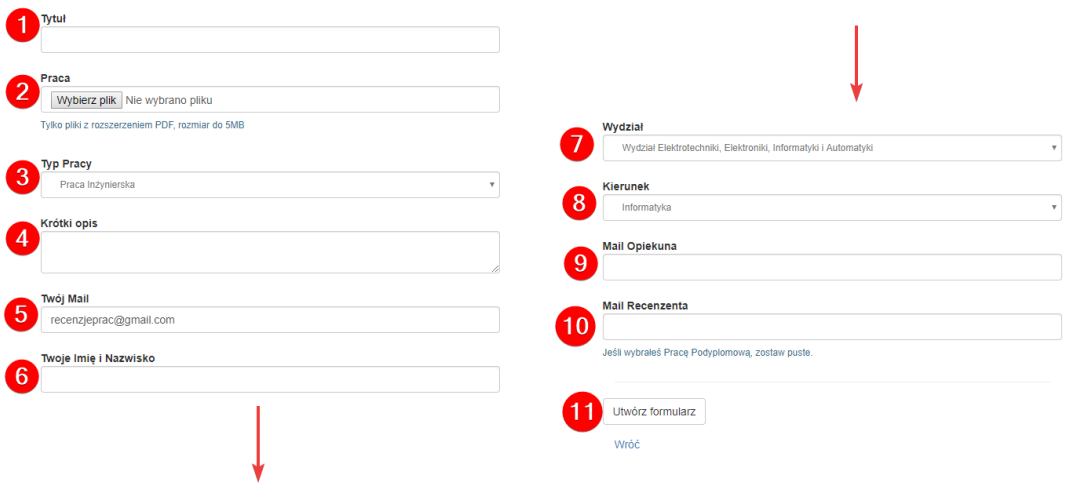
Na tej stronie (Rys. 12) odwiedzający może zobaczyć listę (1) obecnych w bazie pracowników. Każdy zarejestrowany jest automatycznie dodawany i wyświetlany na tej podstronie. Co ważne, jeśli danego pracownika nie można tutaj znaleźć, nie będzie możliwe wysłanie zgłoszenia na podany mail. Warto zawczasu więc sprawdzić, czy recenzent znajduje się już w bazie. Dostępna jest tutaj wyszukiwarka (2), która szuka danego tekstu w każdej kolumnie, ale można też użyć domyślnego szukania w przeglądarce skrótem control+F.



Rys. 12 Lista pracowników

3.1.6 Strona Formularza

Główna funkcja dostępna dla odwiedzającego, skierowana dla studentów uczelni. Na tej podstronie (Rys. 13) wysłać można pracę dyplomową i zgłoszenie odnośnie jej recenzji. W domyśle w zgłoszeniu umieszczony jest abstrakt oraz praca dyplomowa w celu identyfikacji, wysyłany jest poprzez kliknięcie na przycisk i wybranie pliku. Przyjmowane są tylko dokumenty w formacie pdf ważące nie więcej niż 5Mb. Na samej stronie znajduje się kilka pól do wypełnienia. Tytuł (1) oznacza tytuł pracy dyplomowej. Praca (2) to pole do wrzucenia pliku z dokumentem. Typ pracy (3) określa rodzaj pracy, gdzie znajdziemy opcje: inżynierska, licencjacka, magisterska oraz podyplomowa. Wybór ten określa dokładny wzór recenzji, gdyż na wielu uczelniach poszczególne prace mają inne formularze dla recenzentów. Pole z krótkim opisem (4) zarezerwowane jest na abstrakt, który powinienem znaleźć się w każdej pracy. „Twój mail” (5) pozwala odwiedzającemu wpisać swój adres mailowy. Na podana tutaj skrzynkę będą przychodzić wiadomości powiadamiające, od linku z widokiem recenzji pracy po zawiadomienie o zakończeniu oceniania przez recenzentów. W następnym polu (6) student podaje swoje imię i nazwisko, wydział w polu z wydziałem (7) oraz kierunek studiów (8), które mają pomóc zidentyfikować studenta i jego pracę. Ostatnie dwa pola to odpowiednio mail opiekuna (9) i recenzenta. (10) Podanym tutaj użytkownikom zostanie wysłane zgłoszenie.



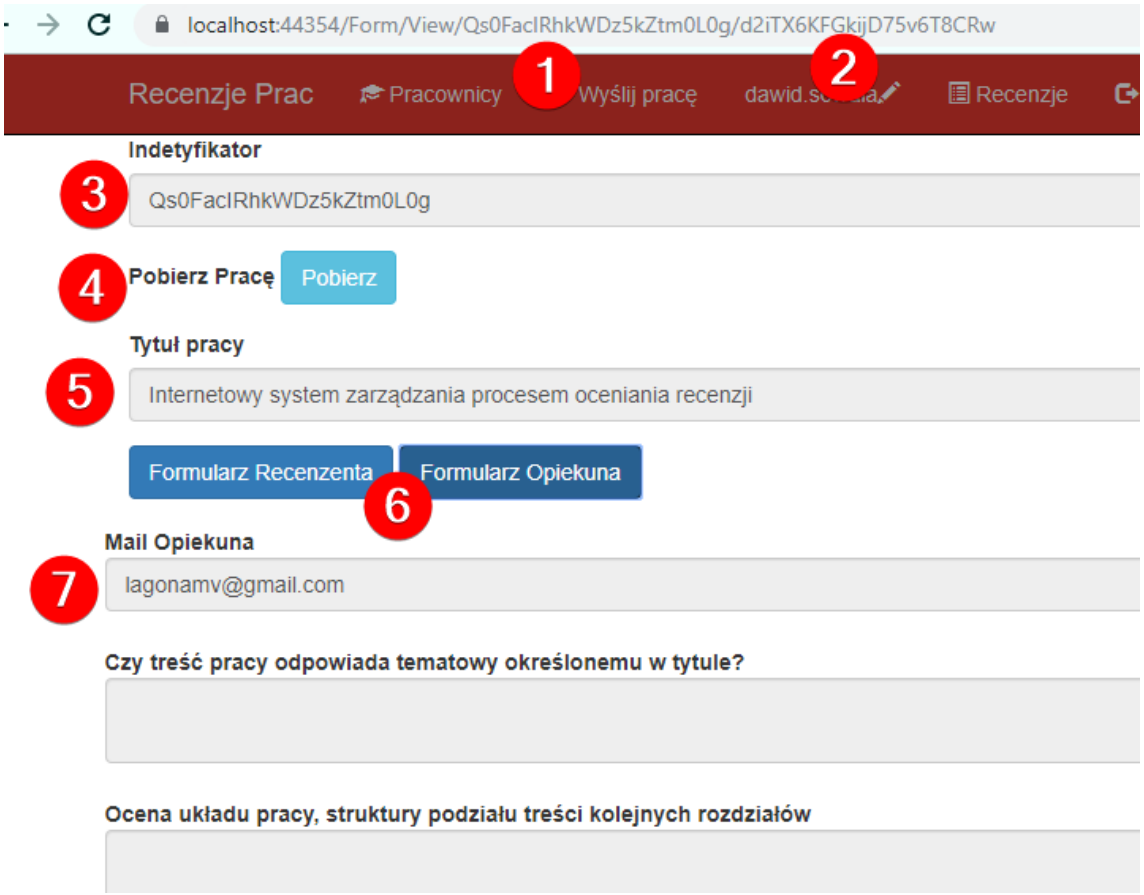
Rys. 13 Formularz wysłania pracy do recenzji

Nim formularz zostanie wysłany, aplikacja dokona walidacji danych następujących po kliknięciu w „Utwórz Formularz” (11). Istotnym jest, iż każde pole jest wymagane, wyjątkiem jest mail recenzenta, który przy wyborze pracy dyplomowej znika z widoku. Jeśli walidacja pól się nie uda, strona o tym poinformuje odpowiednim komunikatem.

W przypadku prawidłowego wpisania danych, zostaniemy przeniesieni na główną stronę, a na maila podanego w polu „Twój Mail” przyjdzie wiadomość z linkiem.

