POLITECHNIKA ŚLĄSKA

WYDZIAŁ AUTOMATYKI, ELEKTRONIKI I INFORMATYKI

Praca dyplomowa inżynierska

Aplikacja internetowa wspomagająca komunikację pomiędzy dyplomantem a promotorem

autor: Bartosz Borys

kierujący pracą: dr inż. Michał Maćkowski

Gliwice, styczeń 2019

załącznik nr 2 do zarz. nr 97/08/09

Oświadczenie

Wyrażam zgodę / Nie wyrażam zgody* na udostępnienie mojej pracy dyplomowej / rozprawy doktorskiej*.

Gliwice, dnia 7 stycznia 2019

.....
(podpis)

.....
(poświadczenie wiarygodności
podpisu przez Dziekanat)

* podkreślić właściwe

Oświadczenie promotora

Oświadczam, że praca „Aplikacja internetowa wspomagająca komunikację pomiędzy dyplomantem a promotorem" spełnia wymagania formalne pracy dyplomowej inżynierskiej.

Gliwice, dnia 7 stycznia 2019

.....

(podpis promotora)

Spis treści

Rozdział 1

Wstęp

Podczas pisania pracy dyplomowej niezbędna jest częsta komunikacja dyplomanta ze swoim promotorem. Dyplomant może chcieć uzyskać pomoc od swojego promotora w rozwiązaniu problemów związanych z pracą dyplomową, takich jak logika pracy lub forma jej opisu. Najczęściej jednak dyplomant przekazuje promotorowi opisową część pracy dyplomowej w celu jej weryfikacji. Podczas całego procesu, promotor musi często kilkukrotnie ją przeczytać. Po przeczytaniu może mieć uwagi dotyczące pracy, w takim przypadku powinien przekazać je swojemu dyplomantowi, a jeżeli ich nie ma, po prostu zatwierdzić pracę. Z tego powodu potrzebne byłoby narzędzie, które umożliwi w szybki i prosty sposób przesłanie przez dyplomanta pracy w celu weryfikacji jej przez promotora, a w przypadku błędów, zapewni możliwość przesłania komentarzy dyplomantowi. Aktualnie jest wiele możliwości realizacji takiego zadania. Przykładem może być e-mail, repozytorium kontroli wersji (GitHub), fizyczne dostarczenie dokumentu na konsultacje. Takie rozwiązania posiadają jednak wady, które mogą utrudnić przebieg całego procesu.

Po przemyśleniu powyższego, rozsądnym posunięciem jest próba stworzenia systemu, który posiadałby taką funkcjonalność. Do tego celu autor pracy musiał się zastanowić, jak taki system miałby działać oraz zaprojektować go w taki sposób, aby był łatwo dostępny dla każdego dyplomanta oraz promotora. Dobrym rozwiązaniem w erze internetu okazała się aplikacja internetowa, która do działania wymaga jedynie przeglądarki.

Do zaprojektowania, a następnie zaimplementowania tej aplikacji, przydatna okazała się wiedza zdobyta przez autora podczas studiów. Między innymi niezbędne okazały się umiejętności projektowania systemu przy użyciu diagramów UML (ang. *Unified Modeling Language*) oraz wiedza z zakresu programowania komputerów od zmiennych i pętel zaczynając, a kończąc na szablonach i wzorcach projektowych. Dodatkowym i wartościowym źródłem wiedzy okazały się książki związane z tematyką programowania. Podsumowując, powyższa wiedza znacząco ułatwiła osiągnięcie celu pracy.

Autor zaprojektował i zaimplementował aplikację samodzielnie.

1.1 Cel pracy

Niniejsza praca inżynierska przedstawia proces projektowania, a następnie implementacji aplikacji internetowej, która ma na celu usprawnienie komunikacji między promotorem a dyplomantem.

1.2 Plan pracy

Proces tworzenia aplikacji jest obszerny, z tego powodu został zastosowany podział na rozdziały. Jednym z nich jest analiza podjętego tematu (rozdział 2), podczas której temat zostanie objaśniony i będzie porównany do podobnych, aktualnie istniejących systemów. Następnym jest rozdział „wymagania i narzędzia” (rozdział 3), w którym między innymi zostaną przedstawione wymagania funkcjonalne oraz niefunkcjonalne aplikacji. Kolejnym rozdziałem jest „specyfikacja zewnętrzna” (rozdział 4), w której zostanie poruszona kwestia obsługi aplikacji przez potencjalnego użytkownika. W rozdziale „specyfikacja wewnętrzna” (rozdział 5) zostanie opisana techniczna strona aplikacji, na przykład to, które zewnętrzne biblioteki zostały użyte. Przedostatni rozdział „weryfikacja i walidacja” opisywać będzie zakres testowania i sprawdzania poprawności działania aplikacji. Natomiast ostatni rozdział jest rozdziałem podsumowującym (rozdział 7), gdzie znajdą się wnioski i przemyślenia autora, a także zostanie przedstawiony możliwy dalszy rozwój aplikacji.

Rozdział 2

Analiza tematu

Tematem pracy jest stworzenie aplikacji internetowej służącej do komunikacji między promotorem a dyplomantem. Taka aplikacja powinna zawierać szereg różnych funkcjonalności, które ułatwią komunikację podczas sprawdzania pracy dyplomowej. W takiej aplikacji powinna istnieć funkcjonalność dodawania pliku pracy dyplomowej w formacie PDF (ang. *Portable Document Format*). Promotor i dyplomant muszą mieć możliwość komunikowania się przy pomocy czatu czasu rzeczywistego, a promotor dodatkowo mógłby mieć możliwość wysłania ważnych wiadomości, przykładowo organizacyjnych, do wszystkich swoich dyplomantów. Ważnym zagadnieniem z punktu widzenia informatyki jest to, że aplikacja ta powinna być aplikacją internetową. Należało więc wybrać technologię oraz architekturę. Zdecydowano się na korzystanie z architektury REST (ang. *Representational State Transfer*). Zapewnia ona przesyłanie informacji w sposób czysty i zgodny ze standardem. Ważne było również podzielenie aplikacji na mniejsze części — Frond-End, czyli to, co widzi użytkownik, Back-End, czyli wszystkie obliczenia, odczyt, zapis i edycja danych na bazie oraz sama baza danych.

2.1 Wprowadzenie do dziedziny

Dziedziną, jak zostało określone, jest programowanie aplikacji internetowych z zastosowaniem architektury REST API (ang. *Application Programming Interface*). Aplikacje tego typu są podzielone na dwie części: Front-End i Back-end.

Front-End jest graficznym interfejsem użytkownika, reprezentuje on dane, które zostają pobrane z bazy danych poprzez internetowy interfejs. Do implementacji interfejsu graficznego wykorzystane mogą zostać platformy programistyczne takie jak Angular, React, Vue. Natomiast Back-End jest to internetowy interfejs, przy tworzeniu którego mogą okazać się pomocne platformy programistyczne, takie jak ASP.NET, Django, Spring, Node czy Symfony. Wykorzystanie platform programistycznych przy implementacji aplikacji znacząco przyśpiesza pracę, ponieważ zapewnia gotowe rozwiązania podstawowych problemów.

Interfejs graficzny aplikacji komunikuje się poprzez zapytania HTTP (ang. *Hypertext Transfer Protocol*) z internetowym interfejsem, aby otrzymać dane. Zapytania mogą posiadać różnego rodzaju parametry lub nagłówki. Parametry wysyłane są w formacie JSON (ang. *Javascript Object Notation*), które są odczytywane przez Back-End i w zależności od nich podejmowane są odpowiednie akcje, a następnie wysyłana jest odpowiedź. Nagłówki są wysyłane w takim samym formacie jak parametry. Mogą one określać między innymi zawartość zapytania poprzez wartość klucza Content-Type lub token autoryzacyjny. Więcej na temat nagłówków można dowiedzieć się z dokumentacji [4]. Każde z zapytań jest zdefiniowane pod konkretnym adresem oraz z konkretną metodą HTTP. One również, zgodnie z architekturą REST, muszą być stworzone w określony sposób. Adresy powinny być określone przez rzeczowniki, jednoznacznie wyznaczające zasoby, jakie pod tym adresem można znaleźć. Czynność, jaką chcemy wykonać na danym zasobie — po prostu odczytać, zmienić, a może dodać nowy rekord — określona jest z kolei poprzez metodę HTTP. Ważne jest, by w adresie nie używać czasowników, ponieważ powoduje to niepotrzebne duplikowanie informacji.

2.2 Opis znanych rozwiązań

Jedno z podobnych rozwiązań stanowi strona prac dyplomowych Politechniki Śląskiej, jednakże jest to oficjalna strona, na którą powinno się dodawać już końcowe rozwiązania. Dodatkowo, oficjalna strona z Politechniki Śląskiej nie zapewnia płynnej komunikacji z promotorem. Istnieją też inne aplikacje, które zapewniającą komunikację wraz z możliwością przesyłania plików, takie jak Gmail lub Messenger od firmy Facebook. Aplikacje te jednak posiadają bardzo dużo różnych użytkowni-

ków, nie tylko dyplomantów, co może być uciążliwe. Dodatkowo promotor w takich aplikacjach nie ma możliwości ustawienia ocen danemu studentowi.

Rozdział 3

Wymagania i narzędzia

Przed implementacją aplikacji, na początku trzeba określić jej wymogi. Można to zrobić przez analizę wymagań funkcjonalnych oraz niefunkcjonalnych, a także przypadków użycia, wykonując odpowiedni diagram UML. Kolejną ważną rzeczą, po określeniu wcześniej wymienionych, jest wybranie narzędzi i metodyk pracy, z użyciem których tworzona będzie aplikacja.

3.1 Wymagania funkcjonalne i niefunkcjonalne

Określenie wymagań jakie ma spełniać dany system znacząco ułatwia proces projektowania aplikacji, ponieważ pozwala jednoznacznie określić, co musi zostać stworzone w jej ramach. Na podstawie wymagań łatwiej też określić interfejs użytkownika, ponieważ wiadomym jest, co dokładnie powinno się na nim znaleźć. W opisywanym systemie przewidywane są następujące wymagania funkcjonalne i niefunkcjonalne.

