Politechnika Śląska

Wydział Matematyki Stosowanej
Informatyka, sem. VI

Aplikacje bazodanowe
Dokumentacja Projektu

eRegistry

Autorzy:
Karolina Chrząszcz
Szymon Górnioczek
Wiktor Gruszczyński
Jarosław Kania
Tomasz Kryg

Repozytorium

15 września 2017

Spis treści

1 Temat projektu

Naszym zadaniem jest stworzenie elektronicznego dziennika szkolnego. Dziennik ten powinien umożliwić, uczniom i ich opiekunom, wgląd do ocen ucznia z przedmiotów, na które uczęszcza oraz dać możliwość nauczycielom wprowadzania tych ocen.

Dzięki projektowi rodzice na bieżąco będą mieli wgląd w szkolne życie swoich dzieci oraz ułatwi to komunikację z nauczycielami.

2 Opis ogólny wymagań

Na podstawie przeprowadzonej analizy zagadnienia przygotowaliśmy listę podstawowych funkcjonalności, którą powinien realizować nasz projekt.

Każda osoba, mająca dostęp do dziennik, będzie nazywana użytkownikiem. Każdy użytkownik będzie miał stworzone własne konto, dzięki któremu będzie miał wgląd do odpowiednich zasobów dziennika. Zostanie mu również przypisana *data ważności* (data, po której powinien zostać usunięty dostęp do dziennika) oraz informacja o tym, czy jest *aktywnym* użytkownikiem.

W trakcie tworzenia konta, administrator wprowadzi dane osobowe użytkownika oraz przydzieli mu odpowiednie uprawnienia. Po zalogowaniu się, użytkownicy poza wyświetleniem odpowiednich informacji będą mogli edytować swoje dane, oraz zmienić hasło. W celu ułatwienia opisu funkcjonalności podzieliliśmy użytkowników na podane niżej grupy:

2.1 Uczniowie

Każdy uczeń posiadający konto, powinien mieć wgląd do swoich ocen, przedmiotów na które uczęszcza, informacji o sobie, prowadzących zajęcia oraz klasy do której należy.

Oceny będą dzielić się na:

- ocena częściowa – pomniejsza ocena związana z pewną aktywnością ucznia na zajęciach. Może to być ocena za sprawdzian, odpowiedź lub zadanie domowe. Oprócz otrzymanego stopnia będzie wyświetlona waga oceny, informacja za co została wystawiona oraz data wystawienia.

- ocena semestralna – ocena podsumowująca pracę ucznia przez cały semestr. Jest ona wyliczana na podstawie ocen częściowych ucznia,

osobno dla każdego przedmiotu.

- ocena końcowa – główna ocena wystawiana z każdego przedmiotu po całym roku zajęć. Jest to średnia z dwóch ocen semestralnych: za pierwszy oraz za drugi semestr.

Uczeń z przedmiotu, jako ocenę końcową, może uzyskać między innymi ocenę niedostateczną. Oznacza ona, że nie uzyskuje on promocji do następnej klasy. W takim wypadku zostaje on przepisany do innej klasy (rok wcześniej) i kontynuuje swoją naukę w tej klasie.

Uczniowie należą do klas. Klasa posiada swój profil oraz wychowawcę. Między klasami z tego samego rocznika mogą występować grupy. Grupa jest to większa ilość uczniów uczestniczących razem w konkretnych zajęciach. Jedna klasa może się dzielić na grupy (na przykład w $IIIa$ mogą być dwie grupy z języka angielskiego różniące się między sobą poziomem zaawansowania) lub jedna grupa może łączyć w sobie osoby z różnych klas (na przykład dziewczyny z $IIIa$ i $IIIb$ mogą razem uczęszczać na zajęcia z wychowania fizycznego).

W przypadku uczniów *data ważności* będzie rozumiana jako przewidywana data końca jego edukacji w konkretnej placówce (w przypadku nieotrzymania promocji do następnej klasy, data ta zostanie przedłużona o rok).

2.2 Opiekun

Z perspektywy uprawnień i funkcjonalności, opiekun jest prawie takim samym użytkownikiem, co uczeń. Jedyna różnica polega na tym, że po zalogowaniu zostaje wyświetlona lista uczniów, którymi się opiekuje. Po wyborze swojego dziecka zostaną wyświetlone takie same informacje, co uczniowi (zajęcia z ich prowadzącymi na, które uczęszcza dziecko i jego oceny wraz z ich opisem). Opiekun ma również możliwość edycji danych osobowych swojego podopiecznego.

Data ważności w przypadku opiekuna to przewidywana data końca nauki jego podopiecznego (w przypadku kilku podopiecznych zostanie wzięta pod uwagę bardziej odległa data).

2.3 Nauczyciel

Osobą pełniącą główną pieczę nad ocenami jest nauczyciel. Może być przypisany do każdej z klas jako jej wychowawca. Dodatkowo każdy z nauczycieli

prowadzi zajęcia z pewnego przedmiotu. Po zalogowaniu na swoje konto, powinien móc wyświetlić wszystkie grupy, w których prowadzi zajęcia. Po wyborze konkretnej grupy, zostanie wyświetlona lista uczniów należących do niej oraz oceny, jakie do tej pory im wystawił.

Po wybraniu konkretnego ucznia nauczyciel będzie mógł edytować dotychczasowe oceny oraz dodawać nowe (wraz z odpowiednim opisem - legendą, wagą i datą). Nauczyciel samodzielnie wystawia każdą z ocen, na podstawie osiągnięć uczniów z danego przedmiotu.

W przypadku nauczyciela, *data ważności* rozumiana jest, jako data końca umowy. W przypadku przedłużenia umowy, data ulega zmianie. W przypadku dłuższego urlopu lub braku prowadzonych zajęć w danym semestrze, użytkownik będzie oznaczony jako *nieaktywny*.

2.4 Wybór technologii

Na podstawie powyższej analizy postanowiliśmy napisać osobną aplikację dla nauczycieli i uczniów - aplikacja na urządzenia mobilne z systemem Android. Natomiast panel administratorski i nauczycielski postanowiliśmy zrealizować w wersji webowej. Dla wszystkich użytkowników połączenie z internetem będzie koniecznym warunkiem korzystania z dziennika.

3 Aplikacja na Androida dla opiekuna i ucznia

Aplikacja kliencka jest napisana w androidzie aby umożliwić swobodny dostęp do danych. Aplikacja kierowana dla uczniów oraz ich opiekunów dlatego wymagania rozpisano dla obu grup.

3.1 Wymagania ogólne

1. Możliwość zalogowania (zapamiętaj login/hasło).
2. Możliwość zmiany hasła (trzeba ustawić przy pierwszym. uruchomieniu, hasło generowane przy tworzeniu użytkownika).
3. Obliczanie średniej ważonej (Wyświetla na kafelku przedmiotu).
4. Wyświetlanie przedmiotu, ocen cząstkowych z datami. wystawienia
5. Zmiana danych (mail, telefon).