3.1.7 Strona Widoku Recenzji

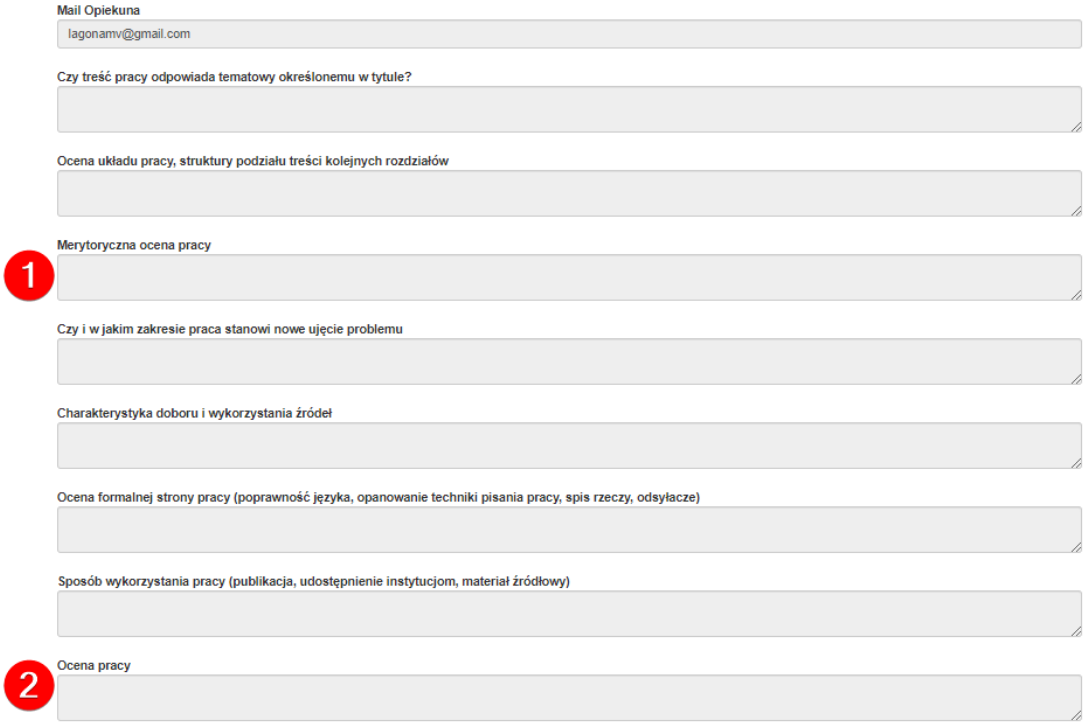
Ta podstrona (Rys. 14) różni się nieco od reszty, bo o ile jest ona dostępna dla niezalogowanego, to musi on posiadać specjalny link.



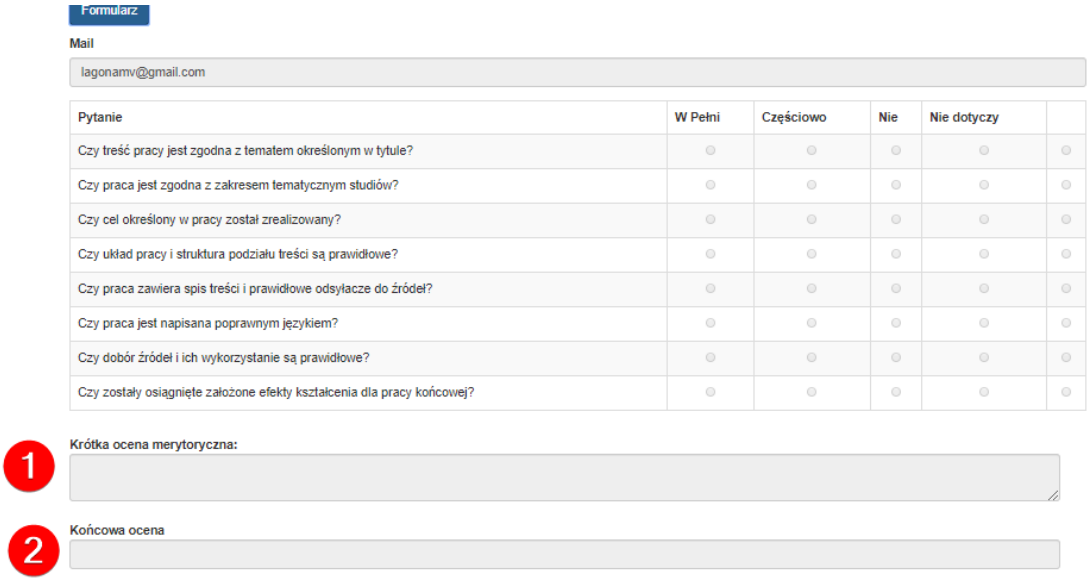
Rys. 14 Formularz studenta - pola główne

Adres ten zawiera w sobie dwie ważne części. 1 to specjalny identyfikator, którego można zobaczyć jako rezultat prawidłowego formularza. Istotną jest druga część (2) - jest to specjalne hasło, które to dostajemy w adresie URL zawartym w mailu powiadamiającym o stworzeniu formularza. Hasło to ma na celu uniemożliwienie przeglądania osobom postronnym w tym pracownikom. Recenzenci również posiadają pierwszą część linku i bez hasła, mogliby zobaczyć pracę opiekuna lub opiekun recenzenta. Poszczególne pola w widoku mogą się różnić, więc opisane zostaną te widoczne na każdym wzorze. Pierwsze pole zawiera identyfikator (3), przycisk (4) do pobrania wrzuconej wcześniej pracy, następnie i tytuł pracy dyplomowej (5). Następnie zauważyć można 1 lub 2 przyciski (6). Odsłaniają one formularz opiekuna lub recenzenta, jeśli praca posiada takowego, a ich ponownie kliknięcie chowa określony formularz.

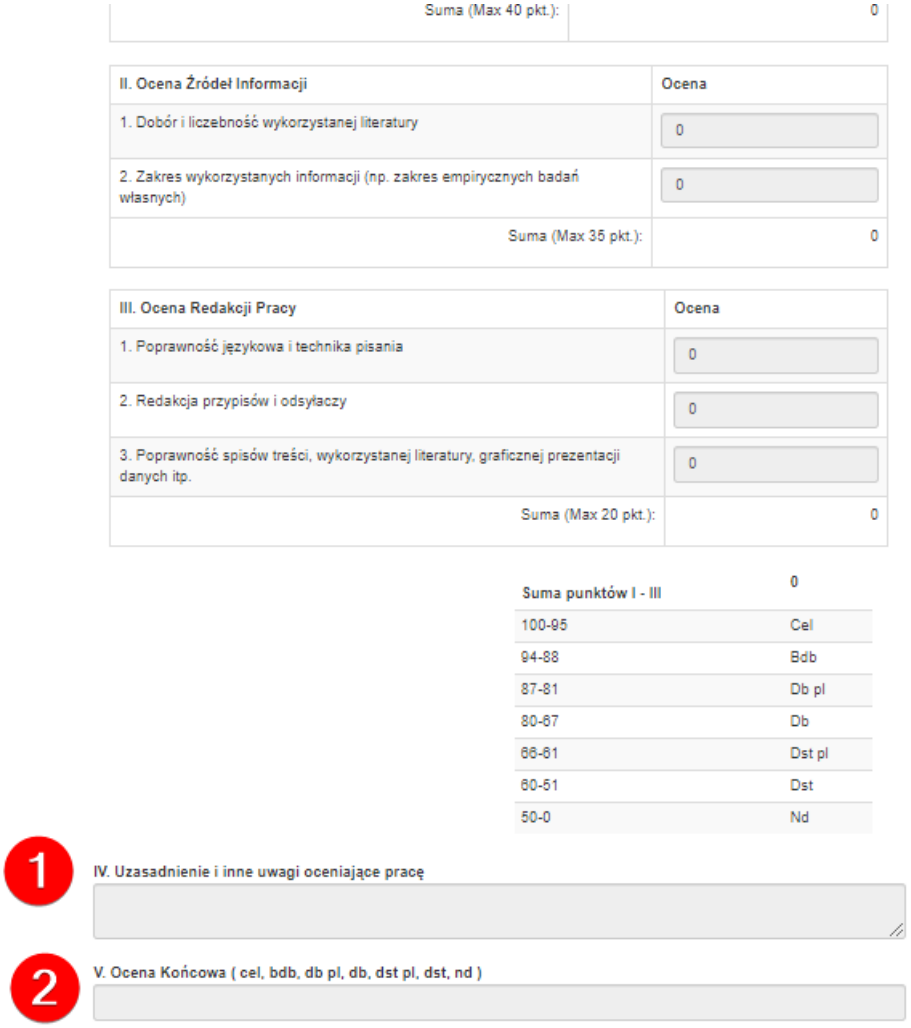
W odsłoniętym wzorze (Rys. 15) (Rys. 16) (Rys. 17) istotny jest mail recenzenta/opiekuna (7). Sam formularz będzie się różnił zależnie od rodzaju pracy dyplomowej, ale ich punktem wspólnym jest merytoryczna ocena pracy (1) oraz końcowa ocena pracy (2).



Rys. 15 Formularz dla pracy inżynierskiej i licencjackiej



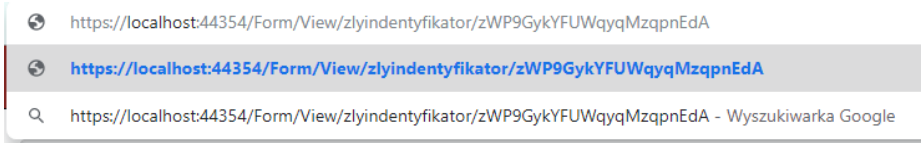
Rys. 16 Formularz dla pracy podyplomowej



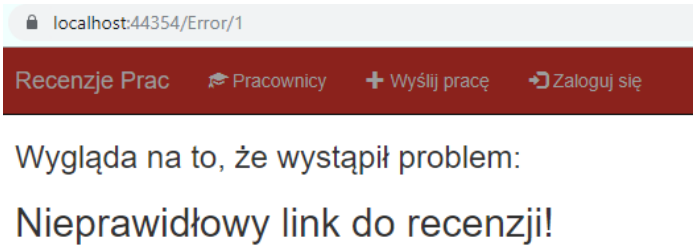
Rys. 17 Formularz dla pracy magisterskiej

3.1.8 Strona Błędu

Strona, która wyświetla się, gdy aplikacja napotka jakiś problem. Treść komunikatu zmienia się zależnie od napotkanego błędu. Poniżej strona (Rys. 19) wywołana nieprawidłowym linkiem (Rys. 18).



