UNIWERSYTET JANA KOCHANOWSKIEGO

W KIELCACH

WYDZIAŁ MATEMATYCZNO-PRZYRODNICZY

Kierunek: Informatyka

Patryk Gola

Numer albumu 129855

Praca inżynierska

Projekt oprogramowania do zarządzania elektronicznym indeksem

Promotor pracy:

dr inż. Przemysław Ślusarczyk

Praca przyjęta pod względem

merytorycznym i formalnym

w formie papierowej i elektronicznej

…………………………….

/data i podpis promotora/

**KIELCE 2021**

# Spis treści

[Spis treści 2](#_Toc68870781)

[Spis rysunków 4](#_Toc68870782)

[Spis tabel 5](#_Toc68870783)

[Spis wydruków 6](#_Toc68870784)

[Wstęp 7](#_Toc68870785)

[1. Analiza dziedziny i specyfikacja wymagań 8](#_Toc68870786)

[1.1. Sformułowanie zadania projektowego 8](#_Toc68870787)

[1.2. Wymagania projektowe 8](#_Toc68870788)

[1.2.1. Wymagania funkcjonalne. 8](#_Toc68870789)

[1.2.2. Wymagania niefunkcjonalne. 9](#_Toc68870790)

[1.3. Przegląd gotowych rozwiązań 10](#_Toc68870791)

[1.3.1. System do zarządzania wirtualną uczelnią wu.ujk.edu.pl 10](#_Toc68870792)

[1.3.2. System do zarządzania wirtualną uczelnią cas.usos.tu.kielce.pl 11](#_Toc68870793)

[1.3.3. Porównanie dostępnych rozwiązań 12](#_Toc68870794)

[1.4. Analiza wymagań funkcjonalnych 13](#_Toc68870795)

[1.4.1. Diagram przypadków użycia 13](#_Toc68870796)

[1.4.2. Wykaz przypadków użycia dla projektowanego systemu 18](#_Toc68870797)

[2. Projekt oprogramowania 30](#_Toc68870798)

[2.1. Architektura oprogramowania 30](#_Toc68870799)

[2.2. Analiza struktur danych 31](#_Toc68870800)

[2.2.1. Wewnętrzne struktury danych 31](#_Toc68870801)

[2.2.2. Zewnętrzne struktury danych 31](#_Toc68870802)

[2.2.3. Struktura bazy danych 32](#_Toc68870803)

[2.2.4. Schemat powiązań w bazie danych 34](#_Toc68870804)

[2.3. Struktura logiczna oprogramowania 35](#_Toc68870805)

[2.3.1. Diagramy klas 35](#_Toc68870806)

[2.4. Analiza dynamiki oprogramowania 39](#_Toc68870807)

[2.5. Projekt interfejsu użytkownika 44](#_Toc68870808)

[2.5.1. Założenia 44](#_Toc68870809)

[2.5.2. Struktura 44](#_Toc68870810)

[2.5.3. Przykłady elementów interfejsu 45](#_Toc68870811)

[3. Implementacja oprogramowania 48](#_Toc68870812)

[3.1. Charakterystyka wykorzystanych technologii 48](#_Toc68870813)

[3.1.1. Języki programowania 48](#_Toc68870814)

[3.1.2. Narzędzia wspomagające 48](#_Toc68870815)

[3.2. Opis implementacji 50](#_Toc68870816)

[3.2.1. Wykaz plików źródłowych 50](#_Toc68870817)

[3.2.2. Omówienie wybranych fragmentów kodu 52](#_Toc68870818)

[3.2.3. Obsługa błędów i sytuacji wyjątkowych 56](#_Toc68870819)

[3.3. Opis użytkowania 57](#_Toc68870820)

[3.4. Testowanie oprogramowania 61](#_Toc68870821)

[Zakończenie 66](#_Toc68870822)

[Bibliografia 67](#_Toc68870823)

# Spis rysunków

[Rys. 1.1. Strona główna panelu studenta wu.ujk.edu.pl 10](#_Toc68870824)

[Rys. 1.2. Strona główna panelu studenta cas.usos.tu.kielce.pl 11](#_Toc68870825)

[Rys. 1.3. Diagram przypadków użycia dla obszaru OA1 Logowanie 13](#_Toc68870826)

[Rys. 1.4. Diagram przypadków użycia dla obszaru OA2 Zarządzanie studentami 14](#_Toc68870827)

[Rys. 1.5. Diagram przypadków użycia dla obszaru OA3 Zarządzanie zajęciami 15](#_Toc68870828)

[Rys. 1.6. Diagram przypadków użycia dla obszaru OA4 Zarządzanie zadaniami 16](#_Toc68870829)

[Rys. 1.7. Diagram przypadków użycia dla obszaru OA2 Zarządzanie notatkami 17](#_Toc68870830)

[Rys. 2.1. Architektura projektowanego oprogramowania 30](#_Toc68870831)

[Rys. 2.2. Schemat powiązań w bazie danych 34](#_Toc68870832)

[Rys. 2.3. Diagram klas dla funkcjonalności studenta 35](#_Toc68870833)

[Rys. 2.4. Diagram klas dla funkcjonalności zajęć 36](#_Toc68870834)

[Rys. 2.5. Diagram klas dla funkcjonalności zadań 37](#_Toc68870835)

[Rys. 2.6. Diagram klas dla funkcjonalności notatek 38](#_Toc68870836)

[Rys. 2.7. Diagram aktywności dla OA1: Logowanie 39](#_Toc68870837)

[Rys. 2.8. Diagram aktywności dla OA2: Zarządzanie studentem 40](#_Toc68870838)

[Rys. 2.9. Diagram aktywności dla OA3: Zarządzanie zajęciami 41](#_Toc68870839)

[Rys. 2.10. Diagram aktywności dla OA4: Zarządzanie zadaniami 42](#_Toc68870840)

[Rys. 2.11. Diagram aktywności dla OA5: Zarządzanie notatkami 43](#_Toc68870841)

[Rys. 2.12. Diagram ilustrujący strukturę interfejsu graficznego 44](#_Toc68870842)

[Rys. 2.13. Projekt interfejsu graficznego strony głównej 45](#_Toc68870843)

[Rys. 2.14. Projekt interfejsu graficznego wykazu studentów 46](#_Toc68870844)

[Rys. 2.15. Projekt interfejsu graficznego wyszukiwarki zajęć 47](#_Toc68870845)

[Rys. 2.16. Projekt interfejsu graficznego dodawania notatek 47](#_Toc68870846)

[Rys. 3.1. Struktura katalogów głównych w projekcie 50](#_Toc68870847)

[Rys. 3.2. Interfejs graficzny wykazu studentów 58](#_Toc68870848)

[Rys. 3.3. Interfejs graficzny dodawania nowych studentów 59](#_Toc68870849)

[Rys. 3.4. Interfejs graficzny aktualizowania danych o studentach 60](#_Toc68870850)

# Spis tabel

[Tabela 1.1. Przypadek użycia UC-01: Logowanie 18](#_Toc68870851)

[Tabela 1.2. Przypadek użycia UC-02: Weryfikacja danych 18](#_Toc68870852)

[Tabela 1.3. Przypadek użycia UC-03: Zresetuj hasło 19](#_Toc68870853)

[Tabela 1.4. Przypadek użycia UC-04: Wyświetl studentów 19](#_Toc68870854)

[Tabela 1.5. Przypadek użycia UC-05: Znajdź studenta 20](#_Toc68870855)

[Tabela 1.6. Przypadek użycia UC-06: Dodaj studenta 20](#_Toc68870856)

[Tabela 1.7. Przypadek użycia UC-07: Zaktualizuj studenta 21](#_Toc68870857)

[Tabela 1.8. Przypadek użycia UC-08: Usuń studenta 21](#_Toc68870858)

[Tabela 1.9. Przypadek użycia UC-09: Pobierz dane z bazy 22](#_Toc68870859)

[Tabela 1.10. Przypadek użycia UC-10: Dodaj zajęcia 22](#_Toc68870860)

[Tabela 1.11. Przypadek użycia UC-11: Wyświetl zajęcia 23](#_Toc68870861)

[Tabela 1.12. Przypadek użycia UC-12: Wyszukaj zajęcia 23](#_Toc68870862)

[Tabela 1.13. Przypadek użycia UC-13: Zaktualizuj zajęcia 24](#_Toc68870863)

[Tabela 1.14. Przypadek użycia UC-14: Usuń zajęcia 24](#_Toc68870864)

[Tabela 1.15. Przypadek użycia UC-15: Wyświetl zadania 25](#_Toc68870865)

[Tabela 1.16. Przypadek użycia UC-16: Znajdź zadanie 25](#_Toc68870866)

[Tabela 1.17. Przypadek użycia UC-17: Dodaj zadanie 26](#_Toc68870867)

[Tabela 1.18. Przypadek użycia UC-18: Zaktualizuj zadanie 26](#_Toc68870868)

[Tabela 1.19. Przypadek użycia UC-19: Usuń zadanie 27](#_Toc68870869)

[Tabela 1.20. Przypadek użycia UC-20: Wyświetl notatki 27](#_Toc68870870)

[Tabela 1.21. Przypadek użycia UC-21: Znajdź notatkę 28](#_Toc68870871)

[Tabela 1.22. Przypadek użycia UC-22: Dodaj notatkę 28](#_Toc68870872)

[Tabela 1.23. Przypadek użycia UC-23: Zaktualizuj notatkę 29](#_Toc68870873)

[Tabela 1.24. Przypadek użycia UC-24: Usuń notatkę 29](#_Toc68870874)

[Tabela 2.1. Struktura tabeli students\_db 32](#_Toc68870875)

[Tabela 2.2. Struktura tabeli classes\_db 32](#_Toc68870876)

[Tabela 2.3. Struktura tabeli note\_db 33](#_Toc68870877)

[Tabela 2.4. Struktura tabeli task\_db 33](#_Toc68870878)

[Tabela 2.5. Struktura tabeli exercise\_db 33](#_Toc68870879)

[Tabela 3.1. Opis zawartości plików wchodzących w skład projektu oprogramowania 51](#_Toc68870880)

[Tabela 3.2. Przypadki testowe dla obszaru OA1 – Logowanie 61](#_Toc68870881)

[Tabela 3.3. Przypadki testowe dla obszaru OA2 – Zarządzanie studentami 62](#_Toc68870882)

[Tabela 3.4. Przypadki testowe dla obszaru OA3 – Zarządzanie zajęciami 63](#_Toc68870883)

[Tabela 3.5. Przypadki testowe dla obszaru OA4 – Zarządzanie zadaniami 64](#_Toc68870884)

[Tabela 3.6. Przypadki testowe dla obszaru OA5 – Zarządzanie notatkami 65](#_Toc68870885)

# Spis wydruków

[Wydruk 3.1. Wydruk kodu źródłowego SecurityConfig.java 52](#_Toc68870886)

[Wydruk 3.2. Wydruk kodu źródłowego SecurityConfig.java 53](#_Toc68870887)

[Wydruk 3.3. Wydruk kodu źródłowego ExerciseService.java 54](#_Toc68870888)

[Wydruk 3.4. Wydruk kodu źródłowego ExerciseController.java 55](#_Toc68870889)

[Wydruk 3.5. Przykładowy format JSON żądania wysyłanego na serwer 56](#_Toc68870890)

# Wstęp

Celem niniejszej pracy jest opracowanie projektu i implementacja wybranych funkcjonalności aplikacji webowej służącego do zarządzania elektronicznym indeksem. Projekt tego systemu udostępni użytkownikom zarządzanie informacjami gromadzonymi w aplikacji m.in. dodawanie nowych wpisów, odczytywanie danych, modyfikowanie oraz usuwanie. Implementacja systemu zostanie zrealizowana w środowisku Java Enterprise Edition w oparciu o bibliotekę Spring Framework.

Większość studentów w Polsce korzysta z wirtualnych indeksów na wielu platformach od komputerów stacjonarnych, aż po urządzenia mobilne. Aż 46 procent badanych studentów w 2019 roku korzystających z elektronicznych indeksów jest niezadowolona z funkcjonowania wirtualnego dziekanatu na swojej uczelni [14]. Jest to dobra inicjatywa, aby sporządzić jednolity system uzupełniony o brakujące funkcjonalności informujące chociażby o odwołanych zajęciach, żeby uniknąć komunikacji mailowej. Potencjalny odbiorca systemu za jego pośrednictwem będzie miał bezpośredni dostęp do wielu funkcjonalności jak konwersowanie z innymi użytkownikami, weryfikacją swojego planu uczelnianego jak i ocen, tworzeniem notatek oraz wiele innych możliwości z poziomu jednej aplikacji webowej, która w finalnej wersji będzie miała dodatkowo niezależną wersje nowatorskiej aplikacji na systemy takie jak Android, bądź iOS.

Praca składa się z trzech rozdziałów. Pierwszy zawiera opis specyfikacji wymagań stawianych projektowanej aplikacji. Gdzie zamieszczona jest specyfikacja wymagań oraz analiza wymagań funkcjonalnych i niefunkcjonalnych aplikacji webowej. W drugim rozdziale przedstawiono projekt oprogramowania w postaci diagramów UML, struktura baz danych z wyszczególnieniem na relacje zachodzące między nimi, architekturę systemu i interfejs użytkownika. Trzeci rozdział jest poświęcony na szczegółowy opis implementacji kodu źródłowego, z ogólną charakterystyką użytych technologii i narzędzi wspomagających, wykaz wszystkich plików źródłowych wchodzących w skład implementacji oraz omówienie wybranych fragmentów kodu.

# Analiza dziedziny i specyfikacja wymagań

## Sformułowanie zadania projektowego

Technologie używane w tworzeniu innowacyjnych systemów oraz aplikacji rozwijają się w błyskawicznym tempie, co prowadzi do zwiększenia wymogów związanych z już istniejącymi zastosowaniami na rynku oraz dążą one do polepszenie komfortu użytkowania owego oprogramowania jak i dostępności w każdym miejscu.

Dziedziną problemu jest zrealizowanie projektu i implementacji oprogramowania do zarządzania elektronicznym indeksem, które ułatwi użytkownikom na komunikacje, aktualizowanie, przeglądanie treści zawartych oraz aktualizowanie informacji dotyczących poszczególnych zagadnień uczelnianych.

System zostanie zrealizowany w dwóch początkowych oraz końcowych stadiach procesu front-end oraz back-end. Będzie on opierał się na przedstawianiu informacji wprowadzonych do relacyjnej bazy danych, dodawaniu oraz modyfikowaniu zależnie od zagadnienia z poziomu ucznia, wykładowcy, bądź moderatora, a także usuwaniu rejestrów dodanych do bazy. Realizacja tego systemu zostanie wykonana w formie aplikacji webowej.

## Wymagania projektowe

### Wymagania funkcjonalne.

Oprogramowania do zarządzania elektronicznym indeksem będzie realizować następujące funkcjonalności:

* W pełni funkcjonalny system rejestracji/logowania z podziałem na role,
* Dodawanie oraz aktualizowanie zajęć do wybranych kolekcji kierunków,
* Dodawanie nowych użytkowników oraz nadawanie im pewnej roli,
* Tworzenie oraz aktualizowanie listy zadań do zrobienia użytkownika,
* Tworzenie oraz aktualizowanie notatek użytkownika,
* Tworzenie i aktualizowanie zadań oraz projektów dla użytkowników,
* Przeglądanie spisu zajęć użytkowników zależnie od kryteriów,
* Przeglądanie notatek użytkowników po konkretnych parametrach,
* Przeglądanie zadań do zrobienia użytkowników według normy,
* Przeglądanie zajęć danego kierunku i wykładowcy,
* Wyszukiwanie informacji po danych wytycznych,
* Aktualizowanie ocen użytkowników.

### Wymagania niefunkcjonalne.

Projekt powinien być zrealizowany z normami zawartymi w ustawie z dnia 10 maja 2018r. o ochronie danych osobowych, czyli z dbałością o bezpieczeństwo przechowywanych informacji w relacyjnych bazach danych .

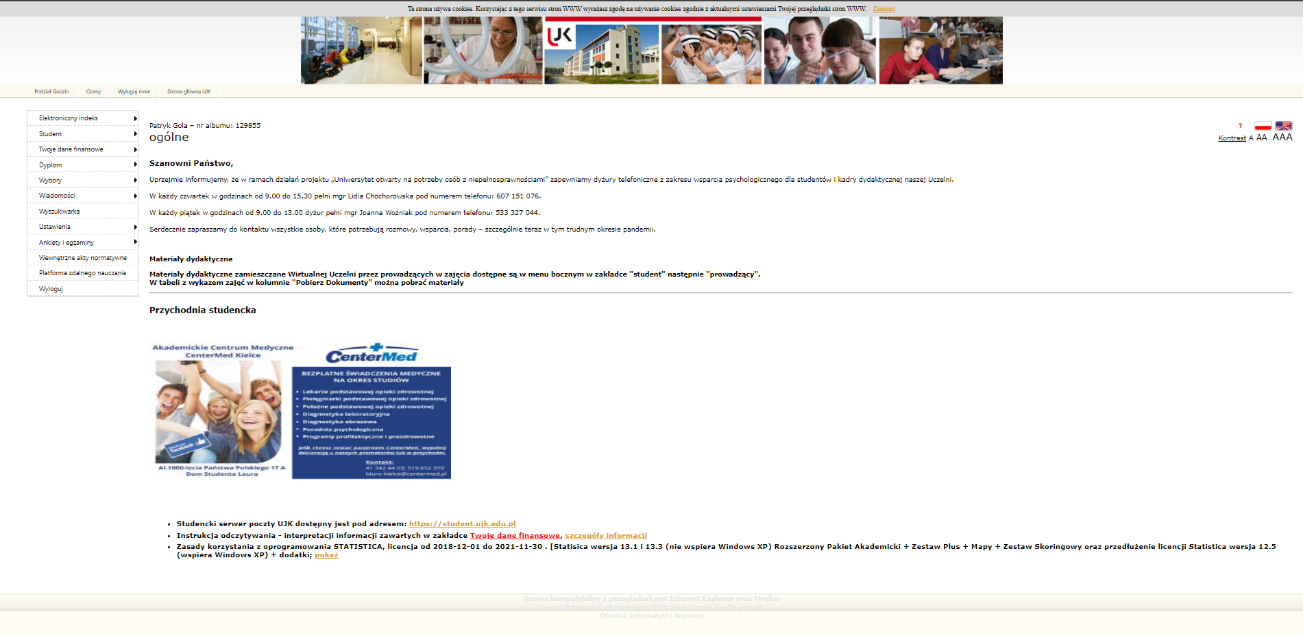
Realizowany system będzie posiadał następujące wymagania niefunkcjonalne:

* Oprogramowanie musi działać responsywnie,
* Oprogramowanie musi być bezpieczne,
* Oprogramowanie będzie wymagało ciągłego połączenia do sieci internetowej,
* Oprogramowanie będzie zrealizowane w dwóch procesach początkowych oraz końcowych,
* W celu uruchomienia serwera oprogramowania będzie wymagane zainstalowanie JVM, czyli wirtualnej maszyny Java w wersji powyżej Java 8 i narzędzie automatyzujące budowę oprogramowania Maven.
* Oprogramowanie będzie działać na wszystkich systemach operacyjnych,
* Do uruchomienia aplikacji wymagany będzie skompilowane projekt jako plik JAR,
* Oprogramowanie będzie bazowało na ogólnodostępnych bibliotekach,
* Oprogramowanie będzie miało połączenie z zewnętrzną relacyjną bazą danych,
* Oprogramowanie będzie zrealizowane od samych podstaw zgodnie z konwencją tworzenia oprogramowania aplikacji webowych.