3.1.1 Funkcjonalności opiekuna

1. Wybór podopiecznego dla którego wyświetlać oceny.
2. Zmiana adresu zamieszkania opiekuna oraz podopiecznych. (zastosuj dla podopiecznych).
3. Wyświetlanie danych wychowawcy, możliwość wykonania połączenia z poziomu aplikacji.

3.1.2 Funkcjonalności opcjonalne

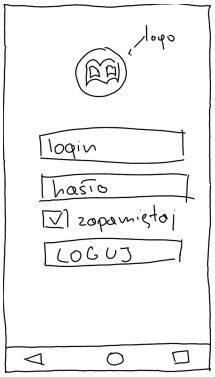
1. Oznaczenie o nowych ocenach od ostatniego logowania.
2. W zależności od płci podopiecznego w menu wyboru odpowiednia ikona.
3. W dniu urodzin życzenia.
4. Wyświetlanie pomocy.

3.2 Opis interfejsu

Po uruchomieniu aplikacji ukaże się ekran logowania. Po zalogowaniu użytkownik zostanie przeniesiony do menu głównego z którego można nawigować do poszczególnych funkcjonalności.

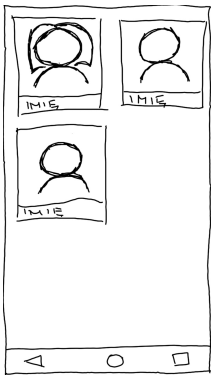
3.3 Uwagi

1. Podwójne kliknięcie wstecz w menu głównym wychodzi z aplikacji (po pierwszym kliknięciu 'toast' z informacją o tym)
2. Opiekun może tylko wyświetlić mail oraz nr. telefonu podopiecznego. może edytować jego adres zamieszkania.
3. Uczeń nie może edytować miejsca zamieszkania, może edytować mail oraz nr. telefonu.
4. Na kafelku wyboru przedmiotu wyświetla się aktualna średnia ważona przedmiotu. (Może w zależności od stopnia zmieniać kolor)



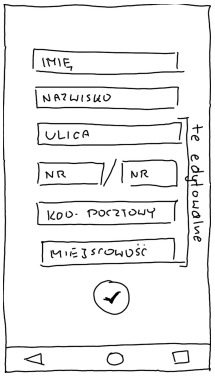
Rysunek 1: Ekran logowania

Rysunek 2: Menu główne (dla opiekuna)



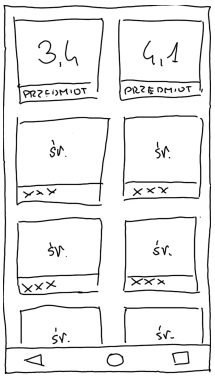
Rysunek 3: Wybór podopiecznego (dla opiekuna)

Rysunek 4: Menu główne (dla ucznia)

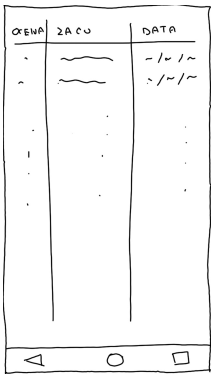


Rysunek 5: Zmiana adresu (dla opiekuna)

Rysunek 6: Zmiana hasła



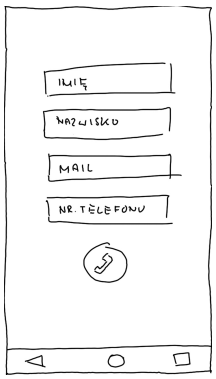
Rysunek 7: Widok główny ocen



Rysunek 8: Widok ocen z przedmiotu

Rysunek 9: Zmiana maila

Rysunek 10: Zmiana numeru



Rysunek 11: Widok danych wychowawcy

3.4 Technologia

Aplikacja kliencka działająca w systemie Android napisana w języku Java. Będzie łączyć się z serwerem za pomocą klasy HttpURLConnection. Wybór języka programowania na podstawie wspólnych preferencji oraz doświadczenia członków zespołu.

4 Aplikacja webowa dla nauczyciela oraz administratora.

4.1 Technologia

Aplikacja zostanie wykonana w technologii webowej, gdyż pozwala to na dostęp z każdego urządzenia posiadającego przeglądarkę internetową.
Skorzystamy z bootstrapa, ponieważ są gotowe szablony dla stron administratorskich.

4.2 Wymagania

- Wymagania dla panelu administratora:
  - tworzenie użytkowników oraz ich profili: nauczyciela, ucznia bądź opiekuna;
  - tworzenie klas i grup;

  - przypisywanie:
    * nauczyciela do klasy jako wychowawcy;
    * nauczyciela, grupy oraz przedmiotu do danej lekcji;
    * uczniów do klas i grup;
  - przeglądanie ocen oraz danych

- Wymagania dla panelu nauczyciela:
  - zmiana danych własnych;
  - przeglądanie, dodawanie oraz modyfikowanie ocen z własnych zajęć;
  - przeglądanie danych dot. uczniów własnej klasy oraz ich opiekunów;
  - przeglądanie ocen uczniów swojej klasy z innych zajęć;

4.3 Interfejs

Dwa różne interfejsy:

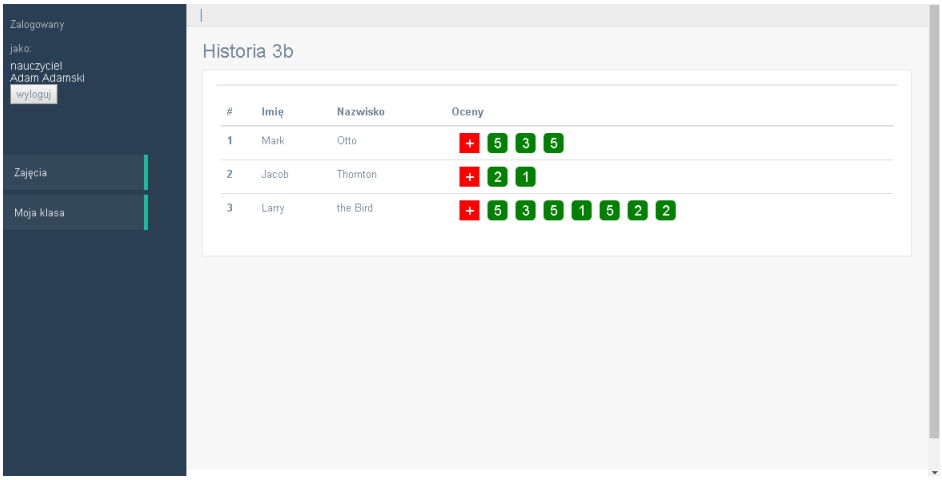
- dla nauczyciela,
- dla administratora.