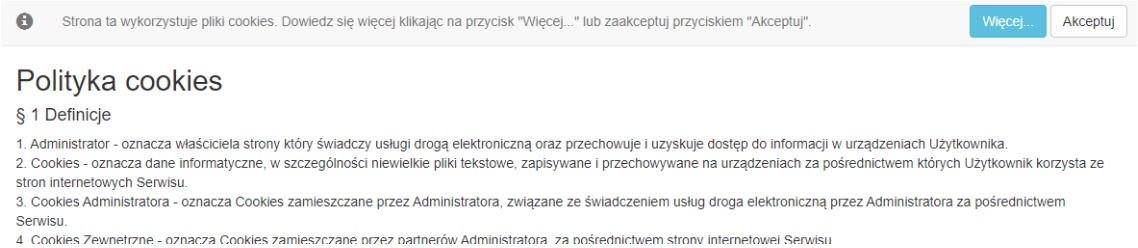
Rys. 18 Nieprawidłowy link do widoku recenzji



Rys. 19 Strona błędu

3.1.9 Strona Polityka Cookies

Ostatnią stroną w trybie gościa/studenta to to witryna zawierająca Politykę Cookies (Rys. 20), przedstawia one odpowiednie prawną dokumentację wykorzystywania plików w przeglądarce. Dzięki nim strona pamięta np. zalogowanym użytkowniku. Zawiadomienie to wyskoczy przy pierwszym uruchomieniu strony na konkretnej przeglądarce i po zatwierdzeniu, nie powinno się już wyświetlać.



Rys. 20 Strona z polityką cookies wraz z komunikatem

3.2 Użytkownik

Aby uzyskać ten tryb dostępu, wymagane jest zalogowanie się za pomocą odpowiednich danych. Można to zrobić używając linku „Zaloguj się” z paska nawigacji, na głównej stronie lub posiadając link do strony, która nie jest normalnie dostępna dla odwiedzającego - zamiast się do niej dostać, użytkownik przeniesiony zostanie do witryny logowania. Jest to tryb przeznaczony dla pracowników uczelni i aby zalogować się, musimy dostać dane logowania od administratora, lub poprosić o nie przez wcześniej wspominaną stronę zgłoszenia użytkownika. Wejście na poniższe strony nie będąc zalogowanym, ale posiadając odpowiedni, przenosi odwiedzającego do witryny z logowaniem, a zalogowanie się do wcześniej wpisanej podstrony. W tym segmencie skupiono się na delikatnych zmianach w istniejących już wcześniej stronach oraz na zupełnie nowych.

3.2.1 Zmiany

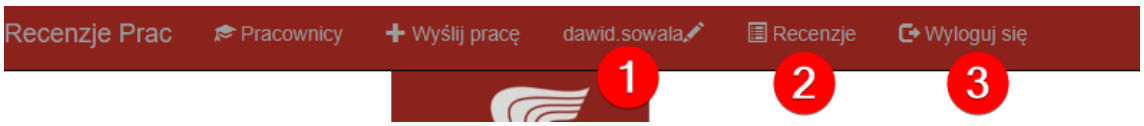
3.2.1.1 Strona Główna

Tutaj (Rys. 21) zauważyć można zmianę przycisku „Zaloguj się”, który obecnie ma wpisaną wartość „Recenzje” (1), przenoszącą pracownika do listy zgłoszeń.

Rys. 21 Strona główna po zalogowaniu

3.2.1.2 Pasek Nawigacyjny

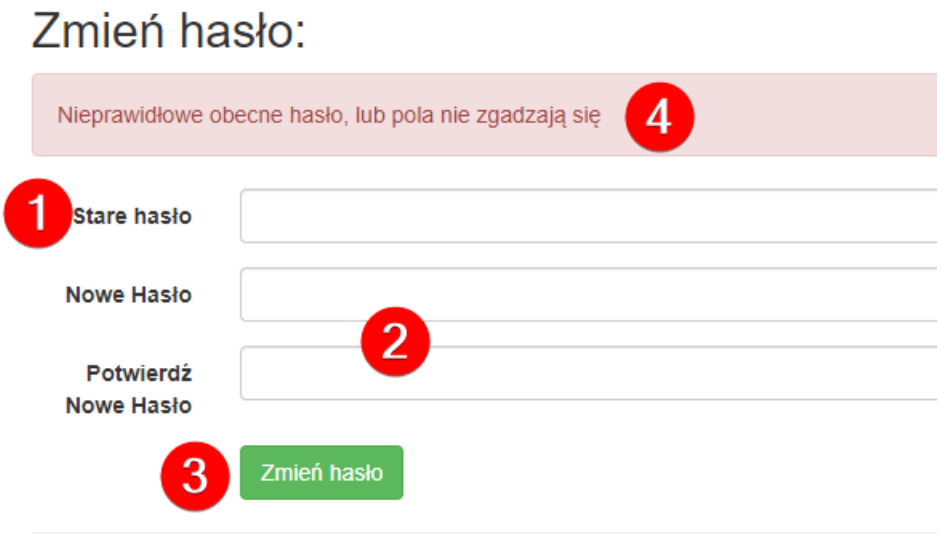
Na pasku (Rys. 22) po zalogowaniu, link do logowania zostaje zastąpiony i przeniesiony na sam koniec jako opcja „Wyloguj się”. Użytkownik dostaje trzy nowe opcje. Pierwszą jest jego unikalna nazwa użytkownika (1), która przenosi do zmiany hasła. Druga to „Recenzje” (2), przekierowująca pracownika do listy zgłoszeń. Trzecia i ostatnia umożliwia wylogowanie się (3).



Rys. 22 Pasek po zalogowaniu

3.2.2 Strona Ustawienia

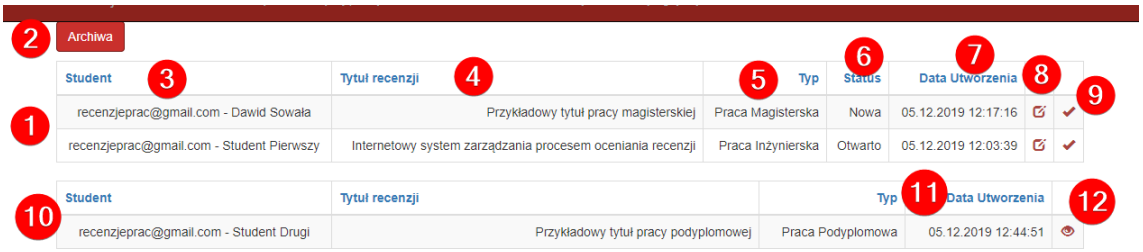
Na tej stronie (Rys. 23) użytkownik może zmienić hasło do swojego konta. W pola trzeba wpisać swoje dotychczasowe hasło (1), w dwóch następnych (2) nowe hasło oraz jego potwierdzenie. Przycisk „Zmień hasło” (3) wyśle odpowiednie zapytanie i przy sukcesie nastąpi przekierowanie do strony głównej, na której ujrzeć można komunikat sukcesu. Błędna walidacja nie przeniesie użytkownika i pokaże mu komunikat błędu (4).



Rys. 23 Ustawienia

3.2.3 Strona Recenzje

Centrum działań pracownika. Na tej stronie (Rys. 24) można znaleźć zgłoszenia studentów oraz archiwalne recenzje. Na pierwszy plan wychodzi lista ze zgłoszeniami (1) oraz przycisk Archiwa (2). Lista może być oczywiście pusta, natomiast jeśli znajdą się w niej jakieś dane - odpowiednie kolumny opiszą je. Pole „Student" (3) wyświetla jego dane w postaci imienia i nazwiska oraz adresu mailowego, który docelowo powinien zawierać również jego indeks. Tytuł recenzji (4) oraz Typ (5) zawierającą dane odnośnie samej pracy dyplomowej, które student wypełnił przy składaniu formularza. Status (6) daje pracownikowi informację czy recenzje jest już przejrzana czy też nie. Data utworzenia (7) zawiera datę wysłania formularza przez studenta. Dwie następne kolumny zawierają w sobie specjalne funkcje. Pierwsza z ikonką edycji (8), przenosi użytkownika do formularza edycji recenzji, druga zaś pozwala mu na zakończenie oceniania (9). Kliknięcie w nagłówki kolumn pozwala sortować listę.