## Przegląd gotowych rozwiązań

### System do zarządzania wirtualną uczelnią wu.ujk.edu.pl

Wirtualna uczelnia to serwis internetowy, skupiający w sobie informacje dotyczące różnych użytkowników (studentów, wykładowców) z dokładnym opisem aktywności uniwersyteckich. Dla każdego studenta sporządzony jest oddzielny panel do weryfikowania swoich danych z dostępem do elektronicznego indeksu, ocen, planu zajęć, wiadomości, ankiet oraz wielu innych funkcjonalności. Podczas nauki zdalnej jak i stacjonarnej jest to narzędzie niezbędne do realizacji całego programu nauczania, bez którego w ostatnich latach byłoby to niemożliwe [21].

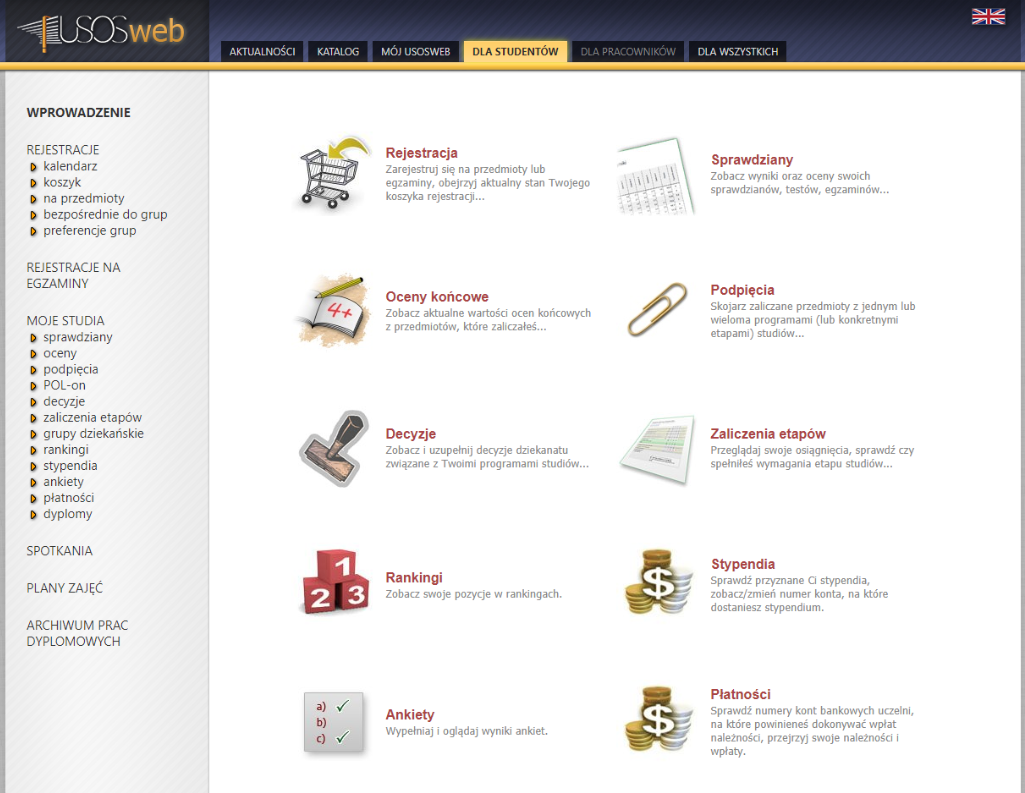


Rys. .. Strona główna panelu studenta wu.ujk.edu.pl

Platforma przedstawiona na zrzucie ekranu jest to startowa strona panelu Wirtualnej Uczelni Uniwersytetu Jana Kochanowskiego, gdzie w górnym interfejsie umieszczone są odnośniki do podziału godzin, wglądu do ocen studenta, przycisk pełniący funkcję wylogowania się oraz powrót do strony głównej UJK. Boczny panel menu zawiera uogólnione zakładki, które po najechaniu kursorem myszki rozwijają się dając do kolejnych odnośników związanych z perypetiami akademickimi. Centrum strony wyświetla informacje z aktualnego położenia użytkownika w panelu wirtualnej uczelni. W dolnej części witryny widoczna jest stópka z wiadomością dotyczącą kompatybilności strony z poszczególnymi przeglądarkami internetowymi. Platforma Wirtualnej Uczelni jest niezbyt dobrze czytelna, co wynika z nieaktualnej konwencji oraz braku intuicyjnego interfejsu użytkownika z nowoczesnym zastosowaniem wyglądu. Dużym niedociągnięciem tej witryny jest błędne dostosowanie pod urządzenia mobilne, na których wymagana jest zmiana trybu przeglądarki na poziom komputerowy, co daje dostęp do pełnej funkcjonalności systemu.

### System do zarządzania wirtualną uczelnią cas.usos.tu.kielce.pl

USOS web jest to serwis internetowy skupiający się na zarządzaniu wirtualną uczelnią pod wymogi akademickie dostarczający użytkownikom niezbędne informacje dotyczące sprawdzianów, ocen końcowych, decyzji na temat złożonych dokumentów do dziekanatu, ankiet. Serwis działa w bardzo podobny sposób, jak system Uniwersytetu Jana Kochanowskiego w Kielcach [2].



Rys. .. Strona główna panelu studenta cas.usos.tu.kielce.pl

Na rysunku 1.2. przedstawiono stronę główną panelu studenta dla zalogowanego użytkownika. W górnej części znajduje się menu główne z takimi opcjami jak „Aktualności”, „Dla studentów”. Boczny panel znajdujący się po lewej stronie platformy pełni funkcję paska nawigacyjnego umożliwiającego studentowi przemieszczanie się po znacznej ilości korzyści płynącej z USOS web. Interfejs jest prosty w obsłudze oraz intuicyjny, ale nie wszystkie z widocznych możliwości użycia funkcjonalności działają sprawnie np. plan zajęć znajdujący się w menu po lewej stronie nie działa, lecz wymaga oddzielnego arkuszu do wglądu w zajęcia.

### Porównanie dostępnych rozwiązań

Opisane gotowe rozwiązania w poprzednich dwóch podrozdziałach są stworzone w analogiczny sposób, z tą samą koncepcją mającą na głównym celu dostarczanie użytkownikowi niezbędnych informacji, które zastępują wersję papierową indeksu studenta. Obydwie witryny posiadają wiele funkcjonalności systemu, z których odbiorcy nie korzystają, bądź nie maja możliwości z nich korzystać. Jednym z takich przykładów w przypadku platformy cas.usos.tu.kielce.pl jest korespondowanie studenta z osobą prowadzącą zajęcia. Mogą tylko odczytywać wiadomości od wykładowców, lecz wszelkie odpowiedzi na komunikat trzeba wykonywać za pośrednictwem zewnętrznych serwisów pocztowych. System wu.ujk.edu.pl posiada normalną możliwość na korespondowanie przez platformę, w celu skontaktowania się tylko z dziekanatami różnych wydziałów lub działem księgowości.

Wersja mobilna na urządzeniach korzystających z platformy iOS oraz Android w przypadku platformy Uniwersytetu Jana Kochanowskiego wymaga zmiany sposobu przeglądania na tryb komputerowy, aby mieć możliwość na pełną interakcje z przyciskami pełniącymi jakąś funkcjonalność. Logując się w normalnym trybie do platformy nie ma możliwości wykorzystać jakiejkolwiek dogodności systemu. Cały design strony został przystosowany pod użytkowanie go z poziomu normalnych przeglądarek internetowych na komputerach i laptopach. Platforma Politechniki Świętokrzyskiej została dostosowana w lepszy sposób do odbioru jej z poziomu urządzeń mobilnych.

USOS web posiada bardziej przejrzystą szatę graficzna niż wirtualna uczelnia, ale ciągle odbiegają one z nowymi standardami stron, które tworzone są przez profesjonalne grupy programistów zajmujących się serwisami webowymi. Oba systemu mają wiele funkcjonalności „widmo”, jakie zostały dodane tylko, w celu polepszenia wizualnego odbioru jednej oraz drugiej platformy. Na witrynie wu.ujk.edu.pl jedną z takich korzyści dla użytkownika, która jest jako część informacyjna, bądź wizualna nie pełniąc swojej funkcji to zakładka „sylabus”, gdzie funkcjonalność dopiero możliwa jest po kontakcie z informatykiem wydziałowym przez korespondencje mailową, a odnośnik do programu studiów jest dostępny tylko po podłączeniu do sieci UJK. W przypadku strony internetowej cas.usos.tu.kielce.pl takim udogodnieniem jest zakładka „sprawdziany”, która nie jest wykorzystywana przez wykładowców, co obliguje studentów do uczestniczenia w kolokwiach i egzaminach za pośrednictwem oddzielnej witryny.

## Analiza wymagań funkcjonalnych

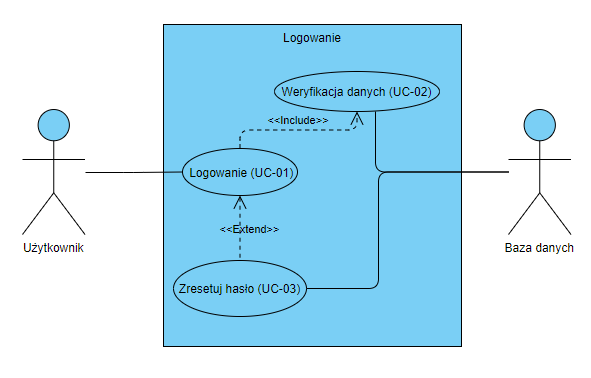
W tym paragrafie omówione zostały diagramy przypadków użycia (*ang. Use Case Diagram*) i przewidziane scenariusze użytkowania. Zostały one opracowane w celu określenia kontekstu systemu, uchwycenia wymagań systemowych, do zarządzania cyklem tworzenia implementacji i generowania przypadków testowania. [19]

### Diagram przypadków użycia

Funkcjonalność oprogramowania do zarządzania elektronicznym indeksem została podzielona na następujące obszary aktywności:

* OA1 – Logowanie
* OA2 – Zarządzanie danymi studentów
* OA3 – Zarządzanie zajęciami
* OA4 – Zarządzanie zadaniami
* OA5 – Zarządzanie notatkami

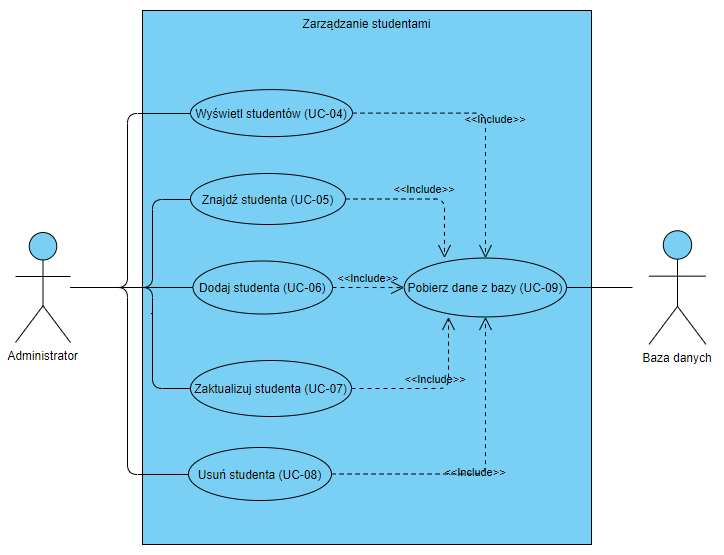
Przypadki użycia systemu w obszarze aktywności OA1: Logowanie zostały przedstawione na diagramie (rys. 1.3).

**

Rys. .. Diagram przypadków użycia dla obszaru OA1 Logowanie

Obszar aktywności „Logowanie” przedstawiony na powyższym rysunku, aktor Użytkownik, którym może być zarówno student jak i wykładowca. Posiada on możliwość zalogowania się do systemu, z którego aktor może wybrać opcję resetowania hasła. Przypadek użycia „Weryfikacja danych” i „Zresetuj hasło” odnosi się do aktora „Baza danych”, który odpowiada za zewnętrzne przechowywanie informacji.

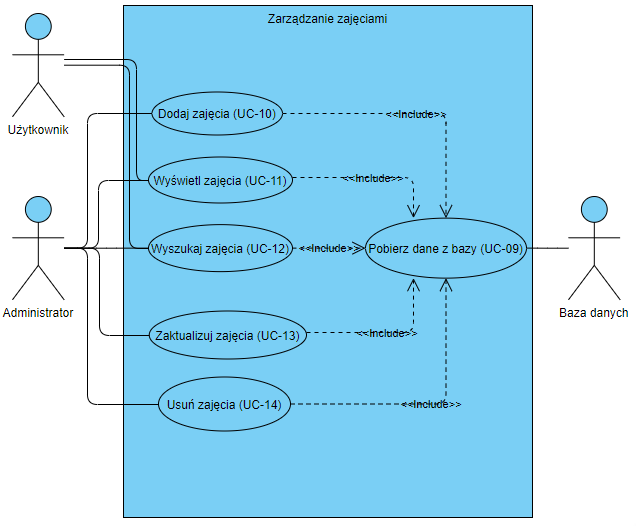
OA2: Obsługa studentów została ukazana na diagramie przypadków użycia „zarządzanie studentami gdzie umieszczone są przypadki użycia: UC-06, UC-07, UC-08, UC-09.



Rys. .. Diagram przypadków użycia dla obszaru OA2 Zarządzanie studentami

Rysunek 1.4 przedstawia obszar aktywności „zarządzanie studentami”. Odpowiada on za rządzenie studentami danej uczelni przez aktora „Administrator”, którym jest wykładowca z odpowiednimi uprawnieniami. W systemie znajduje się przypadek użycia „wyświetl studentów”, „znajdź studenta” zależnie od parametru wyszukiwania, „dodaj studenta” co prezentuję rejestrowanie nowych użytkowników w oprogramowaniu, „zaktualizuj studenta”, gdzie błędnie wpisane dane mogą zostać skorygowane oraz „usuń studenta”, w przypadku zrezygnowania przez osobę z toku nauczania lub wynikające   
z zaniedbywania obowiązków studenckich. Wszystkie wymienione przypadki użycia   
są zawarte w bazie danych, gdzie wszelkie informacje są aktualizowane i trafiają do aktora „baza danych”, prezentującego zewnętrzną bazę danych MySQL.

OA3: Obsługa zajęć została zilustrowana na diagramie przypadków użycia „zarządzanie zajęciami”, gdzie umieszczone są przypadki użycia: UC-09, UC-10,   
UC-11, UC-12, UC-13, UC-14.

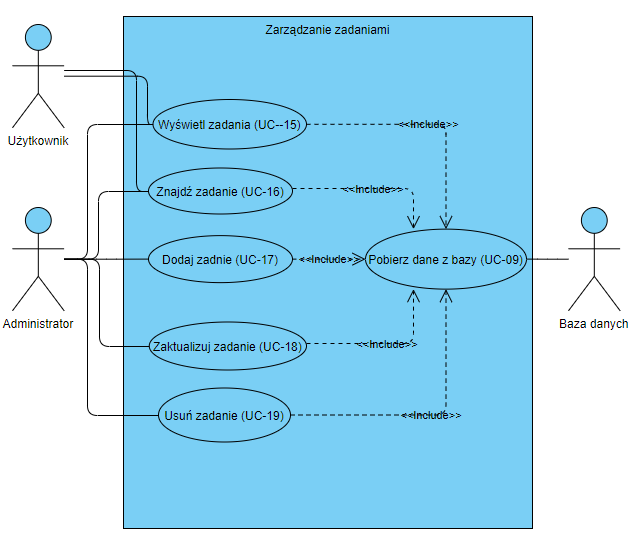


Rys. .. Diagram przypadków użycia dla obszaru OA3 Zarządzanie zajęciami

Diagram przypadków użycia na rysunku 1.5 przedstawia obszar aktywności zarządzania zajęciami, polega na administrowaniu zajęciami dydaktycznymi na uczelni, gdzie aktor „administrator” posiada pełne uprawnienia do zarządzania przypadkami użycia takimi jak „dodaj zajęcia”, „wyświetl zajęcia”, „wyszukaj zajęcia” według danego parametru, „zaktualizuj zajęcia”, aby wprowadzić korektę lub zmianę w zajęciach   
oraz „usuń zajęcia”, jeżeli zostały one wprowadzone omylnie do bazy danych lub wyszły z toku nauczania.

Aktor „użytkownik”, w tym przypadku to student, który posiada wgląd do zajęć i wyszukiwanie poszczególnych zajęć zależnie od wprowadzonego parametru.  
Aktor „baza danych” składuje informacje dotyczące zajęć w zewnętrznej bazie danych, gdzie są one dodawane, pobierane, aktualizowane, usuwane.

OA4: Obsługa zadań jest przedstawiona na diagramie przypadków użycia „zarządzanie zadaniami”, gdzie umieszczone są przypadki użycia: UC-09, UC-15,   
UC-16, UC-17, UC-18, UC-19.

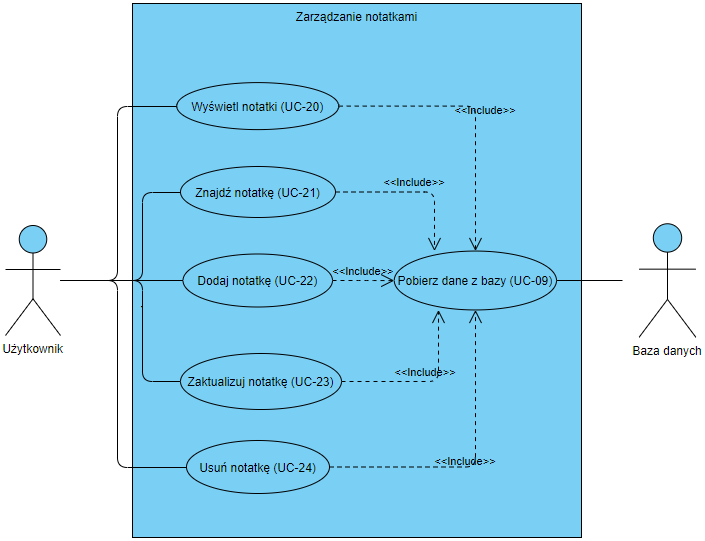


Rys. .. Diagram przypadków użycia dla obszaru OA4 Zarządzanie zadaniami

Obszar aktywności „zarządzanie zadaniami” przedstawiony na powyższym rysunku 1.6 pełni funkcję komunikacyjną między wykładowcą, a studentami, gdzie umieszczane są zadania do zrealizowania przez studentów w ramach uzyskania zaliczenia lub oceny.