Oba opierają się na idei nawigatora po lewej stronie.

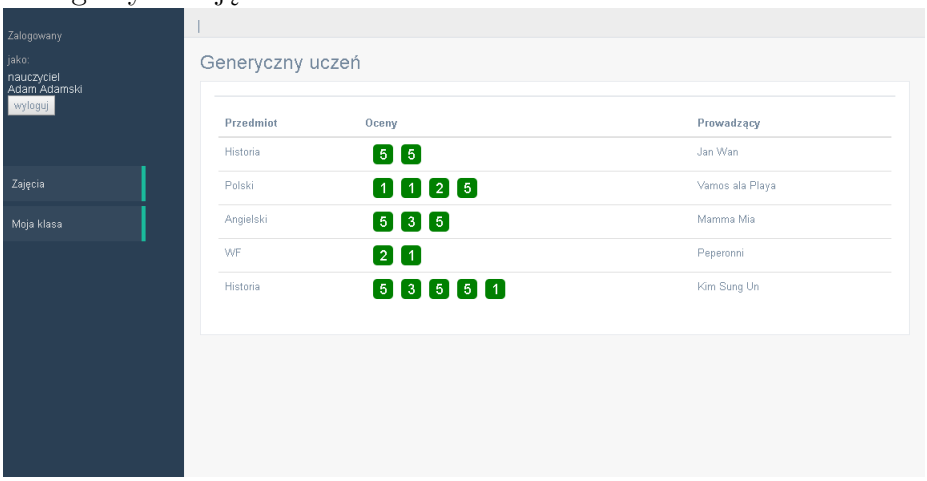
4.3.1 Interfejs nauczyciela

Interfejs nauczyciela powinien posiadać zakładki:

- Zajęcia - dostęp do zajęć prowadzonych przez danego nauczyciela. Po kliknięciu na daną lekcję odsyłany jest on do strony z tabelą z ocenami.

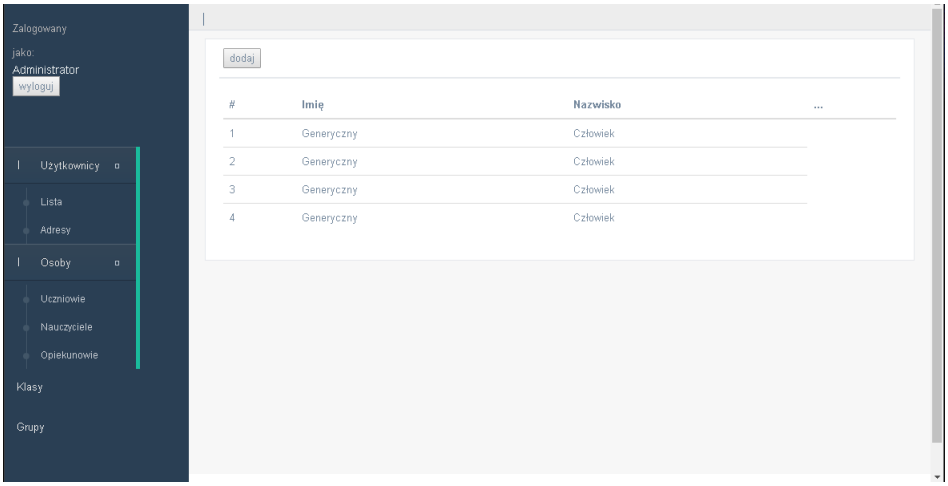


- Moja klasa - klasa, której wychowawcą jest dany nauczyciel - jeżeli nie jest wychowawcą żadnej klasy to zakładka ta nie pojawia się. Tutaj nauczyciel ma prawo przejrzeć swoich uczniów oraz ich oceny z poszczególnych zajęć.



4.3.2 Interfejs administratora

Administrator ma dostęp w pasku nawigacyjnym do listy użytkowników jak i osób, klas i grup.



5 Baza danych

5.1 Technologia

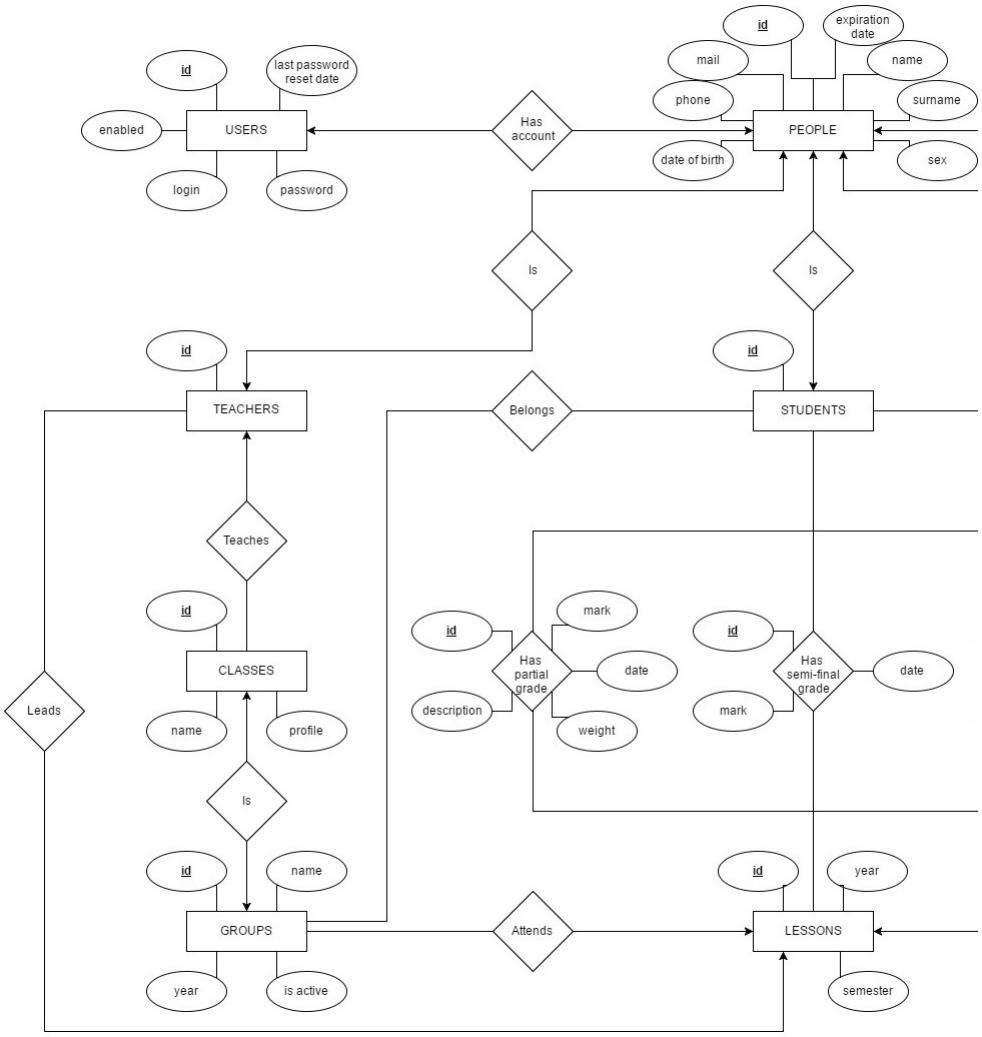
Baza danych została napisana w MySQL. Jest to ogólnodostępny system, umożliwiający zarządzanie relacyjnymi bazami danych. Zdecydowaliśmy się na niego, ponieważ jest dostępny na wielu platformach, darmowy i posiada rozbudowaną dokumentację.

