

Bazy Danych 2 - Projekt

Dziennik szkolny

Natalia Rusin - 252725
Aleksandra Tłałka - 252803
Mikołaj Chmielecki - 252723

28 stycznia 2022

Prowadzący: dr inż. Roman Ptak
Termin zajęć: piątek 11:15 - 12:45
Nr grupy: 4

Spis treści

1 Wstęp	3
1.1 Cel projektu	3
1.2 Zakres projektu	3
2 Analiza wymagań	4
2.1 Opis działania systemu	4
2.2 Wymagania funkcjonalne	4
2.3 Wymagania niefunkcjonalne	5
2.3.1 Wykorzystywane technologie i narzędzia	5
2.3.2 Wymagania dotyczące rozmiaru bazy danych	5
2.3.3 Wymagania dotyczące bezpieczeństwa systemu	5
2.4 Dodatkowe założenia projektowe	6
3 Projekt bazy danych	7
3.1 Diagram przypadków użycia i diagram aktywności	7
3.2 Model bazy danych	9
3.3 Indeksy	11
3.4 Procedury składowane	11
3.4.1 <i>Wprowadzanie oceny</i>	11
3.4.2 <i>Usprawiedliwianie nieobecności</i>	11
3.4.3 <i>Zapowiadanie sprawdzianu</i>	12
3.4.4 <i>Archiwizowanie klasy</i>	12
3.4.5 <i>Wpisywać obecności</i>	12
3.4.6 <i>Sprawdzenie zajęć nauczyciela</i>	12
3.4.7 <i>Sprawdzenie zajęć ucznia</i>	13
3.4.8 <i>Sprawdzenie ocen ucznia</i>	13
3.4.9 <i>Sprawdzenie ocen nauczyciela</i>	13
3.4.10 <i>Wyświetl sprawdziany ucznia</i>	13
3.4.11 <i>Wyświetl sprawdziany nauczyciela</i>	13
3.4.12 <i>Wyświetl nieobecności ucznia</i>	13
3.4.13 <i>Wyświetl lekcje ucznia</i>	14
3.4.14 <i>Wyświetl lekcje nauczyciela</i>	14
3.5 Triggery	14
3.5.1 <i>Limit uczniów na klasę</i>	14

3.5.2	<i>Limit klas na rocznik</i>	14
3.5.3	<i>Czy wychowawca</i>	14
3.5.4	<i>Czy ocenić</i>	14
3.5.5	<i>Czy nowy symbol</i>	14
3.6	Mockupy aplikacji webowej	15
4	Implementacja systemu bazy danych	19
4.1	Tworzenie tabel i definiowanie ograniczeń	19
4.1.1	Tworzenie tabel	19
4.1.2	Tworzenie ograniczeń	20
4.2	Implementacja mechanizmów przetwarzania danych	21
4.2.1	Indeksy	21
4.2.2	Widoki	22
4.2.3	Triggery	22
4.2.4	Procedury składowane	23
4.3	Implementacja uprawnień i innych zabezpieczeń	23
4.3.1	Użytkownicy	24
4.3.2	Uprawnienia użytkowników	24
4.4	Testowanie bazy danych na przykładowych danych	25
4.4.1	Procedura składowana	25
4.4.2	Widok	27
4.4.3	Ograniczenia	27
4.5	Triggery	27
4.5.1	Testy wydajnościowe	28
5	Implementacja i testy aplikacji	29
5.1	Instalacja i konfigurowanie systemu	29
5.2	Instrukcja użytkowania aplikacji	29
5.2.1	User flow uczeń	29
5.2.2	User flow nauczyciel	29
5.3	Testowanie opracowanych funkcji systemu	33
5.3.1	Testy automatyczne	33
5.3.2	Testy manualne	34
5.4	Omówienie wybranych rozwiązań programistycznych	34
5.4.1	Implementacja interfejsu i dostępu do bazy danych	34
5.4.2	Implementacja wybranych funkcjonalności systemu	39
5.4.3	Implementacja mechanizmów bezpieczeństwa	40
6	Pliki źródłowe	43
7	Podsumowanie i wnioski	43

1 Wstęp

1.1 Cel projektu

Celem projektu jest nabycie umiejętności projektowania bazy danych i tworzenia aplikacji dostępowych, komunikujących się z bazą danych. Do realizacji projektu wykorzystamy różne biblioteki, ułatwiające zarządzanie zbiorami informacji.

1.2 Zakres projektu

Zakres naszego projektu obejmuje zaprojektowanie bazy danych służącej do obsługi elektronicznego dziennika szkolnego w Akademickim Liceum Ogólnokształcącym Politechniki Wrocławskiej. Baza ta będzie skoncentrowana na zarządzaniu ocenami, frekwencją, sprawdzianami i lekcjami. Ponadto, jednym z zadań będzie również zarządzanie uczniami, klasami, tworzenie planów zajęć. Projekt bazy danych będzie połączony z projektem aplikacji dostępowej, pozwalającej na zarządzanie przechowywanymi danymi. Użytkownicy będą posiadali własne konta dostępowe z odpowiednimi uprawnieniami, od których będzie zależeć ich zakres w zarządzaniu bazą danych. Użytkownicy będą mieli dostęp do serwisu poprzez aplikację webową.

2 Analiza wymagań

2.1 Opis działania systemu

System działa w oparciu o 4 różne rodzaje użytkowników, z których każdy posiada określony zakres uprawnień. Uczeń powinien mieć możliwość wglądu w zapowiedziane dla niego sprawdziany, oceny oraz frekwencję. Rodzic posiada wszystkie te uprawnienia co uczeń, a ponadto ma możliwość usprawiedliwiania nieobecności swojego dziecka. Nauczyciel ma możliwość wpisywania tematów lekcji oraz frekwencji na nich, zapowiadania sprawdzianów, wystawiania ocen z własnego przedmiotu uczniom, z którymi ma zajęcia. Ponadto, każdy z powyższych użytkowników ma możliwość wglądu w plan zajęć. Administrator bazy danych ma możliwość edycji danych użytkowników. Może tworzyć i zmieniać klasy oraz plany zajęć.

2.2 Wymagania funkcjonalne

Nauczyciel:

- wpisywanie, zmiana i usuwanie tematów lekcji,
- zaznaczanie i odznaczanie nieobecności uczniów na zajęciach,
- wgląd w swój plan zajęć oraz w plan lekcji klasy,
- wpisywanie, edycja i usuwanie ocen swoim uczniom,
- zapowiadanie, zmiana terminu, odwoływanie i przeglądanie zapowiedzianych przez siebie sprawdzianów,
- usprawiedliwianie nieobecności uczniów.

Uczeń:

- przeglądanie własnych ocen,
- sprawdzenie swojego planu zajęć,
- przeglądanie zapowiedzianych dla niego sprawdzianów,
- przeglądanie tematów swoich lekcji,
- sprawdzenie swojej frekwencji.

Rodzic:

- przeglądanie ocen swojego dziecka,
- sprawdzanie i usprawiedliwianie nieobecności dziecka,
- przeglądanie tematów lekcji dziecka,
- sprawdzenie planu zajęć dziecka,
- przeglądanie zapowiedzianych dla dziecka sprawdzianów.

Administrator:

- dodawanie, usuwanie i edycja klas,
- archiwizowanie klas, zajęć i użytkowników,
- tworzenie i zmiana planów zajęć,
- przypisywanie nauczycieli do zajęć,
- dodawanie użytkowników,
- edycja danych użytkowników.

2.3 Wymagania niefunkcjonalne

- Użytkownicy mają dostęp do swoich funkcjonalności za pomocą aplikacji webowej.
- Każdy użytkownik posiada konto zabezpieczone unikalnym loginem i hasłem.
- Baza danych dopuszcza polskie znaki.

2.3.1 Wykorzystywane technologie i narzędzia

- Baza danych skonfigurowana i uruchomiona w środowisku MySQL
- Aplikacja dostępową napisana w języku Java 11
- Komunikacja pomiędzy bazą i aplikacją dostępową za pośrednictwem Java Hibernate
- Aplikacja webowa zaprojektowana z wykorzystaniem Java Spring
- Mockupy zaprojektowane przy użyciu aplikacji Figma
- Interfejs graficzny wykonany w oparciu o Vaadin

2.3.2 Wymagania dotyczące rozmiaru bazy danych

Na rozmiar bazy danych mają wpływ następujące parametry:

- dane są przechowywane przez okres 10 lat,
- rok szkolny ma około 190 dni,
- w szkole jest 16 klas,
- w każdej klasie jest 30 uczniów,
- dziennie uczniowie mają średnio 7 godzin lekcyjnych zajęć dydaktycznych,
- uczniowie otrzymują średnio 2 oceny tygodniowo,
- rada pedagogiczna składa się z 50 nauczycieli.

Szacując rozmiar bazy danych, bierzemy pod uwagę rozmiar tabel z największą liczbą rekordów. Będą to:

- obecności, liczba rekordów: $10 \cdot 190 \cdot 16 \cdot 30 \cdot 7 \cdot (4 + 4 + 1)B = 6'384'000 \cdot 9B = 57,5MB$
- tematy lekcji, liczba rekordów: $10 \cdot 190 \cdot 16 \cdot 7 \cdot (250 + 50)B = 212'800 \cdot 300B = 63,8MB$
- oceny, liczba rekordów: $10 \cdot 190 \div 5 \cdot 16 \cdot 30 \cdot (4 + 4 + 4 + 200)B = 182'400 \cdot 212B = 38,7MB$

Podsumowując, otrzymujemy 160 MB danych zawartych w najliczniejszych tabelach. Ze względu na dodatkowy rozmiar indeksów założonych na tabelach oraz niepełne wypełnienie stron dyskowych przez pliki bazy danych, zakładamy, że rozmiar bazy danych nie przekroczy 300 MB.

2.3.3 Wymagania dotyczące bezpieczeństwa systemu

- Dostęp do bazy danych dla użytkowników jest możliwy poprzez aplikację dostępową.
- Z aplikacji dostępowej użytkownicy mogą korzystać w każdym miejscu z dostępem do internetu.
- Aplikacja webowa komunikuje się z serwerem w sposób zaszyfrowany.
- Aplikacja dostępową jest dostępna publicznie.
- Aplikacja dostępową komunikuje się z serwerem, który jest uruchomiony na stacji bazowej znajdującej się w budynku szkoły.
- Szkoła posiada publiczny adres IP.

- Stacja bazowa serwera znajduje się w pomieszczeniu klimatyzowanym i chronionym.
- Stacja bazowa ma zapewnione awaryjne zasilanie.
- Serwer z bazą danych znajduje się w sieci lokalnej szkoły i nie ma do niego bezpośredniego dostępu z zewnątrz.
- Każdy użytkownik będzie posiadał login oraz tajne hasło, które w bazie będzie przechowywane w sposób zaszyfrowany.
- Baza danych jest zaszyfrowana i zabezpieczona hasłem.

2.4 Dodatkowe założenia projektowe

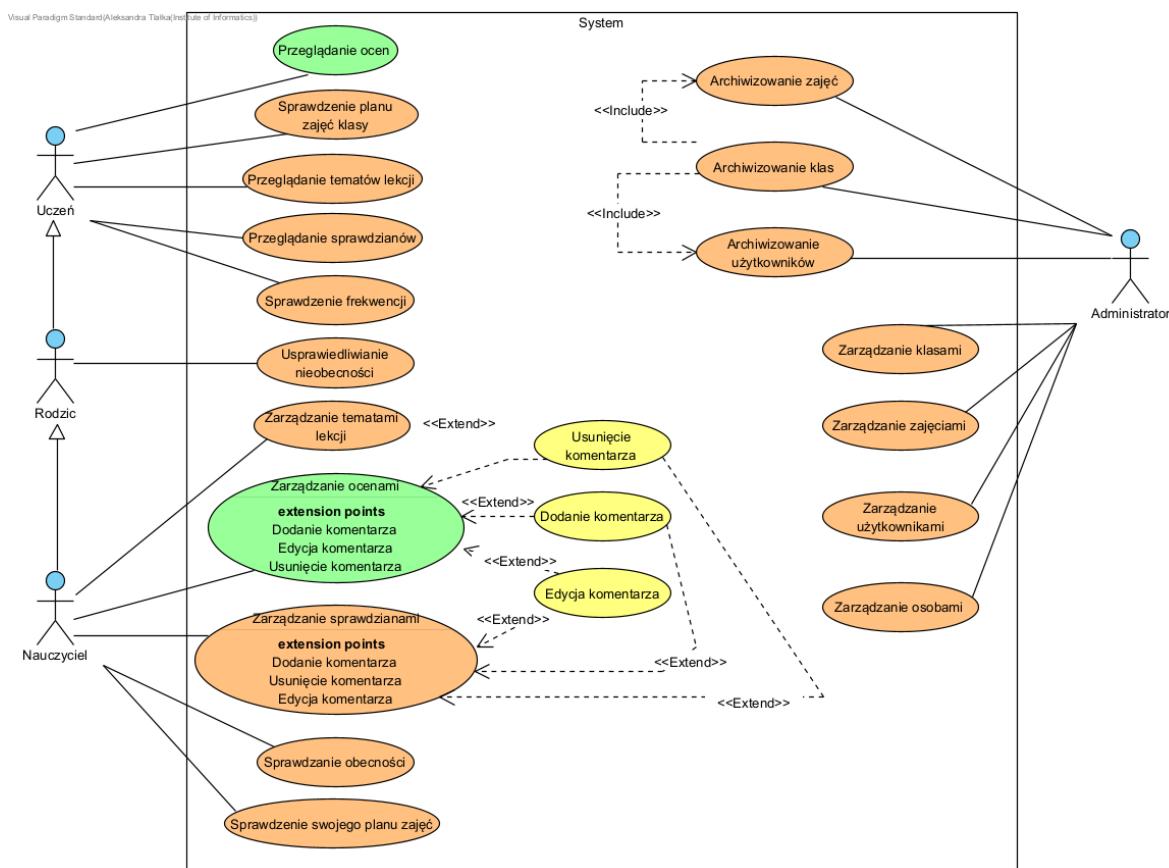
- Do danego dziecka przypisane jest tylko jedno konto rodzica, ale rodzic może mieć przypisanych kilkoro dzieci.
- Hasło do konta zna tylko jego właściciel.
- Archiwizujemy dane w okresie 10 lat.
- Aplikacja dostępową jest zaimplementowana z wykorzystaniem paradygmatów programowania obiektowego.
- Projekt jest otwarty na rozwój oraz nowe technologie.
- Baza danych posiada indeksy przyśpieszające wybrane operacje na danych.
- Baza jest zaprojektowana w taki sposób, aby nie występowała niepotrzebna redundancja danych.