Aktor „użytkownik”, którym jest student ma dostęp tylko do przypadku użycia „wyświetl zadania” oraz „znajdź zadanie”, czyli posiada pełny wgląd do danych zawartych w tabeli bazy danych dla zarządzania zadaniami. Natomiast aktor „administrator” w roli wykładowcy posiada ten sam dostęp, co „użytkownik”, ale posiada poszerzone uprawnienia o przypadek użycia „dodaj zadanie”, „zaktualizuj zadanie”,  
w celu korekty informacji i „usuń zadanie”, w celu pozbywania się nieaktualnych, przedawnionych zadań. Aktor „baza danych” odpowiada ze przechowywanie wszystkich informacji w zewnętrznej bazie danych.

OA5: Zarządzanie notatkami jest przedstawione na diagramie przypadków użycia „zarządzanie notatkami”, gdzie umieszczone są przypadki użycia: UC-09, UC-20,   
UC-21, UC-22, UC-23, UC-24.



Rys. .. Diagram przypadków użycia dla obszaru OA2 Zarządzanie notatkami

Rysunek 1.7 przedstawia diagram przypadków użycia dla obszaru aktywności „zarządzanie notatkami”, gdzie użytkownik ma pełną możliwość do administrowania prywatnych wpisów, obejmujących szereg informacji związanych z obowiązkami uczelnianymi.

Aktor „użytkownik”, może być zarówno studentem, jak i wykładowcą z dostępem do wszystkich przypadków użycia w tym obszarze aktywności takich jak „wyświetl notatki” pobierającą pełną listę wpisów, „znajdź notatkę” zależnie od własności notatki, „dodaj notatkę”, „zaktualizuj notatkę”, w przypadku wykonania danej notatki, korekty informacji, „usuń notatkę”. Aktor „baza danych” przechowuję wszystkie informacje zawarte w obszarze aktywności zarządzanie notatkami w zewnętrznej bazie danych.

### Wykaz przypadków użycia dla projektowanego systemu

|  |  |
| --- | --- |
| ***Identyfikator*** | UC-01 |
| ***Nazwa*** | Logowanie |
| ***Cel*** | Dostęp do uprawnień dla danej roli |
| ***Aktor główny*** | Użytkownik |
| ***Źródło*** | Patryk Gola |
| ***Warunki początkowe*** | Użytkownik nie jest zalogowany |
| ***Główny scenariusz (podstawowy przepływ)***   1. Użytkownik ma możliwość wpisania nazwy użytkownika i hasła 2. W przypadku zapomnienia hasła użytkownik może wybrać opcje zresetuj hasło 3. Po wprowadzeniu danych system weryfikuje zgodność loginu i hasła. 4. Jeżeli nazwa użytkownika i hasło są poprawne, następuje przekierowanie do panelu użytkownika zależnie od jego roli. | |
| ***Rozszerzenia (przepływy alternatywne)***   1. Jeżeli nazwa użytkownika i hasło są niepoprawne, wyskakuje informacja z błędem. | |

Tabela .. Przypadek użycia UC-01: Logowanie

|  |  |
| --- | --- |
| ***Identyfikator*** | UC-02 |
| ***Nazwa*** | Weryfikacja danych |
| ***Cel*** | Sprawdzenie spójności wpisywanych danych z rejestrem użytkowników |
| ***Aktor główny*** | Baza danych |
| ***Źródło*** | Patryk Gola |
| ***Warunki początkowe*** | Użytkownik loguję się do systemu |
| ***Główny scenariusz (podstawowy przepływ)***   1. System sprawdza zbieżność danych wpisywanych przez użytkownika z znajdującymi się w bazie danych systemu. 2. Jeżeli wpisana nazwa użytkownika i hasło są takie same jak w bazie danych, to system przekierowuje aktora dalej do panelu głównego. | |
| ***Rozszerzenia (przepływy alternatywne)***   1. Proces logowania kończy się niepowodzeniem 2. Wyskakuje informacja o błędzie logowania się | |

Tabela .. Przypadek użycia UC-02: Weryfikacja danych

|  |  |
| --- | --- |
| ***Identyfikator*** | UC-03 |
| ***Nazwa*** | Zresetuj hasło |
| ***Cel*** | Uzyskanie nowego hasła do konta |
| ***Aktor główny*** | Użytkownik |
| ***Źródło*** | Patryk Gola |
| ***Warunki początkowe*** | Użytkownik znajduję się w panelu logowania |
| ***Główny scenariusz (podstawowy przepływ)***   1. Użytkownik wpisuje w pole swój numer indeksu jako nazwę użytkownika 2. Na e-mail podany podczas rekrutacji, zatrudnienia się przychodzi wiadomość z wygenerowanym nowym hasłem tymczasowym 3. Pojawia się wiadomość o pomyślnym przebiegu resetowania hasła | |
| ***Rozszerzenia (przepływy alternatywne)***   1. Jeżeli zresetowanie starego hasła, w celu uzyskania nowego zakończyło się niepowodzeniem, to wyskoczy informacja z błędem. | |

Tabela .. Przypadek użycia UC-03: Zresetuj hasło

|  |  |
| --- | --- |
| ***Identyfikator*** | UC-04 |
| ***Nazwa*** | Wyświetl studentów |
| ***Cel*** | Uzyskanie listy studentów |
| ***Aktor główny*** | Administrator |
| ***Źródło*** | Patryk Gola |
| ***Warunki początkowe*** | Administrator znajduję się w panelu zarządzania studentami |
| ***Główny scenariusz (podstawowy przepływ)***   1. Administrator wybiera opcję wyświetlenia studentów. 2. System pobiera z bazy danych odpowiednią tabele dla studentów 3. System dostarcza spis studentów zawarty w bazie danych | |
| ***Rozszerzenia (przepływy alternatywne)***   1. Jeżeli system nie dostarczy listy studentów, wyskoczy informacja z błędem | |

Tabela .. Przypadek użycia UC-04: Wyświetl studentów

|  |  |
| --- | --- |
| ***Identyfikator*** | UC-05 |
| ***Nazwa*** | Znajdź studenta |
| ***Cel*** | Wyszukanie konkretnego studenta zależnie od parametru |
| ***Aktor główny*** | Administrator |
| ***Źródło*** | Patryk Gola |
| ***Warunki początkowe*** | Administrator znajduje się w panelu zarządzania studentami |
| ***Główny scenariusz (podstawowy przepływ)***   1. Administrator wybiera opcje znajdź student i podaje parametr, po którym chce wyfiltrować studenta, bądź grupę studentów 2. System pobiera z bazy danych listę studentów i filtruje ich, po podanym parametrze 3. System dostarcza studenta lub listę studentów przefiltrowanych z konkretnymi parametrami | |
| ***Rozszerzenia (przepływy alternatywne)***   1. Jeżeli system nie dostarczy listy studentów, wyskoczy informacja z błędem | |

Tabela .. Przypadek użycia UC-05: Znajdź studenta

|  |  |
| --- | --- |
| ***Identyfikator*** | UC-06 |
| ***Nazwa*** | Dodaj studenta |
| ***Cel*** | Dodanie nowego użytkownika do tabeli studentów |
| ***Aktor główny*** | Administrator |
| ***Źródło*** | Patryk Gola |
| ***Warunki początkowe*** | Administrator znajduje się w panelu zarządzania studentami |
| ***Główny scenariusz (podstawowy przepływ)***   1. Administrator wybiera opcje dodawania studentów i uzupełnia dane dotyczącego danej osoby 2. System zapisuje dane do bazy danych tabeli studentów z podanymi parametrami przez administratora 3. Wyskakuje informacja o pomyślnym dodaniu użytkownika do spisu studentów | |
| ***Rozszerzenia (przepływy alternatywne)***   1. Jeżeli system nie zapisze nowego użytkownika do listy studentów, wyskoczy powiadomienie z kodem błędu | |

Tabela .. Przypadek użycia UC-06: Dodaj studenta

|  |  |
| --- | --- |
| ***Identyfikator*** | UC-07 |
| ***Nazwa*** | Zaktualizuj studenta |
| ***Cel*** | Odświeżenie, bądź skorygowanie informacji o studencie |
| ***Aktor główny*** | Administrator |
| ***Źródło*** | Patryk Gola |
| ***Warunki początkowe*** | Administrator znajduje się w panelu zarządzania studentami |
| ***Główny scenariusz (podstawowy przepływ)***   1. Administrator wpisuje odświeżone, bądź skorygowane dane o konkretnym studencie i akceptuje zmiany 2. System nadpisuje informacje znajdujące się w bazie danych dla odpowiedniego identyfikatora 3. Wyświetla się informacja o pomyślnym nadpisaniu użytkownika | |
| ***Rozszerzenia (przepływy alternatywne)***   1. Jeżeli system nie nadpisze studenta, wyskoczy informacja z błędem | |

Tabela .. Przypadek użycia UC-07: Zaktualizuj studenta

|  |  |
| --- | --- |
| ***Identyfikator*** | UC-08 |
| ***Nazwa*** | Usuń studenta |
| ***Cel*** | Usunięcie użytkownika z listy studentów |
| ***Aktor główny*** | Administrator |
| ***Źródło*** | Patryk Gola |
| ***Warunki początkowe*** | Administrator znajduje się w panelu zarządzania studentami |
| ***Główny scenariusz (podstawowy przepływ)***   1. Administrator wybiera studenta do usunięcia po indywidualnym numerze tożsamości 2. System znajduję w bazie danych numer id danego studenta i usuwa go 3. Wyświetla się informacja o pomyślnym usunięciu studenta | |
| ***Rozszerzenia (przepływy alternatywne)***   1. Jeżeli system nie usunie studenta, wyskoczy informacja z błędem | |

Tabela .. Przypadek użycia UC-08: Usuń studenta

|  |  |
| --- | --- |
| ***Identyfikator*** | UC-09 |
| ***Nazwa*** | Pobierz dane z bazy |
| ***Cel*** | Integracja bazy danych z web service |
| ***Aktor główny*** | Baza danych |
| ***Źródło*** | Patryk Gola |
| ***Warunki początkowe*** | Użycie metod http: operacji CRUD |
| ***Główny scenariusz (podstawowy przepływ)***   1. Aplikacja wysyła żądanie http do bazy danych przez zewnętrzny interfejs lub aplikacje Postman [13] w celu pobrania informacji 2. Program wczytuje dane z relacyjnej bazy danych MySQL | |
| ***Rozszerzenia (przepływy alternatywne)***   1. Pojawienie się informacji błędnym żądaniem lub innym kodem błędu | |

Tabela .. Przypadek użycia UC-09: Pobierz dane z bazy

|  |  |
| --- | --- |
| ***Identyfikator*** | UC-10 |
| ***Nazwa*** | Dodaj zajęcia |
| ***Cel*** | Uzupełnienie listy zajęć o nowy wpis |
| ***Aktor główny*** | Administrator |
| ***Źródło*** | Patryk Gola |
| ***Warunki początkowe*** | Administrator znajduje się w panelu zarządzania zajęciami |
| ***Główny scenariusz (podstawowy przepływ)***   1. Administrator wybiera opcje dodawania zajęć i uzupełnia dane dotyczącego zajęć 2. System zapisuje dane do bazy danych tabeli zajęć z podanymi parametrami przez administratora 3. Wyskakuje informacja o pomyślnym dodaniu rekordu do spisu zajęć | |
| ***Rozszerzenia (przepływy alternatywne)***   1. Jeżeli system nie zapisze nowego użytkownika do listy zajęć, wyskoczy powiadomienie z kodem błędu | |

Tabela .. Przypadek użycia UC-10: Dodaj zajęcia

|  |  |
| --- | --- |
| ***Identyfikator*** | UC-11 |
| ***Nazwa*** | Wyświetl zajęcia |
| ***Cel*** | Wgląd do spisu zajęć |
| ***Aktor główny*** | Użytkownik |
| ***Źródło*** | Patryk Gola |
| ***Warunki początkowe*** | Użytkownik znajduję się w panelu zarządzania zajęciami |
| ***Główny scenariusz (podstawowy przepływ)***   1. Użytkownik wybiera opcję wyświetlenia zajęć. 2. System pobiera z bazy danych odpowiednią tabele zajęć 3. System dostarcza spis zajęć zawarty w bazie danych | |
| ***Rozszerzenia (przepływy alternatywne)***   1. Jeżeli system nie dostarczy listy studentów, wyskoczy informacja  z błędem | |

Tabela .. Przypadek użycia UC-11: Wyświetl zajęcia

|  |  |
| --- | --- |
| ***Identyfikator*** | UC-12 |
| ***Nazwa*** | Wyszukaj zajęcia |
| ***Cel*** | Odnalezienie konkretnych zajęć zależnie od parametrów |
| ***Aktor główny*** | Użytkownik |
| ***Źródło*** | Patryk Gola |
| ***Warunki początkowe*** | Użytkownik znajduję się w panelu zarządzania zajęciami |
| ***Główny scenariusz (podstawowy przepływ)***   1. Użytkownik wybiera opcje wyszukaj zajęcia i podaje parametr, po którym chce wyfiltrować zajęcia 2. System pobiera z bazy danych listę zajęć i filtruje ich, po podanym parametrze 3. System dostarcza zajęcia lub listę zajęć przefiltrowanych z konkretnymi parametrami | |
| ***Rozszerzenia (przepływy alternatywne)***   1. Jeżeli system nie dostarczy listy zajęć, wyskoczy informacja z błędem | |

Tabela .. Przypadek użycia UC-12: Wyszukaj zajęcia

|  |  |
| --- | --- |
| ***Identyfikator*** | UC-13 |
| ***Nazwa*** | Zaktualizuj zajęcia |
| ***Cel*** | Odświeżenie, bądź skorygowanie danych o zajęciach |
| ***Aktor główny*** | Administrator |
| ***Źródło*** | Patryk Gola |
| ***Warunki początkowe*** | Administrator znajduje się w panelu zarządzania zajęciami |
| ***Główny scenariusz (podstawowy przepływ)***   1. Administrator wpisuje odświeżone, bądź skorygowane dane o konkretnym studencie i akceptuje zmiany 2. System nadpisuje informacje znajdujące się w bazie danych dla odpowiedniego identyfikatora 3. Wyświetla się informacja o pomyślnym nadpisaniu użytkownika | |
| ***Rozszerzenia (przepływy alternatywne)***   1. Jeżeli system nie nadpisze studenta, wyskoczy informacja z błędem | |

Tabela .. Przypadek użycia UC-13: Zaktualizuj zajęcia

|  |  |
| --- | --- |
| ***Identyfikator*** | UC-14 |
| ***Nazwa*** | Usuń zajęcia |
| ***Cel*** | Usunięcie rekordu z listy zajęć |
| ***Aktor główny*** | Administrator |
| ***Źródło*** | Patryk Gola |
| ***Warunki początkowe*** | Administrator znajduje się w panelu zarządzania zajęciami |
| ***Główny scenariusz (podstawowy przepływ)***   1. Administrator wybiera zajęcia do usunięcia po indywidualnym numerze tożsamości 2. System znajduję w bazie danych numer id danych zajęć i usuwa go 3. Wyświetla się informacja o pomyślnym usunięciu zajęć | |
| ***Rozszerzenia (przepływy alternatywne)***   1. Jeżeli system nie usunie zajęć, wyskoczy informacja z błędem | |

Tabela .. Przypadek użycia UC-14: Usuń zajęcia

|  |  |
| --- | --- |
| ***Identyfikator*** | UC-15 |
| ***Nazwa*** | Wyświetl zadania |
| ***Cel*** | Wgląd do listy zadań |
| ***Aktor główny*** | Użytkownik |
| ***Źródło*** | Patryk Gola |
| ***Warunki początkowe*** | Użytkownik znajduję się w panelu zarządzania zadaniami |
| ***Główny scenariusz (podstawowy przepływ)***   1. Użytkownik wybiera opcję wyświetlenia zadań. 2. System pobiera z bazy danych odpowiednią tabele zadań 3. System dostarcza spis zadań zawarty w bazie danych | |
| ***Rozszerzenia (przepływy alternatywne)***   1. Jeżeli system nie dostarczy listy zadań, wyskoczy informacja z błędem | |

Tabela .. Przypadek użycia UC-15: Wyświetl zadania

|  |  |
| --- | --- |
| ***Identyfikator*** | UC-16 |
| ***Nazwa*** | Znajdź zadanie |
| ***Cel*** | Odnalezienie konkretnego zadania według jego parametrów |
| ***Aktor główny*** | Użytkownik |
| ***Źródło*** | Patryk Gola |
| ***Warunki początkowe*** | Użytkownik znajduję się w panelu zarządzania zadaniami |
| ***Główny scenariusz (podstawowy przepływ)***   1. Użytkownik wybiera opcje znajdź zadanie i podaje parametr, po którym chce wyfiltrować zadanie 2. System pobiera z bazy danych listę zadań i filtruje ich, po podanym parametrze 3. System dostarcza zadania lub listę zadań przefiltrowanych z konkretnymi parametrami | |
| ***Rozszerzenia (przepływy alternatywne)***   1. Jeżeli system nie dostarczy listy zadań, wyskoczy informacja z błędem | |

Tabela .. Przypadek użycia UC-16: Znajdź zadanie

|  |  |
| --- | --- |
| ***Identyfikator*** | UC-17 |
| ***Nazwa*** | Dodaj zadanie |
| ***Cel*** | Uzupełnienie listy zadań o nowy rekord |
| ***Aktor główny*** | Administrator |
| ***Źródło*** | Patryk Gola |
| ***Warunki początkowe*** | Użytkownik znajduję się w panelu zarządzania zadaniami |
| ***Główny scenariusz (podstawowy przepływ)***   1. Administrator wybiera opcje dodaj zadanie i uzupełnia dane dotyczącego zadania 2. System zapisuje dane do bazy danych zadań z podanymi parametrami przez administratora. 3. Wyskakuje informacja o pomyślnym dodaniu zadania do spisu zadań | |
| ***Rozszerzenia (przepływy alternatywne)***   1. Jeżeli system nie zapisze nowego zadania do listy zadań, wyskoczy powiadomienie z kodem błędu | |

Tabela .. Przypadek użycia UC-17: Dodaj zadanie

|  |  |
| --- | --- |
| ***Identyfikator*** | UC-18 |
| ***Nazwa*** | Zaktualizuj zadanie |
| ***Cel*** | Odświeżenie, bądź skorygowanie informacji o zadaniu |
| ***Aktor główny*** | Administrator |
| ***Źródło*** | Patryk Gola |
| ***Warunki początkowe*** | Użytkownik znajduję się w panelu zarządzania zadaniami |
| ***Główny scenariusz (podstawowy przepływ)***   1. Administrator wpisuje odświeżone, bądź skorygowane dane o konkretnym zadaniu i akceptuje zmiany 2. System nadpisuje informacje znajdujące się w bazie danych dla odpowiedniego identyfikatora 3. Wyświetla się informacja o pomyślnym nadpisaniu zadania | |
| ***Rozszerzenia (przepływy alternatywne)***   1. Jeżeli system nie nadpisze zadania, wyskoczy informacja z błędem | |