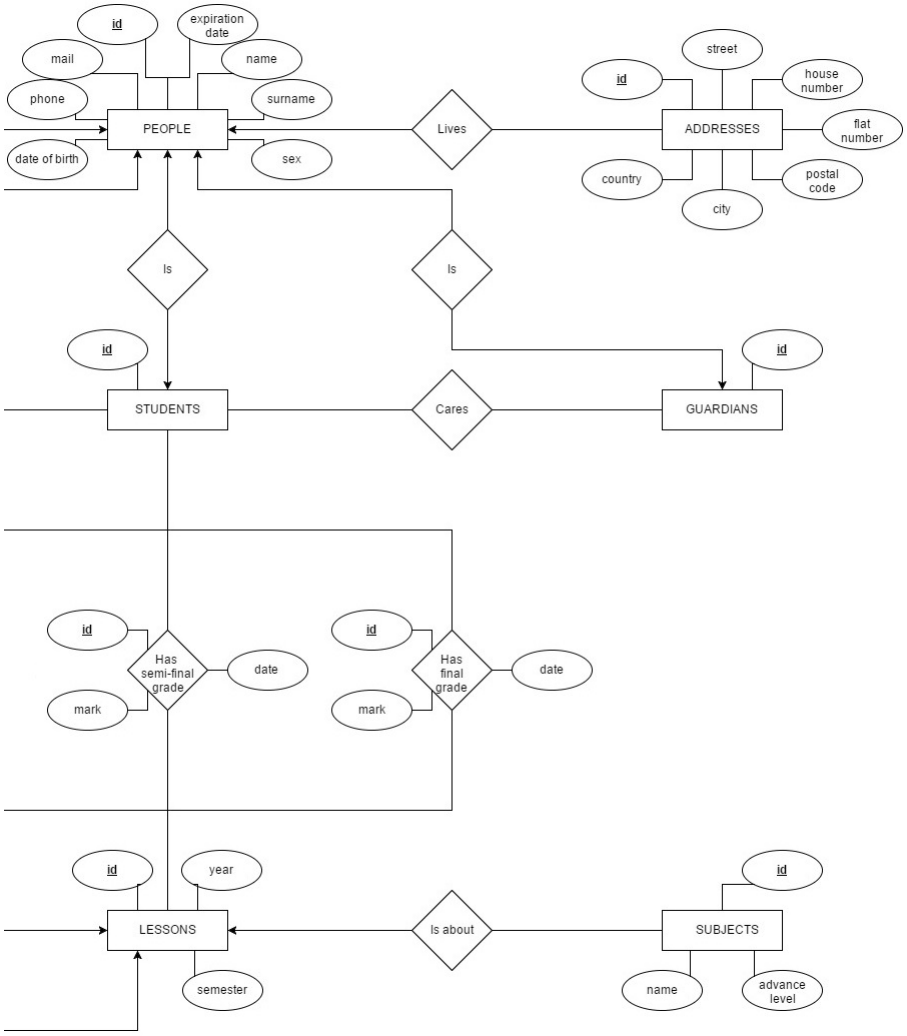
5.2 Narzędzia

Do stworzenia i zarządzania bazą danych wykorzystaliśmy panel administratorski do baz danych na serwerze *home.pl*.

Zapytania oraz połączenie z aplikacją zostały napisane w *IntelliJ*, a do podglądu stworzonych zapytań wykorzystaliśmy dodatek do Google Chrome - aplikację *Postman*.