- Wymagania funkcjonalne:
    1. Logowanie się do systemu z poprawnym loginem i hasłem,
    2. Zarejestrowanie się przy użyciu adresu e-mail,
    3. Komunikacji między promotorem a dyplomantem w formie czatu,
    4. Realizacja odbierania wiadomości w czasie rzeczywistym,
    5. Promotor może wybrać studenta, z którym chce się komunikować,

6. Użytkownik może zmienić hasło,
7. Dyplomant może wybrać swojego promotora,
8. Użytkownik może zmienić swój profil,
9. Promotor może zobaczyć wysłane przez siebie wiadomości globalne,
10. Promotor może wysłać wiadomość globalną,
11. Promotor może usunąć wysłaną wiadomość globalną,
12. Promotor może edytować wysłaną wiadomość globalną,
13. Dyplomant może zobaczyć globalne wiadomości swojego promotora,
14. Promotor może zobaczyć listę swoich dyplomantów,
15. Dyplomant może dodać swoją pracę inżynierską w formacie PDF,
16. Dyplomant może podglądnąć lub pobrać swoją pracę inżynierską,
17. Dyplomant może podglądnąć lub pobrać poprawioną przez promotora pracę inżynierską w formacie PDF,
18. Dyplomant może zobaczyć oceny, średnią oraz końcową ocenę studiów,
19. Promotor może ustawić ocenę końcową ze wszystkich semestrów studiów, ocenę z obrony oraz z pracy dyplomowej, a następnie system obliczy średnią ważoną oraz ocenę końcową studiów,
20. Promotor może podglądnąć, a następnie pobrać pracę inżynierską wybranego dyplomanta,
21. Promotor może wysłać lub podglądnąć wysłaną poprawioną pracę inżynierską wybranego studenta,
22. Promotor może przyjąć lub odrzucić pracę inżynierską, zmieniając aktualny status pracy.

- Wymagania niefunkcjonalne:

1. Hasła w bazie będą przechowywane w postaci zaszyfrowanej,
2. Aplikacja będzie przystosowana do przeglądarki Firefox i Chrome,
3. Interfejs aplikacji będzie się dostosowywał do różnych rodzajów ekranów,

4. Aplikacja nie będzie korzystała z danych, które nie są konieczne do poprawnego działania aplikacji,
5. Dane użytkowników niezarejestrowanych nie będą przechowywane.

3.2 Aktorzy

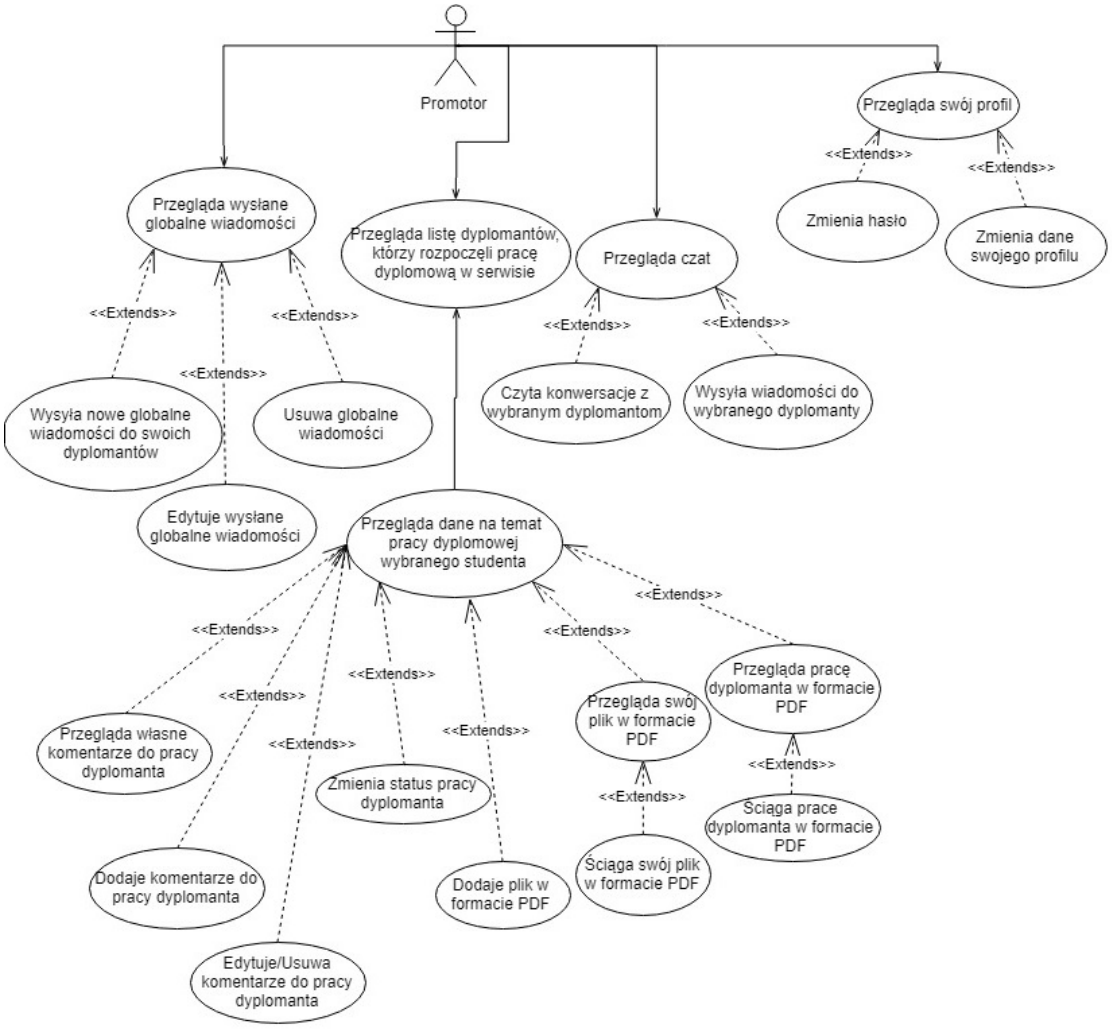
W czasie projektowania aplikacji zostali rozróżnieni trzej aktorzy: Gość, Dyplomant, Promotor. Przy pomocy diagramu UML zostały określone możliwe akcje, które może wykonać dany aktor korzystający z aplikacji.

- Gość — niezalogowany użytkownik serwisu. Dozwolone akcje tego aktora są ograniczone jedynie do zalogowania się oraz rejestracji.
- Dyplomant — zalogowany użytkownik, który posiada podstawowe uprawnienia podczas korzystania z serwisu. Może przeglądać jedynie swoje dane.
- Promotor — zalogowany użytkownik, który posiada, tak jak dyplomant, podstawowe uprawnienia, lecz dodatkowo może przeglądać dane wszystkich swoich dyplomantów, a niektóre z nich zmieniać.

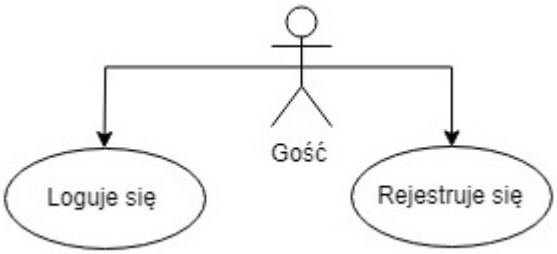
3.3 Przypadki użycia

Podczas projektowania systemu zdefiniowano przypadki użycia, które zostały podzielone odpowiednio na aktorów:

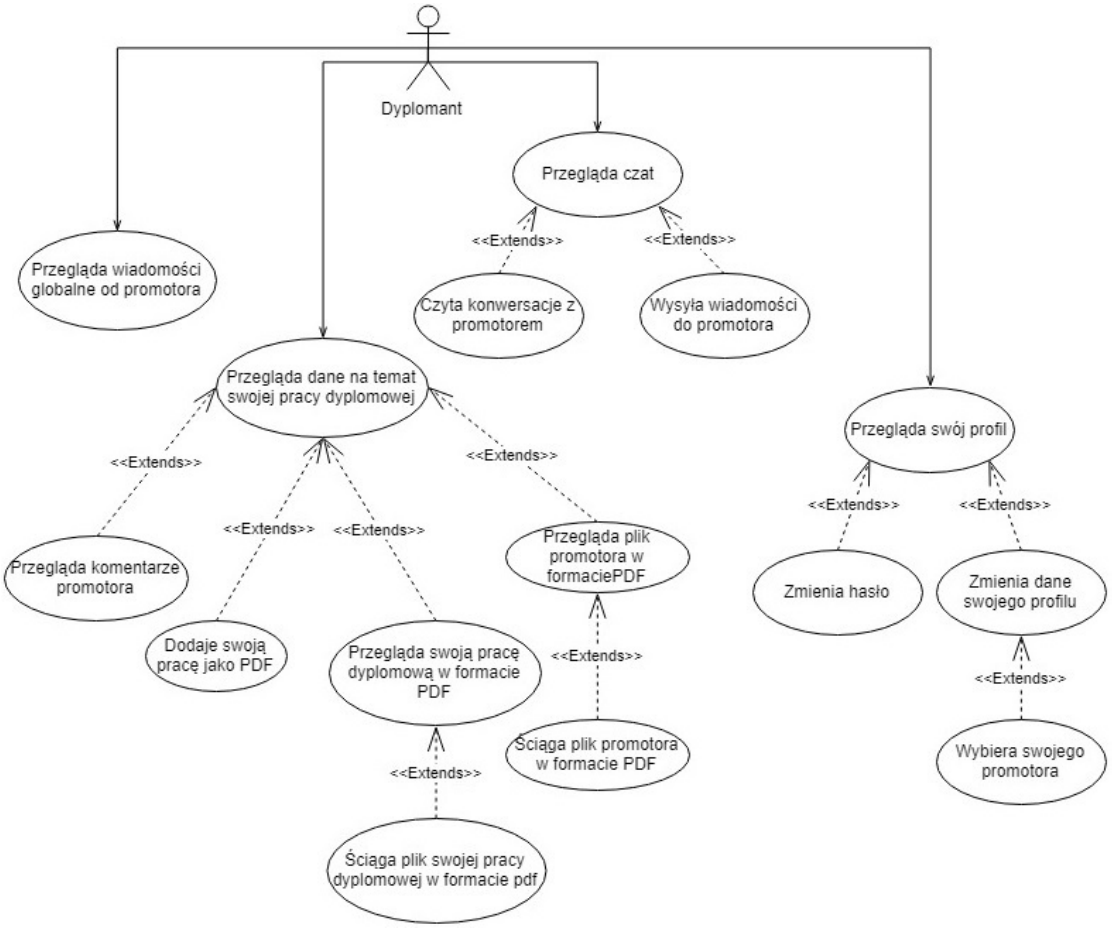
- Promotor (rys. 3.1),
- Gość (rys. 3.2),
- Dyplomant (rys. 3.3).