Tabela .. Przypadek użycia UC-18: Zaktualizuj zadanie

|  |  |
| --- | --- |
| ***Identyfikator*** | UC-19 |
| ***Nazwa*** | Usuń zadanie |
| ***Cel*** | Usunięcie rekordu z listy zadań |
| ***Aktor główny*** | Administrator |
| ***Źródło*** | Patryk Gola |
| ***Warunki początkowe*** | Użytkownik znajduję się w panelu zarządzania zadaniami |
| ***Główny scenariusz (podstawowy przepływ)***   1. Administrator wybiera zadania do usunięcia po indywidualnym numerze tożsamości 2. System znajduję w bazie danych numer id danych zadania i usuwa je 3. Wyświetla się informacja o pomyślnym usunięciu zadania | |
| ***Rozszerzenia (przepływy alternatywne)***   1. Jeżeli system nie usunie zadania, wyskoczy informacja z błędem | |

Tabela .. Przypadek użycia UC-19: Usuń zadanie

|  |  |
| --- | --- |
| ***Identyfikator*** | UC-20 |
| ***Nazwa*** | Wyświetl notatki |
| ***Cel*** | Wgląd do listy notatek |
| ***Aktor główny*** | Użytkownik |
| ***Źródło*** | Patryk Gola |
| ***Warunki początkowe*** | Użytkownik znajduję się w panelu zarządzania notatkami |
| ***Główny scenariusz (podstawowy przepływ)***   1. Użytkownik wybiera opcję wyświetlenia notatek. 2. System pobiera z bazy danych odpowiednią tabele notatek 3. System dostarcza spis notatek zawarty w bazie danych | |
| ***Rozszerzenia (przepływy alternatywne)***   1. Jeżeli system nie dostarczy listy notatek, wyskoczy informacja z błędem | |

Tabela .. Przypadek użycia UC-20: Wyświetl notatki

|  |  |
| --- | --- |
| ***Identyfikator*** | UC-21 |
| ***Nazwa*** | Znajdź notatkę |
| ***Cel*** | Odnalezienie konkretnej notatki według jego parametrów |
| ***Aktor główny*** | Użytkownik |
| ***Źródło*** | Patryk Gola |
| ***Warunki początkowe*** | Użytkownik znajduję się w panelu zarządzania notatkami |
| ***Główny scenariusz (podstawowy przepływ)***   1. Użytkownik wybiera opcje znajdź notatkę i podaje parametr, po którym chce wyfiltrować notatkę 2. System pobiera z bazy danych listę notatek i filtruje ich, po podanym parametrze 3. System dostarcza zadania lub listę notatek przefiltrowanych z konkretnymi parametrami | |
| ***Rozszerzenia (przepływy alternatywne)***   1. Jeżeli system nie dostarczy listy notatek, wyskoczy informacja z błędem | |

Tabela .. Przypadek użycia UC-21: Znajdź notatkę

|  |  |
| --- | --- |
| ***Identyfikator*** | UC-22 |
| ***Nazwa*** | Dodaj notatkę |
| ***Cel*** | Uzupełnienie listy notatek o nowy rekord |
| ***Aktor główny*** | Użytkownik |
| ***Źródło*** | Patryk Gola |
| ***Warunki początkowe*** | Użytkownik znajduję się w panelu zarządzania notatkami |
| ***Główny scenariusz (podstawowy przepływ)***   1. Użytkownik wybiera opcje dodaj notatkę i uzupełnia dane dotyczącego notatkę 2. System zapisuje dane do bazy danych notatkę z podanymi parametrami przez użytkownika 3. Wyskakuje informacja o pomyślnym dodaniu rekordu do spisu notatek | |
| ***Rozszerzenia (przepływy alternatywne)***   1. Jeżeli system nie zapisze nowej notatki do listy notatek, wyskoczy powiadomienie z kodem błędu | |

Tabela .. Przypadek użycia UC-22: Dodaj notatkę

|  |  |
| --- | --- |
| ***Identyfikator*** | UC-23 |
| ***Nazwa*** | Zaktualizuj notatkę |
| ***Cel*** | Odświeżenie, bądź skorygowanie informacji o notatce |
| ***Aktor główny*** | Użytkownik |
| ***Źródło*** | Patryk Gola |
| ***Warunki początkowe*** | Użytkownik znajduję się w panelu zarządzania notatkami |
| ***Główny scenariusz (podstawowy przepływ)***   1. Użytkownik wpisuje odświeżone, bądź skorygowane dane o konkretnej notatce i akceptuje zmiany 2. System nadpisuje informacje znajdujące się w bazie danych dla odpowiedniego identyfikatora 3. Wyświetla się informacja o pomyślnym nadpisaniu notatki | |
| ***Rozszerzenia (przepływy alternatywne)***   1. Jeżeli system nie nadpisze notatki, wyskoczy informacja z błędem | |

Tabela .. Przypadek użycia UC-23: Zaktualizuj notatkę

|  |  |
| --- | --- |
| ***Identyfikator*** | UC-24 |
| ***Nazwa*** | Usuń notatkę |
| ***Cel*** | Usunięcie rekordu z listy notatek |
| ***Aktor główny*** | Użytkownik |
| ***Źródło*** | Patryk Gola |
| ***Warunki początkowe*** | Użytkownik znajduję się w panelu zarządzania notatkami |
| ***Główny scenariusz (podstawowy przepływ)***   1. Użytkownik wybiera notatkę do usunięcia po indywidualnym numerze tożsamości 2. System znajduję w bazie danych numer id danych notatki i usuwa ją 3. Wyświetla się informacja o pomyślnym usunięciu notatki | |
| ***Rozszerzenia (przepływy alternatywne)***   1. Jeżeli system nie usunie notatki, wyskoczy informacja z błędem | |

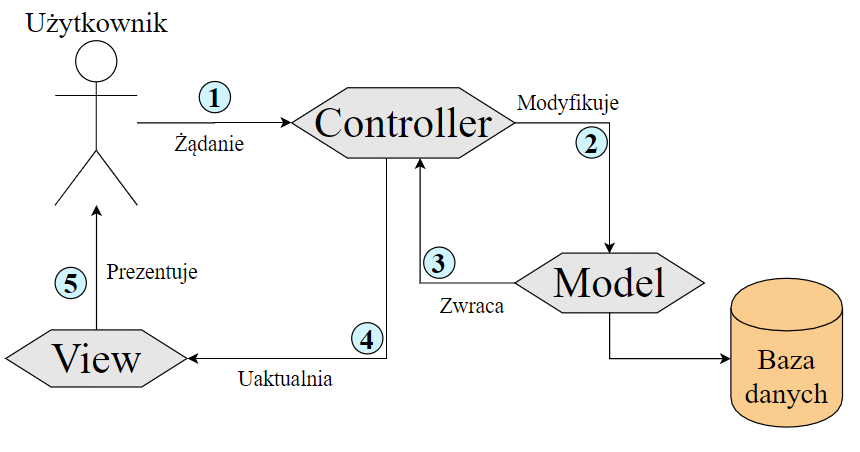
Tabela .. Przypadek użycia UC-24: Usuń notatkę

# Projekt oprogramowania

## Architektura oprogramowania

Projektowane oprogramowanie będzie serwisem webowym bazującym na popularnym trójwarstwowym wzorcu projektowym MVC (ang. Model-View-Controller), przeważnie używanym do tworzenia interfejsu graficznego użytkownika z wygodną możliwością modyfikacji i rozwoju w dalszym stadium implementacji. Jest to standard dla konstrukcji wielu skomplikowanych oprogramowani, z modułami podzielonymi na trzy dominujące części:

* Model – Ze strony serwera jest to widok całego modelu, który jest zlokalizowany na serwerze. Odpowiada za zarządzanie danymi, logiką oraz procedurami aplikacji.
* View – odpowiada za wyświetlanie elementów na stronie w postaci tabel, wykresów, diagramów, które kontroluje widok serwera za pośrednictwem żądań HTTP, którego główną część stanowi strona HTML.
* Controller – zarządza widokiem i modelem, odpowiada na żądanie i przeprowadza interakcje z obiektem modelu danych. W wyniku sprawdzenia dokładności danych wejściowych, przekazuje je do modelu.



Rys. .. Architektura projektowanego oprogramowania

Współpraca trójwarstwowego wzorca projektowego MVC została stworzona   
z myślą o ułatwieniu implementacji i pełnej integracji struktury relacyjnych   
i nierelacyjnych baz danych, obsługę danych wprowadzanych do serwera oraz wyjściowych, co wiąże się z interakcją użytkownika.

Serwis webowy będzie w pełni komunikował się za pośrednictwem architektury REST, czyli reprezentacyjnego transferu stanów przez co można nazywać serwis jako RESTful Web Service. Jest to lekki i intuicyjny serwis oparty na architekturze REST, który udostępnia interfejs API z aplikacja w pełni bezpieczny sposób. Podstawowym protokołem REST jest protokół http.

Po wysłaniu żądania na serwer za pośrednictwem kontrolera REST, w odpowiedzi otrzymujemy dane z API, które zapisane są w formacie JSON (JavaScript Object Notation). Jest on często stosowany ostatnimi czasy, przez co wypycha starszy format XML (Exstensible Markup Language) z użytkowania.

## Analiza struktur danych

### Wewnętrzne struktury danych

Oprogramowanie korzysta z frameworku Spring Boot, który dostarcza odpowiednie zależności odpowiadające za stronę serwera bez większej ingerencji programisty. Wykorzystuje obiekty Bean (pol. ziarna), zarządzające całą kondycją frameworku. Ziarna pozwalają programiście na wskazywanie konkretnego miejsca do  strzyknięcia, przez odpowiednią adnotacje w kodzie źródłowym. Jest to jeden z wzorców projektowych Dependency Injection (wstrzykiwanie zależności), który jest używany w celu wstrzyknięcia danej zależności do konkretnego kontenera, zarządzającego ziarnami. Pozwala to w trakcie implementacji na wygodę testowania różnych opcji zastosowania konkretnych zależności z ułatwieniem wyboru priorytetowego konteneru w strukturze frameworku Spring.

### Zewnętrzne struktury danych

Dobór odpowiednich relacyjnych, bądź nierelacyjnych baz danych przebiega za pośrednictwem domyślnie tworzonego pliku z właściwościami *application.properties*. Zawarta jest w nim wszelka konfiguracja aplikacji Springowej pozwalająca na łatwe łączenie się z zewnętrznymi bazami danych takimi jak MySQL [17]. Podczas testowania bez podłączania się do baz danych można stworzyć lokalne pole testowe danych *H2,* które dostarczane jest przez frameworku Hibernate [5]. Rozszerzenie *.properties*, może być zastąpione .*yml* , lecz obydwa dają przyzwolenie programiście na zarządzanie profilami, czyli nie jesteśmy zobligowani do posiadania jednego pliku z takim rozszerzeniem. Możemy testować wszelkie opcje chociażby pod produkcje lub lokalny hosting.

### Struktura bazy danych

W projekcie oprogramowania przepływ danych, będzie opierał się na popularnym, systemie zarządzania relacyjnymi bazami danych MySQL. Dane będą segregowane za  pośrednictwem tabel, poza nimi niezbędne będą interakcje w aplikacji z plikiem .properties i nadanie właściwego odniesienia do źródła baz danych.

Opisywane struktury relacyjnych baz danych są przedstawione poniżej:

Tabela students\_db przechowuje dane o konkretnym studencie.

|  |  |  |
| --- | --- | --- |
| students\_db | | |
| Kolumna | **Typ** | **Opis** |
| Id | Int(11) | Identyfikator studenta |
| Name | Varchar(255) | Imię użytkownika |
| Last\_name | Varchar(255) | Nazwisko użytkownika |
| Field\_of\_study | Varchar(255) | Kierunek studiów studenta |
| Semestr | Int(11) | Semestr studenta |
| Phone | Int(11) | Numer telefonu studenta |

Tabela .. Struktura tabeli students\_db

Tabela classes\_db przechowuje dane o zajęciach dydaktycznych.

|  |  |  |
| --- | --- | --- |
| classes\_db | | |
| Kolumna | **Typ** | **Opis** |
| Id | Int(11) | Identyfikator zajęć |
| Subject | Varchar(255) | Nazwa zajęć |
| Lecturer | Varchar(255) | Prowadzący zajęcia |
| Form | Varchar(255) | Forma zajęć |
| Pass | Tinyint(1) | Zaliczenie z zajęć |

Tabela .. Struktura tabeli classes\_db

Tabela note\_dbprzechowuje dane o notatkach.

|  |  |  |
| --- | --- | --- |
| note\_db | | |
| Kolumna | **Typ** | **Opis** |
| Id | Int(11) | Identyfikator notatek |
| Title | Varchar(255) | Tytuł notatki |
| description | Varchar(255) | Treść notatki |

Tabela .. Struktura tabeli note\_db

Tabela task\_db przechowuje dane o zadaniach do zrobienia.

|  |  |  |
| --- | --- | --- |
| task\_db | | |
| Kolumna | **Typ** | **Opis** |
| Id | Int(11) | Identyfikator zadania do zrobienia |
| Description | Varchar(255) | Treść zadania do zrobienia |
| Done | Tinyint(1) | Status wykonania zadania |

Tabela .. Struktura tabeli task\_db

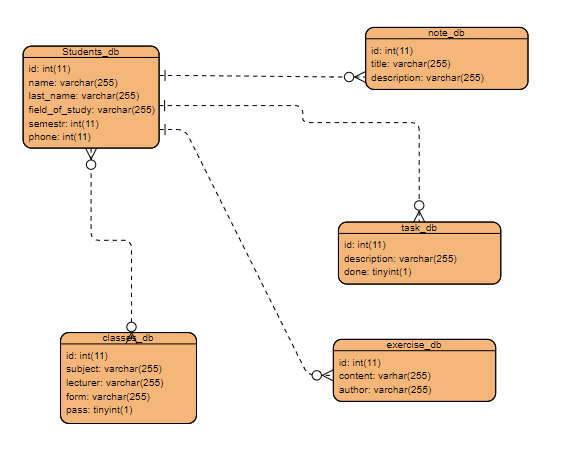
Tabela exercise\_db przechowuje dane o ćwiczeniach.

|  |  |  |
| --- | --- | --- |
| exercise\_db | | |
| Kolumna | **Typ** | **Opis** |
| Id | Int(11) | Identyfikator ćwiczenia |
| Content | Varchar(255) | Treść ćwiczenia |
| Author | Varchar(255) | Autor ćwiczenia |

Tabela .. Struktura tabeli exercise\_db

### Schemat powiązań w bazie danych

Rysunek 2.2. ilustruje strukturę bazy danych w systemie oraz wzajemne relacje między nimi w formie diagramu ERD:



Rys. .. Schemat powiązań w bazie danych

## Struktura logiczna oprogramowania

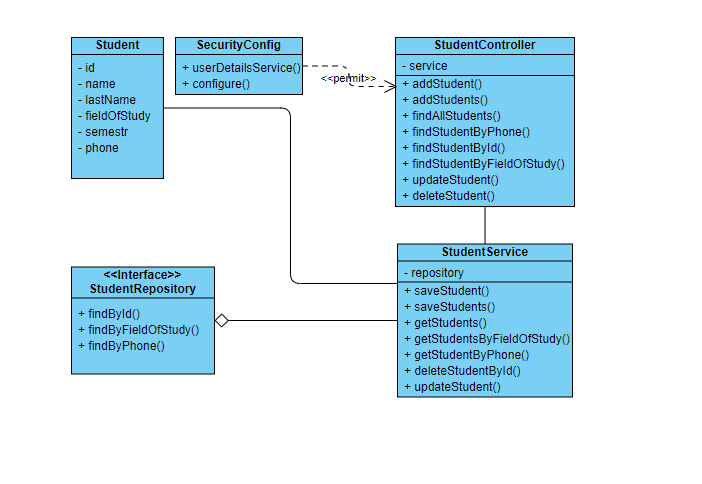
Oprogramowanie zrealizowano w dwóch początkowych i końcowych stadiach procesu front-end oraz back-end, gdzie jedna z nich odpowiada za wyświetlanie danych w warstwie widoku, a druga odpowiada za wszystkie funkcjonalności w systemie.

Część wykonana w etapie back-endu przy użyciu frameworku Spring realizuje funkcjonalności za pośrednictwem komunikacji klas korzystając z adnotacji, które nadają pewien szyk poszczególnym plikom, korzystając z ścisłej konwencji, aby kod był łatwy do odczytu dla innych. Znajdują się w nim wyszczególnione pakiety z podzieleniem na jednostkę modelu, repozytorium odpowiadające dla danego modelu, kontroler oraz serwis przystosowany pod cala architekturę. REST pozwala na wygodniejsze komunikację w Springu informując aplikacje o jego obecności przez adnotacje *@RestController* w pakiecie kontrolerów dla każdej klasy znajdującej się w tym folderze. Poszerza on możliwości systemu o protokół HTTP.

Za ogólną konfigurację serwera odpowiada plik *applications.properties*, gdzie umieszczone są między innymi dane dotyczące łączenia z bazą danych, konfiguracja portu dla hosta, nazwa użytkownika i hasło wymagane do połączenia z bazą danych.

### Diagramy klas

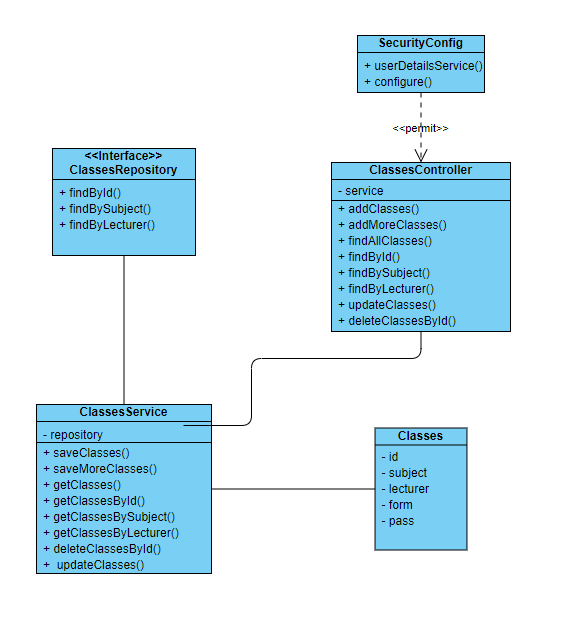
Diagram z rysunku 2.3. ilustruje relacje między klasami pełniącymi funkcjonalność pełnego tworzenia, odczytu, aktualizowania i usuwania studenta czyli proces CRUD (ang. Create-Read-Update-Delete). Przedstawiają pełen schemat przepływu informacji w implementacji od jednostki (ang. Entity), repozytorium rozszerzającym *JPARepository* (Java-Persistence-Api-Repository), przez serwis (ang. Service), aż do kontrolera RESTowego (ang. Rest Controller), gdzie wykorzystywany są poprzedni klasy interfejs.