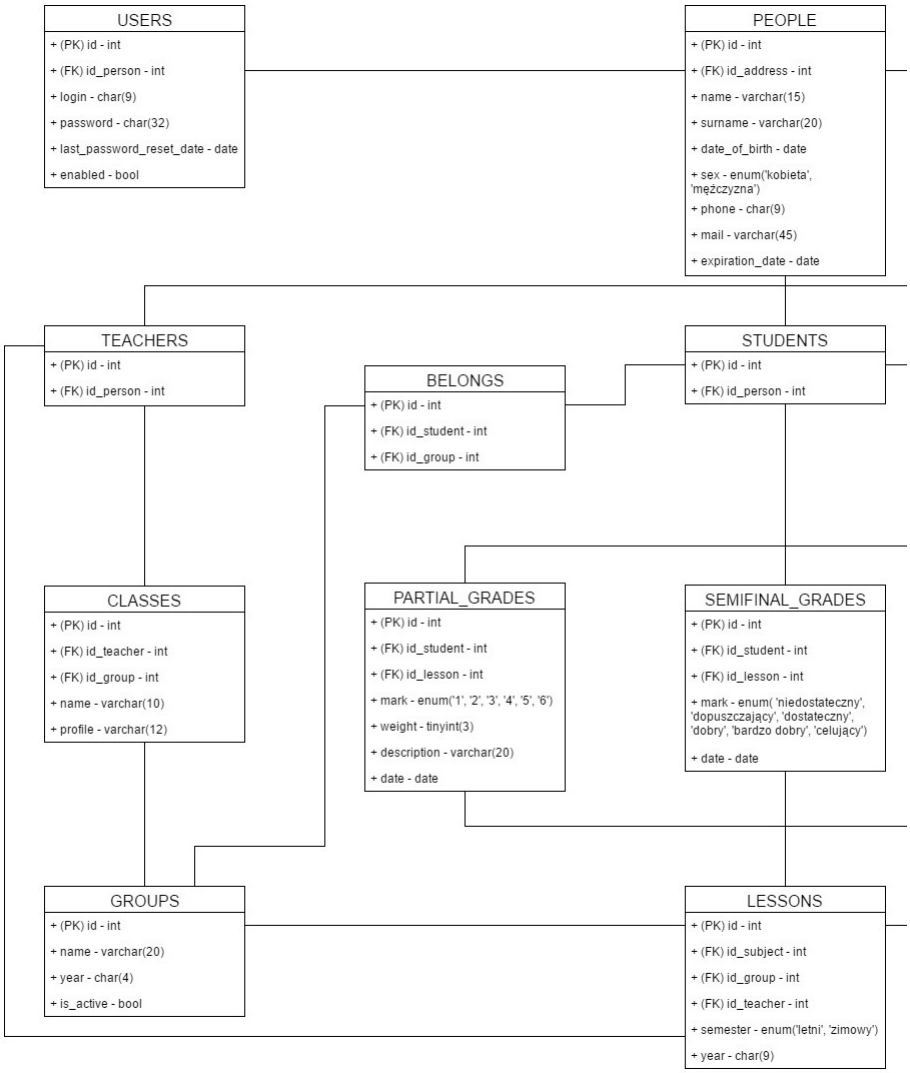
5.3 Diagram

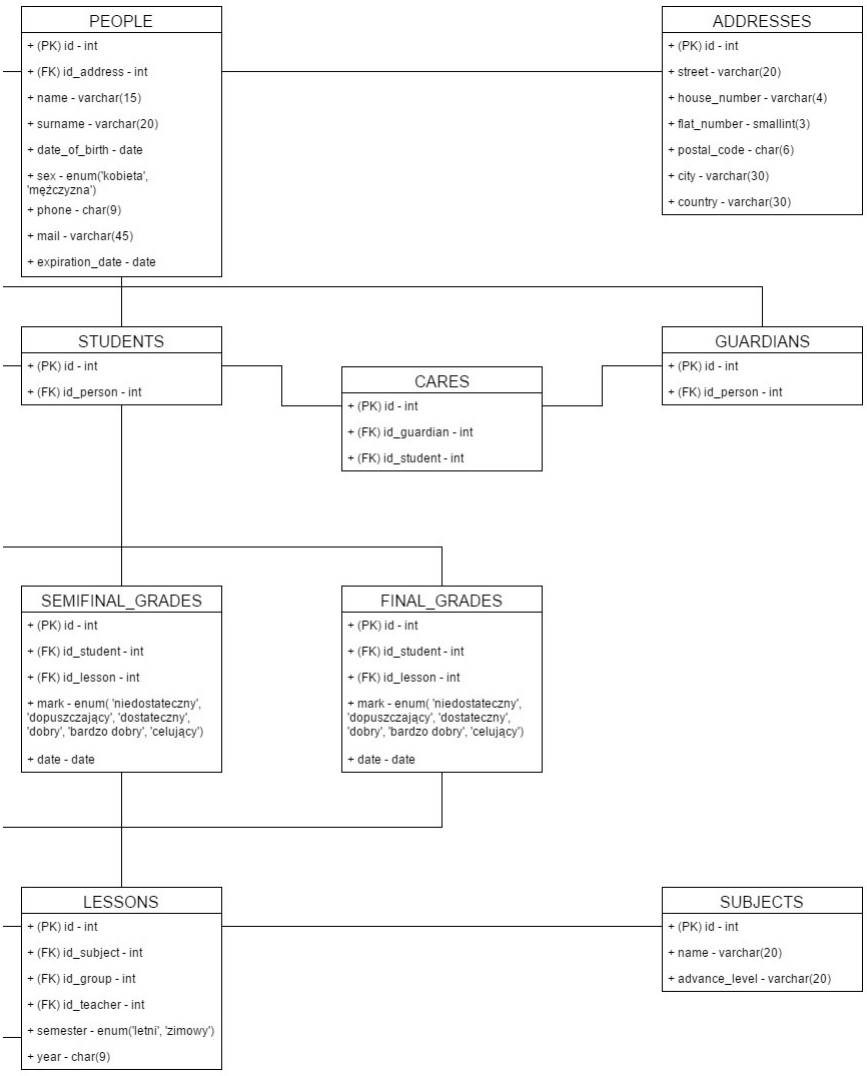

id
expiration date
mail
name
phone
surname
PEOPLE
date of birth
sex
id
last password reset date
enabled
USERS
login
password
Has account
Is
Is
id
TEACHERS
Belongs
id
STUDENTS
Teaches
id
CLASSES
name
profile
Leads
Is
id
name
GROUPS
year
is active
id
mark
Has partial grade
date
description
weight
id
Has semi-final grade
date
mark
Attends
id
year
LESSONS
semester



Przedstawione diagramy zostały przygotowane z użyciem darmowego edytora *draw.io*.

5.4 Relacyjny model bazy danych





5.5 Dane, o które pytamy

1. W trakcie logowania użytkownika:
   - czy podany login jest w bazie
   - czy dla podanego loginu użytkownik wprowadził poprawne hasło
   - kim jest osoba logująca – admin, nauczyciel, uczeń, opiekun
2. W trakcie tworzenia użytkownika przez admina:

- dodawanie nowego użytkownika, wprowadzenie nowego adresu i danych osobowych
- przypisanie użytkownika do klasy/zajęć/dziecka/opiekuna....

3. Zalogowany nauczyciel:
   - lista zajęć które prowadzi (przedmiot, grupa)
   - lista uczniów, którzy są w danej grupie
   - dodawanie ocen (uczniowi z konkretnych zajęć, data, opis, waga, stopień)
   - po dodaniu oceny aktualizacja średniej ważonej ocen
   - edycja ocen/usuwanie
4. Zalogowany opiekun/uczeń:
   - lista zajęć na które uczęszcza uczeń (nazwa przedmiotu, prowadzący)
   - lista ocen z poszczególnych zajęć (data, opis, ocena)
   - średnia ważona obliczona na podstawie wystawionych dotychczas stopni
   - dane wychowawcy klasy, do której należy uczeń
   - dla opiekuna lista jego podopiecznych
5. Zalogowany użytkownik:
   - wgląd do swoich danych z możliwością edycji (adres, mail, telefon, ...)

6 Serwer

6.1 Technologia

Usługa została napisana w języku Java, framework Spring Boot. Użyliśmy Spring Boota ze względu na dobrą dokumentacje oraz stosunkowo krótki czas konfiguracji. System logowania oparty jest na tokenach w standardzie JSON Web Token, jest to standard wolnego dostępu, dobrze udokumentowany.

6.2 Narzędzia

Korzystamy ze środowiska IntelliJ do programowania serwisu. Do komunikacji z serwerem wykorzystujemy wtyczkę do przeglądarki Google Chrome - Postman.

6.3 Opis endpointów

Z aplikacją serwerową można komunikować się poprzez restowe API.