Rysunek 3.1: Diagram przypadków użycia promotora



Rysunek 3.2: Diagram przypadków użycia gościa



Rysunek 3.3: Diagram przypadków użycia dyplomanta

1. Gość — posiada jedynie dwa przypadki użycia 3.2. W pierwszym przypadku gość loguje się, aby uzyskać token potrzebny do korzystania z systemu. Logowanie może się nie powieść, wtedy otrzyma stosowny komunikat o błędzie. W drugim przypadku gość rejestruje się, aby utworzyć konto potrzebne do zalogowania się i otrzymania tokenu. Podczas rejestracji użytkownik podaje swoje dane. W przypadku błędu danych, zostanie również wyświetlony odpowiedni komunikat o błędzie. Po udanej rejestracji użytkownik może zalogować się do systemu z wcześniej podanym loginem i hasłem.

2. Dyplomant — dla użytkownika z tą rolą zostały określone przypadki użycia na rysunku 3.3. Użytkownik może jedynie przeglądać wiadomości globalne od swojego promotora. Nie posiada innych uprawnień niż odczytywanie tych wiadomości. Może się zdarzyć sytuacja, w której promotor nie wysłał żadnych wiadomości. W takim przypadku nie zostanie wyświetlona żadna wiadomość, a użytkownik zobaczy pusty ekran z panelem nawigacji u góry. Następnie student może przejrzeć dane swojej pracy dyplomowej. Użytkownik może dalej, rozszerzając czynność swojej pracy dyplomowej, przeglądnąć komentarze promotora. Może także dodać swoją pracę dyplomową w formacie PDF, przeglądać ją oraz ściągnąć. Te same czynności, poza dodaniem i edycją, może również dokonać na pracy promotora. Kolejnym przypadkiem użycia jest przeglądanie czatu. Dyplomant może nawiązać konwersację jedynie ze swoim promotorem. Rozszerzając ten przypadek, istnieje możliwość czytania konwersacji oraz wysyłania wiadomości do promotora. Ostatnim przypadkiem użycia założonym dla dyplomanta jest przeglądanie swojego profilu. Przeglądanie profilu można rozszerzyć o zmianę swoich danych osobowych, gdzie dodatkowo można wybrać swojego promotora. Podczas przeglądania swojego profilu, można również zmienić swoje hasło.

3. Promotor — Konto promotora posiada najbardziej rozbudowaną funkcjonalność względem pozostałych ról. Funkcjonalność dla promotora jest bardzo zbliżona do funkcjonalności dyplomanta. Można stwierdzić, że konto promotora jest rozszerzeniem konta dyplomanta. W przypadku użycia globalnych wiadomości promotor może przeglądać wiadomości wysłane przez siebie samego. Dodatkowo ten przypadek użycia ma możliwość rozszerze-

nia o wysłanie globalnych wiadomości do swoich dyplomantów, edytowanie oraz usuwanie ich. Następnie, w przypadku użycia przeglądania prac dyplomowych, promotor, zamiast konkretnej pracy, widzi listę wszystkich prac swoich dyplomantów. Każdy element listy posiada skrótowy opis zawierający informacje na temat imienia oraz nazwiska dyplomanta, oceny, statusu oraz daty ostatniej akcji dokonanej przez promotora na konkretnej pracy. Promotor może wejść w szczegóły wybranej pracy z listy, co stanowi rozszerzenie poprzedniego przypadku użycia. Tam, podobnie jak student, może przeglądać dane pracy dyplomowej, takie jak: status, nazwa, oceny cząstkowe, średnia ocen, ocena końcowa, plik pracy dodany przez studenta oraz plik dodany przez siebie samego. Można edytować niektóre dane, edycja jest rozszerzeniem tego przypadku użycia. Każdy z tych plików, podobnie jak dyplomant, promotor może ściągnąć na dysk. Może także zaakceptować lub odrzucić pracę. Ostatnią rzeczą, jakiej może dokonać na szczegółach pracy jest dodanie krótkich komentarzy, zbyt małych, by warto było dodawać je w pliku PDF. Każdy z wcześniej dodanych komentarzy może również edytować lub usunąć. Reszta przypadków użycia jest niemal identyczna, jak te które zostały opisane w przypadku dyplomanta. Sekcja czatu różni się tym, że promotor posiada kilku dyplomantów, z tego powodu ma możliwość wyboru dyplomanta, z którym chce nawiązać konwersację. Edycja informacji na profilu także się różni, na przykład brakiem możliwość wybrania swojego promotora co w tym przypadku byłoby nonsensowne, dlatego edycja została ograniczona do zmiany imienia i nazwiska. Promotor ma identyczną jak dyplomant możliwość zmiany hasła.

3.4 Opis narzędzi

Po zaprojektowaniu systemu przyszedł czas na wybranie technologii, przy użyciu których miałaby zostać stworzona aplikacja. Po zapoznaniu się z aktualnie dostępnymi technologiami oraz wyciągnięciu kilku wniosków wybrane zostały następujące narzędzia:

- Angular 7

Został wybrany do utworzenia interfejsu graficznego aplikacji. Jest on popularny, rozwijany, bardzo dobrze opisany w dokumentacji [1]. Dodatkowo jest zintegrowany z językiem TypeScript, który jest kompilowany. Etap kompilacji zmniejsza ryzyko występowania błędów w kodzie, co jest ogromnym atutem.

- ASP.NET Core v2.1

  Ta technologia została wybrana do napisania internetowego interfejsu aplikacji. Używa, tak jak w przypadku platformy programistycznej Angular, języka kompilowanego, którym jest C# [3]. Język ten jest dostatecznie szybki, a jego składnia jest bardzo przyjemna. Wartościowe informacje na temat platformy ASP.NET można zaczerpnąć bezpośrednio z dokumentacji na stronie firmy Microsoft [2].

- MySql

  Na bazę wybrany został MySql z uwagi na to, że łatwo podłączyć go do serwera aplikacji przy użyciu Entity Framework. Jest to jedna z popularnych baz danych, z tego powodu istnieje duża szansa na znalezienie rozwiązania ewentualnego problemu.

3.5 Metodyka pracy nad projektowaniem i implementacją

Podczas pracy nad projektem zostały wykorzystane niektóre popularne metodyki programowania:

- TDD (ang. *Test Driven Development*). Jest to metodyka, w której tworzenie kodu rozpoczyna się od pisania testów, które przedstawiają funkcjonalność danego fragmentu. Testy z początku nie zwracają rezultatu pozytywnego, jednakże z kolejnymi liniami implementacji, zaczynają kolejno zwracać wyniki pozytywne. Takim sposobem można uzyskać kod spełniając wszystkie założone wymagania, oraz zabezpieczenie go testami. Odwołując się do tekstu zapisanego w książce [10]: „Clean code that works, in Ron Jeffries' pithy

phrase, is the goal of Test-Driven Development (TDD). Clean code that works is a worhtwhile goal for a whole bunch of reasons.", można dowiedzieć się, że celem tej metodyki jest zapewnienie czystości kodu. Sama metodyka TDD nie generuje czystego kodu, tylko tworzy elementy, czyli testy, które wspomagają ten proces. W tym miejscu warto odnieść się do fragmentu z innej książki [11]: „(...)programiści piszący frameworki wiedzą, że pierwotny efekt ich pracy nie będzie idealny — musi ewoluować, w miarę jak nabierają oni doświadczenia. Wiedzą także, że kod częściej będzie odczytywany i zmieniany niż pisany od nowa. W związku z tym kluczowe znaczenie ma zapewnienie czytelności kodu i możliwości jego modyfikowania. Rozwiązaniem jest refaktoryzacja.", można dowiedzieć się, że kod, który będzie rozszerzany musi być czysty i zrozumiały. Z tego powodu nieustannie trzeba dbać o jego jakość. Natomiast nieczysty kod powinno się poddać procesowi refaktoryzacji, zwłaszcza, gdy jest on pokryty testami. Po zmianach w kodzie można sprawdzić, czy wszystko działa zgodnie z założeniami uruchamiając wcześniej napisane testy. Owe testy powinny potwierdzić, bądź nie, poprawność działania danego kodu.

- SOLID jest to zbiór pięciu zasad [8], z użyciem jakich powinno tworzyć się klasy w programowaniu orientowanym obiektowo. Są to kolejno: zasada jednej odpowiedzialności, zasada otwarte-zamknięte, zasada podstawienia liskov, zasada segregacji interfejsu, zasada odwrócenia zależności. Podczas tworzenia aplikacji starano się przestrzegać tych zasad.

Rozdział 4

Specyfikacja zewnętrzna

Każda aplikacja internetowa musi zapewnić interakcję z użytkownikiem, która powinna zostać wcześniej zaprojektowana, a następnie zaimplementowana. Użytkownik musi również zostać uświadomiony, jak powinien korzystać z aplikacji, co jest wymagane do korzystania z niej. Natomiast potencjalna osoba, która chciałaby wykorzystać aplikację lub ją rozszerzyć, powinna zostać poinformowana, jak skonfigurować aplikację.

4.1 Wymagania sprzętowe i programowe

Do obsługi klienta, czyli aplikacji dla użytkownika systemu, wystarczy komputer z dostępem do internetu potrafiący obsłużyć przeglądarki takie jak Chrome, Opera, Firefox. Dodatkowo do uruchomienia części Back-End lokalnie wymagana jest jeszcze platforma .NET, której środowisko trzeba pobrać, a następnie zainstalować (.NET SDK wersja testowana 2.1.6). Wymagane jest również zainstalowanie, a następnie uruchomienie bazy danych oraz wygenerowanie tabel z pliku SQL (ang. *Structured Query Language*). Baza danych, która została wykorzystana przez autora to MySQL.

4.2 Sposób instalacji

Przed przystąpieniem do instalacji, w zależności od portów, jakie zostaną ustawione poszczególnym częściom systemu, należy skonfigurować tak zwane „Connection Strings". Po ich skonfigurowaniu, należy przebudować obie części aplikacji. Po kompilacji, aby uruchomić aplikację, pliki odpowiedzialne za interfejs graficzny należy przenieść, w przypadku systemu Windows, do głównego katalogu dysku „C". Następnie przy pomocy komendy „dotnet" uruchomić plik DLL (ang. *Dynamic-Link Library*) odpowiedzialny za Back-End, to znaczy w wierszu poleceń (command line, powershell) należy wpisać: „dotnet nazwa pliku.dll ". Przy uruchamianiu należy upewnić się, czy baza jest włączona i komunikuje się poprawnie z interfejsem.