Rys. .. Diagram klas dla funkcjonalności studenta

Diagram klas dla funkcjonalności zajęć przedstawiony na poniższym rysunku 2.4 pokazuję relację zachodzącą między klasą Classes.java, która przechowuję pełen szereg zmiennych dotyczących zajęć wraz z metodami pobierania i ustawiania wartości, a innymi klasami takimi jak ClassesRepository.java, interfejsem zawierającym metody używane do wyszukiwania poszczególnych zajęć.

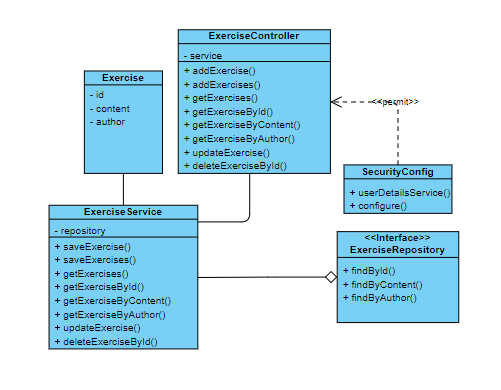
Do tej relacji należy jeszcze klasa ClassesController.java, pełniącej funkcję kontrolera RESTowego (ang. Rest Controller), gdzie znajduję się mapowanie do poszczególnych adresów webowych url (ang. Uniform Resource Locator) zawierający uprawnienia do konkretnych stron na podstawie klasy SecurityConfig.java,  
a to wszystko połączone jest w pełną funkcjonalność w klasie ClassesService.java.



Rys. .. Diagram klas dla funkcjonalności zajęć

Przedstawiony poniżej diagram klas dla funkcjonalności zadań, przedstawia zależność między klasami zachodzi proces CRUD (ang. Create-Read-Update-Delete), który spełnia funkcjonalność realizacji zadań w systemie. Klasa ExerciseService.java, pobiera niezbędne zależności od innych klas, które połączone w całość pozwalają na płynne funkcjonowanie kontrolera RESTowego (ang. Rest Controller), co robi z naszego serwisu webowego, aplikacje RESTful.

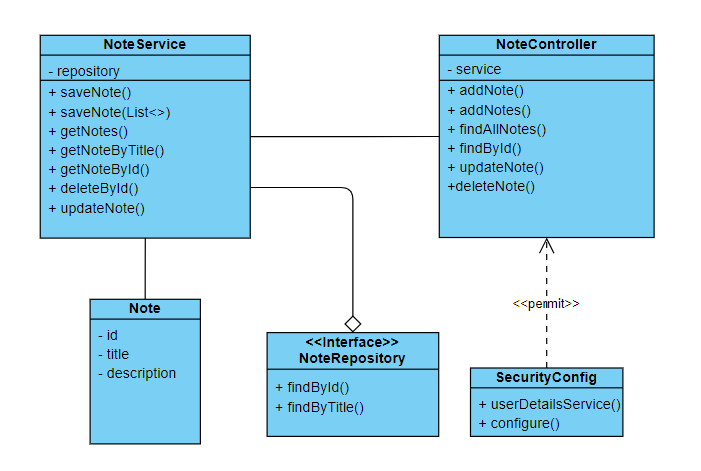
Przepływ implementacji zaczyna się od jednostki (ang. Entity) klasy Exercise.java, gdzie zawarte są podstawowe zmienne z szeregiem metod GET, SET, z których korzysta interfejs ExerciseRepository.java posiadający metody do wyszukiwania zadań zależnie od parametru oraz klasa ExerciseController.java, która jako kontroler RESTowy posiada adnotacje mapowania dla skonfigurowanego adresu URL, dla metod zapisywania, pobierania, aktualizowania i usuwania zadań.

****

Rys. .. Diagram klas dla funkcjonalności zadań

Diagram klas dla funkcjonalności notatek przebiega w analogiczny sposób do poprzednich diagramów studenta, zajęć oraz zadań. Przedstawia on zależność klas, które pełnią funkcjonalność notatek w systemie. Zawierają one dodawanie, usuwanie, aktualizowanie oraz usuwanie wpisów zawartych w bazie danych.

Klasa NoteService.java jest ostatnia w procesie implementacji, oznacza   
to że pobiera zmienne i metody z klas, rozszerza interfejs NoteRepository.java, który korzysta z repozytorium JPARepository (ang. Java Persistence-Api-Repository), przy czym używa kontrolera RESTowego posiadającego metody niezbędne do obsługi mapowania adresu witryny internetowej dla poszczególnych funkcjonalności CRUD.



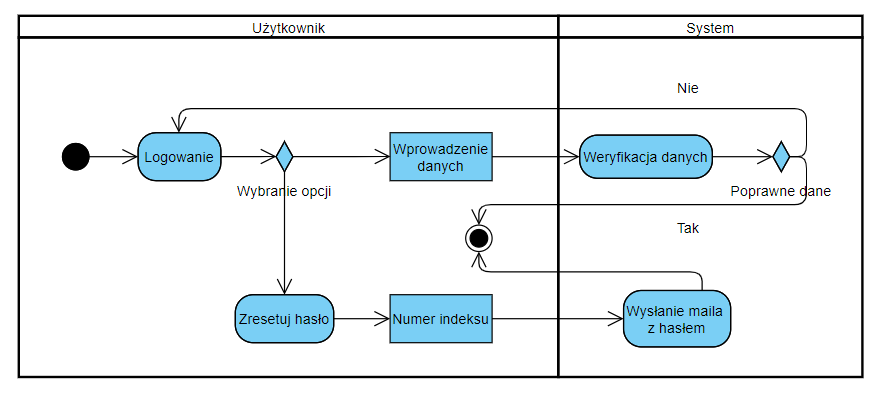
Rys. .. Diagram klas dla funkcjonalności notatek

Wszystkie obiegi zależności klas sprowadzają się do wspólnej zewnętrznej bazy danych MySQL, gdzie wyszczególnione są odpowiednie tabele do przechowywania informacji oraz ich wymiany między użytkownikiem, a systemem.

## Analiza dynamiki oprogramowania

Rozdział 2.4 poświęcony jest zilustrowaniu zakresu odpowiedzialności, przedstawiających każdą czynność wykonaną w konkretnym fragmencie oprogramowania.

Diagram z rysunku 2.7. obrazuje aktywności odwołuje się do OA1: Logowanie (Rys. 1.3.).

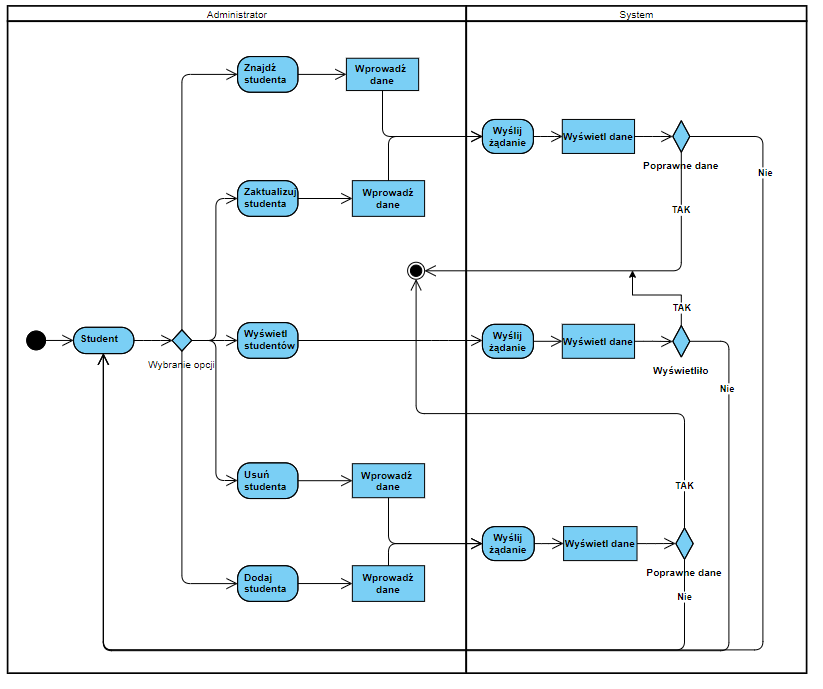


Rys. .. Diagram aktywności dla OA1: Logowanie

Na powyższym rysunku przedstawiony jest diagram aktywności (ang. Activity diagram) dla obszaru aktywności logowanie, na którym znajduje się wymodelowany dynamiczny przebieg krokowy w systemie.

Funkcją tego rysunku jest pokazanie etapów, przez które system przechodzi, po wyborze danej funkcji podczas interakcji użytkownika. W tym przypadku użytkownik następnie, po wybraniu opcji logowania się będzie miał opcje do wprowadzenia swoich danych niezbędnych do wejścia w system lub zresetowania hasła, w przypadku zapomnienia go. Kolejnym etapem jest weryfikacja przez system wprowadzonych danych lub wysłanie maila z hasłem zwracając odpowiedź do węzła końcowego.

Diagram aktywności z rysunku 2.8. odwołuje się do OA2: Zarządzanie studentami (Rys. 1.4.).

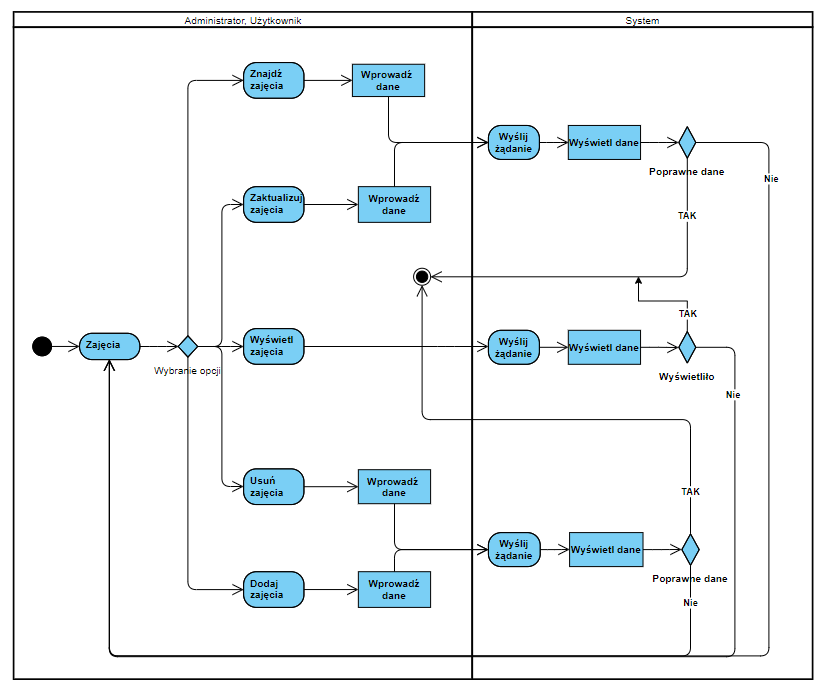


Rys. .. Diagram aktywności dla OA2: Zarządzanie studentem

Rysunek 2.8 przedstawia diagram czynności dla obszaru aktywności „zarządzanie studentem”, gdzie znajduję się sekwencja kroków, przez które przechodzi administrator oraz system.

Administrator, w roli wykładowcy wybiera opcje student, po czym posiada możliwość wybrania następnej czynności od znalezienia poszczególnego studenta,  
aż po dodanie lub usunięcie go. Później wpisując dane zależnie od wyboru, dalsze kroki zależą od systemu, w którym wysyłane są żądania i sprawdzenie wyświetlanych danych wracając do węzła końcowego albo początkowego panelu studenta.

Diagram aktywności z rysunku 2.9. odwołuje się do OA3: Zarządzanie zajęciami   
(Rys. 1.5.).

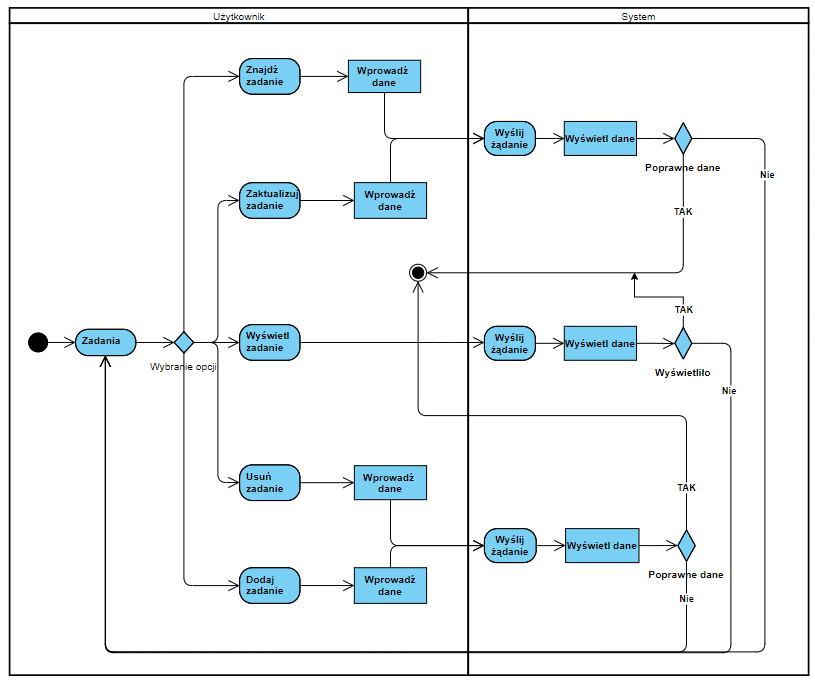


Rys. .. Diagram aktywności dla OA3: Zarządzanie zajęciami

Powyższy rysunek 2.9 będący diagramem czynności dla obszaru aktywności zarządzanie zajęciami pokazuje przebieg krokowy wykonywania danych funkcjonalności w systemie przez aktora administrator, użytkownik oraz system. Diagram aktywności UML (ang. Unified Modeling Language) pokazuje szczegółowy przepływ czynności   
dla przypadków użycia w obszarze aktywności „Zarządzanie zajęciami”.

Ukazane są etapy od startu aktora administrator i użytkownika przez panel zajęć   
z wyborem konkretnych funkcjonalności, po weryfikacje wprowadzonych danych przez system wraz z informacja zwrotną podczas pomyślnego sprawdzenia do węzła końcowego, w przeciwnym przypadku powrotu do panelu wyboru dogodności systemu.

Diagram aktywności z rysunku 2.10. odwołuje się do OA4: Zarządzanie zadaniami (Rys. 1.6.).

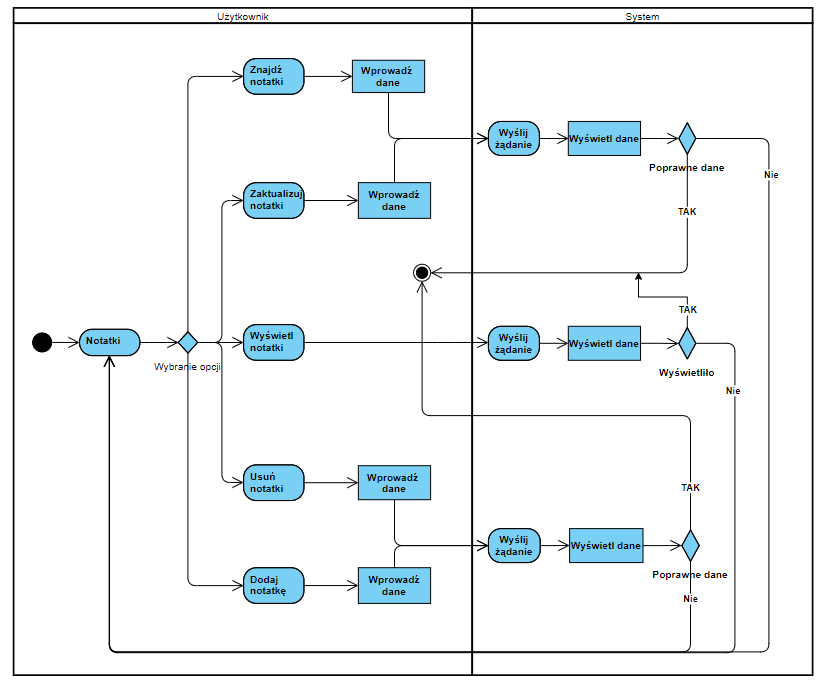


Rys. .. Diagram aktywności dla OA4: Zarządzanie zadaniami

Diagram aktywności UML (ang. Unified Modeling Language) przedstawiony na rysunku 2.10 ukazuje komunikację zdarzeń w procesie biznesowym dla obszaru aktywności „zarządzanie zadaniami”.

Posługuje się on do zwizualizowania tego za pomocą zestawu symboli, zaczynając od węzła początkowego po wybór działalności „zadanie”, skąd następnym krokiem jest węzeł decyzyjny z rozgałęzieniem do pięciu opcji CRUDowych (ang. Create-Read-Update-Delete) z możliwością działania przez wprowadzenie danych, skąd system przyjmuje żądanie, następnie weryfikując dane, po czym znowu następuje węzeł decyzyjny, z którego po poprawnej weryfikacji przechodzi do węzła końcowego, a w przeciwnym wypadku powraca do symbolu działalności zadania.

Diagram aktywności z rysunku 2.11. odwołuje się do OA5: Zarządzanie notatkami (Rys. 1.7.).



Rys. .. Diagram aktywności dla OA5: Zarządzanie notatkami

Diagram czynności przedstawiony na rysunku 2.11 dla obszaru aktywności

„zarządzanie notatkami” ilustruje przepływ elementów w systemie, pokazujący przejście między węzłami.

Punktem startowym użytkownika jest panel notatek, skąd węzeł decyzyjny rozszerzany jest o pięć opcji wyboru. Kolejnym krokiem są reprezentacje działań, jak znajdź notatki, zaktualizuj notatki, usuń notatki. Następnie dla czterech przypadków poza wyświetleniem notatek użyte są symbole działań, które prowadzą do weryfikacji żądania wykonanego przez użytkownika, po czym trwa weryfikacja podobszaru działalności „wyświetl dane” z czego trafia ponownie do węzła decyzyjnego, gdzie poprawne dane prowadzą do węzła końcowego, a niepoprawne wracają do panelu notatek.