3 Projekt bazy danych

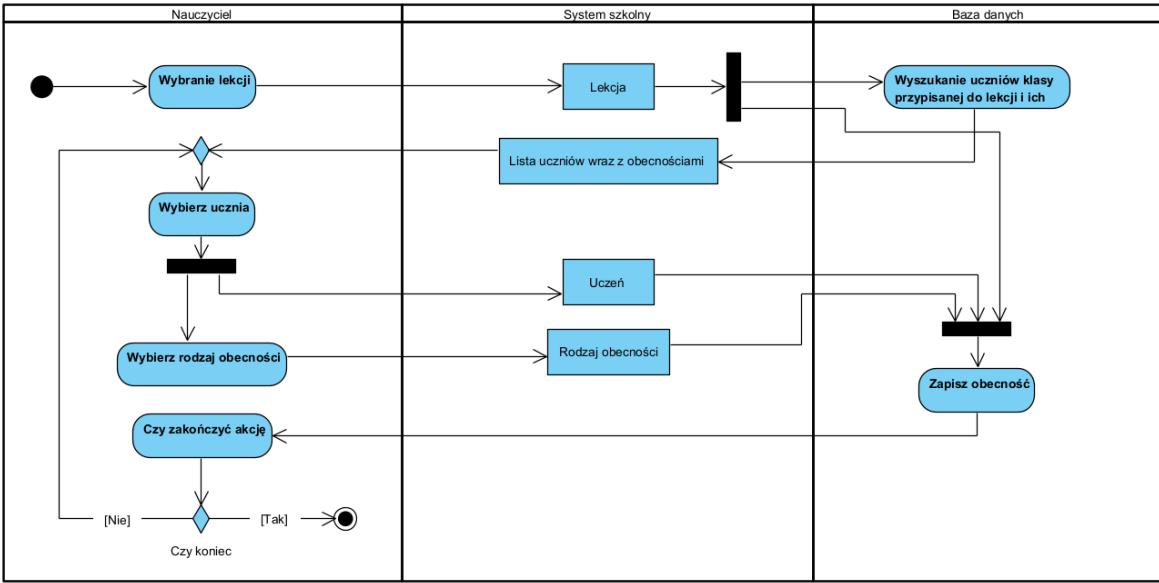
Przed pisaniem kodu SQL wykonaliśmy diagramy przypadków użycia i aktywności oraz model koncepcyjny, logiczny, a także fizyczny. Następnie rozplanowaliśmy działanie mechanizmów przetwarzania danych, które chcemy zaimplementować w naszej bazie. W projekcie przewidujemy triggery, procedury składowane oraz indeksy. Ostatnim krokiem etapu projektowania było stworzenie mockupów przedstawiających docelowy interfejs użytkownika.

3.1 Diagram przypadków użycia i diagram aktywności

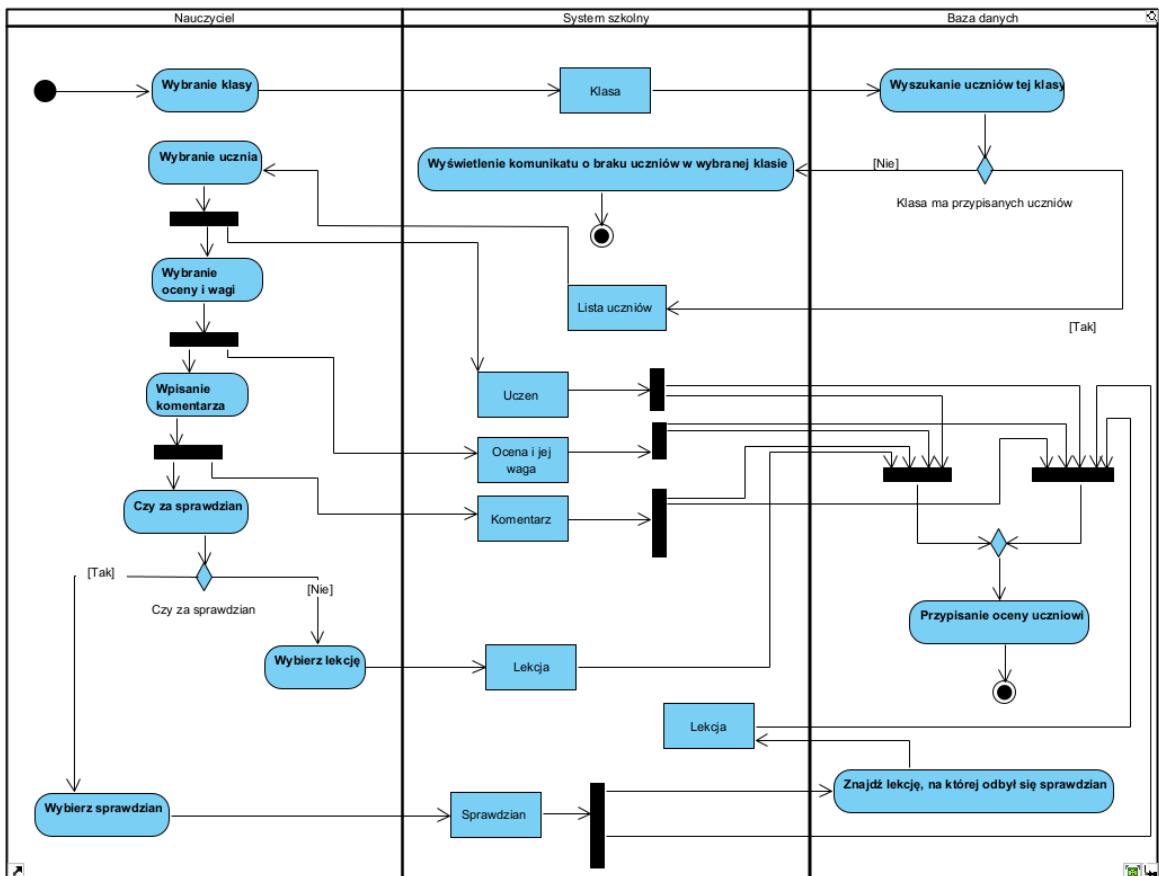
Na poniższym diagramie przypadków użycia prezentowane są wszystkie funkcjonalności gotowej aplikacji. Dla czytelności diagramu zebraliśmy wszystkie czynności dotyczące CRUD'a na danej encji w jednym przypadku użycia opisany zarządzanie. Poprzez CRUD rozumiemy tworzenie, odczyt, edycję oraz usunięcie obiektu (rekordu). Ze względu na ograniczenie czasowe nie zaimplementowaliśmy wszystkich funkcjonalności. Przypadki pokolorowane na zielone zostały zrealizowane całkowicie, a te pokolorowane na żółto - częściowo, w tym przypadku są one zrealizowane dla ocen, ale nie dla sprawdzianów.



Rysunek 1: Diagram przypadków użycia.

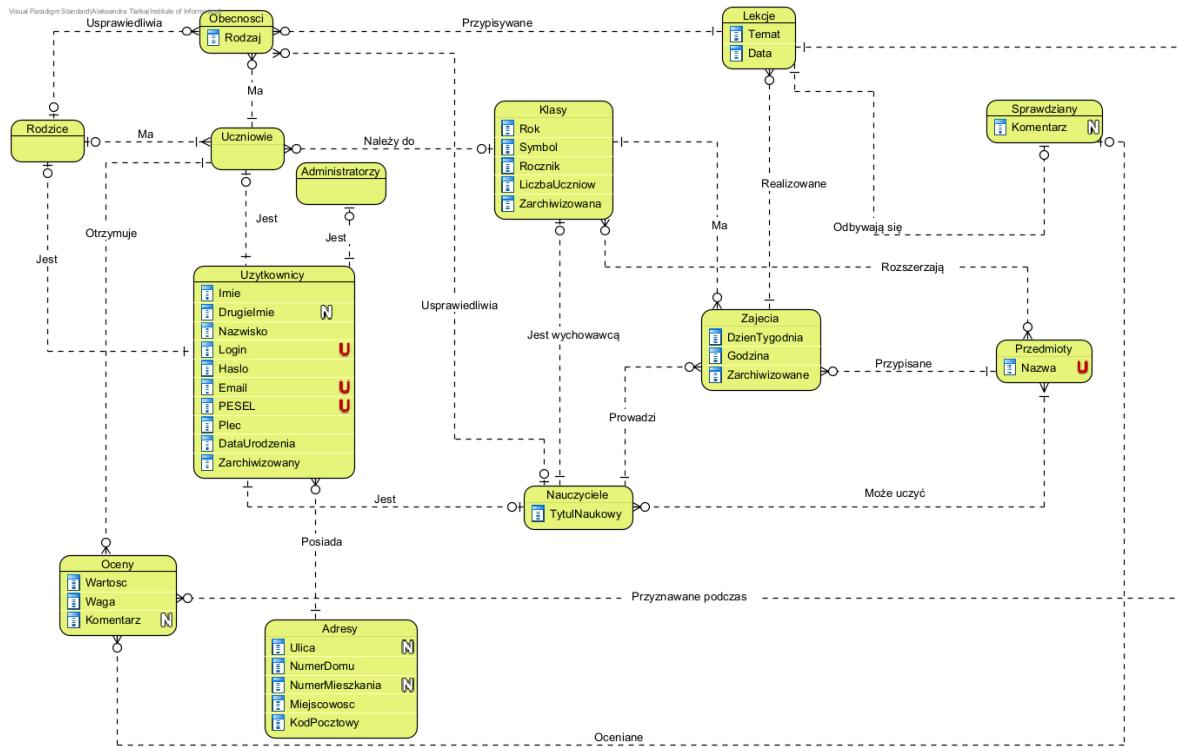


Rysunek 2: Diagram aktywności dla PU Sprawdzenie obecności

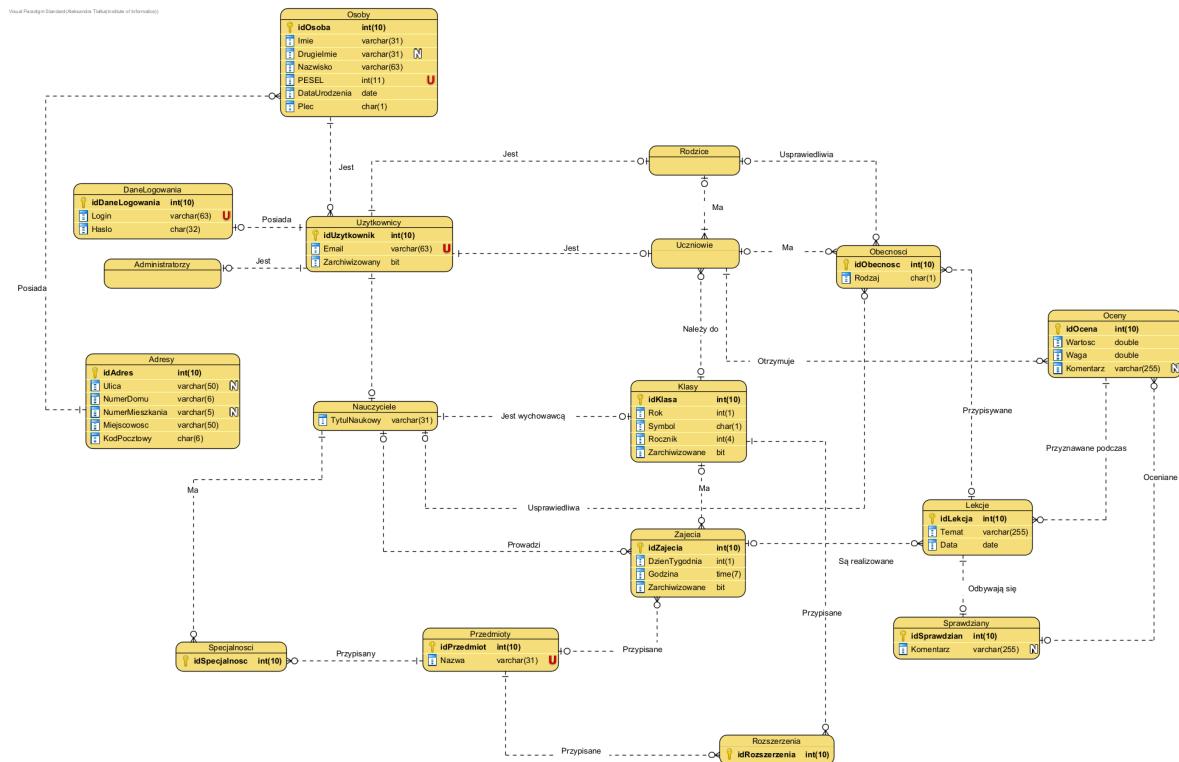


Rysunek 3: Diagram aktywności dla PU Wpisanie oceny

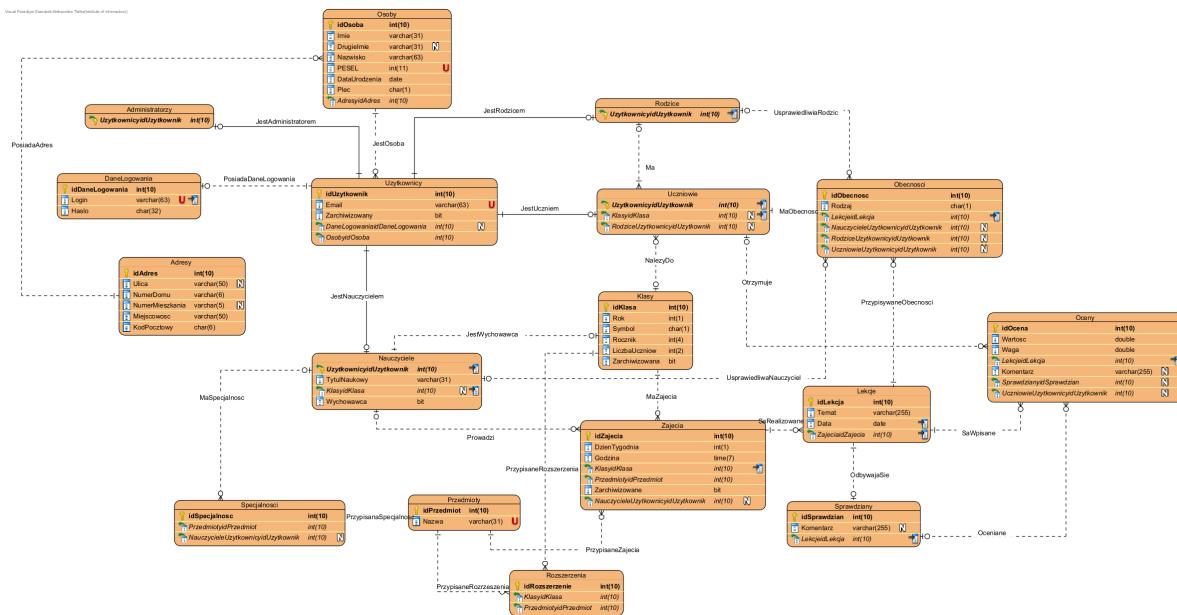
3.2 Model bazy danych



Rysunek 4: Model konceptualny



Rysunek 5: Model logiczny



Rysunek 6: Model fizyczny



Rysunek 7: Widoki

3.3 Indeksy

Tabela 1: Indeksy

Tabela	Indeks	Typ
Osoby	1. Imie, DrugieImie, Nazwisko	inc
Uzytkownicy	1. DaneLogowaniaidDaneLogowania	inc
DaneLogowania	1. Login	inc
Rodzice	1. UzytkownicyidUzytkownik	inc
Nauczyciele	1. UzytkownicyidUzytkownik	inc
Klasy	1. NauczycieleidNauczyciel	inc
Uczniowie	1. UzytkownicyidUzytkownik	inc
	2. RodziceidRodzic	inc
	3. KlasyidKlasa	inc
Lekcje	1. ZajeciaidZajecia	inc
	2. Data	inc
	3. Data	dec
Obecnosci	1. LekcjeidLekcja	dec
	2. UczniowieidUczen	inc
Oceny	1. UczniowieidUczen	inc
	2. LekcjeidLekcja	dec
Zajecia	1. NauczycieleidNauczyciel	inc
	2. KlasyidKlasa	inc
Sprawdziany	1. LekcjeidLekcja	dec

3.4 Procedury składowane

3.4.1 Wprowadzanie oceny

Parametry wejściowe:

- ocena
 - waga
 - idLekcja
 - idUczen
 - idSprawdzian
 - Komentarz
1. Nauczyciel wprowadza ocenę, wagę, wybiera lekcję i ucznia oraz ewentualnie sprawdzian i komentarz.
 2. Do tabeli "Oceny" dodawany jest rekord o podanych wartościach atrybutów, w atrybucie id lekcji wpisywany jest id lekcji tożsamy z wybraną lekcją, to samo dotyczy ucznia i sprawdzianu.