4.3 Kategorie użytkowników

W systemie występują trzy rodzaje użytkowników: Promotor, Dyplomant oraz Gość. Promotor jest to osoba, która prowadzi prace inżynierskie — ocenia je, zmienia stan, komentuje, doradza. Promotor może wysłać globalną wiadomość do swoich dyplomantów. Dyplomant może natomiast dodawać swoją pracę dyplomową, czytać komentarze dotyczące pracy oraz komunikować się z promotorem poprzez czat. Może także podejrzeć wszystkie swoje oceny. Gość natomiast posiada najmniejsze uprawnienia. Może się on tylko zalogować i zarejestrować.

4.4 Sposób obsługi

Aplikację obsługuje się przy użyciu myszki i klawiatury oraz przeglądarki internetowej. Przetestowane przeglądarki to Mozilla Firefox i Google Chrome. Internet Explorer nie jest kompatybilny z aplikacją, ponieważ nie wspiera nowych standardów wprowadzonych w technologiach takich jak: HTML, CSS, Javascript. Po uruchomieniu wszystkich elementów aplikacji należy uruchomić plik HTML (ang. *Hypertext Markup Language*), który powinien znajdować się w miejscu, gdzie umieszczone zostały pliki interfejsu użytkownika, o których mowa w podpunkcie 4.2.

4.5 Administracja systemem

System nie posiada roli/konta dla administratora. W wypadku ewentualnych problemów wymagana jest ingerencja w kod aplikacji czy bazę danych. Można również zrestartować serwer lub bazę posiadając odpowiednie uprawnienia w systemie operacyjnym.

4.6 Kwestie bezpieczeństwa

System uwzględnia aspekty bezpieczeństwa. Hasła użytkowników są przechowywane w bazie danych jako zaszyfrowany ciąg znaków przy użyciu algorytmu B-Crypt. Zapytania SQL są wykonywane poprzez Entity Framework, co zabezpiecza przed ewentualnymi atakami SQL-Injection. Do autoryzacji wykorzystywany jest JWT [6] (ang. *JSON web token*), który użytkownik otrzymuje po poprawnym zalogowaniu się. Taki token posiada określony czas ważności, po upływie którego użytkownik musi go odświeżyć poprzez ponowne zalogowanie się, lub automatycznie jest odświeżany w ramach aktualnej sesji. Wszystkie zapytania wysyłane do internetowego interfejsu poza logowaniem i rejestracją wymagają autoryzacji wspomnianym tokenem. Dodatkowo sprawdzana jest poprawność wszystkich parametrów z otrzymywanych przez interfejs zapytań. Niektóre informacje dostępne są jedynie dla użytkowników konkretnej roli, aby na przykład dyplomant nie mógł podejrzeć wyników innego dyplomanta.

4.7 Przykład działania

Użytkownik przed pierwszym kontaktem z aplikacją najprawdopodobniej nie posiada konta. W takim wypadku użytkownik, który aktualnie jest gościem, musi się zarejestrować. Rejestracja przedstawiona została na rysunku 4.2. Po rejestracji gość posiada już konto, na które może się zalogować (rysunek 4.1). W zależności od użytego adresu e-mail podczas rejestracji, użytkownik po zalogowaniu posiada rolę dyplomanta lub promotora. Zalogowany użytkownik może poruszać się bezproblemowo po aplikacji, jednakże poszczególne sekcje w zależności od roli różnią się.

W pierwszej kolejności opisany zostanie użytkownik z kontem dyplomanta. Po zalogowaniu dyplomant odsyłany jest do sekcji głównej (rysunek 4.3), gdzie może przeglądać wiadomości globalne wysłane przez jego promotora. Wiadomości mają różne priorytety w zależności od koloru. W kolejnej sekcji dyplomant może podejrzeć status swojej pracy inżynierskiej. W przypadku pierwszej wizyty na tej sekcji wymagane jest potwierdzenie rozpoczęcia swojej pracy dyplomowej poprzez kliknięcie przycisku „rozpocznij". Gdy praca dyplomowa jest już rozpoczęta (rysunek 4.6), można podejrzeć jej status, dodać swoją pracę w formie PDF. Warto tutaj zauważyć, że dodany przez siebie plik PDF dyplomant może również podejrzeć oraz pobrać, tak samo jak plik wysłany przez promotora. W tej sekcji dyplomant może przejrzeć komentarze wysłane przez promotora. Dalej, w sekcji wiadomości dyplomant może komunikować się ze swoim promotorem w formie czatu czasu rzeczywistego. Po otwarciu sekcji, widoczny jest czat z promotorem, jak przedstawiono na rysunku 4.7. Wiadomości aktualnego użytkownika oznaczone są kolorem niebieskim, natomiast wysłane przez drugą osobę, kolorem żółtym. Wiadomości można wysłać poprzez wpisanie treści w pole tekstowe znajdujące się pod wiadomościami, a następnie trzeba wcisnąć przycisk „Enter" lub przycisnąć przycisk „wyślij". Ostatnia sekcja zawiera informacje o profilu — rysunek 4.8. Tutaj dyplomant może zmienić hasło, imię, nazwisko, promotora i inne swoje informacje.

Aplikacja z perspektywy promotora trochę się różni. Po zalogowaniu się, promotor, tak samo jak dyplomant, zostaje przekierowany na stronę główną aplikacji (rysunek 4.3). Tutaj promotor może przeglądnąć wiadomości wysłane przez siebie, może je usunąć bądź edytować, klikając w przyciski znajdujące się w nagłówku wiadomości. Dodatkowo promotor może wysłać wiadomość do swoich dyplomantów, poprzez kliknięcie w lewym górnym rogu przycisku koperty. W menu, które się pojawi (rysunek 4.4), promotor może wybrać status wiadomości, wpisać nagłówek oraz jej treść. Okienko edycji wygląda identycznie jak okno dodania nowej wiadomości. W sekcji prac dyplomowych po wejściu pojawia się lista swoich dyplomantów (rysunek 4.5). Po kliknięciu w wiersz z dyplomantem otworzy się widok z danymi na temat jego pracy inżynierskiej (rysunek 4.6), jest on bardzo podobny do tego, które widzi użytkownik z kontem dyplomanta. Poza funkcjonalnością, jaką posiada osoba z kontem dyplomanta, promotor dodatkowo może: edytować ocenę, zaakceptować lub odrzucić pracę. W sekcji wiadomości (rysunek 4.7), promotor

może wybrać dyplomanta, z którym chce się skomunikować. Po wybraniu go, pojawi się okno czatu z wcześniejszą konwersacją dyplomanta z promotorem, o ile wystąpiła. W ostatniej sekcji (rysunek 4.8), promotor, tak samo jak dyplomant, może zmienić swoje hasło i dane swojego profilu.

4.8 Scenariusze korzystania z systemu

Podczas korzystania z systemu, użytkownik może wykonać różnego rodzaju akcje. Akcje te mogą zakończyć się różnie w zależności od podjętych przez użytkownika decyzji, bądź wprowadzonych przez niego danych. Kilka scenariuszy takich akcji zostało przedstawionych poniżej:

1. Gość rejestruje się do systemu (rysunek 4.2) z adresem e-mail, który nie jest w domenie Politechniki Śląskiej. Dostaje powiadomienie o konieczności ponowienia próby, z wykorzystaniem adresu, który znajduje się w domenie Politechniki.

2. Gość rejestruje się do systemu, używając adresu e-mail, który już należy do jakiegoś użytkownika. Dostaje powiadomienie o konieczności ponowienia próby.

3. Gość rejestruje się do systemu, używając adresu e-mail odpowiadającego promotorowi. Zakłada konto z uprawnieniami promotora.

4. Gość rejestruje się do systemu, używając adresu e-mail odpowiadającego studentowi. Zakłada konto z uprawnieniami studenta.

5. Gość próbuje zalogować się do systemu. Podaje niepoprawne dane, proszony jest więc o ponowienie próby (rysunek 4.2).

6. Gość loguje się do systemu na konto promotora, aby przejrzeć swoją komunikację ze studentami. Pokazują mu się globalne wiadomości, które wcześniej wysłał do swoich studentów, tak jak widać po prawej stronie zrzutu ekranu na rysunku 4.3. Może z poziomu tego panelu dodać nową wiadomość, określając jej rodzaj, nagłówek oraz treść. Ma możliwość usunięcia lub edycji utworzonej wcześniej wiadomości, uznawszy, że jest nieaktualna. Może także przejść do

panelu „Wiadomości” 4.7, w której to może przejrzeć listę swoich dotychczas wysyłanych ze studentami wiadomości. Może wyszukać studenta, któremu chciałby wysłać wiadomość, a następnie to zrobić. Student taki dostanie tą wiadomość od razu — pojawi mu się ona, nawet jeśli właśnie przegląda swoje wiadomości z promotorem, bez konieczności odświeżania strony.

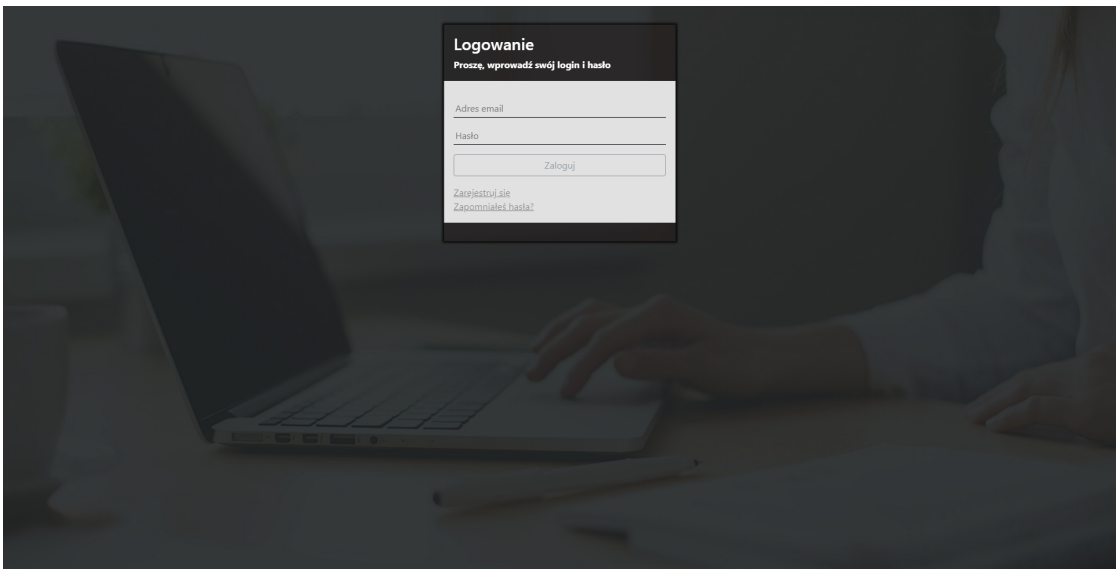
7. Gość loguje się do systemu na konto promotora, aby przejrzeć prace swoich studentów. Przechodzi w sekcję „Prace”, w której znajduje się lista studentów wraz ze statusem pracy oraz datą ostatniej modyfikacji (rysunek 4.5). Po kliknięciu w wybrany element listy, może zobaczyć szczegóły pracy oraz jej oceny, jak widać po prawej stronie zrzutu ekranu 4.6. Podgląda pracę studenta, by ją przeczytać, dodaje swój plik z uwagami do pracy, bądź wystawia ocenę. Jeżeli praca nie wymaga uwag, akceptuje ją. Jeżeli ma tylko drobne uwagi, niewymagające wrzucenia całego pliku, może jedynie dodać komentarz do pracy. W przypadku, gdy zmieni zdanie, komentarz taki usuwa lub edytuje.