6.3.1 Logowanie

1. "/auth"
   Opis: Generuje token do autoryzacji zapytań do serwisu na podstawie logina i hasła użytkownika.
   Metoda: POST
   Body: "{"login":"login", "password":"password"}"
   (klasa JwtCredentials) Odpowiedź: {"status":"status odpowiedzi", "token":"token"}
   (klasa JwtAuthenticationResponse)

6.3.2 Użytkownicy

1. "/EregUsers":

Opis:	Zwraca wszystkich użytkowników z bazy. Wymagana rola: TEACHER.
Metoda:	GET
Header:	{"Authorization":"wygenerowany token"}
Odpowiedź:	{"status":"status odpowiedzi", "usersList":[(lista użytkowników)]}
Przykładowa odpowiedź:	{"status": "ok", "usersList": [{ "id": 3, "login": "natbob123", "password": "12345678", "roles": "STUDENT", "lastPasswordResetDate": "2017-06-20 20:14:16", "idPerson": 1, "enable": true}, ...

2. "/EregUsers/enables":

Opis:	Zwraca wszystkich aktywnych użytkowników z bazy. Wymagana rola: TEACHER.
Metoda:	GET
Header:	{"Authorization":"wygenerowany token"}
Odpowiedź:	{"status":"status odpowiedzi", "usersList":[(lista użytkowników)]}

3. "/EregUsers/teachers":

Opis:	Zwraca wszystkich nauczycieli z bazy. Wymagana rola: TEACHER.
Metoda:	GET
Header:	{"Authorization":"wygenerowany token"}
Odpowiedź:	{"status":"status odpowiedzi", "usersList":[(lista użytkowników)]}

4. "/EregUsers/guardians":

Opis:	Zwraca wszystkich opiekunów z bazy. Wymagana rola: TEACHER.
Metoda:	GET
Header:	{"Authorization":"wygenerowany token"}
Odpowiedź:	{"status":"status odpowiedzi", "usersList":[(lista użytkowników)]}

5. "/EregUsers/students":

Opis:	Zwraca wszystkich uczniów z bazy. Wymagana rola: TEACHER.
Metoda:	GET
Header:	{"Authorization":"wygenerowany token"}
Odpowiedź:	{"status":"status odpowiedzi", "usersList":[(lista użytkowników)]}

6. "/EregUsers/id=id":

Opis:	Zwraca użytkownika na podstawie podanego id. Wymagana rola: TEACHER.
Metoda:	GET
Header:	{"Authorization":"wygenerowany token"}
Odpowiedź:	{"status":"status odpowiedzi", "user":{(użytkownik)}}

7. "/EregUsers/id=id":

Opis:	Usuwa użytkownika na podstawie podanego id.
Metoda:	DELETE
Header:	{"Authorization":"wygenerowany token"}
Odpowiedź:	{"status":"status odpowiedzi", "message":"Removed eReg user id: id" lub informacja o błędzie}

8. "/EregUsers/newPassword":

Opis:	Zmiana hasła użytkownika na podstawie loginu pobranego z tokena.
Metoda:	POST
Header:	{"Authorization":"wygenerowany token"} {"Content-Type":"application/json"}
Body:	Json ze starym i nowym hasłem, np: { "oldPassword":"12345678", "newPassword":"noweHaslo" }
Odpowiedź:	{ "status": "Ok", "message":"Login: ..., new pass: ..." lub informacja o błędzie, "token": wygenerowany token dla nowego hasła}

9. "/EregUsers":

Opis:	Dodanie nowego użytkownika o unikalnym idPerson.
Metoda:	POST
Header:	{"Authorization":"wygenerowany token"} {"Content-Type":"application/json"}
Body:	Użytkownik, na przykład: { "id": 1, "login": "karolina", "password": "12345678", "roles": "STUDENT", "lastPasswordResetDate": null, "idPerson": 40, "enable": true }
Odpowiedź:	{"status":"status odpowiedzi", "message":"Inserted user" lub informacja o błędzie}

6.3.3 Ludzie

1. "/People":

Opis:	Zwraca wszystkie osoby z bazy.
Metoda:	GET
Header:	{"Authorization":"wygenerowany token"}
Odpowiedź:	{"status":"status odpowiedzi", "people":[(lista osób)]}
Przykładowa odpowiedź:	{"status": "ok", "people": [{ "id": 1, "idUser": 3, "name": "Natalia", "surname": "Bobek", "dateOfBirth": "2010-03-08", "sex": "kobieta", "phone": "514568222", "mail": "nat.bobek@gmail.com", "expirationDate": "2019-08-30", "idAddress": 1}, ...

2. "/People/myChildren":

Opis:	Zwraca dzieci zalogowanego opiekuna. Id użytkownika wyciągane z tokenu.
Metoda:	GET
Header:	{"Authorization":"wygenerowany token"}
Odpowiedź:	{"status":"status odpowiedzi", "people":[(lista osób)]}

3. "/People/myPersonalData":

Opis:	Zwraca dane osobowe zalogowanego użytkownika. Id użytkownika wyciągane z tokenu.
Metoda:	GET
Header:	{"Authorization":"wygenerowany token"}
Odpowiedź:	{"status":"status odpowiedzi", "personalData":{(dane użytkownika)}}
Przykładowa odpowiedź:	{"status": "ok", "personalData": { "idPerson": 4, "name": "Paweł", "surname": "Mamek", "dateOfBirth": "1966-08-14", "sex": "mężczyzna", "phone": "514847817", "mail": null, "expirationDate": "2019-08-31", "idAddress": 2, "street": "Zwycięstwa", "houseNumber": "68", "flatNumber": 3, "postalCode": "44-100", "city": "Gliwice", "country": "Polska" } }

4. "/People/Person/id=idPerson":

Opis:	Zwraca dane osoby na podstawie podanego id.
Metoda:	GET
Header:	{"Authorization":"wygenerowany token"}
Odpowiedź:	{"status":"status odpowiedzi", "person":{(osoba)}}

5. "/People/Address/id=idAddress":

   Opis: Zwraca adres na podstawie podanego id.

   Metoda: GET

   Header: {"Authorization":"wygenerowany token"}

   Odpowiedź: {"status":"status odpowiedzi",
   "address":{(adres)}}

6. "/People/Person/id=idPerson":

   Opis: Usuwa osobę na podstawie podanego id.

   Metoda: DELETE

   Header: {"Authorization":"wygenerowany token"}

   Odpowiedź: {"status":"status odpowiedzi",
   "message": "Removed person id: id" lub informacja o błędzie.}

7. "/People/Address/id=idAddress":

   Opis: Usuwa adres na podstawie podanego id.