3.4.2 Usprawiedliwianie nieobecności

Parametry wejściowe:

- idLekcja
- idUczen
- idNauczyciel
- idRodzic

1. Rodzic bądź Nauczyciel wybiera ucznia oraz lekcję, na której jego nieobecność ma zostać usprawiedliwiona.
2. W rekordzie w tabeli "Obecnosci" o id ucznia tożsamym z id wybranego ucznia oraz id lekcji równej id wybranej lekcji następuje update rodzaju z "n" na "u".

3.4.3 Zapowiadanie sprawdzianu

Parametry wejściowe:

- idLekcja
 - Komentarz
1. Nauczyciel wybiera lekcję, na której ma się odbyć sprawdzian i wprowadza ewentualny komentarz do sprawdzianu.
 2. Do tabeli "Sprawdziany" dodawany jest rekord o id lekcji odpowiadającym wybranej lekcji oraz o wprowadzonym komentarzu.

3.4.4 Archiwizowanie klas

Parametry wejściowe:

- idKlasa
1. Administrator archiwizuje klasę poprzez wybranie jej.
 2. W rekordzie o id klasy zgodnym z wybraną klasą w tabeli "Klasy" wykonywany jest update i atrybut "Zarchizowana" ustawiany jest na wartość "1".
 3. We wszystkich rekordach w tabeli "Zajecia", gdzie id klasy jest tożsame z id wybranej klasy, robimy update i atrybut "Zarchiwizowane" ustawiany jest na "1".
 4. Z tabeli "Uczniowie" pobierane są wszystkie id użytkownika z rekordów, których id klasy było zgodne z wybraną klasą. Dla każdego z tych id, szukany jest taki rekord w tabeli "Uzytkownicy", gdzie id użytkownika jest tożsame ze znalezionym. Na takim rekordzie wykonywany jest update, w którym atrybut "Zarchiwizowany" ustawiany jest na "1".

3.4.5 Wpiswanie obecności

Parametry wejściowe:

- idLekcja
 - idUczen
 - Rodzaj obecności
1. Nauczyciel wybiera lekcję, ucznia oraz rodzaj obecności.
 2. Do tabeli "Obecnosci" dodawany jest nowy rekord, w którym id lekcji, id ucznia oraz rodzaj są tożsame z wybranymi.

3.4.6 Sprawdzenie zajęć nauczyciela

Parametry wejściowe:

- idNauczyciel
1. Nauczyciel podaje swoje id.
 2. Wyświetlane są wszystkie zajęcia, które prowadzi nauczyciel o podanym id, na które składa się nazwa przedmiotu, dzień tygodnia, godzina oraz klasa.

3.4.7 Sprawdzenie zajęć ucznia

Parametry wejściowe:

- idUczen
1. Uczeń podaje swoje id lub rodzic podaje idUcznia, który jest jego dzieckiem.
 2. Wyświetlane są wszystkie zajęcia, na które jest zapisany uczeń o podanym id. Wyświetlane są one w postaci nazwa przedmiotu, dzień tygodnia, godzina, przedmiot oraz nauczyciel, który je prowadzi.

3.4.8 Sprawdzenie ocen ucznia

Parametry wejściowe:

- idUczen
1. Uczeń podaje swoje id lub rodzic podaje idUcznia, który jest jego dzieckiem.
 2. Wyświetlane są wszystkie oceny, które otrzymał uczeń o podanym id. Wyświetlane są one w postaci ocena, waga, komentarz, przedmiot, data oraz nauczyciel, który wpisał ocenę.

3.4.9 Sprawdzenie ocen nauczyciela

Parametry wejściowe:

- idNauczyciel
1. Nauczyciel podaje swoje id.
 2. Wyświetlane są wszystkie oceny, które wpisał nauczyciel o podanym id. Wyświetlane są one w postaci nazwa przedmiotu, uczeń, ocena, waga, data oraz komentarz.

3.4.10 Wyświetl sprawdziany ucznia

Parametry wejściowe:

- idUczen
1. Uczeń podaje swoje id lub rodzic podaje idUcznia, który jest jego dzieckiem.
 2. Wyświetlane są wszystkie sprawdziany, które zostały zapowiadane dla ucznia o podanym id. Wyświetlane są one w postaci przedmiot, data, nauczyciel, który zapowiedział sprawdzian oraz komentarz do sprawdzianu, w kolejności malejącej po dacie.

3.4.11 Wyświetl sprawdziany nauczyciela

Parametry wejściowe:

- idNauczyciel
1. Nauczyciel podaje swoje id.
 2. Wyświetlane są wszystkie sprawdziany, które zostały zapowiadane przez nauczyciela o podanym id. Wyświetlane są one w postaci przedmiot, data, klasa oraz komentarz do sprawdzianu, w kolejności malejącej po dacie.

3.4.12 Wyświetl nieobecności ucznia

Parametry wejściowe:

- idUczen
1. idUcznia jest podawane przez ucznia, jego rodzica lub wychowawcę.
 2. Wyświetlane są nieobecności ucznia o podanym id. Wyświetlane są one w postaci data, przedmiot oraz rodzaj, w kolejności malejącej po dacie.

3.4.13 Wyświetl lekcje ucznia

Parametry wejściowe:

- idUczelnia

1. Uczeń podaje swoje id lub rodzic podaje idUcznia, który jest jego dzieckiem.
2. Wyświetlane są wszystkie lekcje, na których powinien być uczeń o podanym id. Wyświetlane są one w postaci temat, data, przedmiot, nauczyciel, w kolejności malejącej po dacie.

3.4.14 Wyświetl lekcje nauczyciela

Parametry wejściowe:

- idNauczyciel

1. Nauczyciel podaje swoje id.
2. Wyświetlane są wszystkie lekcje, które prowadził nauczyciel o podanym id. Wyświetlane są one w postaci klasa, temat, data oraz przedmiot, w kolejności malejącej po dacie.

3.5 Triggery

3.5.1 Limit uczniów na klasę

Wywoływany przed INSERT na tabeli Uczniowie.

Sprawdza, czy klasa, do której chcemy przypisać ucznia nie jest pełna, czyli czy liczba uczniów wynosi 30. Jeśli tak, to akcja jest przerwana.

3.5.2 Limit klas na rocznik

Wywoływany przed INSERT na tabeli Klasy.

Sprawdza liczbę klas na roczniku, który chcemy przypisać do klasy. Jeśli liczba wynosi 4, to akcja jest przerwana.

3.5.3 Czy wychowawca

Wywoływany przed INSERT i UPDATE na tabeli Klasy.

Sprawdza, czy nauczyciel, którego chcemy przypisać klasie jako wychowawcę, nie jest już wychowawcą innej klasy. Jeśli jest, to czynność zostaje przerwana.

3.5.4 Czy ocenić

Wywoływany przed INSERT i UPDATE na tabeli Oceny.

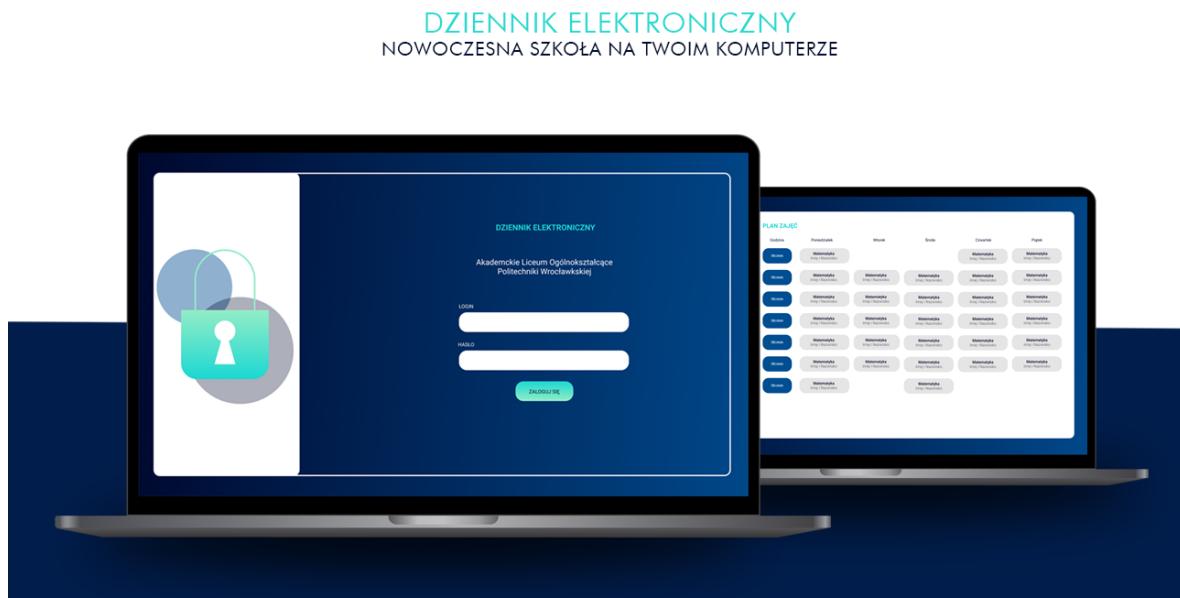
Sprawdza, czy uczeń, któremu chcemy wpisać ocenę z danej lekcji, był na niej obecny. Jeśli był to ocena zostaje wpisana, w przeciwnym wypadku działanie zostaje przerwane.

3.5.5 Czy nowy symbol

Wywoływany przed INSERT i UPDATE na tabeli Klasy.

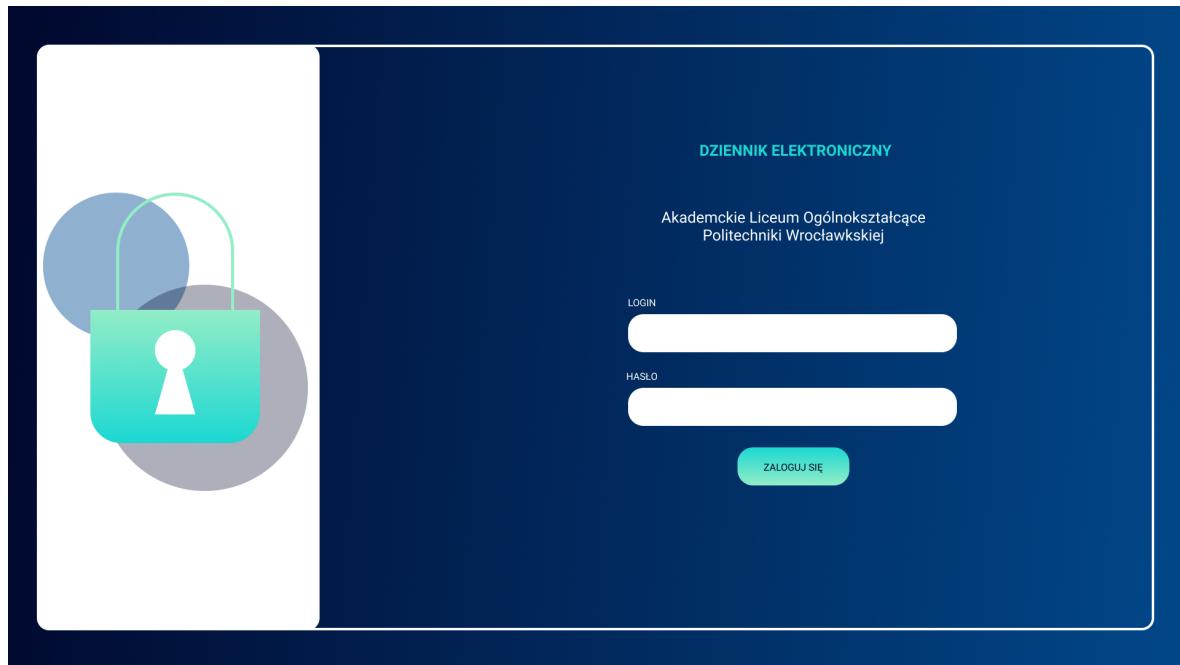
Sprawdza czy na danym roku nie ma jeszcze klasy o tym samym symbolu, w przeciwnym wypadku działanie zostaje przerwane.

3.6 Mockupy aplikacji webowej

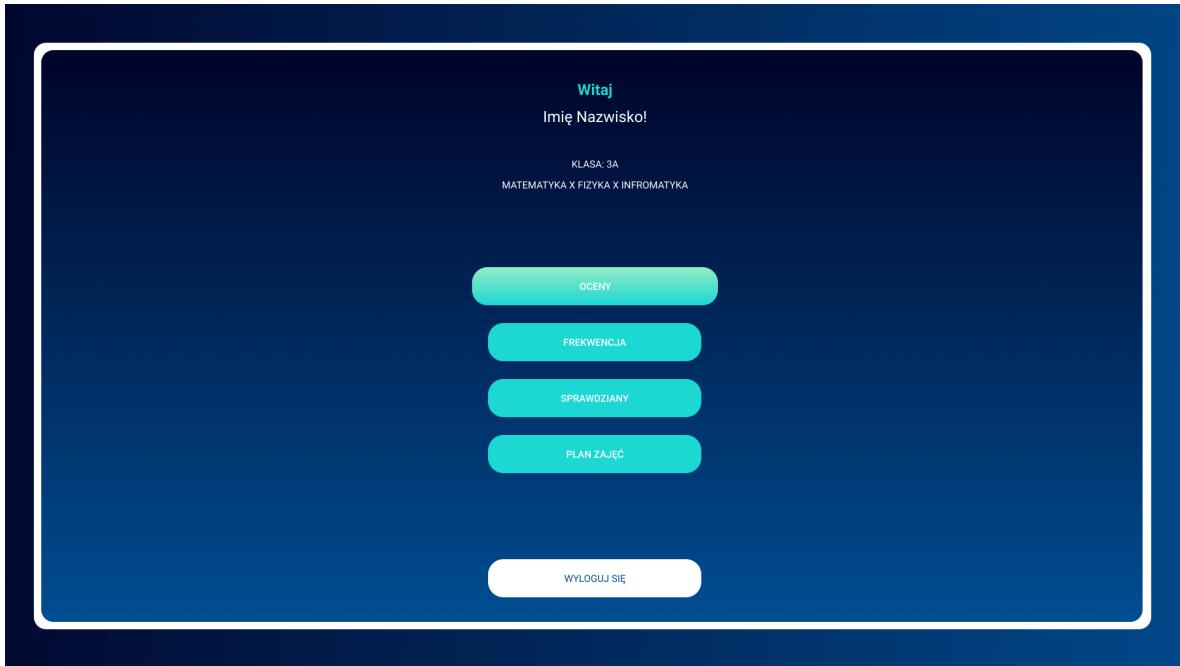


Rysunek 8: Dziennik Elektroniczny - Nowoczesna Szkoła Dzisiaj

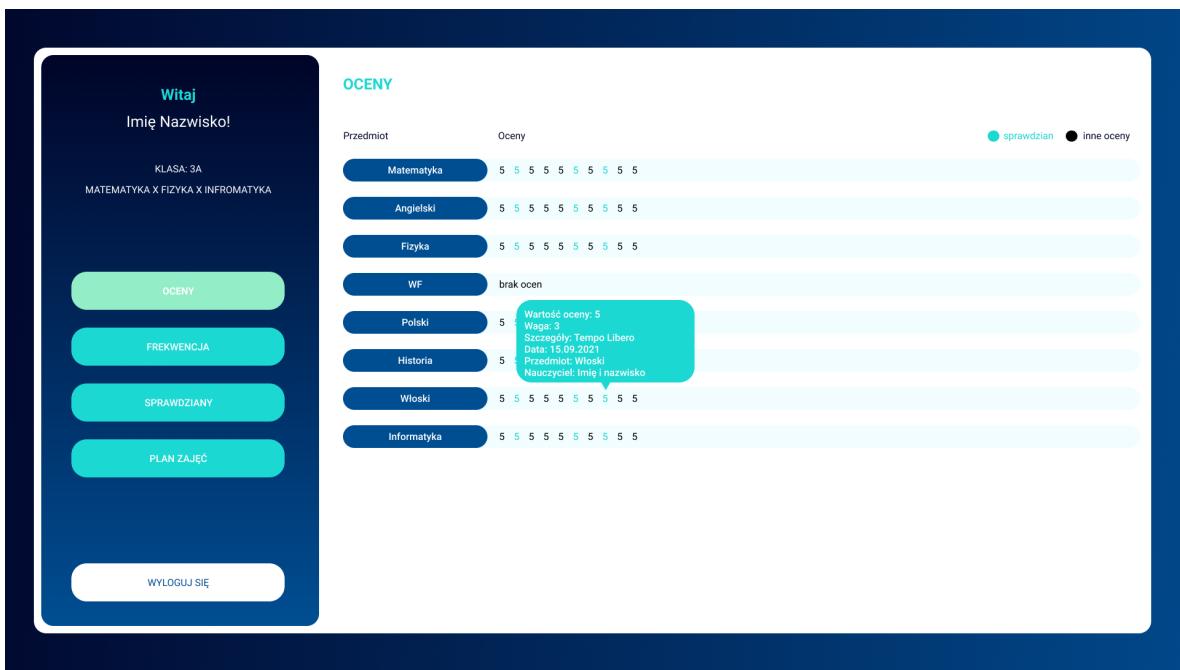
Klikalny Prototyp Aplikacji



Rysunek 9: Strona Logowania



Rysunek 10: Menu Głównego



Rysunek 11: Oceny

Rysunek 12: Frekwencja

Rysunek 13: Sprawdziany



Rysunek 14: Plan Zajęć

4 Implementacja systemu bazy danych

Baza danych została zaimplementowana w MySQLServer 8.0.