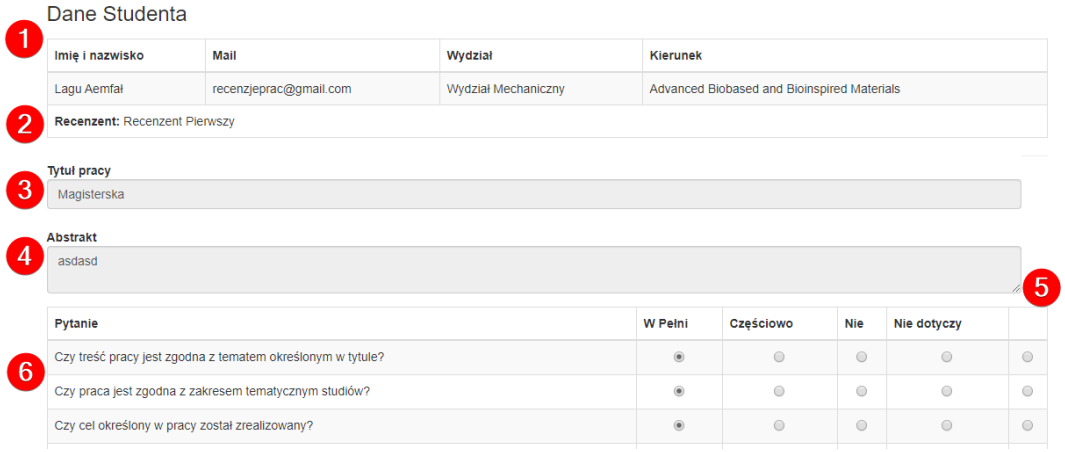


Rys. 24 Lista zgłoszeń recenzji

Przycisk Archiwa (2) odsłania drugą tablicę (10), która zawiera w sobie ocenione już wcześniej prace. Można tam dostrzec małe zmiany w stosunku do pierwszej tabelki. Data utworzenia (11) zmienia się na datę zakończenia oceniania recenzji. W miejscu dwóch ikonek jest teraz jedna, która pozwala na przejrzenie formularza (12).

Strona Edycji Formularza

Witryna, na której pracownik może podjąć się oceny pracy inżynierskiej przez jej recenzje (Rys. 25) (Rys. 26). Domyślnie dostępne są 3 wzory, nie oddają one faktycznych z Politechniki Łódzkiej, ale ich większa ilość ma pokazać do czego zdolna jest aplikacja. Na formularzu oprócz pól do ocen, znaleźć można pola, które znajdują się w każdym formularzu. Podobnie jak w widoku przejrzenia recenzji przez studenta, jest tutaj pole z tytułem pracy (3). Dodatkowe pola to natomiast tablica „Dane Studenta" (1) zawierające jego imię, nazwisko, mail z indeksem, wydział oraz kierunek, Następnie imię i nazwisko recenzenta. Występuje też tutaj wstęp, zatytułowany jako „Abstrakt" (4), który można rozciągnąć (5). Następnie zauważyć można sam formularz z pytaniami i odpowiedziami (6), na obrazku wzór dla pracy magisterskiej. Na samym dole przyciski „Pobierz" (7) – pozwalający pobrać pracę dyplomową, „Zapisz" (8) - zapisujące obecne zmiany i ustawiający status recenzji na „Otwarto"; „Cofnij" (9) wracający do listy zgłoszeń; „Drukuj" (10) otwierający okno drukowania przeglądarki; „Zakończ ocenienia" (11) otwierający okno alertu, którego potwierdzenie, zakończy ocenianie i uniemożliwi dalszą edycję ustawiając status „Oceniono".



Rys. 25 Strona edycji formularza 1/2

Rys. 26 Strona edycji formularza 2/2

Innym rezultatem sfinalizowania recenzji może być wrzucenie recenzji do archiwum. Jest to jednak zależne od tego czy druga osoba w postaci recenzenta/opiekuna również ukończyła wystawienie opinii. Jeśli tego nie zrobiła, recenzja trawi do archiwum, gdy już to zrobi. Istnieje możliwość wrzucenia recenzji do archiwum. Jeżeli już oceniony przez pracownika formularz ma więcej niż 60 dni, w widoku (Rys. 27) pojawi się przycisk umożliwiający przeniesienie recenzji do archiwum (11). Czynność ta jednak uczyni to dla dwóch recenzujących, zalecane jest więc skontaktowanie z drugim recenzentem

Rys. 27 Archiwizacja formularza

3.3 Administrator

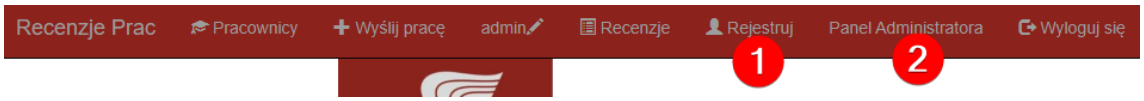
Ten dostęp dzieli się na 2 typy. Najważniejszy użytkownik to tzw. Administrator Główny. To najważniejsze konto posiadające wszystkie możliwości, konto to nie jest wpisane do listy pracowników co oznacza, że nie może ono przyjmować zgłoszeń.

Główny administrator to konto, które zostaje stworzone na przez programistę na dany przez klienta mail. Aktywuje się ono przy pierwszym uruchomieniu, wysyłając na pocztę elektroniczną hasło i login. Po tym administrator może rejestrować użytkowników. Domyślnie nie ma odwiedzający nie ma funkcji rejestracji i tylko administratorzy mogą takowe utworzyć, jednakże można wysłać zgłoszenie braku konta poprzez Stronę Zgłoszenia Użytkownika. Przy tworzeniu użytkownika można nadać mu rolę administratora. Konta takie mają dostęp do stron, na których mogą robić raporty pracownicze czy resetować hasła pracownikom. Podobnie jak poprzednio pobieżnie zostaną opisane zmiany w dotychczasowych elementach strony by następnie przejść do nowych komponentów. Dostanie się do stron przeznaczonych dla administratora skutkuje przeniesieniem użytkownika do strony błędów z komunikatem o braku uprawnień, gdy pracownik jest zalogowany lub gdy nie jest - przeniesieniem do witryny logowania.

3.3.1 Zmiany

3.3.1.1 Pasek Nawigacyjny

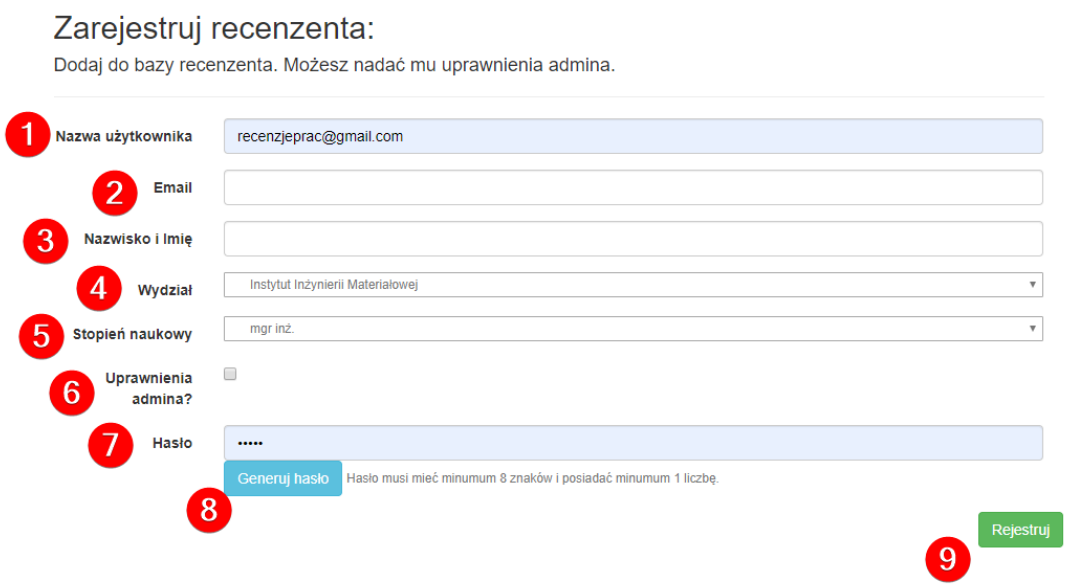
Pasek (Rys. 28) ma w trybie administratora dwie nowe opcje w stosunku do zalogowanego pracownika. Pierwsza to „Rejestruj” (1), która umożliwia rejestrowanie nowych użytkowników, przenosząc do odpowiedniej podstrony. Druga to panel administratora (2), gdzie znajduje się większa część nowych funkcjonalności.



Rys. 28 Pasek nawigacyjny u administratora

3.3.2 Strona Rejestracji

Rejestrowanie pracowników jest możliwe właśnie na tej podstronie (Rys. 29). Wszystkie pola są wymagane, a ich nie wpisanie skutkuje komunikatem o błędnie wprowadzonych danych. W odpowiednie pola administrator musi wpisać: nazwę użytkownika (1), jego mail (2), imię i nazwisko (3), stopień naukowy (4) oraz jego wydział (5). Następnie nadać lub nie uprawnienia administratora (6). Na końcu wymagane jest hasło (7). Jeśli nie chcemy znać hasła użytkownika, istnieje opcja wygenerowania hasła (8). Dane te łącznie z hasłem zostaną wysłane na podany mail po naciśnięciu „Rejestruj” (9). Po zarejestrowaniu, nowy pracownik powinien automatycznie dodać się na listę pracowników dostępną w pasku nawigacji.