   Metoda: DELETE

   Header: {"Authorization":"wygenerowany token"}

   Odpowiedź: {"status":"status odpowiedzi",
   "message":"Removed address id: id" lub informacja o błędzie}

8. "/People/updatePhone":

Opis:	Zmiana telefonu użytkownika.
Metoda:	POST
Header:	{"Authorization":"wygenerowany token"} {"Content-Type":"application/json"}
Body:	Json z id osoby i nowym numerem, np: { "idPerson": 1, "newPhone": "111111" }
Odpowiedź:	{ "status": "Ok", "message":"Updated phone." lub informacja o błędzie}

9. "/People/updateMail":

Opis:	Zmiana maila użytkownika.
Metoda:	POST
Header:	{"Authorization":"wygenerowany token"} {"Content-Type":"application/json"}
Body:	Json z id osoby i nowym mailem, np: { "idPerson": 1, "newMail": "marysia@gmail.com" }
Odpowiedź:	{ "status": "Ok", "message":"Updated mail." lub informacja o błędzie}

10. "/People/updateExpirationDate":

Opis:	Zmiana "daty ważności" użytkownika. Na przykład daty końca umowy (nauczyciel).
Metoda:	POST
Header:	{"Authorization":"wygenerowany token"} {"Content-Type":"application/json"}
Body:	Json z id osoby i nową datą, np: { "idPerson": 1, "expirationDate": "2018-07-15" }
Odpowiedź:	{ "status": "Ok", "message":"Updated expiration date." lub informacja o błędzie}

11. "/People/updateIdAddress":

Opis:	Zmiana id adresu użytkownika.
Metoda:	POST
Header:	{"Authorization":"wygenerowany token"} {"Content-Type":"application/json"}
Body:	Json z id osoby i nowym id adresu, np: { "idPerson": 1, "newIdAddress": "4" }
Odpowiedź:	{ "status": "Ok", "message":"Updated id address." lub informacja o błędzie}

12. "/People/updateAddress":

Opis:	Zmiana adresu.
Metoda:	POST
Header:	{"Authorization":"wygenerowany token"} {"Content-Type":"application/json"}
Body:	Json z id adresu i nowymi danymi, np: { "idAddress":1, "street":"Pierwszego Maja", "houseNumber":"22b", "flatNumber":5, "postalCode":"44-100", "city":"Gliwice", "country":"Polska" }
Odpowiedź:	{ "status": "Ok", "message":"Updated address." lub informacja o błędzie}

13. "/People/newPersonWithAddress":

Opis:	Dodanie nowej osoby wraz z adresem.
Metoda:	POST
Header:	{"Authorization":"wygenerowany token"} {"Content-Type":"application/json"}
Body:	Osoba oraz adres, na przykład: { "address": { "idAddress":1, "street":"Dolnych Wałów", "houseNumber":"22b", "flatNumber":5, "postalCode":"44-100", "city":"Gliwice", "country":"Polska" }, "person": { "id": 1, "name": "Bartłomiej", "surname": "Kaczkowski", "dateOfBirth": "1982-04-28", "sex": "kobieta", "phone": "511511511", "mail": "dwdwdwd", "expirationDate": "2019-12-12", "idAddress": 4 } }
Odpowiedź:	{"status":"status odpowiedzi", "message":"Inserted person with address." lub informacja o błędzie}

14. "/People/newAddress":

Opis:	Dodanie nowego adresu.
Metoda:	POST
Header:	{"Authorization":"wygenerowany token"} {"Content-Type":"application/json"}
Body:	Adres.
Odpowiedź:	{"status":"status odpowiedzi", "message":"Inserted address." lub informacja o błędzie}

15. "/People/newPerson":

Opis:	Dodanie nowej osoby.
Metoda:	POST
Header:	{"Authorization":"wygenerowany token"} {"Content-Type":"application/json"}
Body:	Osoba.
Odpowiedź:	{"status":"status odpowiedzi", "message":"Inserted person." lub informacja o błędzie}

6.3.4 Grupy

1. "/Groups/teacher/myGroups":

Opis:	Lista grup, w których użytkownik prowadzi zajęcia. Id użytkownika wyciągane z tokenu.
Metoda:	GET
Header:	{"Authorization":"wygenerowany token"}
Odpowiedź:	{"status":"status odpowiedzi", "groups":[(lista grup)]}
Przykładowa odpowiedź:	{"status": "ok", "groups": [{ "id": 1, "name": "IB", "year": null, "active": false}, ...

2. "/Groups/attendingStudents/idGroup":

Opis:	Lista uczniów należących do danej grupy.
Metoda:	GET
Header:	{"Authorization":"wygenerowany token"}
Odpowiedź:	{"status":"status odpowiedzi", "people":[(lista osób)]}

3. "/Groups/myClass":

Opis:	Klasa, do której należy uczeń. Id użytkownika wyciągane z tokenu.
Metoda:	GET
Header:	{"Authorization":"wygenerowany token"}
Odpowiedź:	{"status":"status odpowiedzi", "groupClass":{(klasa}}
Przykładowa odpowiedź:	{"status": "ok", "groupClass": [{ "name": "IVA", "profile": null, "idTutor": 7, "tutorName": "Andrzej", "tutorSurname": "Maciejewski", "tutorPhone": "506481354", "tutorMail": "andrzej.maciejewski@o2.pl"}, ...

4. "/Groups/userClass/userId=userId":

Opis:	Klasa, do której należy podany użytkownik.
Metoda:	GET
Header:	{"Authorization":"wygenerowany token"}
Odpowiedź:	{"status":"status odpowiedzi", "groupClass":{(klasa)}}

6.3.5 Lekcje

1. "/Lessons":

Opis:	Zwraca wszystkie lekcje.
Metoda:	GET
Header:	{"Authorization":"wygenerowany token"}
Odpowiedź:	{"status":"status odpowiedzi", "lessons":[(lista lekcji)]}
Przykładowa odpowiedź:	{"status": "ok", "lessons": [{ "id": 1, "year": "2016/2017", "semester": "letni", "idTeacher": 4, "teacherName": "Ziemowit", "teacherSurname": "Chmielewski", "idGroup": 1, "groupName": "IA", "idSubject": 1, "subjectName": "język polski" }, ...