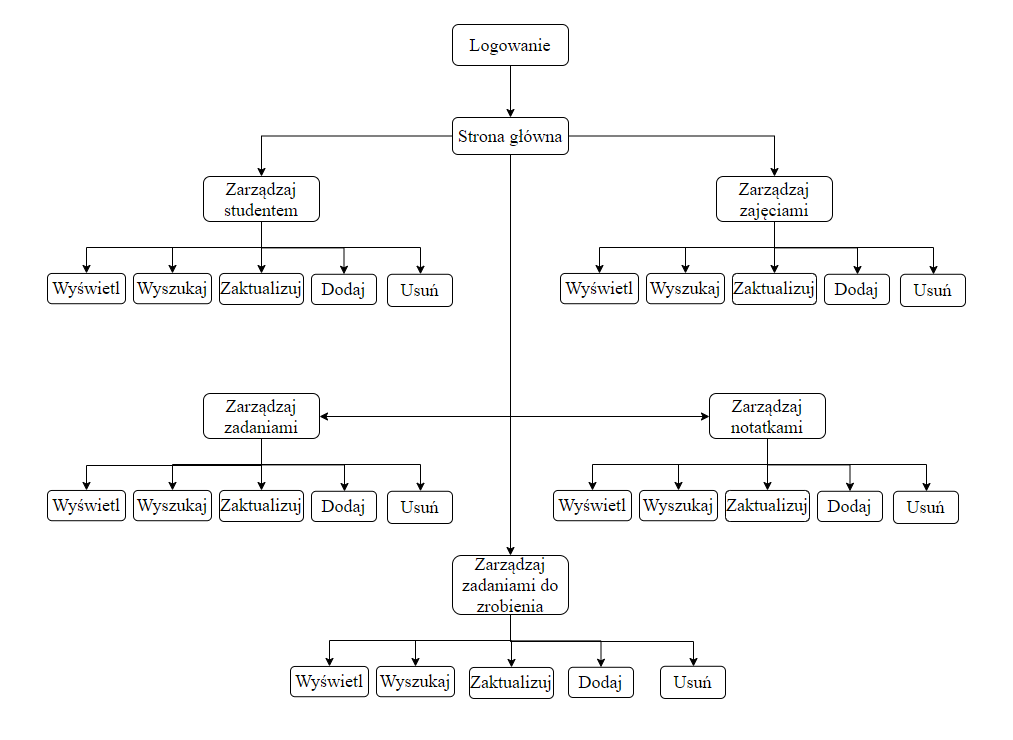
## Projekt interfejsu użytkownika

### Założenia

Interfejs graficzny użytkownika w oprogramowaniu ma na celu być intuicyjny, przejrzysty i w odświeżonej, nowoczesnej szacie graficznej. Wszystkie podstrony systemu będą bazować na podobnym lub identycznym kaskadowym arkuszu stylu,   
w responsywny sposób zostaną dostosowane pod użytkowanie wieloplatformowe zarówno na jednostkach stacjonarnych, przenośnym i mobilnych jak smartphony działające na systemie Android oraz iOS.

### Struktura

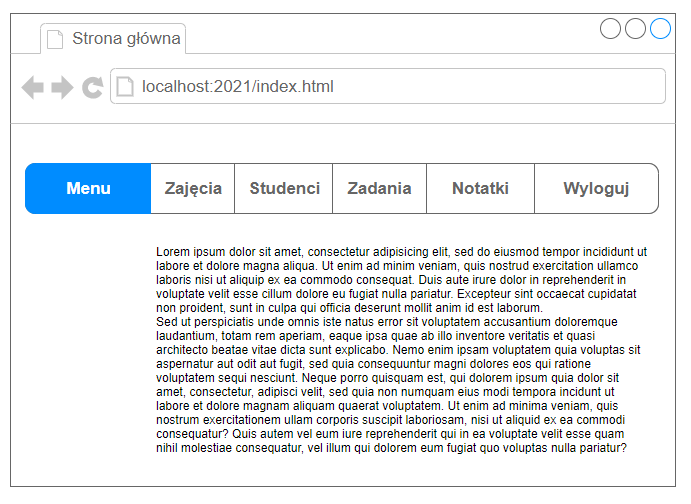
Ten podrozdział jest poświęcony przedstawieniu struktury interfejsu graficznego w formie diagramów. Diagram z rysunku 2.12. przedstawia wizję struktury:



Rys. .. Diagram ilustrujący strukturę interfejsu graficznego

### Przykłady elementów interfejsu

W tym rozdziale zostaną pokazane rysunki, które ilustrują przykładowe interfejsy graficzne oprogramowania dla użytkownika. Rysunek 2.13 przedstawia projekt interfejsu dla strony głównej:



Rys. .. Projekt interfejsu graficznego strony głównej

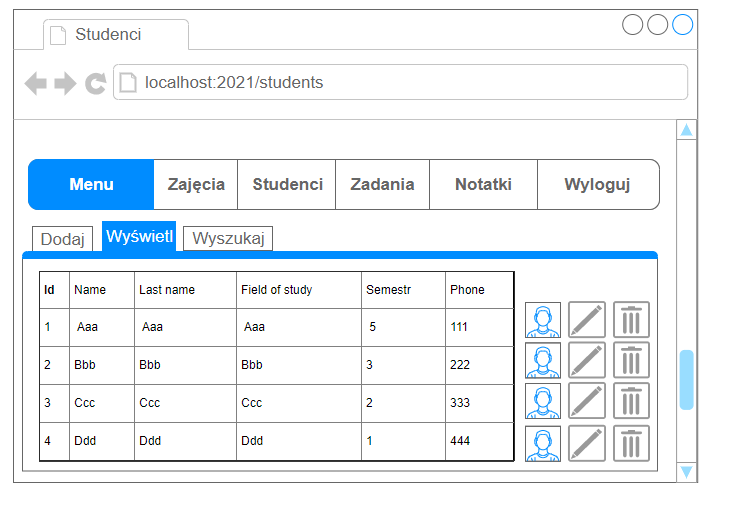
Przedstawiony powyżej rysunek 2.13 przedstawia projekt interfejsu graficznego dla strony głównej systemu, która jest początkową stroną oprogramowania, gdzie użytkownik posiada pełną możliwość przejścia do pełnej funkcjonalności systemu.

Na widocznym przykładzie pokazany jest główny pasek nawigacyjny, na którym znajduję się odnośnik do menu, zajęć, studentów, zadań, notatek oraz możliwość wylogowania się z konta. W zapełnionym polu tekstowym będą znajdować się ogłoszenia lub informacje o tematyce związanej z uczelnia albo przypominającej użytkownikowi   
o danej czynności.

Rysunek 2.14 przedstawia projekt interfejsu dla podstrony studentów, do którego dostęp będzie miał tylko administrator z pełnym dostępem do korzystania z wszystkich funkcjonalności systemu dla tej zakładki.

W panelu studenta jest przewidywany wgląd do pełnej listy studentów w formie tabeli z podziałem na różne parametry danej jednostki. Po prawej stronie tabeli będą znajdować się trzy przyciski, pierwszy od lewej strony za wyświetlenie danego studenta na oddzielnym adresie url, drugi w kolei przycisk pełni funkcję aktualizowania osoby w tabeli, a ostatni będzie odpowiadał za usuwanie persony z rejestru studentów.

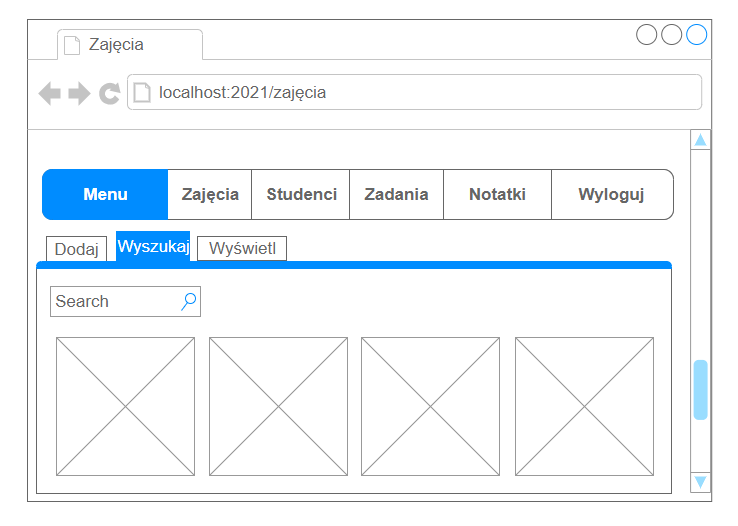
Domyślnie panel studenta będzie posiadał w górnej części strony pasek nawigacyjny taki sam jak na stronie startowej z odnośnikami do innych funkcjonalności systemu.



Rys. .. Projekt interfejsu graficznego wykazu studentów

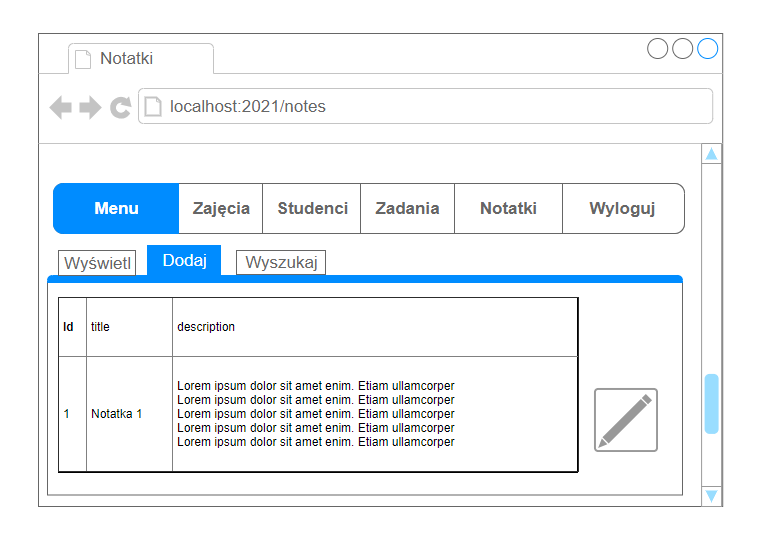
Zrzut ekranu przedstawiony na rysunku 2.15 przedstawia projekt interfejsu graficznego dla wyszukiwarki zajęć uczelnianych. Składa się on z domyślnego dla każdej podstrony paska nawigacyjnego w górnej części witryny, oddzielnej zakładki wyszukiwania, gdzie znajduję się pasek wyszukiwania oraz pod nim wyniki wyszukiwania.

Dostęp do tej opcji będzie miał każdy użytkownik, a informacje dotyczące poszczególnych zajęć będą zawarte w zewnętrznej bazie danych MySQL.   
W celu dostania się do tej zakładki użytkownik będzie musiał zalogować się, następnie przejść do podstrony zajęć, gdzie domyślnie będzie wyświetlana lista studentów, kolejnym krokiem będzie wybranie opcji „wyszukaj” na podrzędnym pasku nawigacyjnym.



Rys. .. Projekt interfejsu graficznego wyszukiwarki zajęć

Rysunek 2.16 przedstawia projekt interfejsu graficznego dodawania notatek, który posiada analogiczny wygląd do poprzednich rysunków makiet dla strony głównej, wykazu studentów oraz wyszukiwarki zajęć. Interfejs posiada domyślny pasek nawigacyjny w górnej części witryny z odnośnikami do konkretnych funkcjonalności. Dodawanie nowych notatek będzie polegało na uzupełnieniu informacji dotyczącej danego wpisu zależnie od parametrów podanych w tabeli i kliknięcie przycisku po prawej stronie tabeli. Dostęp do tej podstrony będą posiadali wszyscy użytkownicy.



Rys. .. Projekt interfejsu graficznego dodawania notatek

# Implementacja oprogramowania

## Charakterystyka wykorzystanych technologii

### Języki programowania

W całym procesie implementacji projektu oprogramowania brało udział kilka technologii oraz związane z nimi frameworki:

* Java – jest popularnym językiem programowania obiektowego i platformą programistyczną. Znajduję ona swoje zastosowania od laptopów, po centra danych, od konsol do gier, po telefony komórkowe. Pozostaje ona platforma programistyczną najchętniej wybieraną przez duże firmy i deweloperów, a zawdzięcza to szybkości, bezpieczeństwu i niezawodności. [9]
* SQL – (ang. Structured Query Language) jest to język strukturalny zapytań, używany do tworzenia, modyfikowania, umieszczania i pobierania danych z baz danych. Należy on do języków deklaratywnych. Bazy danych oparte na SQL pozwalają na efektywne tworzenie zapytań i statystyk, a także ułatwiają pracę ze skomplikowanymi, relacyjnymi danymi. [17]
* HTML – (ang. HyperText Markup Language) jest to język znaczników hipertekstu używany do tworzenia struktury umieszczonej w stronie internetowej. Pozwala on przedstawić konkretny wygląd dokumentu z poziomu przeglądarki internetowej. [6]
* CSS – (ang. Cascading Style Sheets) jest to język wykorzystywany do tworzenia i ustalania wyglądu dokumentów HTML. Pozwala on na przechowywanie informacji dotyczących wszelkich zmian koloru, kształtu itp. w oddzielnym pliku. [3]

### Narzędzia wspomagające

Podczas rozwoju implementacji kodu źródłowego, weryfikacji poprawności wyników i zarządzania bazą danych skorzystano z następujących narzędzi:

* IntelliJ IDEA – jest to zintegrowane środowisko programistyczne dla Javy zrobione przez firmę JetBrains. W intuicyjny sposób ułatwia tworzenie pełnego kodu źródłowego przez odpowiednie podpowiadanie metod oraz importowanie wszelkich bibliotek. Jedną z największych zalet tego środowiska jest integrowanie podstawowego języka programowania z dodatkowymi narzędziami i bazami danych. [7]
* Maven – jest to ogólnodostępne narzędzie stworzone do zarządzania kompilacją, umożliwia on na wdrożenie różnych zależności do projektu oraz pluginów. [11]
* Spring – jest to najpopularniejszy framework dla języka Java, głównie dlatego ponieważ skupia się on na szybkości i prostocie produktywności. Sprawia, że programowanie w Javie jest szybsze, łatwiejsze i bezpieczniejsze dla każdego użytkownika. [16]
* Hibernate – jeden z wolno dostępnych frameworków w języku Java, który wykorzystywany jest do kreowania warstwy dostępu do danych (ang. Persistence layer). Integruje on projekt z relacyjnymi i nierelacyjnymi bazami danych. [5]
* Postman – narzędzie dostępne w wersji przeglądarkowej jako rozszerzenie i aplikacja pulpitowa, pozwalające na kolaboracje tworzenia API (ang. Application programming interface). Użytkownik za pośrednictwem tego narzędzia jest w stanie wysłać zapytania http. [13]
* Bootstrap – jest to biblioteka CSS pozwalająca na błyskawiczne tworzenie interfejsu graficznego strony. W swoim wachlarzu możliwości zawiera on wszelkie elementy jak przyciski, formularze, tabele, paski nawigacyjne oraz wiele innych komponentów możliwych do przechowywania na stronie. [1]
* XAMPP Control Panel – jest to narzędzie do zarządzania, które pozwala na kontrolowanie elementów serwerowych takich jak bazę danych. Dostępne na wielu platformach np. na Windowsie, Linuxie. [22]
* phpMyAdmin – jest to wolno dostępne oprogramowanie, służące do zarządzania bazą danych MySQL z poziomu administratora. Można za pośrednictwem tego narzędzia tworzyć, usuwać, edytować tabele oraz ustalać relacje między nimi. Zawiera ono dodawanie nowych użytkowników bazy danych i podtrzymywanie serwera.
* GitHub – serwis internetowy odpowiada za przechowywanie projektów i repozytorium w różnych językach programowania. Korzysta on z Git, czyli systemu kontroli wersji, przez co w wygodny sposób można wrócić do stanu projektu przed wprowadzeniem danej zmiany oraz kontrolowania rozwoju implementacji. [4]

## Opis implementacji

### Wykaz plików źródłowych

Ten podrozdział poświęcony jest przedstawieniu wszystkich plików źródłowych implementacji w formie struktury oraz tabeli.

Rysunek 3.1. przedstawia strukturę klas zawartych w katalogu głównym, gdzie znajdują się pliki zapisane w ścisłej konwencji dla Javy:

<katalog główny> ─┐

├─ \main ─┐

│ ├─ \config ─┐

│ │ ├─ SecurityConfig.java

│ │ └─ TaskConfiguration.java

│ │

│ ├─ \controller ─┐

│ │ ├ ClassesController.java

│ │ ├ ExerciseController.java

│ │ ├ InfoController.java

│ │ ├ NoteController.java

│ │ ├ StudentController.java

│ │ ├ TaskController.java

│ │ ├ TestController.java

│ │ └─ WebController.java

│ ├─ \entity ─┐

│ │ ├─ Classes.java

│ │ ├─ Exercise.java

│ │ ├─ Note.java

│ │ ├─ Student.java

│ │ └─ Task.java

│ │

│ ├─ \repository ─┐

│ │ ├ ClassesRepository.java

│ │ ├ ExerciseRepository.java

│ │ ├ NoteRepository.java

│ │ ├ SqlTaskRepository.java

│ │ ├ StudentRepository.java

│ │ └─ TaskRepository.java

│ └─ \service ─┐

│ ├─ ClassesService.java

│ ├─ ExerciseService.java

│ ├─ NoteService.java

│ └─ StudentService.java

│

└─ ThesisApplication.java

Rys. 3.1. Struktura katalogów głównych w projekcie

Tabela 3.1. przedstawia wykaz klas wchodzących w skład projektu:

|  |  |
| --- | --- |
| ***Nazwa pliku*** | ***Opis zawartości*** |
| ThesisApplication.java | Główny plik zawierający funkcję main() odpalający aplikację Springową |
| SecurityConfig.java | Plik konfiguracyjny zawierający ustawienia z dostępem do poszczególnych podstron serwisu |
| TaskConfiguration.java | Konfiguracja odpowiadająca za zadania do zrobienia |
| ClassesController.java | Kontroler RESTowy zajęć mapujący podstrony CRUD’u |
| ExerciseController.java | Kontroler RESTowy zadań mapujący podstrony CRUD’u |
| InfoController.java | Kontroler RESTowy mapujący podstronę */info* |
| *NoteController.java* | Kontroler RESTowy notatek mapujący podstrony CRUD’u |
| StudentController.java | Kontroler RESTowy studenta mapujący podstrony CRUD’u |
| TaskController.java | Kontroler RESTowy zadań do zrobienia mapujący podstrony CRUD’u |
| WebController.java | Kontroler mapujący podstrony obsług wyjątków i innych stron zawartych w serwisie webowym |
| Classes.java | Klasa zawierająca dane na temat modelu zajęć |
| Exercise.java | Klasa zawierająca dane na temat modelu zadań |
| Note.java | Klasa zawierająca dane na temat modelu notatek |
| Student.java | Klasa zawierająca dane na temat modelu studenta |
| Task.java | Klasa zawierająca dane na temat modelu zadań do zrobienia |
| ClassesRepository.java | Interfejs repozytorium zajęć przechowujący metody używane w serwisie |
| ExerciseRepository.java | Interfejs repozytorium zadań przechowujący metody używane w serwisie |
| NoteRepository.java | Interfejs repozytorium notatek przechowujący metody używane w serwisie |
| SqlTaskRepository.java | Interfejs repozytorium rozszerzające *TaskRepository*, o bibliotekę *Java Persistence API.* |
| StudentRepository.java | Interfejs repozytorium przechowujący metody używane w serwisie |
| TaskRepository.java | Interfejs repozytorium zadań do zrobienia przechowujący metody używane w kontrolerze |
| ClassesService.java | Plik zawierający metody zajęć wykorzystywane  w kontrolerze RESTowym |
| ExerciseService.java | Plik zawierający metody zadań wykorzystywane  w kontrolerze RESTowym |
| NoteService.java | Plik zawierający metody notatek wykorzystywane  w kontrolerze RESTowym |
| StudentService.java | Plik zawierający metody studentów wykorzystywane w kontrolerze RESTowym |
| 401.html | Podstrona html dla kodu błędu 401 |
| 404.html | Podstrona html dla kodu błędu 404 |
| Index.html | Główna strona serwisu webowego |

Tabela .. Opis zawartości plików wchodzących w skład projektu oprogramowania

### Omówienie wybranych fragmentów kodu

Na wydruku 3.1, przedstawiona jest przykładowa konfiguracja adaptera WebSecurityConfigurerAdapter. Metoda userDetailsService() dodaje do systemu konto studenta z odpowiednią dla niego rolą.