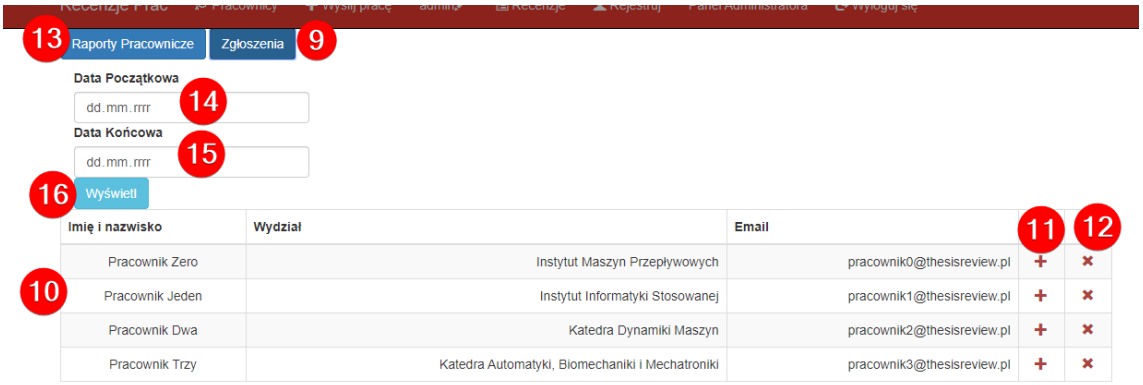


Rys. 29 Strona rejestracji nowego użytkownika

3.3.3 Panel Administratora

W panelu (Rys. 30) (Rys. 31) znaleźć można główne funkcje administratorskie. Na pierwszy plan wychodzi lista użytkowników (1). W tabeli tej będzie brakowało konta zalogowanego użytkownika by ten nie mógł się usunąć. W kolumnach możemy znaleźć podstawowe dane w postaci imienia i nazwiska (2), wydział (3) oraz mail (4). Dwie ikony na końcu umożliwiają kolejno reset hasła (5) i dezaktywowanie konta (6). Jeśli konto jest już dezaktywowane, ikonka zmieni się (7), a kliknięcie w takowy umożliwi ponowną aktywację konta. Na górze dwa przyciski (8) rozwijają odpowiadające im pola.

Rys. 30 Panel administratora 1/2



Rys. 31 Panel administratora 2/2

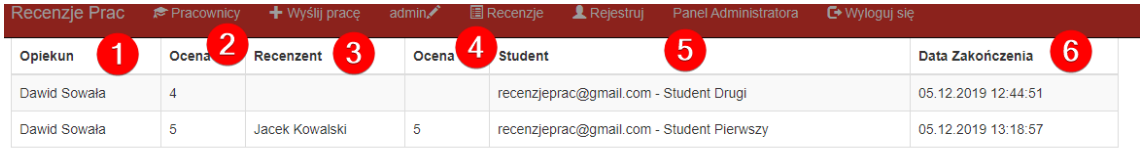
Przycisk „Zgłoszenia” (9) odsłania tablicę (10) ze zgłoszeniami użytkowników. Naciskając plus (11), aplikacja przechodzi do strony rejestracji automatycznie wpisując odpowiednie dane w polach. Przycisk „x” usuwa zgłoszenie (12).

Przycisk „Raporty Pracownicze” (13) ukazuje dwa pola pozwalające wybrać datę startową (14) i końcową (14). Wybranie ich i zaakceptowanie przyciskiem (16) przekierowuje na stronę z raportami pracowniczymi.

3.3.4 Strona Raporty Pracownicze

Witryna (Rys. 32) dostępna jest tylko po wcześniejszym wpisaniu w panelu administratora dat. Tutaj można uzyskać raport dotyczący zakończonych i ocenionych recenzji na przestrzeni wybranych dni. Strona powinna wyświetlić tablice z 6 kolumnami oznaczających:

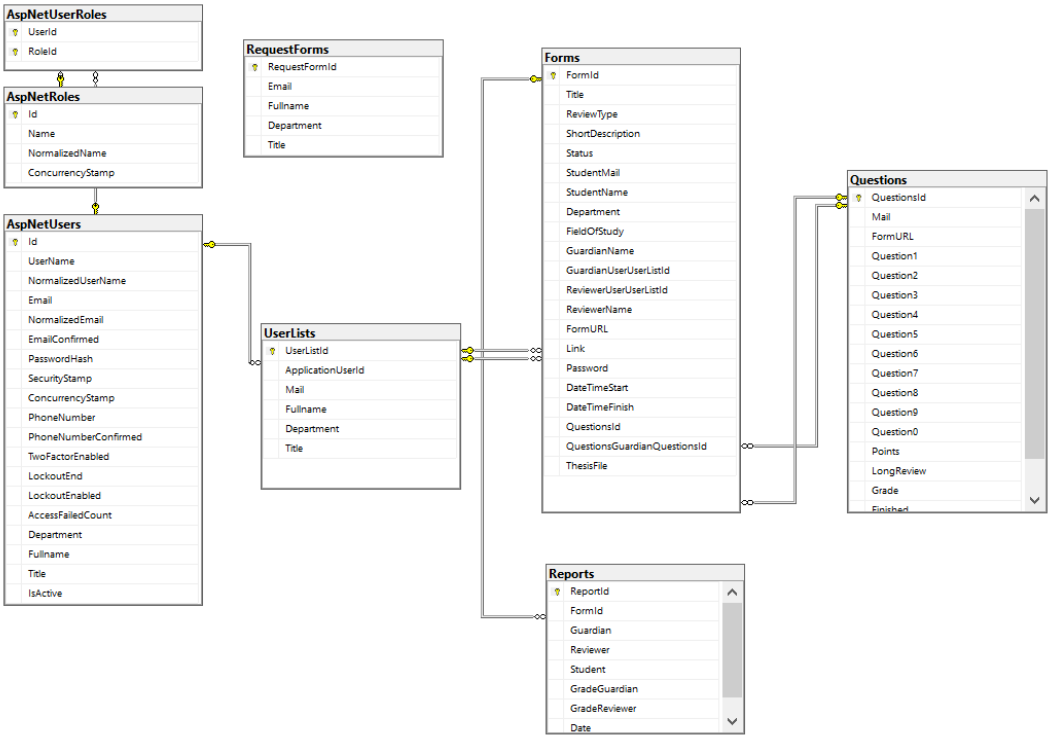
- Opiekun (1) /Recenzent (3) - kolumna zawierająca ich imiona i nazwiska.
- Ocena (2) (4) - oceny wystawione kolejno przez opiekuna i recenzenta.
- Student (5) - mail studenta
- Data Zakończenia (6) - data wystawienia finalnej oceny.



Rys. 32 Raport pracowniczy

3.4 Baza Danych

Jednym z kluczowych zadań w projekcie był stworzenie modelu bazy danej. Schemat bazy stworzonej w programie Microsoft SQL Server Management Studio można zobaczyć poniżej (Rys. 33).



Rys. 33 Schemat bazy

Baza była zmieniania podczas pisania aplikacji dzięki zastosowaniu Entity Framework Core. Pozwolił on na tworzenie tablic w bazie opartych na używanych w aplikacji klasach. Dodając odpowiednie adnotacje nad polami klasie, ustalić można było, czy pole jest np. wymagane. Zaprojektować można było również relacje. Takie rozwiązanie pozwoliło na skupieniu się na aplikacji, gdyż baza byłą dostosowywana pod potrzeby autora pracy.
Każda z tablic posiada unikalny identyfikator, który wymagany jest przez SQL W pracy pominięto pola, których zastosowanie jest oczywiste i skupiono się na opisaniu tych, które według autora są warte opisania.

3.4.1 Forms

Form odpowiada za formularz stworzony przez studenta. Można w nim znaleźć dane studenta oraz opiekuna i recenzenta.
GuardianName/ReviewerName - maila pracowników, pozwalające na pokazanie formularza odpowiedniemu pracownikowi.
Link - link do wglądu w recenzję. To ten sam adres, który wysyłany jest do studenta po stworzeniu formularza.
FormURL - Identyfikator zgłoszenia. Występuje on w linkach do przeglądania czy edytowania formularzu. Stworzony jest przed GUID, dający mu id niemożliwe do wpisania przez przypadek w przeciwieństwie do np. 1,2,3 itd.
Password - specjalne hasło dostępne dla studenta, potrzebne do przejrzenie formularza.
DateTimeStart/Finish - daty, które kolejno wrzucane są przy stworzeniu recenzji przez studenta i ocenieniu jej przez recenzentów.
QuestionId, QuestionGuardianQuestionId - wskazują na kolumnę QuestionsId z tablicy Questions, pozwalając odnieść się do recenzji i jej odpowiedzi.
ThesisFile - plik z pracą zapisany w postaci tablicy bitów.
GuardianUserUserListId/ReviewerUserUserListId - wskazują na pracownika i są powiązane z tablicą UserLists.

3.4.2 Questions

Question zawiera w sobie odpowiedzi na odpowiednie pola z formularza recenzji. Wszystko co zapisze recenzent zapisane jest właśnie w tej tablicy

Question0-9 - reprezentują odpowiedzi w polach formularza. Nie wszystko pola wykorzystane są w każdym typie recenzji.

LongReview - reprezentuje krótką ocenę merytoryczną, znajdującą się w każdym formularzu

Grade - końcowa ocena, która wyświetlana jest potem na raporcie oraz w mailu do użytkownika przy zakończeniu recenzji.

Finished - typ logiczny, prawdziwy, jeśli recenzent zakończył ocenianie.

Status - status recenzji, ma on 3 warianty, „Nowa" dla nowych prac, „Otwarta" dla prac z zaczętą już edycją recenzji oraz „Oceniono", który podobno jak Finished, oznacza sfinalizowanie oceniania.

3.4.3 Reports

Report to tablica tworzącą się po zakończeniu przez recenzentów oceniania pracy dyplomowej. Te dane wyświetlane są przy tworzeniu raportów pracowniczych.

FormId - wskazuje na odpowiadający mu formularz.