Dzięki stworzeniu modeli i diagramów w Visual Paradigm, część kodu mogliśmy wygenerować automatycznie - tabele oraz połączenie między nimi. Pozostałą część pisaliśmy samodzielnie - widoki, triggery, procedury składowane oraz indeksy, ponieważ pomimo zadeklarowania ich w Visual Paradigm, wygenerowany kod nie spełniał norm składniowych MySQL. Musieliśmy także poprawić constrainty w tabelach, gdyż nie zostały one wygenerowane wraz z tabelami.

Dane testowe były generowane na 2 sposoby:

1. generacja za pomocą ogólnodostępnych generatorów internetowych na powszechnie dane, których liczebność nie była duża - imiona, nazwiska, pesel
2. generacja za pomocą napisanego przez nas programu w języku Python - szczególnie potrzebna przy przeprowadzaniu testów wydajnościowych, gdyż potrzebowaliśmy 600 tysięcy rekordów

4.1 Tworzenie tabel i definiowanie ograniczeń

4.1.1 Tworzenie tabel

Tabele zostały utworzone na podstawie założeń przedstawionych na diagramie fizycznym bazy danych. Poniżej przedstawiono zapis utworzonych przez nas tabel.

Listing 1: Skrypt tworzący tabelę

```
CREATE TABLE Uzytkownicy (idUzytkownik int(10) NOT NULL AUTO_INCREMENT, Email varchar(63) NOT
→ NULL UNIQUE, Zarchiwizowany bit(1) DEFAULT 0 NOT NULL, DaneLogowaniaidDaneLogowania int(10),
→ OsobyidOsoba int(10) NOT NULL, PRIMARY KEY (idUzytkownik)) CHARACTER SET UTF8;

CREATE TABLE Uczniowie (idUzytkownik int(10) NOT NULL, KlasyidKlasa int(10), RodziceidRodzic int(10),
→ PRIMARY KEY (idUzytkownik)) CHARACTER SET UTF8;

CREATE TABLE Rodzice (idUzytkownik int(10) NOT NULL, PRIMARY KEY (idUzytkownik)) CHARACTER
→ SET UTF8;

CREATE TABLE Nauczyciele (idUzytkownik int(10) NOT NULL, TytulNaukowy varchar(31) NOT NULL,
→ PRIMARY KEY (idUzytkownik)) CHARACTER SET UTF8;

CREATE TABLE Obecnosci (idObecnosc int(10) NOT NULL AUTO_INCREMENT, UczniowieidUczten int(10) NOT
→ NULL, LekcjeidLekcja int(10) NOT NULL, Rodzaj char(1) NOT NULL, RodziceidRodzic int(10),
→ NauczycieleidNauczyciel int(10), PRIMARY KEY (idObecnosc), CHECK (Rodzaj IN ('N', 'Z', 'U')),
→ tempUczten int, tempLekcja int) CHARACTER SET UTF8;

CREATE TABLE Klasa (idKlasa int(10) NOT NULL AUTO_INCREMENT, NauczycieleidNauczyciel int(10) NOT
→ NULL, Rok int(1) NOT NULL, Symbol char(1) NOT NULL, Rocznik int(4) NOT NULL, Zarchiwizowana
→ bit(1) DEFAULT 0 NOT NULL, PRIMARY KEY (idKlasa), INDEX (NauczycieleidNauczyciel), CHECK (
→ Rok > 0 AND Rok < 5)) CHARACTER SET UTF8;

CREATE TABLE Lekcja (idLekcja int(10) NOT NULL AUTO_INCREMENT, Data date NOT NULL,
→ ZajeciaidZajecia int(10) NOT NULL, Temat varchar(255) NOT NULL, PRIMARY KEY (idLekcja))
→ CHARACTER SET UTF8;

CREATE TABLE Sprawdziany (idSprawdzian int(10) NOT NULL AUTO_INCREMENT, LekcjeidLekcja int(10)
→ NOT NULL, Komentarz varchar(255), PRIMARY KEY (idSprawdzian)) CHARACTER SET UTF8;

CREATE TABLE Zajecia (idZajecia int(10) NOT NULL AUTO_INCREMENT, KlasaidKlasa int(10) NOT NULL,
→ NauczycieleidNauczyciel int(10) NOT NULL, DzienTygodnia int(1) NOT NULL, Godzina time(6) NOT
→ NULL, PrzedmiotidPrzedmiot int(10) NOT NULL, Zarchiwizowane bit(1) DEFAULT 0 NOT NULL,
→ PRIMARY KEY (idZajecia)) CHARACTER SET UTF8;

CREATE TABLE Przedmioty (idPrzedmiot int(10) NOT NULL AUTO_INCREMENT, Nazwa varchar(31) NOT
→ NULL UNIQUE, PRIMARY KEY (idPrzedmiot)) CHARACTER SET UTF8;

CREATE TABLE Administratorzy (idUzytkownik int(10) NOT NULL , PRIMARY KEY (idUzytkownik))
→ CHARACTER SET UTF8;
```

```

CREATE TABLE Oceny (idOcena int(10) NOT NULL AUTO_INCREMENT, UczniowieidUczen int(10) NOT NULL
    ↪ , LekcjeidLekcja int(10) NOT NULL, Wartosc double NOT NULL, Waga double NOT NULL, Komentarz
    ↪ varchar(255), SprawdzianyidSprawdzian int(10), PRIMARY KEY (idOcena), CHECK (Wartosc >= 1 AND
    ↪ Wartosc <= 6)) CHARACTER SET UTF8;

CREATE TABLE Adresy (idAdres int(10) NOT NULL AUTO_INCREMENT, Ulica varchar(50), NumerDomu
    ↪ varchar(6) NOT NULL, NumerMieszkania varchar(5), Miejscowosc varchar(50) NOT NULL, KodPocztowy
    ↪ char(6) NOT NULL, PRIMARY KEY (idAdres)) CHARACTER SET UTF8;

CREATE TABLE Specjalnosci (idSpecjalnosc int(10) NOT NULL AUTO_INCREMENT, PrzedmiotyidPrzedmiot int
    ↪ (10) NOT NULL, NauczycieleidNauczyciel int(10) NOT NULL, PRIMARY KEY (idSpecjalnosc))
    ↪ CHARACTER SET UTF8;

CREATE TABLE Rozszerzenia (idRozszerzenie int(10) NOT NULL AUTO_INCREMENT, KlasyidKlasa int(10)
    ↪ NOT NULL, PrzedmiotyidPrzedmiot int(10) NOT NULL, PRIMARY KEY (idRozszerzenie)) CHARACTER
    ↪ SET UTF8;

CREATE TABLE DaneLogowania (idDaneLogowania int(10) NOT NULL AUTO_INCREMENT, Login varchar(63)
    ↪ NOT NULL UNIQUE, Haslo char(64) NOT NULL, PRIMARY KEY (idDaneLogowania)) CHARACTER
    ↪ SET UTF8;

CREATE TABLE Osoby (idOsoba int(10) NOT NULL AUTO_INCREMENT, Imie varchar(31) NOT NULL,
    ↪ DrugieImie varchar(31), Nazwisko varchar(63) NOT NULL, PESEL bigint NOT NULL UNIQUE,
    ↪ DataUrodzenia date NOT NULL, Plec char(1) NOT NULL, AdresyidAdres int(10) NOT NULL, PRIMARY
    ↪ KEY (idOsoba), CHECK (Plec IN ('K','M') AND Pesel > 9999999999 AND Pesel < 100000000000))
    ↪ CHARACTER SET UTF8;

```

4.1.2 Tworzenie ograniczeń

Ograniczenia zostały również wygenerowane przez program Visual Paradigm, w którym wcześniej je zdefiniowaliśmy.

Listing 2: Skrypt tworzący połączenia między encjami

```

ALTER TABLE Uczniowie ADD CONSTRAINT Ma FOREIGN KEY (RodziceidRodzic) REFERENCES Rodzice (
    ↪ idUzytkownika);

ALTER TABLE Uczniowie ADD CONSTRAINT NalezyDo FOREIGN KEY (KlasyidKlasa) REFERENCES Klasy (
    ↪ idKlasa);

ALTER TABLE Zajecia ADD CONSTRAINT PrzypisaneZajecia FOREIGN KEY (PrzedmiotyidPrzedmiot)
    ↪ REFERENCES Przedmioty (idPrzedmiot);

ALTER TABLE Zajecia ADD CONSTRAINT MaZajecia FOREIGN KEY (KlasyidKlasa) REFERENCES Klasy (
    ↪ idKlasa);

ALTER TABLE Zajecia ADD CONSTRAINT Prowadzi FOREIGN KEY (NauczycieleidNauczyciel) REFERENCES
    ↪ Nauczyciele (idUzytkownika);

ALTER TABLE Obecnosci ADD CONSTRAINT PrzypisywaneObecnosci FOREIGN KEY (LekcjeidLekcja)
    ↪ REFERENCES Lekcje (idLekcja);

ALTER TABLE Lekcje ADD CONSTRAINT SaRealizowane FOREIGN KEY (ZajeciaidZajecia) REFERENCES
    ↪ Zajecia (idZajecia);

ALTER TABLE Obecnosci ADD CONSTRAINT MaObecnosc FOREIGN KEY (UczniowieidUczen) REFERENCES
    ↪ Uczniowie (idUzytkownika);

ALTER TABLE Oceny ADD CONSTRAINT Otrzymuje FOREIGN KEY (UczniowieidUczen) REFERENCES
    ↪ Uczniowie (idUzytkownika);

ALTER TABLE Specjalnosci ADD CONSTRAINT PrzypisanaSpecjalnosc FOREIGN KEY (PrzedmiotyidPrzedmiot)
    ↪ REFERENCES Przedmioty (idPrzedmiot);

ALTER TABLE Specjalnosci ADD CONSTRAINT MaSpecjalnosc FOREIGN KEY (NauczycieleidNauczyciel)
    ↪ REFERENCES Nauczyciele (idUzytkownika);

ALTER TABLE Obecnosci ADD CONSTRAINT UsprawiedliwiaRodzic FOREIGN KEY (RodziceidRodzic)
    ↪ REFERENCES Rodzice (idUzytkownika);

```

```

ALTER TABLE Obecnosci ADD CONSTRAINT UsprawiedliwaNauczyciel FOREIGN KEY (NauczycieleidNauczyciel
    ↵ ) REFERENCES Nauczyciele (idUzytkownik);

ALTER TABLE Sprawdziany ADD CONSTRAINT OdbywajaSie FOREIGN KEY (LekcjeidLekcja) REFERENCES
    ↵ Lekcje (idLekcja);

ALTER TABLE Oceny ADD CONSTRAINT SaWpisane FOREIGN KEY (LekcjeidLekcja) REFERENCES Lekcje (
    ↵ idLekcja);

ALTER TABLE Rodzice ADD CONSTRAINT JestRodzicem FOREIGN KEY (idUzytkownik) REFERENCES
    ↵ Uzytkownicy (idUzytkownik);

ALTER TABLE Administratorzy ADD CONSTRAINT JestAdministratorem FOREIGN KEY (idUzytkownik)
    ↵ REFERENCES Uzytkownicy (idUzytkownik);

ALTER TABLE Nauczyciele ADD CONSTRAINT JestNauczycielem FOREIGN KEY (idUzytkownik)
    ↵ REFERENCES Uzytkownicy (idUzytkownik);

ALTER TABLE Uczniowie ADD CONSTRAINT JestUczniem FOREIGN KEY (idUzytkownik) REFERENCES
    ↵ Uzytkownicy (idUzytkownik);

ALTER TABLE Oceny ADD CONSTRAINT Oceniane FOREIGN KEY (SprawdzanyidSprawdzian) REFERENCES
    ↵ Sprawdziany (idSprawdzian);

ALTER TABLE Rozszerzenia ADD CONSTRAINT PrzypisaneRozszerzenia FOREIGN KEY (KlasyidKlasa)
    ↵ REFERENCES Klasy (idKlasa);

ALTER TABLE Rozszerzenia ADD CONSTRAINT PrzypisaneRozrzeszenia FOREIGN KEY (
    ↵ PrzedmiotyidPrzedmiot) REFERENCES Przedmioty (idPrzedmiot);

ALTER TABLE Uzytkownicy ADD CONSTRAINT PosiadaDaneLogowania FOREIGN KEY (
    ↵ DaneLogowaniaidDaneLogowania) REFERENCES DaneLogowania (idDaneLogowania);

ALTER TABLE Uzytkownicy ADD CONSTRAINT JestOsoba FOREIGN KEY (OsobyidOsoba) REFERENCES
    ↵ Osoby (idOsoba);

ALTER TABLE Osoby ADD CONSTRAINT PosiadaAdres FOREIGN KEY (AdresyidAdres) REFERENCES Adresy
    ↵ (idAdres);

ALTER TABLE Klasy ADD CONSTRAINT JestWychowawca FOREIGN KEY (NauczycieleidNauczyciel)
    ↵ REFERENCES Nauczyciele (idUzytkownik);

```

4.2 Implementacja mechanizmów przetwarzania danych

4.2.1 Indeksy

Poniżej znajduje się implementacja indeksów przez nas zdefiniowanych. Indeksy implementowaliśmy w oparciu o przypadki użycia oraz na podstawie informacji, po których atrybutach będzie realizowane wyszukiwanie rekordów.