2. "/Lessons/teacher/myLessons":

Opis:	Zwraca lekcje prowadzone przez zalogowanego nauczyciela. Id użytkownika wyciągane z tokenu.
Metoda:	GET
Header:	{"Authorization":"wygenerowany token"}
Odpowiedź:	{"status":"status odpowiedzi", "lessons":[(lista lekcji)]}

3. "/Lessons/attendingGroup/id=idGroup":

Opis:	Zwraca lekcje odbywające się w podanej grupie.
Metoda:	GET
Header:	{"Authorization":"wygenerowany token"}
Odpowiedź:	{"status":"status odpowiedzi", "lessons":[(lista lekcji)]}

4. "/Lessons/student/myLessons":

Opis:	Zwraca lekcje zalogowane studenta. Id użytkownika wyciągane z tokenu.
Metoda:	GET
Header:	{"Authorization":"wygenerowany token"}
Odpowiedź:	{"status":"status odpowiedzi", "lessons":[(lista lekcji)]}

5. "/Lessons/student/lessons/userId=id":

Opis:	Zwraca lekcje studenta na podstawie podanego id użytkownika.
Metoda:	GET
Header:	{"Authorization":"wygenerowany token"}
Odpowiedź:	{"status":"status odpowiedzi", "lessons":[(lista lekcji)]}

6. "/Lessons/id=id":

Opis:	Usuwa lekcję na podstawie podanego id.
Metoda:	DELETE
Header:	{"Authorization":"wygenerowany token"}
Odpowiedź:	{"status":"status odpowiedzi", "message": "Removed lesson id: id" lub informacja o błędzie.}

7. "/Lessons/updateTeacher":

Opis:	Zmiana prowadzącego lekcję.
Metoda:	POST
Header:	{"Authorization":"wygenerowany token"} {"Content-Type":"application/json"}
Body:	Json z id lekcji i nowym id nauczyciela, np: { "idLesson": 40, "idTeacher": "7" }
Odpowiedź:	{ "status": "Ok", "message":"Updated teacher." lub informacja o błędzie}

8. "/Lessons/updateSemester":

Opis:	Zmiana semestu.
Metoda:	POST
Header:	{"Authorization":"wygenerowany token"} {"Content-Type":"application/json"}
Body:	Json z id lekcji i nowym semestrem, np: { "idLesson": 40, "semester": "zimowy" }
Odpowiedź:	{ "status": "Ok", "message":"Updated semester." lub informacja o błędzie}

9. "/Lessons":

Opis:	Dodanie nowej lekcji.
Metoda:	POST
Header:	{"Authorization":"wygenerowany token"} {"Content-Type":"application/json"}
Body:	Lekcja, na przykład: { "id": null, "year": "2016/2017", "semester": "letni", "idTeacher": 4, "teacherName": "", "teacherSurname": "", "idGroup": 1, "groupName": "", "idSubject": 1, "subjectName": "" }
Odpowiedź:	{"status":"status odpowiedzi", "message":"Inserted lesson." lub informacja o błędzie}

6.3.6 Grupy

1. "/Groups/teacher/myGroups":

Opis:	Lista grup, w których użytkownik prowadzi zajęcia. Id użytkownika wyciągane z tokenu.
Metoda:	GET
Header:	{"Authorization":"wygenerowany token"}
Odpowiedź:	{"status":"status odpowiedzi", "groups":[(lista grup)]}
Przykładowa odpowiedź:	{"status": "ok", "groups": [{ "id": 1, "name": "IB", "year": null, "active": false}, ...

2. "/Groups/attendingStudents/idGroup":

Opis:	Lista uczniów należących do danej grupy.
Metoda:	GET
Header:	{"Authorization":"wygenerowany token"}
Odpowiedź:	{"status":"status odpowiedzi", "people":[(lista osób)]}

3. "/Groups/student/myClass":

Opis:	Klasa, do której należy uczeń. Id użytkownika wyciągane z tokenu.
Metoda:	GET
Header:	{"Authorization":"wygenerowany token"}
Odpowiedź:	{"status":"status odpowiedzi", "groupClass":{(klasa}}
Przykładowa odpowiedź:	{"status": "ok", "groupClass": [{ "name": "IVA", "profile": null, "idTutor": 7, "tutorName": "Andrzej", "tutorSurname": "Maciejewski", "tutorPhone": "506481354", "tutorMail": "andrzej.maciejewski@o2.pl"}, ...

4. "/Groups/studentClass/userId=userId":

Opis:	Klasa, do której należy podany użytkownik (student).
Metoda:	GET
Header:	{"Authorization":"wygenerowany token"}
Odpowiedź:	{"status":"status odpowiedzi", "groupClass":{(klasa)}}

5. "/Groups/teacher/myClass":

Opis:	Klasa, którą uczy nauczyciel. Id użytkownika wyciągane z tokenu.
Metoda:	GET
Header:	{"Authorization":"wygenerowany token"}
Odpowiedź:	{"status":"status odpowiedzi", "groupClass":{(klasa)}}

6. "/Groups/teacherClass/userId=userId":

Opis:	Klasa, którą uczy podany użytkownik.
Metoda:	GET
Header:	{"Authorization":"wygenerowany token"}
Odpowiedź:	{"status":"status odpowiedzi", "groupClass":{(klasa)}}

6.3.7 Oceny

1. "/Grades/myPartialGrades":

Opis:	Posortowana (by id lesson) lista cząstkowych ocen zalogowanego użytko Id użytkownika wyciągane z tokenu.
Metoda:	GET
Header:	{"Authorization":"wygenerowany token"}
Odpowiedź:	{"status":"status odpowiedzi", "grades":[(tablica ocen)]}
Przykładowa odpowiedź:	{"status": "ok", "grades": [{ "id": 1, "mark": "5+", "weight": 1, "description": null, "date": "2016-10-19", "idStudent": 5, "idLesson": 4, "subjectName": "język polski"}, ...

2. "/Grades/myPartialGrades/idLesson":

Opis:	Lista cząstkowych ocen zalogowanego użytkownika z danej lekcji. Id użytkownika wyciągane z tokenu.
Metoda:	GET
Header:	{"Authorization":"wygenerowany token"}
Odpowiedź:	{"status":"status odpowiedzi", "grades":[(tablica ocen)]}

3. Analogicznie: "mySemifinalGrades" i "myFinalGrades" dla ocen semestralnych oraz końcowych,

4. "/userPartialGrades/idUser":

Opis:	Lista cząstkowych ocen dla użytkownika o podanym id.
Metoda:	GET
Header:	{"Authorization":"wygenerowany token"}
Odpowiedź:	{"status":"status odpowiedzi", "grades":[(tablica ocen)]}

5. "/userPartialGrades/idUser/lesson/idLesson":

Opis:	Lista cząstkowych ocen dla użytkownika o podanym id z danej lekcji.
Metoda:	GET
Header:	{"Authorization":"wygenerowany token"}
Odpowiedź:	{"status":"status odpowiedzi", "grades":[(tablica ocen)]}

6. Analogicznie "userSemifinalGrades" i "userFinalGrades" dla ocen semestralnych i końcowych.

6.4 Opis usługi

Do aplikacji można się logować jako uczeń ('STUDENT'), opiekun ('GUARDIAN'), nauczyciel ('TEACHER'). Każda z ról ma różny dostęp do API, dostęp do endopintu jest ustawiany za pomocą adnotacji @PreAuthorize("hasRole('TEACHER')").