8. Gość loguje się do systemu na konto studenckie, by przejrzeć komunikację ze swoim promotorem. Na stronie głównej czyta napisane przez niego globalne wiadomości, jak widać po lewej stronie zrzutu ekranu 4.3. Wiadomości te są dla niego jedynie informacyjne, nie może on na nie w żaden sposób odpowiedzieć. W sekcji „Wiadomości” czyta oraz wysyła mu prywatne wiadomości, przykładowo jeśli ma problem w pisaniu pracy, co widać na zrzucie ekranu 4.7. Nie może on komunikować się z nikim poza swoim promotorem, dlatego na liście znajduje się tylko i wyłącznie on.

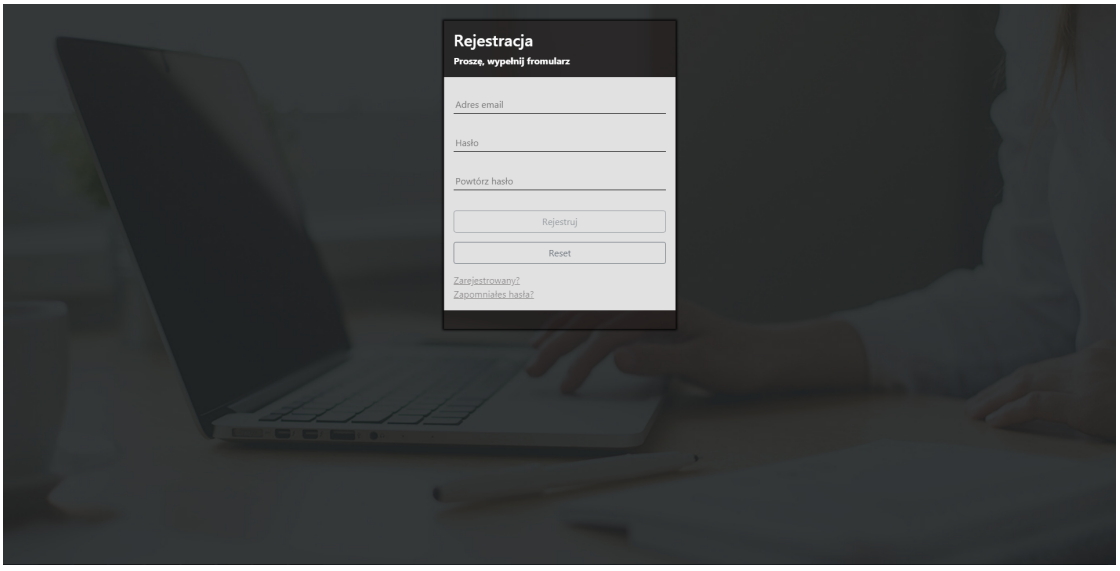
9. Gość loguje się do systemu na konto studenckie, by dodać lub zaktualizować swoją pracę. Przechodzi w zakładkę „Prace”, gdzie może dodać nowy plik PDF, a także zobaczyć opinię promotora, jak widać na zrzucie ekranu 4.6. Widzi, że ten dodał własny plik z uwagami, więc poprawia swoją wersję pracy i dodaje zaktualizowany plik. Może uznać, że konieczne będzie zaktualizowanie pliku również jeśli promotor dodał jakiś komentarz, który widoczny będzie w dolnej części strony.

10. Gość loguje się do systemu. Wchodzi w zakładkę „Konto”, na której edytuje

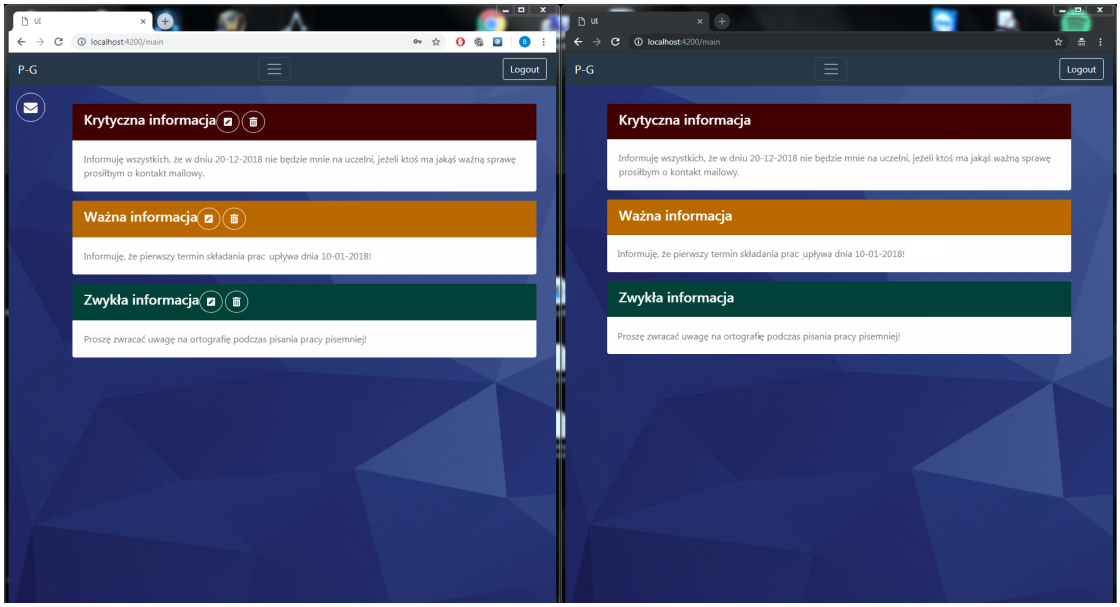
swoje podstawowe informacje. W przypadku, gdy zalogowanym użytkownikiem jest student — wybiera lub zmienia promotora. Jeżeli ma taką potrzebę, zmienia swoje hasło.