Listing 3: Skrypt tworzący indeksy

```

CREATE INDEX OsobyIdx ON Osoby (Imie ASC, DrugieImie ASC, Nazwisko ASC);

CREATE INDEX UzytkownicyIdx ON Uzytkownicy (DaneLogowaniaidDaneLogowania ASC);

CREATE INDEX DaneLogowaniaIdx ON DaneLogowania (Login ASC);

CREATE INDEX KlasyIdx ON Klasy (NauczycieleidNauczyciel ASC);

CREATE INDEX UczniowieRodzicIdx ON Uczniowie (RodziceidRodzic ASC);

CREATE INDEX UczniowieKlasyIdx ON Uczniowie (KlasyidKlasa ASC);

CREATE INDEX LekcjeZajecialIdx ON Lekcje (ZajeciaidZajecia ASC);

CREATE INDEX LekcjeDataAscIdx ON Lekcje (Data ASC);

CREATE INDEX LekcjeDataDescIdx ON Lekcje (Data DESC);

```

```

CREATE INDEX ObecnosciLekcjaIdx ON Obecnosci (LekcjeidLekcja DESC);
CREATE INDEX ObecnosciUczenIdx ON Obecnosci (UczniowieidUczen ASC);
CREATE INDEX OcenyUczenIdx ON Oceny (UczniowieidUczen ASC);
CREATE INDEX OcenyLekcjaIdx ON Oceny (LekcjeidLekcja ASC);
CREATE INDEX ZajeciaNauczycielIdx ON Zajecia (NauczycieleidNauczyciel ASC);
CREATE INDEX ZajeciaKlasaIdx ON Zajecia (KlasyidKlasa ASC);
CREATE INDEX SprawdzianyLekcja ON Sprawdziany (LekcjeidLekcja ASC);

```

4.2.2 Widoki

W naszej bazie danych przewidujemy 3 widoki, które zostały zaimplementowane w następujący sposób:

Listing 4: Skrypt tworzący widoki

```

DROP VIEW if exists ListaNauczycieli;
CREATE VIEW ListaNauczycieli
AS
SELECT
    DISTINCT CONCAT(CONCAT(o.imie, ','), o.nazwisko) "Nauczyciel", p.Nazwa "Przedmiot",
    (SELECT COUNT(idZajecia) FROM Zajecia z WHERE z.NauczycieleidNauczyciel = n.idUzytkownik AND z.
     ↪ PrzedmiotyidPrzedmiot = p.idPrzedmiot) "Liczba_godzin_w_tygodniu", o.Plec "Plec"
FROM
    Zajecia z
JOIN
    Przedmioty p ON p.idPrzedmiot = z.PrzedmiotyidPrzedmiot
JOIN
    Nauczyciele n ON n.idUzytkownik = z.NauczycieleidNauczyciel
JOIN
    Uzytkownicy u ON u.idUzytkownik = n.idUzytkownik
JOIN
    Osoby o ON o.idOsoba = u.OsobyidOsoba;

DROP VIEW if exists ListaKlas;
CREATE VIEW ListaKlas
AS
SELECT
    CONCAT(k.rok, k.symbol) "Klasa", k.rocznik "Rocznik"
FROM
    Klasa k;

DROP VIEW if exists ListaUczniow;
CREATE VIEW ListaUczniow
AS
SELECT
    CONCAT(CONCAT(o.imie, ','), o.nazwisko) "Uczen", CONCAT(k.rok, k.symbol) "Klasa", k.rocznik "Rocznik"
FROM
    Klasa k
JOIN
    Uczniowie d ON k.idKlasa = d.KlasyidKlasa
JOIN
    Uzytkownicy u ON u.idUzytkownik = d.idUzytkownik
JOIN
    Osoby o ON o.idOsoba = u.OsobyidOsoba;

```

4.2.3 Triggery

W naszym systemie posiadamy 8 triggerów. Przykładowe z nich zapisane w MySQL:

Listing 5: Fragment skryptu odpowiedzialnego za tworzenie triggerów

```

DELIMITER /
CREATE TRIGGER CzyOcenicUpdate
BEFORE UPDATE ON Oceny
FOR EACH ROW

```

```

BEGIN
IF (NOT((SELECT Rodzaj
      FROM Obecnosci
     WHERE LekcjeidLekcja = NEW.LekcjeidLekcja AND UczniowieidUczen = NEW.UczniowieidUczen) LIKE 'o' )
    → ) THEN
signal sqlstate '45000' set message_text ='Uczennie_mozeczytac_oceny,_poniewaz_nie_byl_jest_wtedy_obecny.';
END IF;
END;/
DELIMITER ;

DELIMITER /
CREATE TRIGGER CzyWychowawcaUpdate
BEFORE UPDATE ON Klasy
FOR EACH ROW
BEGIN
IF NEW.NauczycieleidNauczyciel IN (SELECT NauczycieleidNauczyciel
      FROM Klasy) THEN
signal sqlstate '45000' set message_text ='Ten_nauczyciel_jest_juz_wychowawca.';
END IF;
END;/
DELIMITER ;

```

4.2.4 Procedury składowane

W naszym systemie przewidzieliśmy, zaprojektowaliśmy i zaimplementowaliśmy 14 procedur składowanych. Przykładowe z nich:

Listing 6: Fragment skryptu odpowiedzialnego za tworzenie procedur składowanych

```

DROP procedure IF EXISTS `archiwizowanieKlasy`;
DELIMITER $$
USE `dziennikszkolny`$$
CREATE PROCEDURE `archiwizowanieKlasy` (idKlasa integer)
BEGIN
    UPDATE Klasy
    SET Zarchiwizowana = 1
    WHERE Klasy.idKlasa = idKlasa;

    UPDATE Uzytkownicy
    SET Zarchiwizowany = 1
    WHERE idUzytkownik IN
        (SELECT idUzytkownik FROM Uczniowie
        WHERE KlasyidKlasa = idKlasa);

    UPDATE Zajecia
    SET Zarchiwizowane = 1
    WHERE KlasyidKlasa = idKlasa;
END$$
DELIMITER ;

DROP procedure IF EXISTS `wprowadzOcene`;
DELIMITER $$
USE `dziennikszkolny`$$
CREATE PROCEDURE `wprowadzOcene` (ocena integer, waga double, idLekcja integer, idUczen integer,
    → idSprawdzian integer, komentarz varchar(255))
BEGIN
    INSERT INTO Oceny (Wartosc, Waga, UczniowieidUczen, LekcjeidLekcja, SprawdzianyidSprawdzian,
    → Komentarz) values (ocena, waga, idUczen, idSprawdzian, komentarz);
END$$
DELIMITER ;

```

4.3 Implementacja uprawnień i innych zabezpieczeń

Skrypt tworzący użytkowników oraz przypisujący im odpowiednie uprawnienia napisaliśmy sami, gdyż nie było efektywnego sposobu na wygenerowanie go.

4.3.1 Użytkownicy

Zgodnie z początkowym założeniem w aplikacji wyróżniamy 4 rodzaje użytkowników - uczniów, rodziców, nauczycieli oraz administratorów. Użytkownicy zostali zaimplementowani w następujący sposób:

Listing 7: Skrypt tworzący użytkowników

```
CREATE USER IF NOT EXISTS Uczen@localhost  
IDENTIFIED BY '1uczen1';  
  
CREATE USER IF NOT EXISTS Rodzic@localhost  
IDENTIFIED BY '2rodzic2';  
  
CREATE USER IF NOT EXISTS Nauczyciel@localhost  
IDENTIFIED BY '3nauczyciel3';  
  
CREATE USER IF NOT EXISTS Administrator@localhost  
IDENTIFIED BY '4admin4';
```

4.3.2 Uprawnienia użytkowników

Dla bezpieczeństwa bazy danych do użytkowników zostały przypisane uprawnienia zgodnie z początkowymi założeniami.

Uprawnienia przypisane uczniowi:

Listing 8: Skrypt przypisujący uprawnienia (granty) uczniowi

```
GRANT EXECUTE  
ON PROCEDURE wyswietlNieobecnosci  
TO Uczen@localhost;  
  
GRANT EXECUTE  
ON PROCEDURE sprawdzenieZajecUcznia  
TO Uczen@localhost;  
  
GRANT EXECUTE  
ON PROCEDURE wyswietlSprawdzianyUcznia  
TO Uczen@localhost;  
  
GRANT EXECUTE  
ON PROCEDURE wyswietlLekcjeUcznia  
TO Uczen@localhost;  
  
GRANT EXECUTE  
ON PROCEDURE sprawdzOcenyUcznia  
TO Uczen@localhost;
```

Rodzic posiada wszystkie uprawnienia ucznia oraz dodatkowo jedno uprawnienie:

Listing 9: Skrypt przypisujący dodatkowe uprawnienie (grant) rodziciowi

```
GRANT EXECUTE  
ON PROCEDURE Usprawiedliwienie  
TO Rodzic@localhost;
```

Nauczyciel ma dosyć dużo uprawnień, m.in ze względu na wpisywanie sprawdzianów, tematów lekcji itd., dlatego poniżej znajdują się tylko przykładowe granty.

Listing 10: Fragment skryptu przypisującego uprawnienia (granty) nauczycielowi

```
GRANT SELECT  
ON dziennikszkolny.listaucznior  
TO Nauczyciel@localhost;  
  
GRANT UPDATE  
ON dziennikszkolny.obecnosci  
TO Nauczyciel@localhost;  
  
GRANT UPDATE  
ON dziennikszkolny.Oceny  
TO Nauczyciel@localhost;
```

```
GRANT DELETE  
ON dziennikszkolny.Oceny  
TO Nauczyciel@localhost;
```

```
GRANT EXECUTE  
ON PROCEDURE Usprawiedliwienie  
TO Nauczyciel@localhost;
```

```
GRANT EXECUTE  
ON PROCEDURE zapowiadanieSprawdzianu  
TO Nauczyciel@localhost;
```

Administrator jest użytkownikiem, który ma najwięcej uprawnień, ponieważ on zarządza bazą danych - tworzy klasy, dodaje osoby itd, dlatego, tak jak w przypadku nauczyciela, w dokumentacji zamieszczamy tylko przykładowe uprawnienia.

Listing 11: Fragment skryptu przypisującego uprawnienia (granty) administratorowi

```
GRANT EXECUTE  
ON PROCEDURE archiwizowanieKlasy  
TO Administrator@localhost;
```

```
GRANT INSERT  
ON dziennikszkolny.uzytkownicy  
TO Administrator@localhost;
```

```
GRANT UPDATE  
ON dziennikszkolny.uzytkownicy  
TO Administrator@localhost;
```

```
GRANT DELETE  
ON dziennikszkolny.uzytkownicy  
TO Administrator@localhost;
```

```
GRANT INSERT  
ON dziennikszkolny.uczniowie  
TO Administrator@localhost;
```

```
GRANT UPDATE  
ON dziennikszkolny.uczniowie  
TO Administrator@localhost;
```

```
GRANT DELETE  
ON dziennikszkolny.uczniowie  
TO Administrator@localhost;
```

4.4 Testowanie bazy danych na przykładowych danych

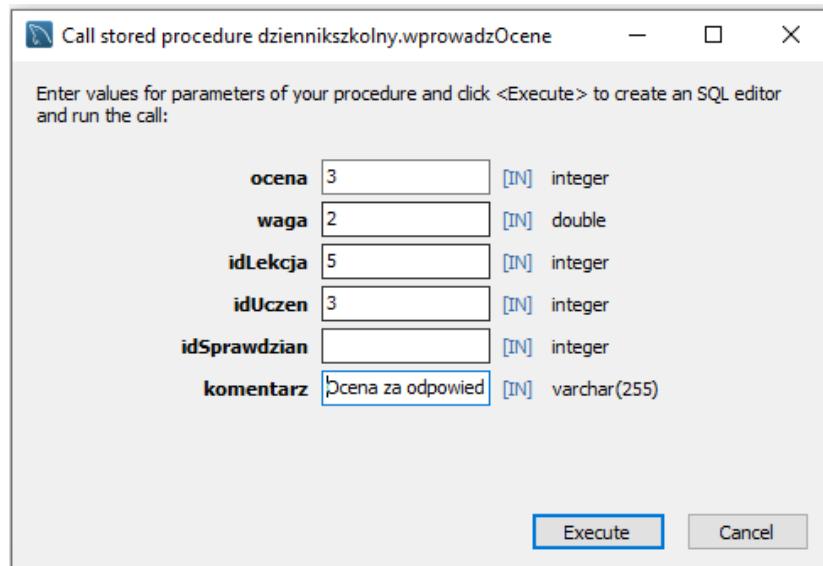
4.4.1 Procedura składowana

Zdecydowano się przetestować procedurę umożliwiającą wprowadzania nowej oceny.

The screenshot shows a database interface with a query editor at the top containing the SQL command: `1 • SELECT * FROM dziennikszkolny.oceny;`. Below the editor is a "Result Grid" table with the following data:

	idOcena	UczniowieidUczeń	LekcjeidLekcja	Wartosc	Waga	Komentarz	SprawdzanyidSprawdzian
▶	1	1	3	5	3	Nunc ac blandit augue.	1
2	2	2	3	4	3	Nunc ac blandit augue.	1
3	3	3	3	5	3	Nunc ac blandit augue.	1
4	4	4	3	3	3	Nunc ac blandit augue.	1
5	5	5	3	5	3	Nunc ac blandit augue.	1
6	6	6	3	6	3	Nunc ac blandit augue.	1
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Rysunek 15: Wszystkie oceny przed wywołaniem procedury dodawania



Rysunek 16: Formularz wywołujący procedurę wprowadzania nowej oceny

	idOcena	UczniowieidUczeln	LekcjeidLekcja	Wartosc	Waga	Komentarz	SprawdzanyidSprawdzian
▶	1	1	3	5	3	Nunc ac blandit augue.	1
2	2	3	4	3	3	Nunc ac blandit augue.	1
3	3	3	5	3	3	Nunc ac blandit augue.	1
4	4	3	3	3	3	Nunc ac blandit augue.	1
5	5	3	5	3	3	Nunc ac blandit augue.	1
6	6	3	6	3	3	Nunc ac blandit augue.	1
7	5	3	3	2	NULL	Ocena za odpowiedź ustna	NULL
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Rysunek 17: Wszystkie oceny po wywołaniu procedury dodawania - widzimy jedną dodatkową ocenę

4.4.2 Widok

	Nauczyciel	Przedmiot	Liczba godzin w tygodniu	Plec
▶	Izabella Nawrocka	Matematyka	12	K
	Izabella Nawrocka	Fizyka	5	K
	Aneta Nowacka	Polski	12	K
	Aneta Nowacka	Włoski	3	K
	Paweł Kalinowski	WF	5	M
	Marzena Stasiak	Historia	6	K
	Marzena Stasiak	WOS	2	K
	Klara Madej	Informatyka	1	K
	Monika Lipińska	angielski	6	K

Rysunek 18: Zestawienie nauczycieli z przedmiotami, których uczą oraz liczbą godzin w tygodniu

4.4.3 Ograniczenia

Zdecydowano się przetestować ograniczenie na rodzaj nieobecności w tabeli "Obecności". Rodzaj nieobecności powinien być znakiem ze zbioru: 'U', 'Z', 'N'. Test ograniczenia odbywa się poprzez próbę dodania rekordu z rodzajem nieobecności nienależącej do zbioru.

#	Time	Action	Message
2330	22:38:42	CREATE PROCEDURE 'wprowadzOcene' (ocena integer, waga d...)	0 row(s) affected
2331	22:38:46	call dziennikszkolny.wprowadzOcene(3, 2, 5, 3, null, 'Ocena za odpowiedź ustna')	Error Code: 1366. Incorrect integer value: 'Ocena za odpowiedź ustna' at position 6
2332	22:39:29	DROP procedure IF EXISTS 'wprowadzOcene'	0 row(s) affected
2333	22:39:29	USE 'dziennikszkolny'	0 row(s) affected
2334	22:39:29	CREATE PROCEDURE 'wprowadzOcene' (ocena integer, waga d...)	0 row(s) affected
2335	22:39:32	call dziennikszkolny.wprowadzOcene(3, 2, 5, 3, null, 'Ocena za od...')	1 row(s) affected
2336	22:39:46	SELECT * FROM dziennikszkolny.oceny LIMIT 0, 1000	7 row(s) returned
2337	22:48:16	SELECT * FROM dziennikszkolny.obecnosci LIMIT 0, 1000	1000 row(s) returned
2338	22:48:42	call dziennikszkolny.wpiswanieObecnosci(2, 3, 'O')	Error Code: 3819. Check constraint 'obecnosc_chk_1' is violated.