GradeReviewer/Guardian - ocena końcowa wystawiona w recenzji.

3.4.4 UserLists

UserList reprezentuje listę pracowników. Duplikuje kolumny AspNetUsers, ale zawiera tylko dane odpowiadające za dane osobowe pracownika.

3.4.5 RequestForms

Formularz tworzony przy zgłoszeniu braku użytkownika wędruje właśnie do tej tablicy, Nie ma ona żadnych relacji z innymi, gdyż nie jest to gotowy użytkownik.

3.4.6 AspNet

Tablicę zawierające „AspNet" stworzone są przez ASP.NET Core w ramach obsługi kont użytkownika. Proces ten dokładnie zostanie opisany w rozdziale z rozwiązaniami. Tutaj skupiono się bardziej na znaczeniu pól. Niektóre z nich nie są wykorzystywane, gdyż nie wpisywały się w określony cel pracy inżynierskiej, zostały one jednak gdyby zdecydowano się na rozwój aplikacji.

AspNetUser zawiera strukturę użytkownika wraz z polami reprezentującymi np. nazwę użytkownika (UserName) czy Email. Z istotnych tutaj pól wymienić można IsActive, który odpowiada za moduł dezaktywacji konta, warto nadmienić, iż to pole wraz z Department, FullName oraz Title nie są domyślnie tworzone przez ASP.NET Core.

AspNetUserRoles oraz AspNetRoles odpowiadają za role użytkowników. W aplikacji mamy normalnego użytkownika z rolą Recenzent oraz administratora z rolą Administrator.

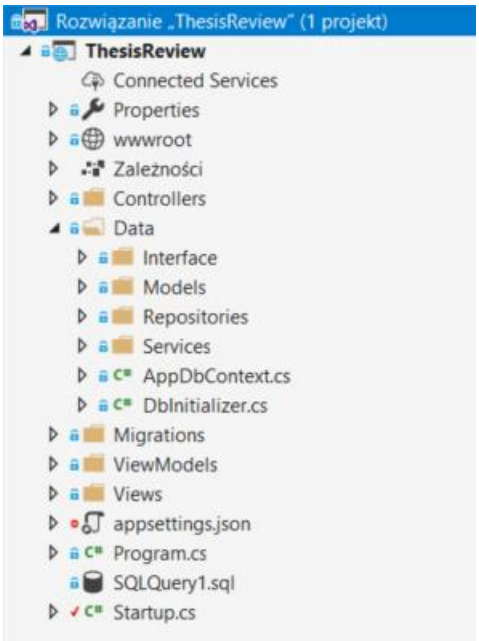
4. Rozwiązania

W tej części pracy, przedstawione zostaną rozwiązania programowe. Zostaną pokazane praktyczne zastosowania technologii i narzędzi opisanych w rozdziale 2.

4.1 ASP.NET Core MVC

Aplikacja została napisana zgodnie z wzorcem projektowym MVC (Model, View Controller). Pozwoliło to na użycie ASP.NET Core MVC - platformy aplikacyjnej, która pozwala budować aplikacje internetowe na wzorcu MVC. Zastosowanie tego rozwiązania widoczne jest w strukturze projektu.

Aplikacja została napisana zgodnie z wzorcem projektowym MVC (Model, View Controller). Pozwoliło to na użycie ASP.NET Core MVC - platformy aplikacyjnej, która pozwala budować aplikacje internetowe na wzorcu MVC. Zastosowanie tego rozwiązania widoczne jest w strukturze projektu (Rys. 34).



Rys. 34 MVC w projekcie

3 najważniejsze foldery - duplikujące nazwy komponentów MVC to:

Models - zawiera w sobie klasy modelu, czyli poszczególne obiekty w aplikacji.

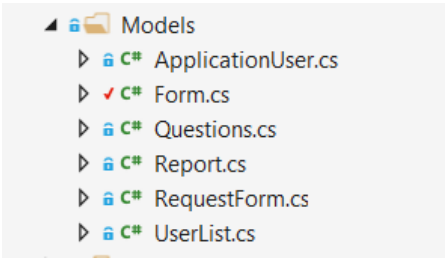
Views - folder z widokami.

Controllers - katalog przechowujący klasy kontrolerów

Jedną z konwencji w ASP.NET Core MVC jest nazywanie kontrolerów podobnie jak ich odpowiadający im folder z widokiem, dodając na końcu nazwy „Controller". Aplikacja dzięki temu zabiegowi potrafi sama odnieść się do odpowiedniego kontrolera.

4.1.1 Model

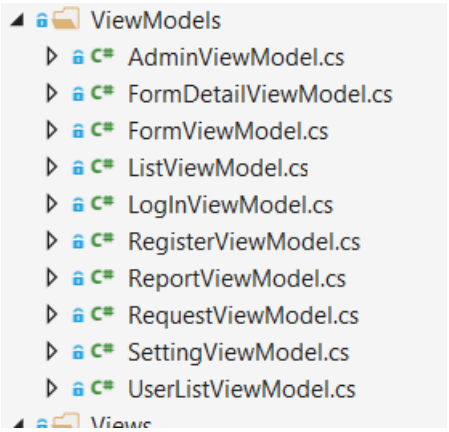
Modele (Rys. 35) w aplikacji wykorzystywane są do półautomatycznego tworzenia relacji w bazach danych poprzez Entity Framework Core. Ich podstawową funkcją jest wysyłanie odpowiednich danych do kontrolera.



Rys. 35 Modele w aplikacji

4.1.2 ViewModel

W kontekście MVC warto opisać ViewModel (Rys. 36). ViewModel to klasa odnosząca się bezpośrednio do zmiennych w Views. Nie ingeruje one bezpośrednio w bazę danych. Poprzez kontroler można wczytać klasy, ale również stworzyć widok, z danymi stworzonymi w kontrolerze danymi.



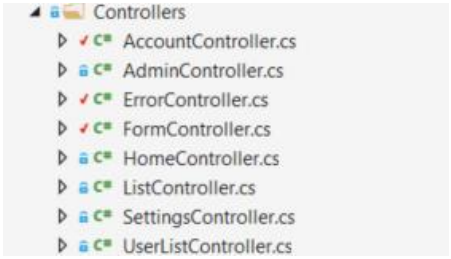
Rys. 36 ViewModel w aplikacji

Kod pokazujący viewmodel w praktyce:

```
public IActionResult Create(FormViewModel fVM)
    {
      //wczytanie danych z FormViewModel do Form
        Form form = new Form
        {
          Title = fVM.Title,
          ReviewType = fVM.ReviewType,
          ShortDescription = fVM.ShortDescription,
          StudentMail = fVM.StudentMail,
          Status = "Nowa",
          ReviewerName = fVM.ReviewerName,
          GuardianName = fVM.GuardianName,
          Department = fVM.Department,
          StudentName = fVM.StudentName
        };
      if (ModelState.IsValid)
      {
        if (!EmailExist(fVM.ReviewerName, fVM.GuardianName,
fVM.ReviewType))
        {
            //Wysłanie danych do FormViewModel w przypadku błędu
walidacji
          fVM.NoError = false;
          fVM.ReviewTypeList = new
SelectList(StringGenerator.ReviewTypesFiller());
          fVM.DepartmentList = new
SelectList(StringGenerator.DepartmentFiller());
          fVM.ErrorMessage = "Brakuje maili w bazie lub mail
opiekuna i recenzenta jest taki sam";
              //Wysłanie ich bezpośrednio do widoku tworzenia
formularza
          return View(fVM);
        }
      [.]
```

4.1.3 Widok

Widoki podzielone są na kontrolery, a każdy kontroler ma określoną ilość widoków.



Rys. 37 Kontrolery w aplikacji

Na obrazku (Rys. 37) można zauważyć katalog, który nosi nazwę kontrolera oraz pliki cshtml, które w tym przypadku odnoszą do metod w kontrolerze oraz domyślnego wyglądu linków w aplikacji. Widok - podstawowa aplikacji internetowej napisany jest podobnie jak wiele innych stron internetowych w HTMLu oraz JavaScript. ASP.NET Core MVC używa silnika Razor. „Razor to silnik renderujący wprowadzony w MVC 3, pozwalający na bardzo łatwe oddzielenie kodu HTML i JavaScript od kodu aplikacji”[3]. Za pomocą symbolu @, silnik wchodzi w specjalny tryb pisania, gdzie możemy wywołać zmienną, czy użyć zapytania warunkowego if.

Na początku można zauważyć wspomniany w poprzednim punkcie ViewModel, który pozwala wyświetlać dane, w tym przypadku tablicę z raportami pracowniczymi.