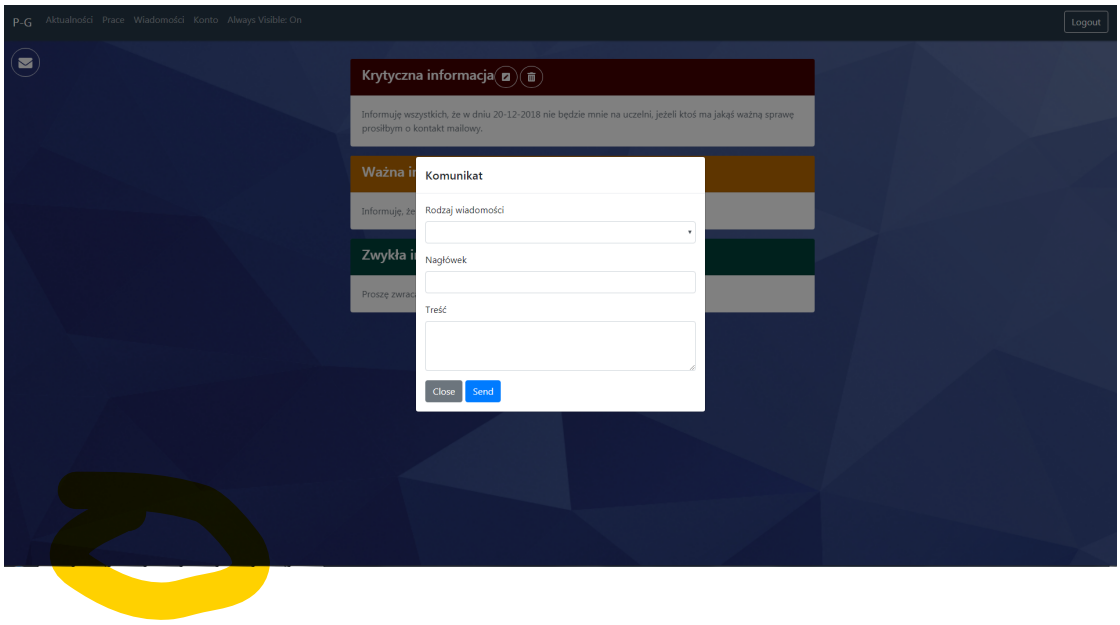
Rysunek 4.1: Zrzut ekranu logowania się



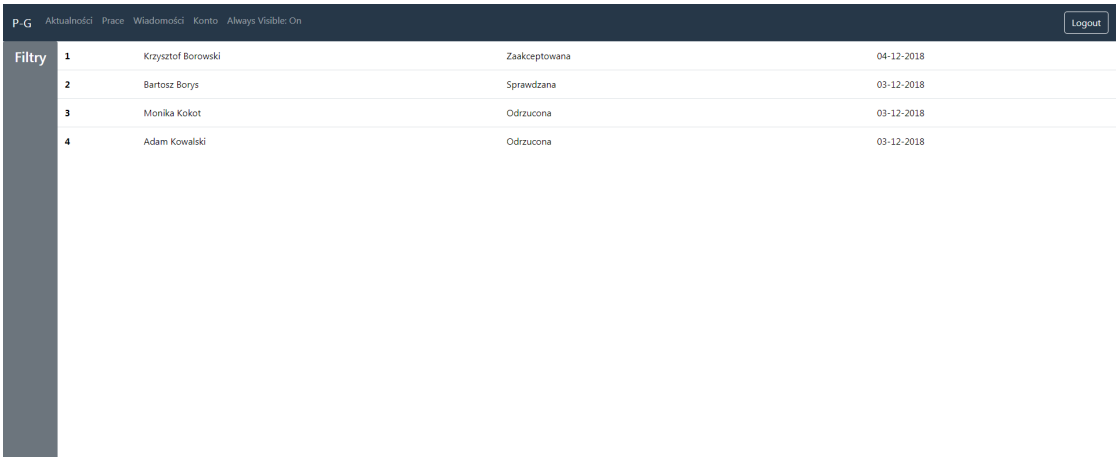
Rysunek 4.2: Zrzut ekranu rejestracji



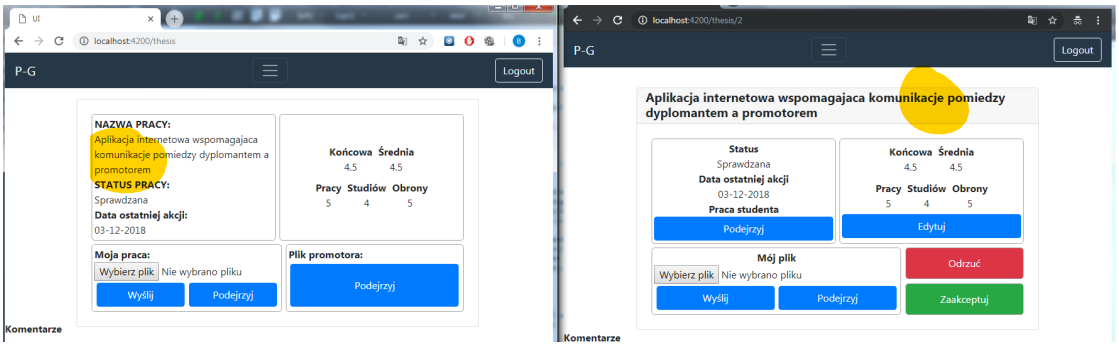
Rysunek 4.3: Zrzut ekranu wiadomości globalnych, wersja dla promotora z lewej, a dla dyplomanta z prawej



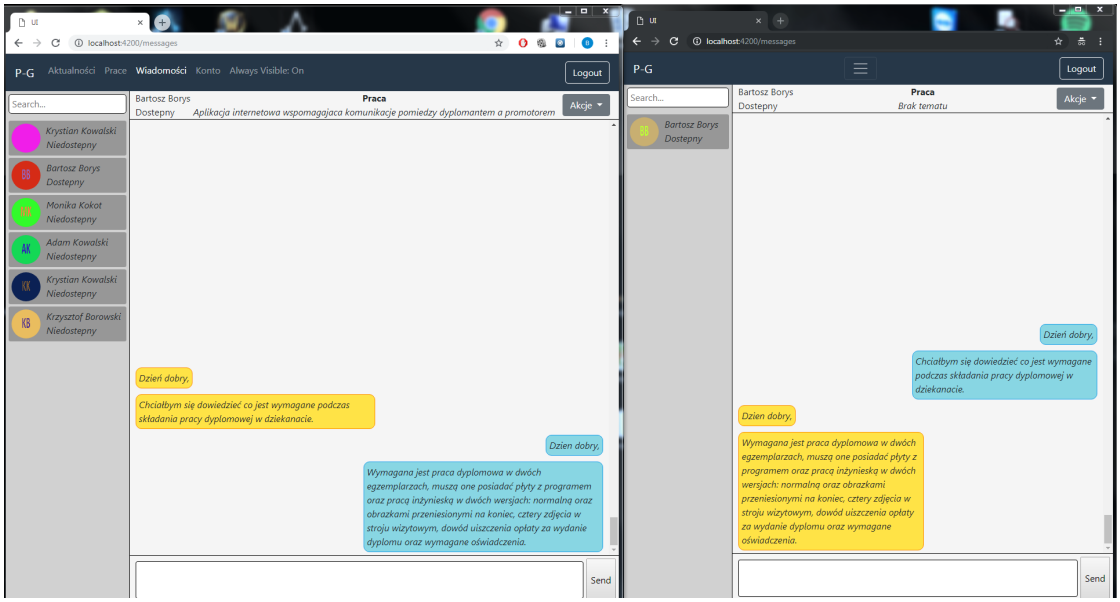
Rysunek 4.4: Zrzut ekranu menu dodawania bądź edycji globalnej wiadomości



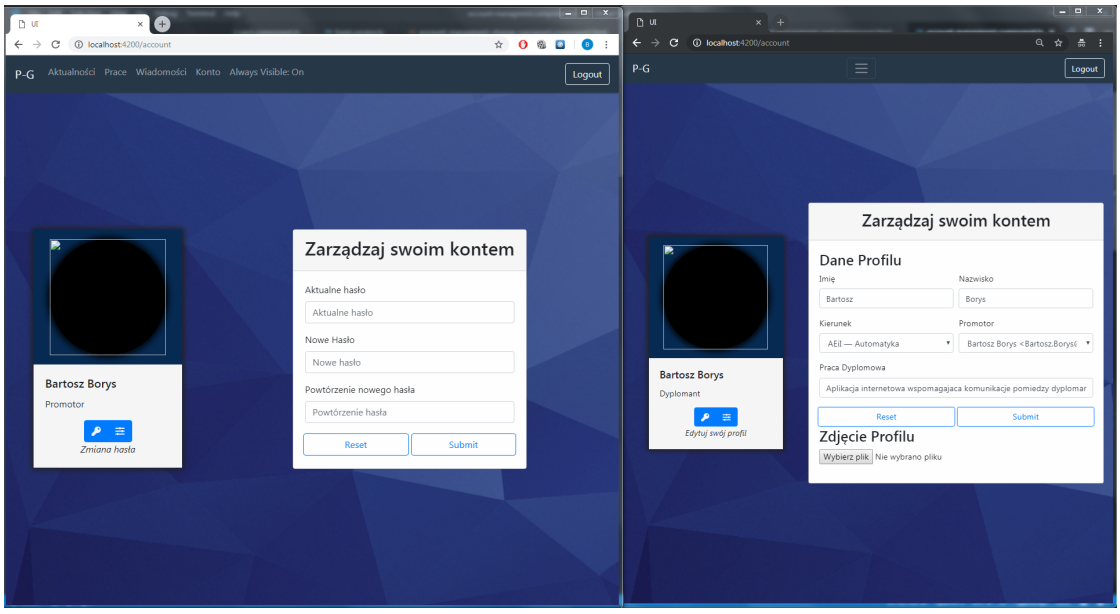
Rysunek 4.5: Zrzut ekranu listy dyplomantów



Rysunek 4.6: Zrzut ekranu szczegółów pracy dyplomowej dla promotora z prawej i dyplomanta z lewej



Rysunek 4.7: Zrzut ekranu czatu promotora z lewej i dyplomanta z prawej



Rysunek 4.8: Zrzut ekranu zmiany hasła z lewej i edycji swojego profilu z prawej

Rozdział 5

Specyfikacja wewnętrzna

Ideą aplikacji internetowej jest usprawnienie procesu promowania przez promotora dyplomanta poprzez uproszczenie komunikacji między obiema stronami. Aplikacja stawia na łatwy dostęp, niskie wymagania sprzętowe, przejrzystość interfejsu użytkownika, bezpieczeństwo, czystość kodu oraz testowanie go. Czystość kodu jest podstawą podczas tworzenia aplikacji, zwłaszcza tych, które mają być w przyszłości rozwijane. Przytaczając fragment z książki [12]: „ I know of one company that, in the late 80s (. . .) They had rushed the product to market and had made a huge mess in the code. As they added more and more features, the code got worse and worse until they simply could not manage it any longer. It was the bad code that brought the company down.” można dowiedzieć się, że powodem upadku pewnej firmy był nieczysty kod, a raczej problemy, jakie spowodował. Dlatego zawsze warto poświęcić czas, na poprawienie jakości kodu aplikacji.

5.1 Architektura systemu

System został podzielony na dwie główne części: interfejs graficzny oraz internetowy interfejs.

- Interfejs graficzny

  Nazywany również Front-End. Jest to część aplikacji, która jest interfejsem graficznym użytkownika. Część ta została napisana przy użyciu platformy Angular. Pisanie interfejsu opiera się o komponenty, które separują logikę,

dzieląc ją na różne poziomy abstrakcji. Komponenty oraz inne części mogą być generowane przez CLI (ang. *Command Language Interpreter*). Składowe komponentu są od siebie odseparowane, to jest HTML, CSS, TS posiadają swoje własne pliki, które podczas kompilacji są łączone, co ułatwia utrzymanie porządku w kodzie. Komunikacja z serwerem następuje poprzez zapytania HTTP, które są wbudowane, co znacznie ułatwia pracę. Dodatkowo w standardzie platformy Angular są testy jednostkowe, których szkielety podczas generacji komponentu są tworzone automatycznie, co znacznie ułatwia pracę. Tutaj zostają w określony sposób wyświetlone dane pobrane z serwera. Zamieszczone są tutaj również wszystkie formularze, dzięki którym użytkownik może komunikować się z serwerem, na przykład logować się, wysyłać wiadomości.

- Internetowy interfejs

  Warstwa biznesowa aplikacji, nazywana często Back-End. Tutaj zamieszczona jest cała logika bezpieczeństwa i dystrybucji danych. Warstwa ta jest pośrednikiem między bazą danych a interfejsem graficznym. Na wskutek zapytań wysyłanych poprzez użytkownika z przeglądarki internetowej wykonuje stosowne akcje odsyłając odpowiedź najczęściej w formacie JSON. Internetowy interfejs został wykonany w oparciu platformę o ASP.NET Core v2.1 oraz architekturę REST, która ułatwia komunikację między poszczególnymi systemami. Z prawie wszystkich elementów internetowego API mogą korzystać jedynie osoby autoryzowane, czyli takie, które posiadają token autoryzacji. Token można uzyskać po zalogowaniu się do systemu. Jest to kluczowe dla zapewnienia bezpieczeństwa, aby dane nie trafiły do niepożądanych osób. Dodatkowo internetowy interfejs komunikuję się z bazą danych. Baza danych jest to kontener na dane użytkowników korzystających z systemu. Komunikacja następuje przy użyciu biblioteki Entity Framework. Dzięki wspomnianej bibliotece, do wyciągania danych z bazy nie wykonuje się bezpośrednio zapytań SQL. Zamiast tego mapuje się tabele na obiekty C#, a następnie, wywołując odpowiednie metody, otrzymuje się dane. Takie rozwiązanie zapobiega ewentualnemu atakowi SQL-Injection, co zwiększa znacząco bezpieczeństwo aplikacji.