Rysunek 19: Wywołanie procedury wpisującej nieobecność z rodzajem nieobecności 'O', która nie należy do zbioru dozwolonych rodzajów nieobecności - widzimy błąd podczas próby dodania rekordu

4.5 Triggers

Sprawdzono trigger, który odpowiada za to, żeby nie można było wprowadzić oceny uczniowi na lekcji, na której nie był obecny

The screenshot shows a database log window with the following details:

```

1  INSERT INTO Oceny (UczniowieidUczen, LekcjeidLekcja, Wartosc, Waga, Komentarz, SprawdzanyidSprawdzian) VALUES (5, 41, 3, 3, "Nowa ocena", null);

Output
Action Output
# Time Action Message
2346 23:04:33 UPDATE Klasa SET NauczycieleidNauczyciel = 1 WHERE idKlasa = 2 Error Code: 1644. Ten nauczyciel jest juz wychowawca.
2347 23:07:41 SELECT * FROM dziennikszkolny.obecnosci LIMIT 0, 1000 1000 row(s) returned
2348 23:09:23 SELECT * FROM dziennikszkolny.oceny LIMIT 0, 1000 7 row(s) returned
2349 23:16:10 INSERT INTO Oceny (UczniowieidUczen, LekcjeidLekcja, Wartosc, Waga, Komentarz, SprawdzanyidSprawdzia... 1 row(s) affected
2350 23:16:47 SELECT * FROM dziennikszkolny.oceny LIMIT 0, 1000 8 row(s) returned
2351 23:17:26 CREATE TRIGGER CzyOcenicInsert BEFORE INSERT ON Oceny FOR EACH ROW BEGIN IF ((SELECT COUNT(*) ... 0 row(s) affected
2352 23:17:26 CREATE TRIGGER CzyOcenicUpdate BEFORE UPDATE ON Oceny FOR EACH ROW BEGIN IF (NOT((SELECT ... 0 row(s) affected
2353 23:17:32 INSERT INTO Oceny (UczniowieidUczen, LekcjeidLekcja, Wartosc, Waga, Komentarz, SprawdzanyidSprawdzia... Error Code: 1644. Uczni nie moze otrzymac oceny, poniewaz nie byl/jest wtedy obecny.

```

Rysunek 20: Błąd podczas wprowadzania uczniowi oceny na lekcji, na której nie był obecny

4.5.1 Testy wydajnościowe

Najbardziej liczną encją jest encja obecności. Zakładamy, że w danym roku do tabeli "Obecnosci" będzie dodawanych około 60000 rekordów. Dane są archiwizowane w okresie 10 lat, zatem łącznie w tabeli będzie znajdować się 600000 rekordów. Z tego względu zdecydowano się przeprowadzić testy wydajnościowe z najczęstszym przypadkiem użycia, czyli wyszukiwaniem nieobecności danego ucznia. W tym celu wygenerowano 600000 rekordów z losowymi danymi. Ze względu na fakt, iż nie wygenerowano odpowiedniej ilości danych losowych uczniów oraz lekcji, dodano dwie kolumny pomocnicze: "tempUczen" oraz "tempLekcja", które umożliwiały wygenerowanie losowych nr id dla uczniów oraz lekcji. Zmierzono czas realizacji pobrania rekordów z klauzulą "WHERE" przed utworzeniem indeksu oraz po utworzeniu indeksu. Uruchomiono polecenie "SELECT * FROM obecnosci WHERE tempUczen = 123;" przed założeniem indeksu. Czas realizacji wyniósł **375 ms**. Czas dodawania nieobecności wynosił **1 ms**. Następnie założono indeks na kolumnie "tempUczen" - "CREATE INDEX ObecnosciIdxTemp ON Obecnosci (tempUczen ASC);". Czas zakładania indeksu wynosił **3313 ms**. Następnie ponownie uruchomiono polecenie "SELECT * FROM obecnosci WHERE tempUczen = 123;". Czas realizacji wyniósł zaledwie **16 ms**. Czas dodawania nieobecności wynosił **31 ms**.

Czas wyszukiwania znaczco się zmniejszył, jednak kosztem czasu wprowadzania i edycji danych.

The screenshot shows a database result grid with the following details:

```

1  select * from obecnosci where tempUczen = 123

```

Result Grid								
	idObecnosci	UczniowieidUczen	LekcjeidLekcja	Rodzaj	RodziceidRodzic	NauczycieleidNauczyciel	tempUczen	tempLekcja
▶	5449	12	39	N	10	1	123	46377
	7568	10	25	N	12	3	123	137112
	10660	4	12	N	9	6	123	31594
	14082	7	21	N	10	1	123	36492
	19941	11	2	N	8	4	123	121163
	23361	1	12	N	9	6	123	91967
	45823	9	26	U	11	5	123	84869
	47710	4	14	N	9	1	123	96566
	48774	10	24	Z	10	3	123	136746
	56122	9	6	N	5	4	123	124469
	58858	11	16	N	1	2	123	135525

Rysunek 21: Wynik wyszukiwania nieobecności ucznia o id = 123

5 Implementacja i testy aplikacji

5.1 Instalacja i konfigurowanie systemu

Do poprawnego działania aplikacji potrzebny jest aktywny serwer MySQL, który przechowuje dane potrzebne do działania aplikacji, Java Development Kit w wersji 15 oraz IDE do uruchomienia programu w wersji deweloperskiej. Podczas tworzenia projektu skorzystaliśmy z narzędzia Maven, które pozwalało nam na dodawanie zewnętrznych zależności, takich jak: Spring Boot, Spring Security, Hibernate oraz Vaadin. Podczas tworzenia aplikacji korzystaliśmy z IDE IntelliJ Community, które pozwalało nam na deweloperskie uruchamianie aplikacji.

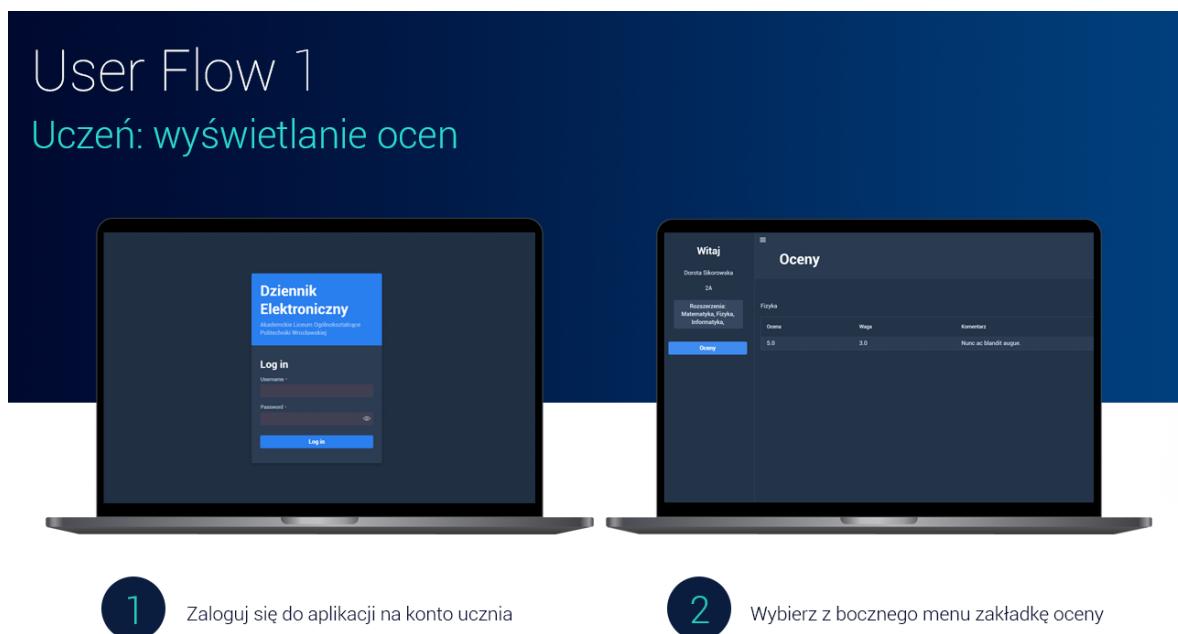
W celu uruchomienia naszej aplikacji na serwerze, należało wykonać polecenie: "mvn clean compile assembly:single". Następnie należy przenieść plik wykonywalny z rozszerzeniem ".jar" na serwer i uruchomić go za pomocą komendy: "java -jar target/ElectronicGrade1.0.0.jar".

5.2 Instrukcja użytkowania aplikacji

5.2.1 User flow uczeń

Przeglądanie ocen dla ucznia

1. Zaloguj się do aplikacji na konto ucznia.
2. Wybierz z bocznego menu zakładkę oceny.



Rysunek 22: User Flow 1 : Uczeń : Wyświetlanie ocen

5.2.2 User flow nauczyciel

Przeglądanie ocen uczniów

1. Zaloguj się do aplikacji na konto nauczyciela.
2. Wybierz z bocznego menu zakładkę oceny.
3. Wybierz przedmiot z listy

4. Wybierz klasę, której oceny chcesz zobaczyć

Dodawanie oceny

1. Zaloguj się do aplikacji na konto nauczyciela.
2. Wybierz z bocznego menu zakładkę oceny.
3. Wybierz z listy przedmiot, z którego chcesz dodać ocenę.
4. Wybierz z listy klasę, dla której ucznia chcesz dodać ocenę.
5. Kliknij przycisk "Dodaj ocenę".
6. W oknie dialogowym wybierz z listy lekcję i ucznia, do których ma być przypisana ocena oraz podaj wartość, wagę i komentarz do oceny.
7. Kliknij przycisk "Zapisz".

Modyfikowanie oceny

1. Zaloguj się do aplikacji na konto nauczyciela.
2. Wybierz z bocznego menu zakładkę oceny.
3. Wybierz z listy przedmiot, z którego ocenę chcesz zmodyfikować.
4. Wybierz z listy klasę, dla której ucznia chcesz zmodyfikować ocenę.
5. Przy wybranej ocenie naciśnij przycisk "Modyfikuj".
6. Popraw odpowiednie pola.
7. Naciśnij przycisk "Zapisz".

Usunięcie oceny

1. Zaloguj się do aplikacji na konto nauczyciela.
2. Wybierz z bocznego menu zakładkę oceny.
3. Wybierz z listy przedmiot, z którego ocenę chcesz usunąć.
4. Wybierz z listy klasę, dla której ucznia chcesz usunąć ocenę.
5. Przy wybranej ocenie naciśnij przycisk "Usuń".
6. Potwierdź usunięcie oceny.

User Flow 2

Nauczyciel: wyświetlanie ocen

1 Zaloguj się do aplikacji na konto nauczyciela

2 Z menu bocznego wybierz opcję oceny, a następnie wybierz z listy przedmiot oraz klasę, której oceny chcesz wyświetlić

Dodawanie oceny

- 3 Dodawanie oceny jest możliwe dopiero po wybraniu przedmiotu oraz klasy. Te dwie akcje aktywują przycisk „Dodaj ocenę”. Aby dodać ocenę należy kliknąć przycisk oraz wypełnić wszystkie pola formularza. W przypadku, gdy wprowadzona wartość oceny lub wagi będzie niepoprawna dodanie oceny nie będzie możliwe. W przypadku chęci anulowania wpisania oceny należy potwierdzić akcję, żeby nie utracić wprowadzonych danych jeśli przycisk był kliknięty przypadkowo

Niepoprawna waga oceny!
Poprawne wartości oceny 1,2,3

OK

Zapisz

Opuszczasz stronę?
Na pewno chcesz opuścić stronę?
Utracisz wprowadzone dane

TAK

Rysunek 23: User Flow 2 : Nauczyciel : Wyświetlanie i dodawanie ocen

Modyfikacja oceny

4

Aby zmodyfikować ocenę należy wybrać ocenę, którą chcemy zmienić i kliknąć przycisk modyfikuj. Wprowadzone zmiany należy zapisać i zamknąć ikoną X. Niepoprawne wartości nie zostaną zapisane.

Imię	Nazwisko	Ocena	Waga	Data	Komentarz	Zapisz	Usuń
Lucyna	Tomczyk	3.0	2.0	2021-11-08	Test1		
Fabian	Kowalczyk	6.0	3.0	2021-11-08	Test2		

Usuwanie oceny

5

Aby usunąć ocenę należy ją wybrać i kliknąć przycisk usuń. Akcję należy potwierdzić.

Przedmiot: WF Klasa: 3B

Zmień przedmiot

Dodaj ocenę

Imię	Nazwisko	Ocena	Waga	Data	Komentarz	Modyfikuj	Usuń
Lucyna	Tomczyk	3.0	2.0	2021-11-08	Test1		
Fabian	Kowalczyk	6.0	3.0	2021-11-08	Test2		

Usuwanie Oceny

Czy na pewno chcesz usunąć ocenę?

NIE **TAK**

Filtrowanie uczniów

6

Istnieje możliwość filtrowania uczniów po imieniu i/lub nazwisku w celu zawężenia wyświetlanych ocen.

Imię	Nazwisko	Ocena	Waga	Data	Komentarz	Modyfikuj	Usuń
Lucyna	Tomczyk	3.0	2.0	2021-11-08	Test1		
Fabian	Kowalczyk	6.0	3.0	2021-11-08	Test2		

Imię	Nazwisko	Ocena	Waga	Data	Komentarz	Modyfikuj	Usuń
	kow						

Rysunek 24: User Flow 2 : Nauczyciel : Modyfikacja, usuwanie, filtracja ocen

5.3 Testowanie opracowanych funkcji systemu

5.3.1 Testy automatyczne

Zostały zaimplementowane testy automatyczne z wykorzystaniem narzędzia Selenium IDE. Testy automatyczne obejmują funkcjonalność logowania. Podczas tworzenia testów automatycznych korzystano z funkcji "Recording", dostarczaną przez narzędzie Selenium IDE. Testy automatyczne testują 3 przypadki:

- logowanie ucznia,
- logowanie nauczyciele,
- błąd logowania.

Search tests...	Command	Target	Value
BladLogowania	1 open	http://localhost:8080/login	
LogowanieNauczyciel	2 type	css=#vaadinLoginUsername > input	dsikorska
LogowanieUczeln*	3 type	css=#vaadinLoginPassword > input	GEuLy5Kv5NaQ
	4 click	css=form > vaadin-button	
	5 click	css=.bg-base > .text-seconda ry	
	6 assert text	xpath=/html/body/vaadin-app- layout/section/vaadin-vertical- layout/label	Dominika Sikorska
	7 assert text	xpath=/html/body/vaadin-app- layout/section/vaadin-vertical- layout/label[2]	3B
	8 assert text	xpath=/html/body/vaadin-app- layout/section/vaadin-vertical- layout/label[3]	Rozszerzenia: Polski, Histori a, WOS,
	9 click	css=.font-medium	
	10 assert element present	xpath=/html/body/vaadin-app- layout/vaadin-vertical-layout/v aadin-button	Oceny

Rysunek 25: Test automatyczny testujący poprawność logowania ucznia. Test wprowadza dane przykładowego ucznia i sprawdza czy po kliknięciu przycisku "zaloguj" logowanie powiedzie się, poprzez sprawdzenie czy na stronie występuje element "Oceny" oraz sprawdza dane personalne ucznia, jego klasę i rozszerzenia

Search tests...		http://localhost:8080/		
		Command	Target	Value
✓	BladLogowania	1 open	http://localhost:8080/login	
✓	LogowanieNauczyciel	2 type	css=#vaadinLoginUsername > input	inawrocka
✓	LogowanieUczen	3 type	css=#vaadinLoginPassword > input	u5dHWdT5q8yp
		4 click	css=form > vaadin-button	
		5 click	css=bg-base > .text-seconda	
		6 assert text	xpath=/html/body/vaadin-app- layout/section/vaadin-vertical- layout/label	Izabella Nawrocka
		7 click	css=.font-medium	
		8 assert element present	xpath=/html/body/vaadin-app- layout/vaadin-vertical-layout/vaadin-button	

Rysunek 26: Test automatyczny testujący poprawność logowania ucznia. Test wprowadza dane przykładowego ucznia i sprawdza czy po kliknięciu przycisku "zaloguj" logowanie powiedzie się, poprzez sprawdzenie czy na stronie występuje element "Oceny", sprawdza dane personalne nauczyciela

Search tests...		http://localhost:8080/		
		Command	Target	Value
✓	BladLogowania	1 ✓ open	http://localhost:8080/login	
✓	LogowanieNauczyciel	2 ✓ type	css=#vaadinLoginU sename > input	blednyLogin
✓	LogowanieUczen	3 ✓ type	css=#vaadinLoginP assword > input	bledneHaslo
		4 ✓ click	css=#vaadinLoginFo rm > vaadin-login-for m-wrapper > form ...	
		5 ✓ verify element pr esent	css=#vaadinLoginFo rm > vaadin-login-for m-wrapper > form ...	