Funkcja withDefaultPasswordEncoder()korzysta z nieaktualnego kodowania hasła, która została zaimplementowana w początkowym stadium implementacji, pod wymogi systemu logowania się.

Dalej opisana jest autoryzacja dla żądań HTTP, z uwzględnieniem na pełny cykl CRUDowy (ang. Create-Read-Update-Delete), czyli tworzenia, odczytywania, aktualizowania oraz usuwania notatek nadając uprawnienia do konkretnych czynności dla poszczególnych ról.

Dalsze metody formLogin()oraz logout()pozwalają wszystkim użytkownikom na korzystanie z możliwości zalogowania się i wylogowania z panelu zarządzania notatkami. Natomiast metoda csrf() (*ang. cross-site request forgery*), jest funkcją chroniącą serwis webowy przed wszelakimi atakami z użyciem aplikacji zewnętrznych. Została ona wyłączona, aby móc wysyłać żądania HTTP,   
na serwer za pośrednictwem narzędzia Postman.

@Bean

public UserDetailsService userDetailsService() {

UserDetails user = User.withDefaultPasswordEncoder()

.username("student")

.password("student1")

.roles("STUDENT")

.build();

http.httpBasic().and().authorizeRequests()

.antMatchers(HttpMethod.GET, "/notes").permitAll()

.antMatchers(HttpMethod.GET, "/notes/{id}").permitAll()

.antMatchers(HttpMethod.PUT, "/notes")

.hasAnyRole("MODERATOR", "WYKLADOWCA")

.antMatchers(HttpMethod.PUT, "/notes/{id}")

.hasAnyRole("MODERATOR", "WYKLADOWCA")

.antMatchers(HttpMethod.POST, "/notes")

.hasAnyRole("MODERATOR", "WYKLADOWCA")

.and()

.formLogin().permitAll()

.and()

.logout().permitAll()

.and()

.csrf().disable();

Wydruk .. Wydruk kodu źródłowego SecurityConfig.java

Wydruk 3.2 przedstawia kod źródłowy klasy *WebController.java* ma za zadanie obsługiwać błędy protokołu HTTP, w różnych przypadkach. Kontroler mapuję dany odnośnik do strony dla sytuacji wyjątkowej.

Między innymi znajduję się status kodu 401, który odsyła do podstrony „/401”, gdzie wyświetlana jest informacja ustawiona przez nas w pliku *401.html.* Ten błąd występuję, jeżeli użytkownik chce wykonać daną czynność przez aplikacje zewnętrzne, do której nie ma nadanych praw dla roli, czyli jest niezautoryzowany. W przypadku mapowania strony „*/404”* użytkownika przekieruje na nią, podczas podania niepoprawnej strony URL. Dodatkowo znajduję się tutaj odniesienie do strony startowej, która jest wyznaczone mapowanie o wartości „*/index”*, gdzie kontroler wczyta stronę, opisaną w pliku *index.html,* będący początkiem serwisu webowego dla nowego odwiedzającego.

@Controller

public class WebController {

@RequestMapping(value = "/index",method = RequestMethod.GET)

public String index() {

return "index";

}

@RequestMapping(value = "/404",method = RequestMethod.GET)

public String error404() {

return "404";

}

@RequestMapping(value = "/401",method = RequestMethod.GET)

public String errror401() {

return "401";

}

}

Wydruk .. Wydruk kodu źródłowego SecurityConfig.java

Kolejny wydruk (wydruk 3.3.) pełnego kodu źródłowego dla klasy *ExerciseService.java*, gdzie znajdują się metody wykorzystywane w dalszym procesie pisania dla kontrolera RESTowego danego schematu. Stworzony został obiekt modelu klasy *Exercise.java* w celu operowania jego zmiennymi na niniejszych metodach.   
Pobiera on zależności zapisane w interfejsie *ExerciseRepository.java*, który jest rozszerzony o klasę *JpaRepository*, co pozwala na korzystanie z gotowych metod wykorzystywanych w cyklu *Create-Read-Update-Delete.*

Z przedstawionych na poniższym wydruku 3.3 metod korzysta kontroler *ExerciseController.java*, gdzie używane są one do nadania konkretnej funkcjonalności.

@Service

public class ExerciseService {

@Autowired

private ExerciseRepository repository;

public Exercise saveExercise(Exercise exercise) {

return repository.save(exercise);

}

public List<Exercise> saveExercises(List<Exercise> exercises) {

return repository.saveAll(exercises);

}

public List<Exercise> getExercises() {

return repository.findAll();

}

public Exercise getExerciseById(int id) {

return repository.findById(id);

}

public Exercise getExerciseByContent(String content) {

return repository.findByContent(content);

}

public Exercise getExerciseByAuthor(String author) {

return repository.findByAuthor(author);

}

public Exercise updateExercise(Exercise exercise) {

return repository.save(exercise);

}

public int deleteExerciseById(int id) {

Exercise e = repository.findById(id);

repository.delete(e);

return id;

}

}

Wydruk .. Wydruk kodu źródłowego ExerciseService.java

Wydruk 3.4 przedstawia większość kodu źródłowego dla klasy kontrolera RESTowego *ExerciseController.java*. Utworzony w nim obiekt prywatny *service* klasy *ExerciseService.java* daje możliwość pełnego korzystania   
z dostępnych metod zapisanych w wyżej wymienionej klasie. Zostały zmapowane konkretne przypadki żądań dla podstrony zadania takie jak dodawanie zadania, przy pomocy adnotacji *@PostMapping* z wartością wskazującą na kontynuacje nazwy witryny. Analogicznie został przeprowadzony cały ciąg metod dla usuwania, aktualizowania, wyszukiwania zadania po konkretnym parametrze oraz usuwania.

Wszystkie mapowania korzystają z kontrolera RESTowego, który inicjalizuję się adnotacja *@RestController* nad samą nazwą klasy. Dostarcza on między innymi mapowania takie jak *@GetMapping* (pobranie danych)*, @PutMapping (*zaktualizowanie danych), *@DeleteMapping* (usunięcie danych).

@Autowired

private ExerciseService service;

@PostMapping("/addexercise")

public Exercise addExercise(@RequestBody Exercise exercise) {

return service.saveExercise(exercise);

}

@PostMapping("/addexercises")

public List<Exercise> addExercises(@RequestBody List<Exercise>

exercises) {

return service.saveExercises(exercises);

}

@GetMapping("/allexercises")

public List<Exercise> getExercises() {

return service.getExercises();

}

@GetMapping("/exercisebyid/{id}")

public Exercise getExerciseById(@PathVariable int id) {

return service.getExerciseById(id);

}

@GetMapping("/exercisebycontent/{content}")

public Exercise getExerciseByContent(@PathVariable String content) {

return service.getExerciseByContent(content);

}

@GetMapping("/exercisebyauthor/{author}")

public Exercise getExerciseByAuthor(@PathVariable String author) {

return service.getExerciseByAuthor(author);

}

@PutMapping("/update")

public Exercise updateExercise(@RequestBody Exercise exercise) {

return service.updateExercise(exercise);

}

@DeleteMapping("/delete/{id}")

public int deleteExerciseById(@PathVariable int id) {

return service.deleteExerciseById(id);

}

Wydruk .. Wydruk kodu źródłowego ExerciseController.java

### Obsługa błędów i sytuacji wyjątkowych

Dla projektu oprogramowania została przewidziana obsługa błędów, która bazuje w większości na komunikacji narzędzia Postman z serwisem webowym.

Daje on możliwość na przeprowadzenie pełnego sprawdzenia sytuacji wyjątkowych przez wysyłanie żądań HTTP, w formacie JSON (*ang. JavaScript Object Notation*), który podczas poprawnego wprowadzania i wysłania zapytania na serwer nie informuje użytkownika o niepoprawnym rezultacie działania tylko zostawia puste klamerki.

Na wydruku 3.5. umieszczone jest przykładowe ciało wysyłane jako żądanie   
na serwer w formacie JSON:

{

"id": 1,

"subject": "Seminarium dyplomowe",

"lecturer": "dr Jan Kowalski",

"form": "Seminarium",

"pass": true

}

Wydruk .. Przykładowy format JSON żądania wysyłanego na serwer

W przypadku niepowodzenia wynikającego z złego podania chociażby zmiennej albo nazwy tej zmiennej serwer zwróci wiadomość do technologii komunikującej się   
z nim Postman, gdzie w polu wyskoczy błąd z dokładnym statusem kodu protokołu HTTP.

Spring framework daje pełną możliwość na dodawanie błędów, np. w przypadku podania pustego pola dla zmiennej, posługując się adnotacją @NotBlank, którą należy umieścić w klasie modelu całego schematu.

## Opis użytkowania

Ten podrozdział został poświęcony przedstawieniu sposobu uruchomienia oprogramowania na każdym urządzeniu oraz instalowania niezbędnych urządzeń i sterowników na urządzeniu. Przykład będzie przeprowadzany na platformie Windows.

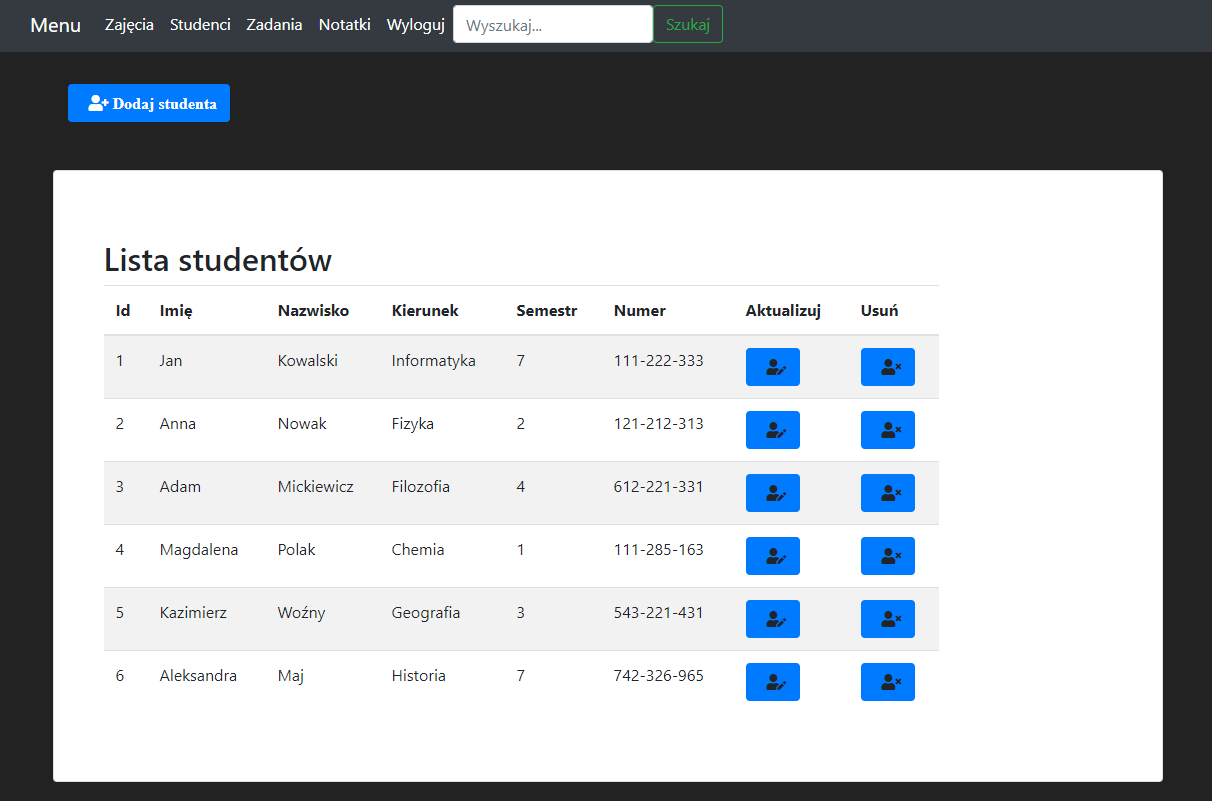
Do uruchomienia aplikacji wymagane jest zainstalowanie następujących narzędzi:

* Środowisko programistyczne *IntelliJ IDEA* w wersji Ultimate z 30 dniowym okresem próbnym, integracyjne środowisko *Eclipse* lub *Visual Studio Code* z wtyczką „*Spring Tools 4”.*
* Pobranie i zainstalowanie panelu kontrolnego *XAMPP Control Panel*.
* Zimportować niezbędne bazy danych do panelu zarządzania bazami pod adresem strony *localhost/phpMyAdmin/* .
* Pobranie i zainstalowanie narzędzia *Postman* do kolaboracji z API.

Do wykorzystania funkcjonalności projektu oprogramowania należy:

* Włączyć jedno z zainstalowanych zintegrowanych środowisk programistycznych np. *IntelliJ Idea Ultimate*. I włączyć serwer przez uruchomienie aplikacji.
* Uruchomić narzędzie XAMPP Control Panel i włączyć moduł MySQL
* W preferowanej przeglądarce internetowej wpisać adres witryny: *localhost:2021/index*
* Przekieruję nas na witrynę, z której mamy dostęp do funkcjonalności systemu.
* Istnieje możliwość dodatkowej komunikacji z serwerem za pośrednictwem zewnętrznego narzędzia Postman.
* Po uruchomieniu *Postmana*, należy podać adres strony zależnie   
  od mapowania kontrolerów i wybrać jedną z pierwszych pięciu opcji żądań, które użytkownik chce uzyskać od serwera.
* Następnym krokiem dla aplikacji *Postman* jest wybranie ciała w zakładce poniżej wpisywania adresu i zmienić ją na ciało, po czym ustawić sposób formatowania informacji na *JSON*.
* W przypadku autoryzacji dla konkretnych funkcjonalności trzeba zalogować się na konto *studenta, wykładowcy* lub *moderatora*. Dla aplikacji Postman istnieje opcja autoryzacji, gdzie użytkownik wpisuje te same dane co podczas procesu logowania przy użyciu przeglądarki internetowej.

W celu wyświetlenia listy studentów, użytkownik musi przejść do zakładki studenci z nawigacyjnego menu położonego w górnej części strony. Po wybraniu podstrony wyświetli się lista studentów znajdujących się w bazie danych.

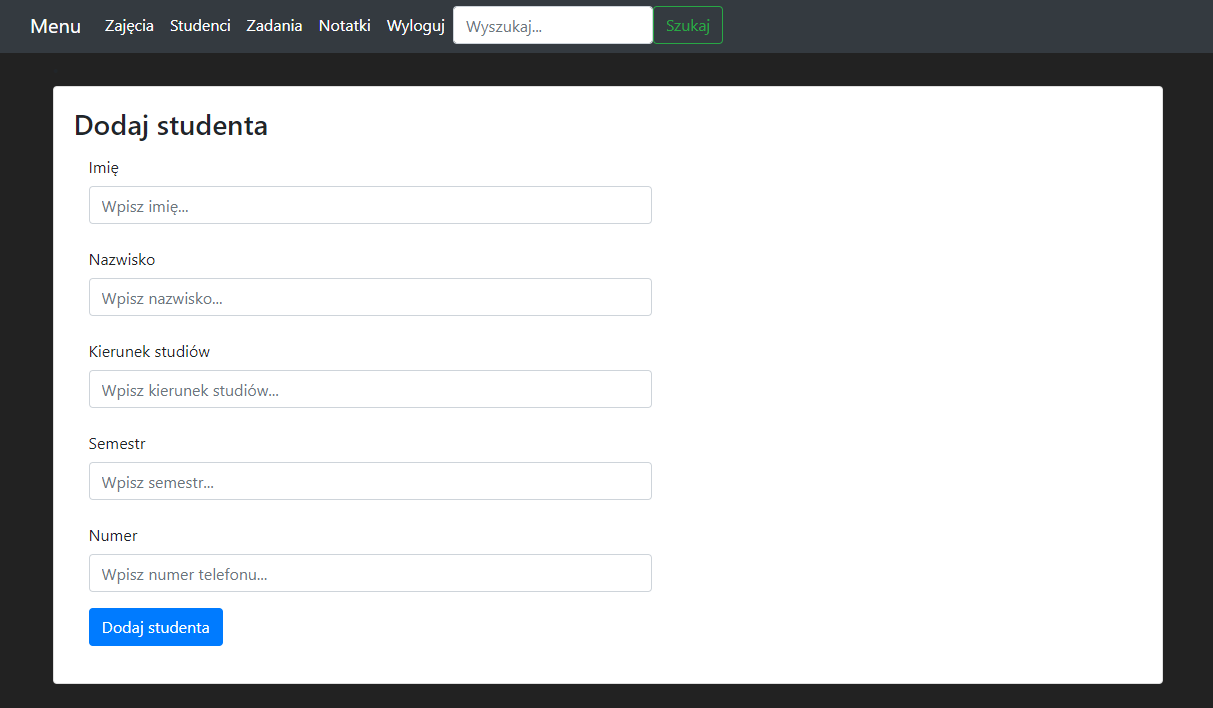


Rys. 3.2. Interfejs graficzny wykazu studentów

Na powyższym rysunku 3.2 interfejsu graficznego wykazu studentów widać omawiane wyświetlanie listy uczniów danej uczelni. Użytkownik posiada dodatkową opcję przejścia do procesu dodawania studentów za pomocą przycisku dodaj studenta. Każdy wpis do tabeli studentów znajdujący się na widoku, posiada możliwość aktualizacji oraz usuwania pojedynczych wierszy.

Lista studentów składa się z uniwersalnego identyfikatora id, który jest niepowtarzalny dla każdej osoby znajdującej się w bazie danych. Kolejną tabelą jest imię, nazwisko, kierunek studiów, semestr oraz numer telefonu konkretnej persony.

Dodawanie nowych użytkowników do tabeli studentów w bazie danych przebiega za pośrednictwem podstrony „Dodaj studenta”, do której można dostać się z panelu listy studentów klikając odpowiedni przycisk funkcyjny.