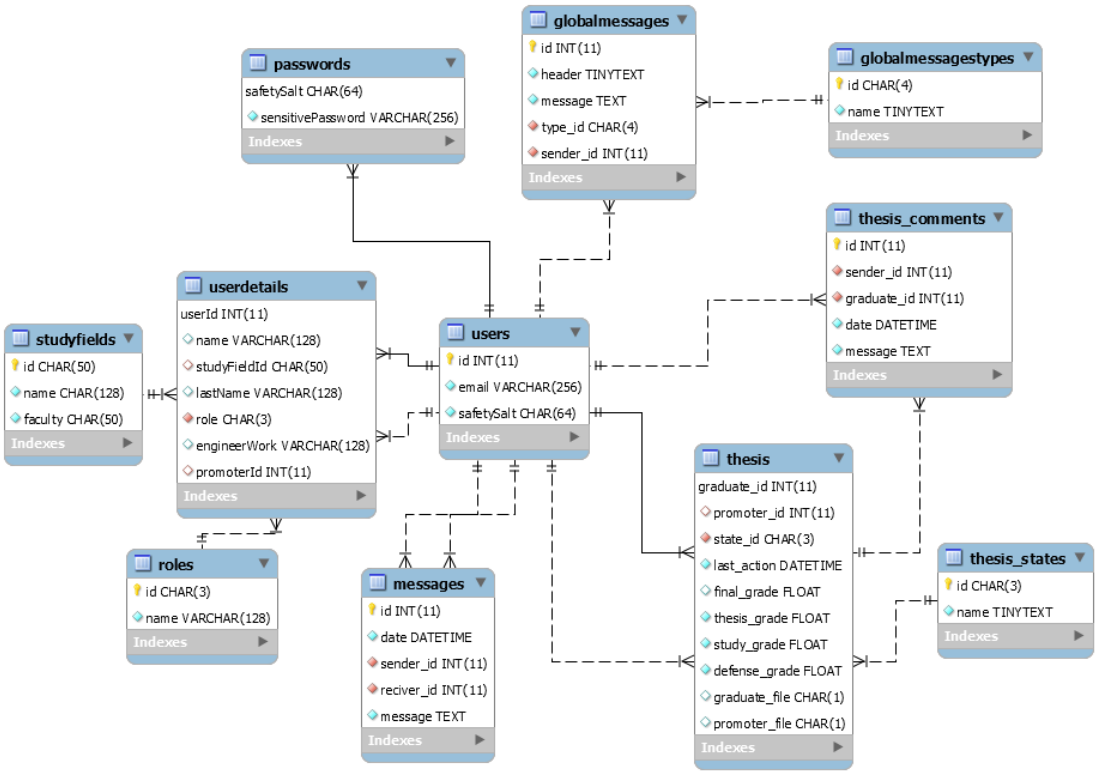
5.2 Opis struktur danych

W aplikacji nie zostały wykorzystane złożone struktury danych. Najczęściej wykorzystywane struktury danych to obiekty w formacie JSON, dynamiczne tablice w postaci list.

5.3 Organizacja bazy danych

Na rysunku 5.1 został przedstawiony schemat bazy danych systemu. Tabele passwords, users, userdetails mogłyby tworzyć jedną całość, lecz zostały odseparowane od siebie ze względu bezpieczeństwa. Tabela passwords przechowuje hasła zaszyfrowane w postaci zahashowanej. Kluczem jest ciąg zaburzający, który jest unikatowy dla każdego użytkownika. Tabela password jest połączona 1 do 1 z tabelą user. Tabela user zawiera adresy e-mail użytkowników oraz wcześniej wspomniany ciąg. Służy on do zabezpieczania hasła przed Rainbow table attack. Tabela user jest połączona relacjami 1 do 1 z tabelami userdetails oraz passwords. Tabela userdetails zawiera informacje na temat użytkownika, takie jak imię, nazwisko, rola czy kierunek studiów. Tabela userdetails jest połączona relacją 1 do N ze słownikiem studyfields oraz roles. Słownik studyfields zawiera informacje na temat kierunków studiów. Natomiast słownik roles zawiera informacje na temat ról dostępnych w systemie. Następna tabela — globalmessages, zawiera wszystkie wysłane przez promotora wiadomości do swoich studentów, a mianowicie ich typ, identyfikator promotora, nagłówek oraz treść. Jest połączona relacją ze słownikiem globalmessagestypes, który dekoduje typy wiadomości. Tabela messages zawiera dane wiadomości wysłanych na czacie. Krotka w tabeli zawiera informacje na temat osoby wysyłającej, odbierającej, datę oraz treść wiadomości. Następnie opisane zostaną tabele związane z pracą dyplomową, zawierające w swojej nazwie słowo kluczowe „thesis". Tabela thesis zawiera informacje na temat promotora, dyplomanta, ocen, stanu pracy czy daty ostatniej akcji wykonanej na pracy przez promotora. Deskryptorem stanów pracy inżynierskiej jest słownik thesis_states, z którym tabela thesis jest w relacji M do N. Ostatnią tabelą, która zostanie opisana jest thesis_comments. Tabela ta zawiera wiadomości wysłane jako komentarze do pracy przez promotora. Zawartość wiersza tabeli zawiera datę, id promotora,

id dyplomanta oraz treść wiadomości.



Rysunek 5.1: Schemat bazy danych

5.4 Komponenty, Moduły i Biblioteki

Do zaimplementowania aplikacji zostały wykorzystane zewnętrzne biblioteki, takie jak: Angular, ASP.NET Core, SignalR, JwtToken, Enitiy Framework, Bootstrap. Poprzez użycie zewnętrznych bibliotek czy platform programistycznych przyspieszona zostaje praca nad projektem. Pozwala się to skupić na najważniejszych aspektach aplikacji. Architektura platformy Angular opiera się o komponenty. Komponenty można używać wielokrotnie dzięki odseparowaniu plików HTML, TypeScript, CSS. Dodatkowo, dzięki istniejącemu w tej platformie strażnikowi można utworzyć klasę opierającą się o wzorzec projektowy wstrzykiwania zależności, która decyduje czy aktualny użytkownik posiada wystarczające upraw-

nienia do danej część aplikacji. Dzięki takiemu podejściu można było oddzielić moduł loginu i rejestracji od właściwej aplikacji. Natomiast po stronie serwera, moduły zostały wydzielone przy użyciu przestrzeni nazw występujących w języku C#. Między innymi klasy, które są zmapowanymi tabelami bazy danych posiadają swoją własną przestrzeń nazw. Dzięki wspomnianemu w punkcie 5.1 narzędziu Entity Framework, nie trzeba pisać zapytań w języku SQL, ponieważ można korzystać z tak zwanego LINQ (ang. *Language INtegrated Query*), dzięki któremu klauzulom języka SQL odpowiadają specjalne metody, czyniąc aplikację czytelniejszą, a także mniej problematyczną w kwestii znajdowania błędów.

5.5 Szczegóły implementacji wybranych fragmentów, zastosowane wzorce projektowe

Podczas projektowania systemu jednym z ważnych elementów było zaprojektowanie podziału na role. Każda z ról w systemie posiada pewne uprawnienia. Na tym etapie rozwoju występują dokładnie trzy role: promotor, dyplomant i gość. Każdy użytkownik posiadający konto może ustawić dane swojego profilu. Jednakże niektóre pola powinny być zarezerwowane dla konkretnej roli. Przykładem może tu być możliwość wybrania swojego promotora, którą powinien mieć jedynie dyplomant. Inna kwestia jest taka, że z punktu widzenia bezpieczeństwa zakłada się, że dane zapytania wysyłanego z przeglądarki mogą być błędne, nawet te z autoryzowanego źródła. Takie zapytanie zawsze można modyfikować. Dlatego po stronie internetowego interfejsu wszystkie zapytania powinny być poddane walidacji. Do sprawdzenia danych profilowych, które przychodzą z zapytania służą specjalne klasy implementujące interfejs `IDetailsValidationHandler`, którego zawartość jest przedstawiona na listingu 5.2. Interfejs ten pełni w tym wypadku rolę strategii. Strategia jest to wzorzec projektowy który definiuje algorytmy wymienne jako osobne byty, czyli w tym wypadku klasy. Każda klasa implementująca ten interfejs odpowiada za walidację danych użytkowników. Jednakże, jak wcześniej zostało wspomniane, każda rola ma inne wymagania, z tego powodu powinna mieć osobny walidator. Do tego celu stworzona została klasa `UserdetailsValidatorFactory`, której implementację przedstawia listing 5.1. Klasa ta zwraca odpowiedni walidator

opierając się o adres e-mail użytkownika. To pozwala na dopasowanie odpowiedniego obiektu do danych z zapytania. W nazwie klasy zostało zamieszczone słowo kluczowe „Factory", które oznacza, że klasa została zaimplementowana w oparciu o wzorzec projektowy faktorii. Jest to wygodne rozwiązanie umożliwiające dodanie walidatorów kolejnych rodzajów e-mail bez konieczności ingerencji w pozostałe.

Listing 5.1: Klasa UserdetailsValidatorFactory

```
1  namespace Backend.Lib.EmailValidators
2  {
3      public class UserdetailsValidatorFactory
4      {
5          private IUserRepository Repository;
6          public UserdetailsValidatorFactory(IUserRepository
               repository)
7          {
8              this.Repository = repository;
9          }
10
11         public IDetailsValidationHandler Get(string email)
12         {
13             string[] dividedEmail = email.Split('@');
14             if (dividedEmail.Length > 2)
15             {
16                 throw new InvalidEmailException(email);
17             }
18
19             switch (dividedEmail[1])
20             {
21                 case "student.polsl.pl":
22                     return new GraduateDetailsValidator(this.
                           Repository, email);
23
24                 case "polsl.pl":
25                     return new PromoterDetailsValidator(email
                           );
26
27                 default:
28                     throw new InvalidEmailException(email);
29             }
30         }
31     }
32 }
```

Listing 5.2: Interfejs IDetailsValidationHandler

```
1       namespace Backend.Lib.EmailValidators
2       {
3           public interface IDetailsValidationHandler
4           {
5               void Validate(Userdetails details);
6           }
7       }
```

Rozdział 6

Weryfikacja i walidacja

W czasie tworzenia aplikacji, ważną sprawą jest jej testowanie, gdyż pozwala to zmniejszyć liczbę błędów w aplikacji. Istnieją różne sposoby testowania, między innymi testowanie poprzez użycie automatycznych testów jednostkowych, testów integracyjnych czy testów funkcjonalnych (end-to-end). Istnieją też mniej profesjonalne rodzaje testów, jak testowanie manualne. Jednak wszystko sprowadza się do sprawdzenia, czy aplikacja działa poprawnie przed i po każdej zmianie. Wspomniane typy testów to tylko kilka z istniejących rodzajów, większą ich liczbę można znaleźć na rysunku 6.1, gdzie uwzględniony jest również podział testów na funkcjonalne i niefunkcjonalne.