Rysunek 27: Test automatyczny testujący przypadek błędnego logowania. Test wprowadza błędne dane niewystępujące w bazie, symuluje kliknięcie przycisku "zaloguj", i sprawdza czy po kliknięciu przycisku "zaloguj" wciąż istnieje, czyli czy logowanie faktyczne nie nastąpiło.

5.3.2 Testy manualne

Pozostałe funkcjonalności były testowane manualnie. Testy manualne polegały na ręcznym testowaniu funkcjonalności w oparci o instrukcję użytkownika zamieszczoną wyżej. W celu usprawnienia testów, wykonywaliśmy je wszyscy wspólnie. Aplikacja webowa była uruchomiona na komputerze jednego z członków grupy. Dzięki publicznemu adresowi IP oraz przekierowaniu portu 8080, wszyscy członkowie grupy mieli dostęp do tej samej aplikacji. Jeden z członków grupy był zalogowany na konto nauczyciela, a pozostała na konta uczniów. Za pomocą panelu nauczyciela, oceny były wprowadzane dla dwóch zalogowanych uczniów, a zalogowani uczniowie odczytywali nowe oceny. W podobny sposób było testowanie usuwanie i modyfikacja ocen.

5.4 Omówienie wybranych rozwiązań programistycznych

5.4.1 Implementacja interfejsu i dostępu do bazy danych

Interfejs został zaimplementowany przy użyciu Vaadina. Framework pozwalana tworzenie strony front-endowej aplikacji przy użyciu języka Java i nie wymaga znajomości takich technologii jak HTML czy

CSS. Komponenty dostępne w Vaadinie są ostylowane i dostępne np. w różnych wersjach kolorystycznych. MainLayout stanowi bazę każdego widoku, a zawartość bocznego menu jest ustawiana w zależności od uprawnień zalogowanego użytkownika. W naszej aplikacji skorzystaliśmy z takich komponentów jak Grid, Layout, Combobox, Drawer czy TextField.

Dostęp do bazy danych uzyskaliśmy dzięki frameworkowi Hibernate. Wszystkie tabele z bazy danych mają swoje odpowiedniki w postaci klas encji z odpowiednimi adnotacjami. Ponadto, do przepływu informacji pomiędzy bazą a aplikacją stworzyliśmy kilka klas Service oraz kilka interfejsów Repository, które rozszerzają JpaRepository.

Listing 12: Plik konfiguracyjny aplikacji ”application.properties”, dostęp do bazy danych, port aplikacji internetowej

```
spring.datasource.url=jdbc:mysql://localhost:3306/dziennikszkolny
spring.datasource.username=root
spring.datasource.password=admin
spring.jpa.hibernate.naming.physical-strategy=org.hibernate.boot.model.naming.
    ↪ PhysicalNamingStrategyStandardImpl

server.port=${PORT:8080}
# Ensure application is run in Vaadin 14/npm mode
vaadin.compatibilityMode = false
logging.level.org.atmosphere = warn
# To improve the performance during development.
# For more information https://vaadin.com/docs/flow/spring/tutorial-spring-configuration.html#special-
    ↪ configuration-parameters
vaadin.whitelisted-packages = com.vaadin.org.vaadin.com.example.application
vaadin.pnpm.enable = false
```

Poniżej znajduje się implementacja głównej klasy aplikacji, która korzysta ze statycznej metody dostarczanej przez framework Spring.

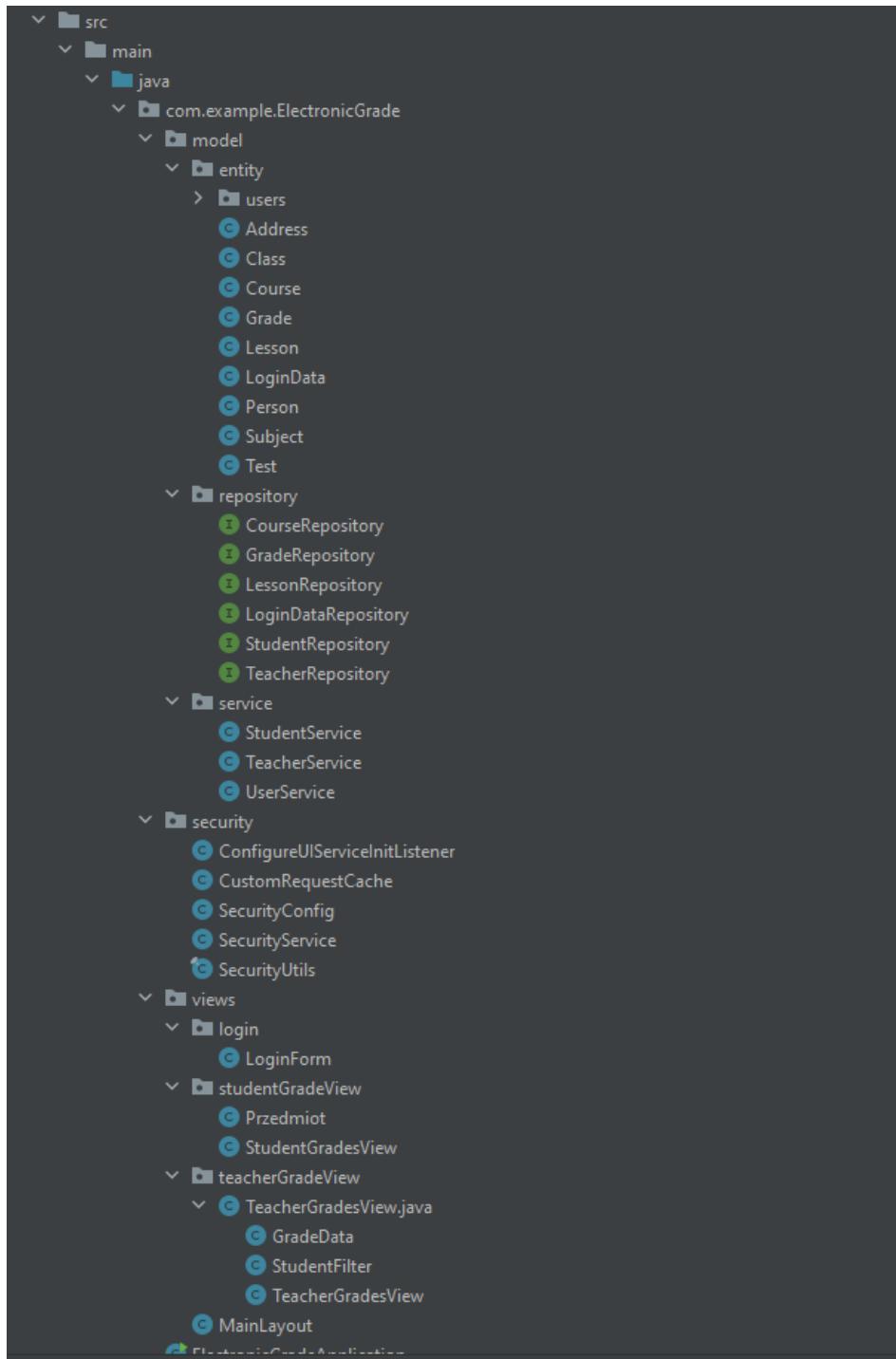
Listing 13: Główna klasa projektu

```
@SpringBootApplication(exclude = ErrorMvcAutoConfiguration.class)
@NoArgsConstructor(value = "lumo-css-framework", version = "^4.0.10")
@NoArgsConstructor(value = "line-awesome", version = "1.3.0")
public class ElectronicGradeApplication {

    public static void main(String[] args) {
        SpringApplication.run(ElectronicGradeApplication.class, args);
    }
}
```

Poniżej znajduje się struktura projektu w oparciu o wzorzec aplikacji korzystających z frameworku Spring. W strukturze projektu wydzieliliśmy następujące pakiety:

- model - zawiera klasy modelujące encje bazy danych,
- repository - zawiera repozytoria dostępu do danych,
- service - zawiera serwisy, umożliwiające pobieranie danych w oparciu o zadane parametry,
- views - widoki dostępne z poziomu serwisu internetowego.



Rysunek 28: struktura plików projektu aplikacji

Przykłady metod powiązanych z interfejsem użytkownika - Vaadin:

Listing 14: Kod metody tworzącej menu boczne, klasa *MainLayout*

```
private Component createDrawerContent() {
    com.vaadin.flow.component.html.Section section = null;
    User user = (User) securityService.getAuthenticatedUser();

    if(user!= null){
        H2 appName = new H2("Witaj");
        Label userName = new Label(user.getName());
        VerticalLayout verticalLayout = new VerticalLayout();
        verticalLayout.setAlignItems(FlexComponent.Alignment.CENTER);
        if(user.getRoles().contains(Role.STUDENT))
        {
            Student studentUser = (Student) securityService.getAuthenticatedUser();
            Label className = new Label(studentUser.getStudentClass().toString());
            TextArea classExtension = new TextArea();
            classExtension.setValue(studentUser.getStudentClass().getExtensionsNames());
            classExtension.addThemeVariants(TextAreaVariant.LUMO_ALIGN_CENTER);
            verticalLayout.add(appName, userName, className, classExtension);
            section = new com.vaadin.flow.component.html.Section(verticalLayout,
                createNavigation(), createFooter());
            section.addClassNames("flex", "flex-col", "items-stretch", "max-h-full", "min-h-full");
        }
        if(user.getRoles().contains(Role.TEACHER))
        {
            verticalLayout.add(appName, userName);
            section = new com.vaadin.flow.component.html.Section(verticalLayout,
                createNavigation(), createFooter());
            section.addClassNames("flex", "flex-col", "items-stretch", "max-h-full", "min-h-full");
        }
    }
    else {
        section = new com.vaadin.flow.component.html.Section(createNavigation(), createFooter());
        section.addClassNames("flex", "flex-col", "items-stretch", "max-h-full", "min-h-full");
    }
    return section;
}
```

Listing 15: Kod metody tworzącej okno dialogowe odpowiedzialne za dodawanie oceny, klasa *Teacher-GradesView*

```
private VerticalLayout createDialogLayout(Dialog dialog){
    H2 headline = new H2("Dodaj_ocenę");
    headline.getStyle().set("margin", "var(--lumo-space-m)_0_0_0")
        .set("font-size", "1.5em").set("font-weight", "bold");

    ComboBox<Student> comboBox = new ComboBox<>("Wybierz_uecnia");
    comboBox.setItems(teacherService.findStudentsByClassId(chosenClass.getId()));
    comboBox.setItemLabelGenerator(Student::toString);

    comboBox.addValueChangeListener(event -> studentFunction(event.getValue()));

    ComboBox<Lesson> lessonComboBox = new ComboBox<>("Wybierz_lekcję");
    lessonComboBox.setItems(teacherService.findLessonsBySubjectAndClassId(chosenSubject.getId(), chosenClass
        .getId()));
    lessonComboBox.setItemLabelGenerator(Lesson::toString);

    lessonComboBox.addValueChangeListener(event -> lessonFunction(event.getValue()));

    TextField valueTextField = new TextField("Wartość_oceny");
    TextField weightTextField = new TextField("Waga_oceny");
    TextField descTextField = new TextField("Komentarz_do_oceny");

    VerticalLayout fieldLayout = new VerticalLayout(comboBox, lessonComboBox, valueTextField,
        weightTextField, descTextField);
    fieldLayout.setSpacing(false);
    fieldLayout.setPadding(false);
```

```

fieldLayout.setAlignItems(FlexComponent.Alignment.STRETCH);

Button cancelButton = new Button("Anuluj", e -> {
    Dialog confirmationDialog = new Dialog();
    confirmationDialog.getElement();
    VerticalLayout confirmationLayout = createConfirmationDialog(confirmationDialog, dialog);
    confirmationDialog.add(confirmationLayout);
    confirmationDialog.open();
    add(confirmationDialog);
});
Button saveButton = new Button("Zapisz");
saveButton.addClickListener(buttonClickEvent -> {
    if(!valueTextField.getValue().equals("") && !weightTextField.getValue().equals("") && Double.
        ↪ parseDouble(valueTextField.getValue()) >= 1.0 && Double.parseDouble(valueTextField.getValue()
        ↪ ()) <= 6.0 && Double.parseDouble(valueTextField.getValue()) % 0.5 == 0 && Double.
        ↪ parseDouble(weightTextField.getValue()) >= 1 && Double.parseDouble(weightTextField.
        ↪ getValue()) <= 3 && Double.parseDouble(weightTextField.getValue()) % 1 == 0){
        Grade g = new Grade(Double.parseDouble(valueTextField.getValue()), Double.parseDouble(
            ↪ weightTextField.getValue()), descTextField.getValue(), chosenLesson, chosenStudent);
        teacherService.saveGrade(g);
        Notification.show("Ocena zostala zapisana");
        updateGrid();
        dialog.close();
    }
    String errorMessage = "";
    if (valueTextField.getValue().equals("") || Double.parseDouble(valueTextField.getValue()) < 1.0 || Double
        ↪ .parseDouble(valueTextField.getValue()) > 6.0 || Double.parseDouble(valueTextField.getValue())
        ↪ % 0.5 != 0) {
        errorMessage = "Niepoprawna wartość oceny!\n\n" +
            "Poprawne wartości oceny to:\n1,1.5,2,2.5,3,3.5,4,4.5,5,5.5,6";
        Dialog errorDialog = new Dialog();
        errorDialog.getElement();
        VerticalLayout errorLayout = createErrorDialog(errorDialog, errorMessage);
        errorDialog.add(errorLayout);
        errorDialog.open();
        add(errorDialog);
    }
    if (valueTextField.getValue().equals("") || Double.parseDouble(weightTextField.getValue()) < 1 || Double.
        ↪ parseDouble(weightTextField.getValue()) > 3 || Double.parseDouble(weightTextField.getValue())
        ↪ % 1 != 0) {
        errorMessage = "Niepoprawna waga oceny!\n\n" +
            "Poprawne wartości oceny 1,2,3";
        Dialog errorDialog = new Dialog();
        errorDialog.getElement();
        VerticalLayout errorLayout = createErrorDialog(errorDialog, errorMessage);
        errorDialog.add(errorLayout);
        errorDialog.open();
        add(errorDialog);
    }
});
saveButton.addThemeVariants(ButtonVariant.LUMO_PRIMARY);
HorizontalLayout buttonLayout = new HorizontalLayout(cancelButton,
    saveButton);
buttonLayout
    .setJustifyContentMode(FlexComponent.JustifyContentMode.END);

VerticalLayout dialogLayout = new VerticalLayout(headline, fieldLayout,
    buttonLayout);
dialogLayout.setPadding(false);
dialogLayout.setAlignItems(FlexComponent.Alignment.STRETCH);
dialogLayout.getStyle().set("width", "400px").set("max-width", "100%");

return dialogLayout;
}