[3] Żydzik K. Rak T. C# 6.0 i MVC 5. Tworzenie nowoczesnych portali internetowych, Gliwice, Wydawnictwo Helion, ISBN 9788324694969, str. 202

```
//Zastosowanie ViewModelu w widoku
@model ReportViewModel
@{
      ViewData["Title"] = "Raporty";
}
<table class="table table-bordered table-striped">
      <thead>
            <tr>
                  <th>Opiekun</th>
                  <th>Ocena</th>
                  <th>Recenzent</th>
                  <th>Ocena</th>
                  <th>Student</th>
                  <th>Data Zakończenia</th>
            </tr>
      </thead>
      <tbody>
      //Pętla foreach korzystająca z dobrodziejstw silnika Razor.
Uwage należy zwórić na @.
            @foreach (var line in Model.Reports)
            {
            <tr>
                  <td>@line.Guardian</td>
                  <td>@line.GradeGuardian</td>
                  <td>@line.Reviewer</td>
                  <td>@line.GradeReviewer</td>
                  <td>@line.Student</td>
                  <td>@line.Date</td>
            </tr>
            }
      </tbody>
</table>
```

4.1.4 Kontroler

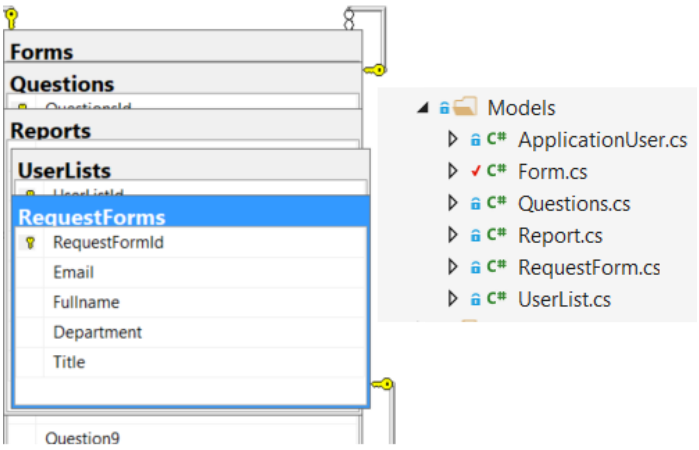
Zdecydowana większa część operacji wykonywana jest w kontrolerach. Reaguje on akcję użytkownika. Przez kontroler, odwiedzający powinien być przenoszony na różne strony. Poszczególne metody w kontrolerze powinny odnosić się do odpowiednich widoków. Odpowiednia nazwa kontrolera pozwala mu się porozumiewać z odpowiadającym mu widokiem automatycznie. „Kontroler nie powinien zawierać logiki biznesowej ani logiki odpowiedzialnej za dostęp do danych”[4], to też w aplikacji większość tych czynności wykonywania jest w repozytoriach. W poniższym kodzie zaobserwować można, że dana akcja dostępna jest tylko dla roli administratora, użytkownicy nie w tej roli, nie będą mieli dostępu do tej metody. Funkcja przyjmuje dwa argumenty, wczytane przez widok.

```
[Authorize(Roles = "Admin")]
public IActionResult Report(string datestart, string datefinish)
{
//Aplikacja tworzy ViewModel z raportami pracowniczymi
  ReportViewModel rVM = new ReportViewModel
  {
      Reports =
_adminRepository.GetReports(Convert.ToDateTime(datestart),
Convert.ToDateTime(datefinish))
  };
  //Aplikacja wysyła ReportViewModel z raportami pracowniczymi do
odpowiadającym im strony
  return View(rVM);
}
```

[4] Żydzik K. Rak T. C# 6.0 i MVC 5. Tworzenie nowoczesnych portali internetowych, Gliwice, Wydawnictwo Helion, ISBN 9788324694969, str. 194

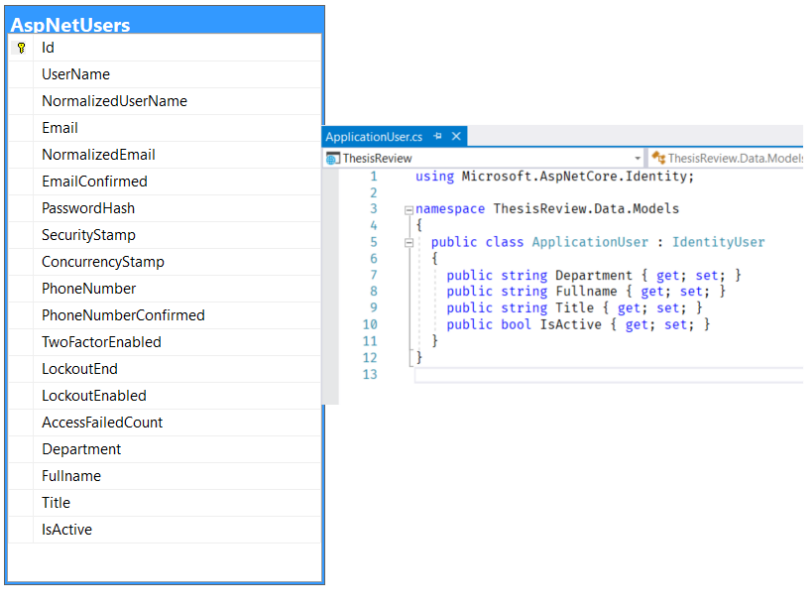
4.2 Entity Framework Core

Początkowo operacje na bazach robione były poprzez stosowanie zapytań bezpośrednio do bazy, pozostałości tego rozwiązanie można znaleźć w klasie DatabaseAction.cs. Do obsługi bazy danych z poziomu aplikacji użyty został Entity Framework Core, który był o wiele wydajniejszy od poprzedniego rozwiązania. Jedną z jego kluczowych funkcji jest automatyczne stworzenie bazy danych na podstawie modelu i zawartych w modelu adnotacji (Rys. 38). Takie rozwiązanie pozwala na szybkie zmiany w bazie. Jak można zauważyć, znaczna część tablic w schemacie bazy ma swojego reprezentanta w postaci klasy modelu.



Rys. 38 Porównanie modelu ze schematem bazy

Model “ApplicationUser” dziedziczy z IdentityUser (Rys. 39Rys. 39), dodając nowe pola.



Rys. 39 Dziedziczenie z IdentityUser

W poniższym kodzie przedstawiono mapowanie przez Entity Framework Core odpowiednich modeli do tablic w bazie

```
//Tworzenie modelu usera na podstawie pokazanego wcześniej modelu ApplicationUser
public class AppDbContext : IdentityDbContext<ApplicationUser>
{


    public AppDbContext(DbContextOptions<AppDbContext> options)
:base(options)
    {


    }


    public DbSet<Form> Forms { get; set; }
    public DbSet<Questions> Questions { get; set; }
    public DbSet<UserList> UserLists { get; set; }
    public DbSet<Report> Reports { get; set; }
    public DbSet<RequestForm> RequestForms { get; set; }
}
```

W Startup.cs, który uruchamiany jest przy starcie aplikacji, przedstawiono funkcję odpowiadającą za poprawne zmapowanie modeli w aplikacji do tablic w bazie na podstawie klasy AppDbContext

```
public void ConfigureServices(IServiceCollection services)
{
      services.AddDbContext<AppDbContext>(options =>

      options.UseSqlServer(_configurationRoot.GetConnectionString(
"DefaultConnection")));
}
```

AppDbContext jest pomostem między aplikacją a bazą danych. Umożliwia to na podstawowe operacje takie jak czytanie z bazy, dodawanie, edycja i usuwanie rekordów.

```
public void DeleteUser(string userId)
    {
      //Czytanie usera z bazy
      var deleteduser = _appDbContext.Users.FirstOrDefault(p =>
p.Email == userId);
        //Deaktywacja jego konta
      deleteduser.IsActive = false;
        //Zapisanie powiązanych z bazą rekordów z pomocą
AppDbContext
      _appDbContext.SaveChanges();
    }
```

Zauważyć można automatyczne nawiązanie relacji między rekordem z bazy a obiektem 'deleteduser'. Zmiany w jego polach są następnie zatwierdzane i zapisywane w bazie.

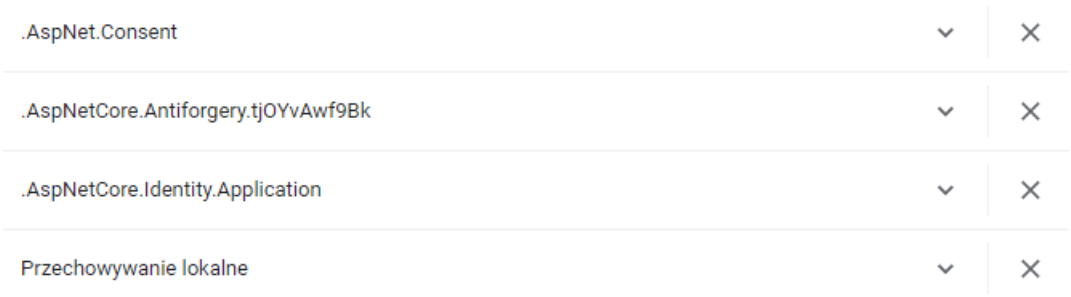
Poniższa metoda w klasie Startup.cs, ukazuje obsługę kont użytkowników wraz z podstawowymi funkcjami jak logowanie czy rejestracja. W szybki i łatwy sposób można za pomocą tego narzędzia zaimplementować logowanie się wbudowaną w nie funkcją.

```
services.AddIdentity<ApplicationUser, IdentityRole>()
            .AddEntityFrameworkStores<AppDbContext>()
            .AddDefaultTokenProviders();
```

Logowanie w aplikacji:

```
public async Task<IActionResult> Login(LogInViewModel
logInViewModel)
    {
      if (!ModelState.IsValid)
        return View(logInViewModel);
      //wyszukanie użytkownika na podstawie maila
      var user = await
_userManager.FindByEmailAsync(logInViewModel.Email);
      if (user != null && user.IsActive)
      {
        //sprawdzenie poprawności hasła oraz wpisanie użytkownika
do przeglądarki w ramach ciasteczek cookies
        var result = await
_signInManager.PasswordSignInAsync(user, logInViewModel.Password,
false, false);
        if (result.Succeeded)
        {
          if (string.IsNullOrEmpty(logInViewModel.ReturnUrl))
            return RedirectToAction("Index", "Home");
          return Redirect(logInViewModel.ReturnUrl);
        }
      }
      ModelState.AddModelError("wrongform", "Nieprawidłowy email
lub hasło");
      return View(logInViewModel);
    }
```

Po zalogowaniu się dane użytkownika zapisywane są do „ciasteczek „w przeglądarce (Rys. 40Rys. 40).