6.1 Sposób testowania w ramach pracy

Podczas pracy nad projektem najczęstszą formą testowania było testowanie manualne. Testowanie manualne polega na ręcznym sprawdzaniu aplikacji. Podczas klikania po graficznym interfejsie aplikacji, sprawdzane jest, czy nowa funkcjonalność przypadkiem nie zmieniła działania innej, zaimplementowanej wcześniej. Może zdarzyć się konflikt, który trzeba jak najszybciej naprawić. Poza sytuacją, gdzie nowo zaimplementowana część systemu potencjalnie może wpłynąć negatywnie na istniejącą, występuję również taki przypadek, w którym trzeba poprawić istniejący kod (optymalizacja). W takim wypadku wymagane jest sprawdzenie, czy dana funkcjonalność działa tak samo jak wcześniej. Najlepiej posiadać testy

Functional testing types include:

- Unit testing
- Integration testing
- System testing
- Sanity testing
- Smoke testing
- Interface testing
- Regression testing
- Beta/Acceptance testing

Non-functional testing types include:

- Performance Testing
- Load testing
- Stress testing
- Volume testing
- Security testing
- Compatibility testing
- Install testing
- Recovery testing
- Reliability testing
- Usability testing
- Compliance testing
- Localization testing

Rysunek 6.1: Rodzaje testów — zrzut ekranu ze strony [9]

automatyczne, które po zmianie sprawdzą, czy działanie danej części systemu się nie zmieniło. Jednakże po stronie interfejsu graficznego nie zostały napisane żadne testy automatyczne. Dlatego podczas jego testowania jedyną metodą było testowanie manualne. Natomiast część aplikacji, którą jest internetowy interfejs została zaopatrzona w testy jednostkowe (ang. *Unit Tests*). Tutaj sytuacja wygląda inaczej, ponieważ elementy systemu, które posiadały testy, nie wymagały testowania manualnego.

6.2 Przypadki testowe, zakres testowania

Testowanie aplikacji sprowadza się do sprawdzenia największej możliwej liczby scenariuszy korzystania z systemu lub konkretnej funkcjonalności. Podczas testowania sprawdzano, czy w czasie korzystania z czatu można bezbłędnie wysłać wiadomość w przypadku, gdy wcześniej już jakieś wiadomości zostały wysłane bądź też nie. Sprawdzano również, czy globalne wiadomości zostały wysłane tylko w obrębie dyplomantów promotora poprzez logowanie się na różne konta. Również sprawdzano, czy elementy widoczne dla użytkownika z konkretną rolą są dostępne dla kogoś z inną rolą. Przykładem może być sekcja prac dyplomowych, gdzie promotor powinien widzieć początkowo listę swoich dyplomantów, natomiast dyplomant powinien widzieć jedynie swój profil. Nie może dojść do sytuacji, gdzie w przypadku błędu, dyplomanta może zobaczyć na przykład prace innych dyplomantów swojego promotora. Dlatego priorytetem było testowanie aplikacji pod kątem zgodności z założeniami funkcjonalności systemu. Gdy system podczas testów spełnia założone funkcjonalności, można stwierdzić, że działa poprawnie.

6.3 Wykryte i usunięte błędy

Podczas tworzenia aplikacji może zajść potrzeba ponownej implementacji, bądź poprawienia istniejących klas, interfejsów lub innych elementów systemu. Jest to często powodowane wykryciem błędu w aplikacji. Podczas implementacji systemu kilkukrotnie zdarzyło się odnaleźć błąd poprzez testowanie manualne czy automatyczne testy jednostkowe. Przykładem może być wysyłanie wiadomości globalnych.

Przy użyciu testowania manualnego, udało się rozwiązać problem z wysyłaniem wiadomości do dyplomantów, którzy nie byli przypisani do danego promotora. Wykryte zostało to poprzez zalogowanie się na konto użytkownika, który nie był dyplomantem tego promotora. Użytkownik z tym kontem otrzymał wiadomość od nie swojego promotora, a zgodnie z założeniami nie powinien. Usterkę tę udało się szybko naprawić. Jednakże błędy nie tylko były wykrywane poprzez testy manualne. Po stronie internetowego interfejsu nastąpiła zmiana klasy odpowiadającej za walidację adresów e-mail. Dzięki testom jednostkowym, jakie posiadała ta klasa, udało się natychmiast znaleźć błąd, który powstał przez pomyłkę, co poskutkowało szybką jego naprawą.

Rozdział 7

Podsumowanie i wnioski

Finalnie autorowi udało się stworzyć aplikację, która działa i może zostać użyta. Starano się zaimplementować wszystkie założone wymagania funkcjonalne oraz niefunkcjonalne. Sam temat pracy jest ciekawy i warty rozwoju, ponieważ brakuje dedykowanego systemu, który wspomoże prowadzenie pracy dyplomowej.

Praca posiada możliwość rozszerzania i przydałyby się jej inne funkcjonalności. Przykładem może być historia poszczególnych etapów pracy inżynierskiej podobnych do tych z kontroli wersji. Mogłyby być zapisywane wszystkie przesłane dokumenty PDF, a nie tylko ostatni. Również dobrym pomysłem jest wsparcie dla innych formatów pliku, takich jak LaTeX, DOC. Bardzo przydatną funkcjonalnością byłyby wiadomości elektroniczne (e-mail) zawierające powiadomienia dotyczące nowej zmiany dokonanej przez promotora. Trzeba wziąć również pod uwagę zaprojektowanie wsparcia dla osób, które grupowo wykonują pracę dyplomową. Jeśli chodzi o aspekt utrzymania oprogramowania, to dobrym posunięciem byłoby rozszerzenie liczby testów jednostkowych, integracyjnych a także funkcjonalnych. Dodawanie różnego rodzaju testów ubezpiecza programistę podczas rozszerzania funkcjonalności czy naprawy systemu. Gdy developer, który rozwija, naprawia, poprawia aplikację posiada pełne pokrycie systemu testami, może edytować kod z prawie zerową szansą wprowadzenia do aplikacji jakichś regresji. Jeżeli jakaś funkcjonalność zacznie działać źle podczas pracy nad kodem, to zgodnie z teorią testy powinny to od razu wykazać. Z tego powodu testy muszą być szybkie, żeby można było je często włączać, aby sprawdzać system w trakcie pracy nad nim.

Jeżeli jednak dana część aplikacji nie posiada testów, to przed dodaniem wcześniej wspomnianych funkcjonalności, najpierw należy dodać testy, aby potem móc tę część zmienić, zmniejszając szanse na wprowadzenie jakiejś regresji.

Podczas pracy nad tym projektem napotkanych zostało wiele problemów związanych najczęściej z integracją zewnętrznych bibliotek. Przykładem takiej biblioteki może być SignalR [7], który umożliwia wykorzystanie gniazd internetowych w aplikacji opartej o platformę ASP.NET. Inna biblioteka, która sprawiła problem to Entity Framework [5] odpowiedzialny za komunikacje z bazą oraz mapowanie tabel z bazy danych na obiekty. Z tego powodu trochę czasu zajęła konfiguracja wielu bibliotek użytych w aplikacji, zanim można było przejść do implementacji funkcjonalności projektu.

Podczas pracy nad projektem autorowi udało się poszerzyć horyzonty zdobywając wiedzę na temat projektowania aplikacji internetowych w architekturze REST, platformy programistycznej Angular, środowiska .NET, testów jednostkowych czy wzorców projektowych.

Bibliografia

[1] angular documentation. https://angular.io/. [data dostępu: 2018-11-27].

[2] asp.net documentation. https://docs.microsoft.com/pl-pl/aspnet/core/?view=aspnetcore-2.2. [data dostępu: 2018-12-4].

[3] c# documentation. https://docs.microsoft.com/en-us/dotnet/csharp/index. [data dostępu: 2018-12-29].

[4] content-type reference. https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Type. [data dostępu: 2018-12-29].

[5] Entity framework documentation. https://docs.microsoft.com/en-us/aspnet/core/data/ef-mvc/?view=aspnetcore-2.2. [data dostępu: 2019-01-1].

[6] jwt documentation. https://jwt.io/. [data dostępu: 2018-11-22].

[7] Signalr documentation. https://www.asp.net/signalr. [data dostępu: 2019-01-1].

[8] s.o.l.i.d principles. https://scotch.io/bar-talk/s-o-l-i-d-the-first-five-principles-of-object-oriented-design. [data dostępu: 2019-01-1].

[9] Types of software testing. https://www.softwaretestinghelp.com/types-of-software-testing/. [data dostępu: 2019-01-1].

[10] Kent Beck. *Test-Driven Development.* Pearson Education, Boston, Massachusetts, USA, 2003.

[11] Martin Fowler, Kent Beck, John Brant, William Opdyke, Don Roberts. *Refaktoryzacja*. HELLION, Gliwice, 2011.

[12] Robert C. Martin. *Clean Code*. Pearson Education, Westford, Massachusetts, USA, 2009.

Dodatki

Spis skrótów i symboli

UML zunifikowany język modelowania (ang. *Unified Modeling Language*)

REST styl architektury oprogramowania (ang. *Representational State Transfer*)

API interfejs programistyczny aplikacji (ang. *Application Programming Interface*)

HTTP protokół przesyłania dokumentów hipertekstowych (ang. *Hypertext Transfer Protocol*)

JSON notacja obiektu w języku javascript (ang. *Javascript Object Notation*)

JWT internetowy token w notacji JSON (ang. *JSON web token*)

TDD metodyka programowania polegająca na rozpoczęciu pracy od pisania testów (ang. *Test Driven Development*)

SQL strukturalny język programowania służący do zarządzania bazami danych (ang. *Structured Query Language*)

DLL biblioteka łączona dynamicznie (ang. *Dynamic-Link Library*)

HTML język znaczników (ang. *Hypertext Markup Language*)

LINQ technologia będąca elementem Microsoft .NET, umożliwiająca dawanie zapytań na bazie danych (ang. Language INtegrated Query)

Zawartość dołączonej płyty

Do pracy dołączona jest płyta CD z następującą zawartością:

- pełna wersja pracy inżynierskiej,
- wersja pracy inżynierskiej bez rysunków,
- źródła programu,

Spis rysunków