```

Listing 16: Kod tworzący layout widoku ocen ucznia, klasa *StudentGradesView*

```

    @PageTitle("Oceny")
    @Route(value = "oceny", layout = MainLayout.class)
    public class OcenyView extends HorizontalLayout {

        private final Student student;

        public OcenyView(SecurityService securityService, @Autowired StudentService studentService) {
            this.student = studentService.findById(((User) securityService.getAuthenticatedUser()).getId()).orElseThrow
                ()::
            ();

            HorizontalLayout header = new HorizontalLayout();
            H2 viewType = new H2();
            header.add(viewType);
            header.setWidth("100%");
            header.addClassNames("bg-base", "border-b", "border-contrast-10", "box-border", "flex", "h-xl", "
                items-center",
                "w-full");

            VerticalLayout verticalLayout = new VerticalLayout();
            verticalLayout.add(header);

            add(verticalLayout);

            Map<Subject, List<Grade>> gradesMap = student.getGradesMap();

            for(Subject s : gradesMap.keySet()){
                ArrayList<Grade> grades = (ArrayList<Grade>) gradesMap.get(s);
                String subjectName = s.getName();

                Label l = new Label(subjectName);

                Grid<Grade> gradeGrid = new Grid<>();
                gradeGrid.addThemeVariants(GridVariant.LUMO_ROW_STRIPES);
                gradeGrid.setItems(grades);

                gradeGrid.addColumn(c->c.getValue()).setHeader("Ocena").setAutoWidth(true);
                gradeGrid.addColumn(c->c.getWeight().toString()).setHeader("Waga");
                gradeGrid.addColumn(c->c.getDesc()).setHeader("Komentarz").setAutoWidth(true);
                gradeGrid.addColumn(c->c.getLesson().getDate().toString()).setHeader("Data");
                gradeGrid.setHeightByRows(true);
                gradeGrid.addThemeVariants(GridVariant.LUMO_WRAP_CELL_CONTENT);
                verticalLayout.add(l);
                verticalLayout.add(gradeGrid);
            }
        }
    }
}

```

5.4.2 Implementacja wybranych funkcjonalności systemu

Ze względu na ograniczenia czasowe, zdecydowaliśmy się na zaimplementowanie przeglądania ocen dla ucznia oraz przeglądania, dodawania, edycji oraz usuwania ocen dla nauczyciela.

W przypadku ucznia, wyświetlamy wszystkie jego oceny zawarte w bazie, dzieląc je na przedmioty dla zwiększenia przejrzystości i podniesienia satysfakcji korzystania z systemu.

Panel ocen nauczyciela różni się znacznie od panelu ucznia. Zawarliśmy tutaj 2 comboboxy, gdzie najpierw w 1. należy wybrać przedmiot - wyświetlane są tylko przedmioty, których uczy dany nauczyciel. Po wybraniu przedmiotu odblokowuje się możliwość wyboru klasy, którą zalogowany nauczyciel uczy wskazanego wcześniej przedmiotu. Następnie wyświetla się lista wszystkich ocen tej klasy wraz z imieniem i nazwiskiem ucznia, który daną ocenę otrzymał oraz szczegół tej oceny. Listę można filtrować po imieniu i nazwisku ucznia. Aktywuje się także przycisk "Dodaj ocenę". Na liście ocen do każdego pola przypisany jest przycisk "Modyfikuj" oraz "Usuń". Modyfikacja oceny polega na zmianie wagi, wartości i/lub komentarza. Można zmiany zapisać, co powoduje aktualizację danej oceny w bazie danych, bądź anulować akcję. W przypadku naciśnięcia usuń, pojawia się komunikat o potwierdzeniu akcji, jeśli została ona potwierdzona to wybrana ocena jest usuwana z bazy i panel zostaje odświeżony. Aby zmienić przedmiot należy kliknąć przycisk "Zmień przedmiot", który aktywuje comboboxa

z przedmiotami. W przypadku dodawania oceny wyświetlane jest okno dialogowe, w którym należy podać najważniejsze dane dotyczące oceny oraz wybrać lekcję i ucznia, których wpisywana ocena ma dotyczyć. Jeśli wprowadzona wartość bądź waga nie zgadza się z przyjętym przez nas formatem, to wyświetlamy komunikat o błędzie i o dostępnych możliwych formatach danego pola.

5.4.3 Implementacja mechanizmów bezpieczeństwa

Zważając na poufność danych i przepisy w naszym kraju, aby móc korzystać z aplikacji wymagana jest autoryzacja w postaci logowania. Wykorzystaliśmy do tego SpringSecurity. Dzięki danym przechowywanym w bazie, jesteśmy w stanie rozróżnić, który użytkownik jest zalogowany, przez co dostosowujemy widoczność i uprawnienia do roli użytkownika.

Klasy związane z bezpieczeństwem (większość z nich jest zaczepiona ze wzorców związanych z frameworkm Spring Security):

Poniżej znajduje się główna klasa odpowiedzialna za zabezpieczenie komunikacji strony klienta i serwera. Ważnym uwagi jest mechanizm szyfrujący. Do szyfrowania haseł korzystamy z szyfratora BCryptPasswordEncoder.

Listing 17: Kod klasy *SecurityConfig* dziedziczącej po WebSecurityConfigurerAdapter

```
@EnableWebSecurity
@Configuration
public class SecurityConfig extends WebSecurityConfigurerAdapter {

    private static final String LOGIN_PROCESSING_URL = "/login";
    private static final String LOGIN_FAILURE_URL = "/login?error";
    private static final String LOGIN_URL = "/login";
    private static final String LOGOUT_SUCCESS_URL = "/login";

    /**
     * Require login to access internal pages and configure login form.
     */
    @Override
    protected void configure(HttpSecurity http) throws Exception {
        // Vaadin handles CSRF internally
        http.csrf().disable()

        // Register our CustomRequestCache, which saves unauthorized access attempts, so the user is redirected
        // after login.
        .requestCache().requestCache(new CustomRequestCache())

        // Restrict access to our application.
        .and().authorizeRequests()

        // Allow all Vaadin internal requests.
        .requestMatchers(SecurityUtils::isFrameworkInternalRequest).permitAll()

        // Allow all requests by logged-in users.
        .anyRequest().authenticated()

        // Configure the login page.
        .and().formLogin()
            .loginPage(LOGIN_URL).permitAll()
            .loginProcessingUrl(LOGIN_PROCESSING_URL)
            .failureUrl(LOGIN_FAILURE_URL)

        // Configure logout
        .and().logout().logoutSuccessUrl(LOGOUT_SUCCESS_URL);
    }

    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws Exception {
        auth.userDetailsService(userDetailsService()).passwordEncoder(passwordEncoder());
    }

    @Bean
    public BCryptPasswordEncoder passwordEncoder() {
```

```

    return new BCryptPasswordEncoder();
}

@Bean
@Override
public UserDetailsService userDetailsService() {
    return new UserService();
}

/**
 * Allows access to static resources, bypassing Spring Security.
 */
@Override
public void configure(WebSecurity web) {
    web.ignoring().antMatchers(
        // Client-side JS
        "/VAADIN/**",
        // the standard favicon URI
        "/favicon.ico",
        // the robots exclusion standard
        "/robots.txt",
        // web application manifest
        "/manifest.webmanifest",
        "/sw.js",
        "/offline.html",
        // icons and images
        "/icons/**",
        "/images/**",
        "/styles/**",
        // (development mode) H2 debugging console
        "/h2-console/**");
}
}

```

Listing 18: Kod klasy *SecurityService*

```

@Component
public class SecurityService {

    private static final String LOGOUT_SUCCESS_URL = "/";

    public UserDetails getAuthenticatedUser() {
        SecurityContext context = SecurityContextHolder.getContext();
        Object principal = context.getAuthentication().getPrincipal();
        if (principal instanceof UserDetails) {
            return (UserDetails) context.getAuthentication().getPrincipal();
        }
        // Anonymous or no authentication.
        return null;
    }

    public void logout() {
        UI.getCurrent().getPage().setLocation(LOGOUT_SUCCESS_URL);
        SecurityContextLogoutHandler logoutHandler = new SecurityContextLogoutHandler();
        logoutHandler.logout(
            VaadinServletRequest.getCurrent().getHttpServletRequest(), null,
            null);
    }
}

```

Listing 19: Kod klasy *SecurityUtils*

```
public final class SecurityUtils {
```

```

private SecurityUtils() {
    // Util methods only
}

static boolean isFrameworkInternalRequest(HttpServletRequest request) {
    final String parameterValue = request.getParameter(ApplicationConstants.REQUEST_TYPE_PARAMETER
        ↗ );
    return parameterValue != null
        && Stream.of(RequestType.values())
            .anyMatch(r -> r.getIdentifier().equals(parameterValue));
}

static boolean isUserLoggedIn() {
    Authentication authentication = SecurityContextHolder.getContext().getAuthentication();
    return authentication != null
        && !(authentication instanceof AnonymousAuthenticationToken)
        && authentication.isAuthenticated();
}

```

W celu weryfikacji danych użytkownika dostosowaliśmy nasz model do klas dostarczanych przez Spring Security, m. in. UserDetails, UserDetailsService czy też GrantedAuthority. Poniżej znajduje się implementacja UserService, która służy do identyfikacji danych użytkownika.

Listing 20: Kod klasy *UserService*

```

@Service
public class UserService implements UserDetailsService {

    @Autowired
    private StudentRepository studentRepository;

    @Autowired
    private TeacherRepository teacherRepository;

    @Override
    @Transactional
    public UserDetails loadUserByUsername(String username) {
        User user = studentRepository.findByUsername(username);
        if (user != null) {
            Hibernate.initialize(((Student) user).getStudentClass().getExtensions());
            Hibernate.initialize(((Student) user).getGradesMap());
        }
        if (user == null) {
            user = teacherRepository.findByUsername(username);
        }

        if (user == null) {
            throw new UsernameNotFoundException("User.not.found.");
        }
        return user;
    }
}

```

6 Pliki źródłowe

<https://github.com/mikolajchmielecki/ElectronicGrade>

7 Podsumowanie i wnioski

Projekt pozwolił nam nie tylko utrważyć dotychczasową wiedzę związaną z bazami danych, ale także ją poszerzyć o takie pojęcia jak procedury składowane czy granty. Nauczyliśmy się również nowych technologii takich, jak Spring, Hibernate czy Vaadin, które przydadzą nam się w pracy zawodowej. Pomimo tego, że aplikacja końcowa nie posiada zaimplementowanych wszystkich przewidzianych funkcjonalności, jesteśmy usatysfakcjonowani pracą, którą włożyliśmy w wykonanie projektu. Gdyby na projekt było przewidziane więcej czasu, zrobiliśmy pozostałe z nich. Przydatne było także pracowanie nad projektem według wyznaczonych etapów, gdyż dzięki temu minimalizowaliśmy liczbę błędów, które wyszłyby na jaw przy etapie późniejszym. Istotną częścią był także model fizyczny, z którym zetknęliśmy się pierwszy raz i do którego wracaliśmy podczas zarówno implementacji bazy jak i aplikacji ze względu na przejrzystość zależności pomiędzy encjami.

Literatura

- [1] Dokumentacja Spring Boot: <https://docs.spring.io/spring-boot/docs/current/api/>
- [2] Dokumentacja Spring Security: <https://docs.spring.io/spring-security/site/docs/current/api/>
- [3] Dokumentacja Hibernate: <https://hibernate.org/orm/documentation/5.6/>
- [4] Dokumentacja Vaadin: <https://vaadin.com/docs/v14/>

Spis rysunków

1	Diagram przypadków użycia.	7
2	Diagram aktywności dla PU Sprawdzenie obecności	8
3	Diagram aktywności dla PU Wpisanie oceny	8
4	Model koncepcyjny	9
5	Model logiczny	9
6	Model fizyczny	10
7	Widoki	10
8	Dziennik Elektroniczny - Nowoczesna Szkoła Dzisiaj	15
9	Strona Logowania	15
10	Menu Główne	16
11	Oceny	16
12	Frekwencja	17
13	Sprawdziany	17
14	Plan Zajęć	18
15	Wszystkie oceny przed wywołaniem procedury dodawania	25
16	Formularz wywołujący procedurę wprowadzania nowej oceny	26
17	Wszystkie oceny po wywołaniu procedury dodawania - widzimy jedną dodatkową ocenę	26
18	Zestawienie nauczycieli z przedmiotami, których uczą oraz liczbą godzin w tygodniu	27
19	Wywołanie procedury wpisującej nieobecność z rodzajem nieobecności 'O', która nie należy do zbioru dozwolonych rodzajów nieobecności - widzimy błąd podczas próby dodania rekordu	27
20	Błąd podczas wprowadzania uczniowi oceny na lekcji, na której nie był obecny	28
21	Wynik wyszukiwania nieobecności ucznia o id = 123	28
22	User Flow 1 : Uczeń : Wyświetlanie ocen	29
23	User Flow 2 : Nauczyciel : Wyświetlanie i dodawanie ocen	31
24	User Flow 2 : Nauczyciel : Modyfikacja, usuwanie, filtracja ocen	32

25	Test automatyczny testujący poprawność logowania ucznia. Test wprowadza dane przykładowego ucznia i sprawdza czy po kliknięciu przycisku "zaloguj" logowanie powiedzie się, poprzez sprawdzenie czy na stronie występuje element "Oceny" oraz sprawdza dane personalne ucznia, jego klasę i rozszerzenia	33
26	Test automatyczny testujący poprawność logowania ucznia. Test wprowadza dane przykładowego ucznia i sprawdza czy po kliknięciu przycisku "zaloguj" logowanie powiedzie się, poprzez sprawdzenie czy na stronie występuje element "Oceny", sprawdza dane personalne nauczyciela	34
27	Test automatyczny testujący przypadek błędnego logowania. Test wprowadza błędne dane niewystępujące w bazie, symuluje kliknięcie przycisku "zaloguj", i sprawdza czy po kliknięciu przycisk "zaloguj" wciąż istnieje, czyli czy logowanie faktycznie nie nastąpiło.	34
28	struktura plików projektu aplikacji	36

Spis tabel

1	Indeksy	11
---	-------------------	----

Listings

1	Skrypt tworzący tabelę	19
2	Skrypt tworzący połączenia między encjami	20
3	Skrypt tworzący indeksy	21
4	Skrypt tworzący widoki	22
5	Fragment skryptu odpowiedzialnego za tworzenie triggerów	22
6	Fragment skryptu odpowiedzialnego za tworzenie procedur składowanych	23
7	Skrypt tworzący użytkowników	24
8	Skrypt przypisujący uprawnienia (granty) uczniowi	24
9	Skrypt przypisujący dodatkowe uprawnienie (grant) rodzicowi	24
10	Fragment skryptu przypisującego uprawnienia (granty) nauczycielowi	24
11	Fragment skryptu przypisującego uprawnienia (granty) administratorowi	25
12	Plik konfiguracyjny aplikacji "application.properties", dostęp do bazy danych, port aplikacji internetowej	35
13	Główna klasa projektu	35
14	Kod metody tworzącej menu boczne, klasa <i>MainLayout</i>	37
15	Kod metody tworzącej okno dialogowe odpowiedzialne za dodawanie oceny, klasa <i>TeacherGradesView</i>	37
16	Kod tworzący layout widoku ocen ucznia, klasa <i>StudentGradesView</i>	38
17	Kod klasy <i>SecurityConfig</i> dziedziczącej po <i>WebSecurityConfigurerAdapter</i>	40
18	Kod klasy <i>SecurityService</i>	41
19	Kod klasy <i>SecurityUtils</i>	41
20	Kod klasy <i>UserService</i>	42