Rys. 40 Ciasteczka tworzone przez aplikacje

Przechowywany lokalnie jest zalogowany użytkownik, ale także czy zaakceptował regulamin. Wylogowanie się usuwa odpowiednie ciasteczka z przeglądarki.

4.3 Routing

Aplikacja dzięki użyciu ASP.NET Core wraz z ASP.NET Core MVC, potrafi sama odnaleźć się w linkach URL. Poprzez odpowiednie użycie kontrolerów, użytkownik zostaje przeniesiony do odpowiednich podstron. Do ścieżek, do których nie używają domyślnych ustawień użyta jest specjalna metoda w Startup.cs.

```
app.UseMvc(routes =>
      {
            //Domyślna ścieżka, wykorzystywa tam, gdzie nie
określona jest konfiguracja.
        routes.MapRoute(
                  name: "default",
                  template:
"{controller=Home}/{action=Index}/{id?}");
        routes.MapRoute(
                  name: "FormEdit",
                  template: "Form/{action}/{id}", defaults: new {
Controller = "Form", action = "Edit", id = "" });
            //Przykład z przeglądaniem recenzji przez studenta.
Przyjmuje id oraz hasło formularza w ścieżce
        routes.MapRoute(
                  name: "FormView",
                  template: "Form/{action}/{id}/{password}",
defaults: new { Controller = "Form", action = "View", id = "",
password = "" });
        routes.MapRoute(
                  name: "UserDelete",
                  template: "Admin/{action}/{id}", defaults: new {
Controller = "Admin", action = "Delete", id = ""});
        routes.MapRoute(
                  name: "UserEdit",
                  template: "Admin/{action}/{id}", defaults: new {
Controller = "Admin", action = "Edit", id = ""});
        routes.MapRoute(
                  name: "List",
                  template: "List/{id}", defaults: new {
Controller = "List", id = "" });
      });
```

4.4 Pierwsze uruchomienie

Podczas startowania aplikacji uruchamiana jest klasa Program.cs. Zawarta jest tutaj metoda Main, w której to aplikacja jest uruchamiana.

```
public class Program
  {
    public static void Main(string[] args)
    {
      CreateWebHostBuilder(args).Build().Run();
    }


    public static IWebHostBuilder CreateWebHostBuilder(string[]
args) =>
        WebHost.CreateDefaultBuilder(args)
            .UseStartup<Startup>().UseSetting("detailedErrors",
"true")
                .CaptureStartupErrors(true);
  }
```

Aplikacja przechodzi do klasy Startup.cs. Tutaj aktywowane są wszystkie odpowiednie usługi wraz z podstawową konfiguracją. Opisać warto podpinanie się do bazy danej za pomocą „connectionstring” czytanego z jsona z podstawową konfiguracją.

```
services.AddDbContext<AppDbContext>(options =>

options.UseSqlServer(_configurationRoot.GetConnectionString("Defau
ltConnection")));
```

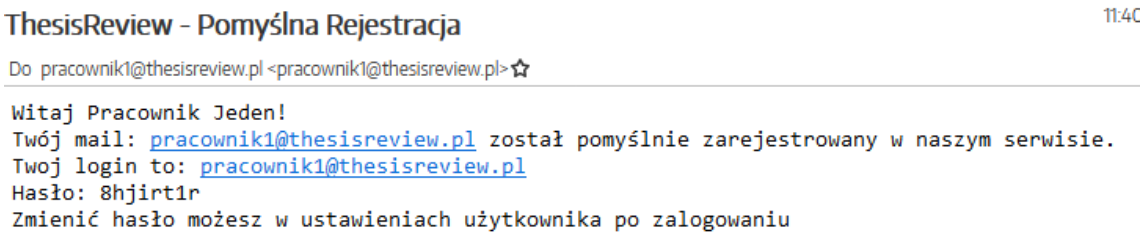
Występują tu również opisane wcześniej obsługi mapowania bazy oraz zarządzanie modułem użytkowników. Na samym końcu uruchamiana jest metoda służąca do sprawdzenia bazy danych i jej ew. stworzenia. Służy do tego funkcja Seed z klasy DbInitializer. Klasa ta podczas uruchamiania ma dwa cele. Pierwszym jest sprawdzenie czy baza danych istnieje. Przy braku bazy wbudowana metoda powinna samodzielnie utworzyć bazę danych, jeżeli istnieje już połączenie przez ConnectionString. Tutaj tworzony jest także główny administrator w oparciu o dane, które powinien dostarczyć przyszły korzystający z aplikacji.

4.5 MailKit

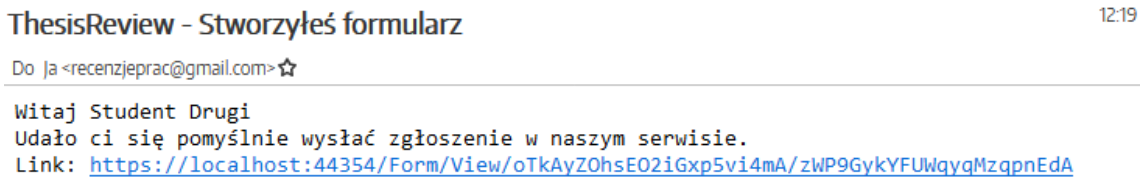
Do wysyłania maili w aplikacji użyto biblioteki MailKit. Podając odpowiednie dane używanej poczty, umożliwiła ona ustawienie nadawcy, tematu oraz treści wiadomości.

```
public static void Send(string receiver, string subject, string
content)
    {
      var message = new MimeMessage();
        //Adres mailowy nadawcy wraz z jego nazwą
      message.From.Add(new MailboxAddress("Recenzje Prac",
MailName));
        //Adres odbiorcy
      message.To.Add(new MailboxAddress(receiver, receiver));
        //Temat maila
      message.Subject = subject;
        //Treści wiadomości
      message.Body = new TextPart("plain")
      {
        Text = content
      };
      //Łączenie z klientem Smtp oraz wysyłka maila
      using (var client = new SmtpClient())
      {
        client.ServerCertificateValidationCallback = (s, c, h, e)
=> true;
        client.Connect(MailSMTP, Int32.Parse(MailPort) , false);
        client.Authenticate(MailName, MailPassword);
        client.Send(message);
        client.Disconnect(true);
      }
    }
```

Przykładowe maile (Rys. 41) (Rys. 42):



Rys. 41 Mail rejestracji



Rys. 42 Mail Stworzenia formularzu

5. Podsumowanie

Autor postawił sobie za cel zdigitalizowania procesu recenzowania prac dyplomowych. Aplikacja spełnia więc wymagania, zarówno pod względem samego celu jak i podstawowych funkcjonalności dla aplikacji internetowej, w skład których wchodzą takie rzeczy jak: logowanie, rejestracja, konto administratorów.

Możliwości rozwoju są jak na aplikacje internetową ogromne. Kluczowym jednak jest, że tego typu rozwiązanie powinno być zaimplementowane na platformach uczelnianych. Przykładem jest chociażby Wikamp Politechniki Łódzkiej czy USOS Uniwersytetu Łódzkiego. Wbudowanie tego w istniejące już środowisko znacznie zwiększyło by potencjał na rozwój. Niestety z oczywistych powodów, Autor mógł pozwolić sobie tylko na zamkniętą aplikację. Podobnie jak inne serwisy tego typu, można byłoby rozwijać aplikację niemal w nieskończoność. W czasie pracy zawsze można było coś dodać, czegoś brakowało. Dobry deweloper powinien umieć wyznaczyć zakres swojego projektu tak, aby nie rozrósł się do przekraczających jego możliwości rozmiarów. Koniec końców to użytkownicy mają znaczący wpływ na aplikację i to na podstawie ich działań powinno się ją dostosowywać.

Bibliografia

1. Freeman A. (2017), ASP.NET Core MVC 2. Zaawansowane programowanie. Wydanie VII, Gliwice, Wydawnictwo Helion, ISBN 9788328346017
2. Oliveira J. Brucher M. (2017), ASP.NET Core 2.0. Wprowadzenie, Gliwice, Wydawnictwo Helion, ISBN 9788328345003
3. Gajda W. (2013), Git. Rozproszony system kontroli wersji, Gliwice, Wydawnictwo Helion, ISBN 9788324673056
4. Żydzik K. Rak T. C# 6.0 i MVC 5. Tworzenie nowoczesnych portali internetowych, Gliwice, Wydawnictwo Helion, ISBN 9788324694969
5. https://docs.microsoft.com/pl-pl/aspnet/core/?view=aspnetcore-2.1
6. https://docs.microsoft.com/pl-pl/aspnet/core/mvc/overview?view=aspnetcore-2.1

Spis rysunków