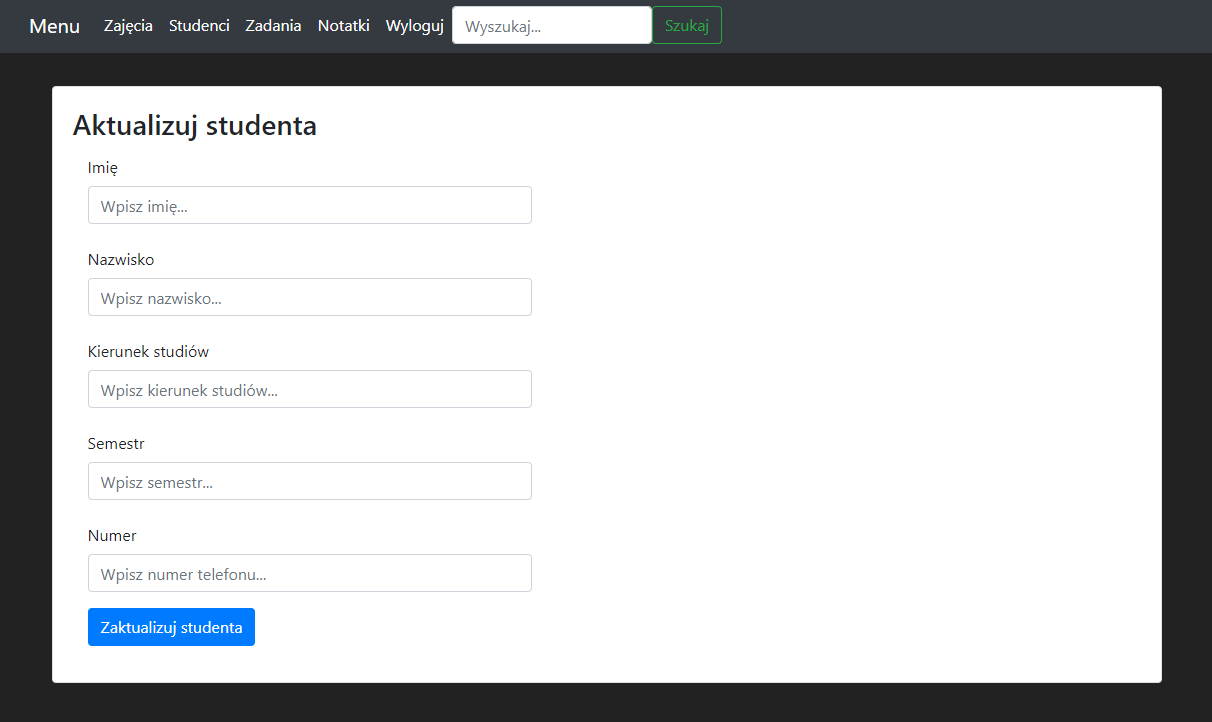


Rys. .. Interfejs graficzny dodawania nowych studentów

Rysunek 3.3. interfejs graficznych dodawania studentów przedstawia widok podstrony odpowiedzialnej za uzupełnianie listy uczniów o nowe wpisy. Wcielanie nowych danych do bazy studentów przebiega przez wypełnienie formularza widocznego na powyższym zrzucie ekranu, gdzie należy wprowadzić dane dotyczące danej jednostki zaczynając od imienia, a kończąc na numerze telefonu.

Kolejność wdrażania nowych informacji do zewnętrznej bazy danych przebiega   
w analogiczny sposób dla zajęć, zadań oraz notatek. Główną różnicą między nimi jest wprowadzanie innych właściwości do formularza dodawania.

W systemie aktualizowanie studentów umieszczonych do zewnętrznej bazy danych przebiega przez ponowne uzupełnienie informacji w formularzu. Użytkownik dostaje   
się do niego przez listę studentów klikając przycisk znajdujący się w kolumnie aktualizuj obok konkretnego ucznia.



Rys. .. Interfejs graficzny aktualizowania danych o studentach

Zrzut ekranu interfejsu graficznego aktualizowania studentów przedstawia cały formularz nadpisywania ucznia, który jest analogiczny do dodawania nowych studentów do bazy danych. Wprowadzenie wszystkich informacji zatwierdzane jest naciśnięciem przycisku funkcyjnego „zaktualizuj studenta”, co wysyła żądanie do bazy danych nadpisujące istniejący wpis.

Podobna funkcjonalność aktualizacji informacji w zewnętrznej bazie danych MySQL zostało użyte dla podstrony zajęć, zadań oraz notatek, gdzie główną różnicą między nimi jest odmienność wprowadzanych instrukcji.

## Testowanie oprogramowania

W tym podrozdziale znajduje się przebieg testowania dla podanych obszarów aktywności w systemie pod aspektami dynamiki oraz stabilności działania podczas różnych sytuacji. Zostały one przedstawione na następujących tabelach 3.2. – 3.7.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Scenariusz** | **Kroki testowania** | **Wprowadzone dane** | **Zamierzony  rezultat** | **Wynik** |
| Logowanie | Użytkownik wprowadza nazwę użytkownika oraz hasło | Nazwa użytkownika, hasło | Zalogowanie | Poprawny |
|  |  | Niepoprawna nazwa użytkownika | Informacja  o niewłaściwej wartości pola | Niepoprawny |
|  |  | Niepoprawne hasło | Informacja  o niewłaściwej wartości pola | Niepoprawny |
| Zresetuj hasło | System sprawdza wprowadzone dane | Numer indeksu | Informacja  o zresetowaniu hasła | Poprawny |
|  |  | Wpisany tekst zamiast indeksu | Informacja  o niewłaściwej wartości pola | Niepoprawny |

Tabela .. Przypadki testowe dla obszaru OA1 – Logowanie

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Scenariusz** | **Kroki testowania** | **Wprowadzone dane** | **Zamierzony rezultat** | **Wynik** |
| Wyświetl studentów | Użytkownik wybiera opcję „wyświetl” w zarzadzaniu studentem | Wprowadzenie dokładnego adresu URL | Wyświetlenie listy studentów | Poprawny |
|  |  | Wprowadzenie błędnego adresu URL | Informacja  o niewłaściwej wartości pola | Niepoprawny |
| Znajdź  studenta | Użytkownik wprowadza ID, nr. Telefonu, kierunek studiów | numer Id, numer telefonu, kierunek studiów | Wyświetlenie danego studenta | Poprawny |
|  |  | Błędny kierunek studiów | Informacja  o niewłaściwej wartości pola | Niepoprawny |
|  |  | Niepoprawny numer Id | Informacja  o niewłaściwej wartości pola | Niepoprawny |
| Dodaj  studenta | Użytkownik wpisuje dane studenta | Id, imię, nazwisko, kierunek studiów, semestr, numer telefonu | Dodanie studenta  do bazy danych | Poprawny |
|  |  | Id, imię, semestr,  numer telefonu | Informacja  o niewłaściwej wartości pola | Niepoprawny |
|  |  | brak | Informacja  o niewłaściwej wartości pola | Niepoprawny |
| Zaktualizuj studenta | Użytkownik wpisuje nadpisane dane studenta | Id, imię, nazwisko, kierunek studiów, semestr, numer telefonu | Nadpisuje studenta  do bazy danych | Poprawny |
|  |  | Id, imię, nazwisko | Informacja  o niewłaściwej wartości pola | Niepoprawny |
|  |  | brak | Informacja  o niewłaściwej wartości pola | Niepoprawny |
| Usuń  studenta | Użytkownik podaję numer id danego studenta | Id | Usunięcie studenta  z bazy danych | Poprawny |
|  |  | brak | Informacja  o niewłaściwej wartości pola | Niepoprawny |

Tabela .. Przypadki testowe dla obszaru OA2 – Zarządzanie studentami

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Scenariusz** | **Kroki testowania** | **Wprowadzone dane** | **Zamierzony rezultat** | **Wynik** |
| Wyświetl  zajęcia | Użytkownik wybiera opcję „wyświetl”  w zarzadzaniu zajęciami | Wprowadzenie dokładnego adresu URL | Wyświetlenie listy zajęć | Poprawny |
|  |  | Wprowadzenie błędnego adresu URL | Informacja  o niewłaściwej wartości pola | Niepoprawny |
| Znajdź  zajęcia | Użytkownik wprowadza ID, przedmiot, forma, zaliczenie | numer Id lub przedmiot lub prowadzącego | Wyświetlenie danych zajęć | Poprawny |
|  |  | Błędny prowadzący | Informacja  o niewłaściwej wartości pola | Niepoprawny |
|  |  | Błędny przedmiot | Informacja  o niewłaściwej wartości pola | Niepoprawny |
| Dodaj  zajęcia | Użytkownik wpisuje dane zajęć | numer Id, przedmiot, prowadzący, forma, zaliczenie | Dodanie zajęć  do bazy danych | Poprawny |
|  |  | numer Id, przedmiot, prowadzący, błędna forma, zaliczenie | Informacja  o niewłaściwej wartości pola | Niepoprawny |
|  |  | brak | Informacja  o niewłaściwej wartości pola | Niepoprawny |
| Zaktualizuj zajęcia | Użytkownik wpisuje nadpisane dane zajęć | numer Id, przedmiot, prowadzący, forma, zaliczenie | Nadpisuje zajęcia  do bazy danych | Poprawny |
|  |  | numer Id, przedmiot, zaliczenie | Informacja  o niewłaściwej wartości pola | Niepoprawny |
|  |  | brak | Informacja  o niewłaściwej wartości pola | Niepoprawny |
| Usuń  zajęcia | Użytkownik podaję numer id danych zajęć | Id | Usunięcie zajęć  z bazy danych | Poprawny |
|  |  | brak | Informacja  o niewłaściwej wartości pola | Niepoprawny |

Tabela .. Przypadki testowe dla obszaru OA3 – Zarządzanie zajęciami

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Scenariusz** | **Kroki testowania** | **Wprowadzone dane** | **Zamierzony rezultat** | **Wynik** |
| Wyświetl  zadania | Użytkownik wybiera opcję „wyświetl”  w zarzadzaniu zadaniami | Wprowadzenie dokładnego adresu URL | Wyświetlenie listy zadań | Poprawny |
|  |  | Wprowadzenie błędnego adresu URL | Informacja  o niewłaściwej wartości pola | Niepoprawny |
| Znajdź  zadanie | Użytkownik wprowadza ID, treść, autor | numer Id lub treść  lub autor | Wyświetlenie danego zadania | Poprawny |
|  |  | Błędny treść | Informacja  o niewłaściwej wartości pola | Niepoprawny |
|  |  | Błędny autor | Informacja  o niewłaściwej wartości pola | Niepoprawny |
| Dodaj  zadanie | Użytkownik wpisuje  dane zadania | numer Id, treść, autor | Dodanie zadania  do bazy danych | Poprawny |
|  |  | numer Id, treść,  błędny autor | Informacja  o niewłaściwej wartości pola | Niepoprawny |
|  |  | brak | Informacja  o niewłaściwej wartości pola | Niepoprawny |
| Zaktualizuj zadanie | Użytkownik wpisuje nadpisane dane zadania | numer Id, przedmiot, prowadzący, forma, zaliczenie | Nadpisuje zadania  w bazy danych | Poprawny |
|  |  | numer Id, błędna treść, autor | Informacja  o niewłaściwej wartości pola | Niepoprawny |
|  |  | brak | Informacja  o niewłaściwej wartości pola | Niepoprawny |
| Usuń zadanie | Użytkownik podaję numer id danych zadań | Id | Usunięcie zadania  z bazy danych | Poprawny |
|  |  | brak | Informacja  o niewłaściwej wartości pola | Niepoprawny |

Tabela .. Przypadki testowe dla obszaru OA4 – Zarządzanie zadaniami

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Scenariusz** | **Kroki testowania** | **Wprowadzone**  **dane** | **Zamierzony**  **rezultat** | **Wynik** |
| Wyświetl  notatki | Użytkownik wybiera opcję „wyświetl”  w zarzadzaniu notatkami | Wprowadzenie dokładnego adresu URL | Wyświetlenie listy notatek | Poprawny |
|  |  | Wprowadzenie błędnego adresu URL | Informacja  o niewłaściwej wartości pola | Niepoprawny |
| Znajdź  notatkę | Użytkownik wprowadza ID, tytuł, opis | numer Id lub tytuł  lub opis | Wyświetlenie danej notatki | Poprawny |
|  |  | Błędny tytuł | Informacja  o niewłaściwej wartości pola | Niepoprawny |
|  |  | Błędny opis | Informacja  o niewłaściwej wartości pola | Niepoprawny |
| Dodaj  notatkę | Użytkownik wpisuje  dane notatki | numer Id, błędna treść, autor | Dodanie notatki  do bazy danych | Poprawny |
|  |  | numer Id, treść, autor | Informacja  o niewłaściwej wartości pola | Niepoprawny |
|  |  | brak | Informacja  o niewłaściwej wartości pola | Niepoprawny |
| Zaktualizuj notatkę | Użytkownik wpisuje nadpisane dane notatki | numer Id, treść, autor | Nadpisuje notatkę  w bazy danych | Poprawny |
|  |  | numer Id, treść,  błędny autor | Informacja  o niewłaściwej wartości pola | Niepoprawny |
|  |  | brak | Informacja  o niewłaściwej wartości pola | Niepoprawny |
| Usuń notatkę | Użytkownik podaję numer id danej notatki | Id | Usunięcie notatki  z bazy danych | Poprawny |
|  |  | brak | Informacja  o niewłaściwej wartości pola | Niepoprawny |

Tabela .. Przypadki testowe dla obszaru OA5 – Zarządzanie notatkami

# Zakończenie

Tworząc projekt oprogramowania do zarządzania elektronicznym indeksem głównym celem było stworzenie w projektu serwisu webowego służącym do korzystania z funkcjonalności uczelni jako wirtualna wersja, gdzie student mógłby wygodnie sprawdzać przyszłe zajęcia, oceny, korespondować z wykładowcami i wiele innych.

Aplikacja powinna służyć jako asystent studenta podczas całego toku nauczania z  możliwością korzystania na każdym urządzeniu, dostępnym na wielu platformach, ponieważ wiele aktualnych alternatyw na rynku nie jest aktualizowane posiadając starszy design odbioru strony internetowej.

Cały proces pracy nad implementacją przebiegł w szybki sposób, przez ułatwienia dzisiejszych narzędzi wykorzystywanych do tworzenia serwisów jak integracyjne środowiska programistyczne naprowadzające użytkownika na popełnione błędy, jakże pomagając kreować sam kod źródłowy przez podpowiadanie potrzebnych metod, funkcji wykorzystywanych w całym cyklu. Nie obeszłoby się to bez nowoczesnych frameworków znacznie ułatwiających cały proces tworzenia serwisu webowego, gdzie   
w przypadku tego projektu oprogramowania wykazał się *Spring Boot*, dostarczający niezbędne zależności oraz gotowy serwer do włączenia, bez zbędnych konfiguracji całego serwera *TomCat.*

System posiada pewne niedociągnięcia, które będą poprawiane podczas dalszych prac nad projektem. Planuję kontynuować rozwój systemu w celu doskonalenia własnych umiejętności albo w formie pracy magisterskiej jako projekt rozbudowany o kolejne funkcjonalności wraz z integracją końcowego stadium rozwoju projektu back-end z front-end, gdzie obydwie warstwy zostaną zaprojektowane z użyciem innowacyjnych narzędzi wliczając w to biblioteki, środowiska programistyczne i zewnętrzne aplikacje służące do testowania kodu.

Nakład pracy i czasu włożony w realizacje tego projektu rozwinął moje umiejętności w zarządzaniu projektami tworzonymi w języku programowania Java, pokazał mi możliwości korzystania z wielu narzędzi podczas tworzenia jednego systemu, scalenia ich w jednolity projekt. Pisanie tej pracy pokazało mi kierunek, w którym chce dążyć podczas mojej ścieżki zawodowej i hobbystycznej.

# Bibliografia

1. Bootstrap [Online] URL: <https://getbootstrap.com/>
2. Wirtualna uczelnia Politechniki Świętokrzyskiej [Online] URL: <https://cas.usos.tu.kielce.pl>
3. CSS [Online] URL: <https://www.w3schools.com/Css/>
4. GitHub [Online] URL: <https://github.com/>
5. Hibernate [Online] URL: <https://hibernate.org/>
6. HTML [Online] URL: <https://www.w3schools.com/html/>
7. IntelliJ IDEA [Online] URL: <https://www.jetbrains.com/idea/>
8. J. Bloch, Java efektywne programowanie, Helion, Gliwice, 2018
9. Java [Online] URL: <https://www.java.com/pl/>
10. Java Dokumentacja [Online] URL: <https://docs.oracle.com/javase/7/docs/api/>
11. Maven [Online] URL: <https://maven.apache.org/>
12. Maven Repozytorium [Online] URL: <https://mvnrepository.com/>
13. Postman URL: <https://www.postman.com/>
14. RP.pl, Studenci: elektroniczny indeks i wirtualny dziekanat to na wielu uczelniach fikcja [Online]. URL: <https://www.rp.pl/Edukacja-i-wychowanie/310239974-Studenci-elektroniczny-indeks-wirutalny-dziekanat-to-na-wielu-uczelniach-fikcja.html>
15. Dokumentacja Spring [Online] URL: [https://docs.spring.io/spring-boot/docs/ current/reference/html/](https://docs.spring.io/spring-boot/docs/%20current/reference/html/)
16. Spring Framework [Online] URL: <https://spring.io/>
17. SQL [Online] URL: <https://www.w3schools.com/sql/>
18. Visual Paradigm [Online] URL: <https://online.visual-paradigm.com/>
19. Visual Paradigm Origin of Use Case [Online] URL: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/>
20. W. Wheeler, J. White „Spring w praktyce” Helion, Gliwice, 2014
21. Wirtualna uczelnia Uniwersytet Jana Kochanowskiego, [Online] URL: <https://wu.ujk.edu.pl>
22. XAMPP Control Panel [Online] URL: [https://www.apachefriends.org/ download.html](https://www.apachefriends.org/%20download.html)

**Patryk Gola**  Kielce, dn. ………………………

imię i nazwisko studenta

**129855**

numer albumu

**Informatyka, Technologie informatyczne**

kierunek, specjalność

**studia stacjonarne I stopnia**

rodzaj studiów, forma studiów

**Oświadczenie**

Przedkładając w roku akademickim 2020/2021 pracę inżynierską pod tytułem:

**„Projekt oprogramowania do zarządzania elektronicznym indeksem’’**

oświadczam, że:

* pracę napisałem samodzielnie,
* praca nie stanowi istotnego fragmentu lub innych elementów cudzego utworu,
* praca nie narusza żadnych innych istniejących praw autorskich,
* wykorzystane w pracy materiały źródłowe zastosowane zostały z zachowaniem zasad prawa cytatu,
* wersja elektroniczna (na nośniku elektronicznym i/lub w systemie Wirtualna Uczelnia) pracy jest tożsama z wersją drukowaną.

Równocześnie oświadczam, że jestem świadomy, iż na podstawie art. 15a ustawy z 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (Dz. U. Nr 24 poz. 83 ze zm.) Uniwersytetowi Jana Kochanowskiego w Kielcach przysługuje pierwszeństwo w opublikowaniu mojej pracy magisterskiej w terminie 6 miesięcy od daty jej obrony.

W przypadku nie skorzystania przez Uniwersytet Jana Kochanowskiego w Kielcach z prawa pierwszeństwa publikacji wyrażam zgodę na udostępnianie mojej pracy magisterskiej przez Uniwersytet Jana Kochanowskiego w Kielcach dla celów naukowych i dydaktycznych.

Prawdziwość powyższego oświadczenia potwierdzam własnoręcznym podpisem.

………………………………………………

czytelny podpis studenta