**AKADEMIA GÓRNICZO-HUTNICZA**

**IM. STANISŁAWA STASZICA W KRAKOWIE**



**Wydział Informatyki, Elektroniki i Telekomunikacji**

Katedra Informatyki

**Dokumentacja Deweloperska**

**Interaktywny system wspomagania układania rozkładu zajęć**

Roger Barlik, Krzysztof Nowakowski,

Krystian Ujma, Krzysztof Wróbel

Promotor: Mgr inż. Witold Rakoczy

OŚWIADCZENIE AUTORÓW PRACY

Oświadczamy, świadomi odpowiedzialności karnej za poświadczenie nieprawdy, że niniejszą pracę dyplomową wykonaliśmy osobiście i samodzielnie (w zakresie wyszczególnionym we wstępie) i że nie korzystaliśmy ze źródeł innych niż wymienione w pracy

Spis treści

[1. Wstęp 5](#_Toc424064777)

[2. Baza danych 5](#_Toc424064778)

[3. Moduł Spring 5](#_Toc424064779)

[4. Obiekty wykorzystywane w systemie 5](#_Toc424064780)

[4.1. Agregat 5](#_Toc424064781)

[4.2. Autoryzacja 6](#_Toc424064782)

[4.3. Typ autoryzacji 6](#_Toc424064783)

[4.4. Awatar 7](#_Toc424064784)

[4.5. Sala 7](#_Toc424064785)

[4.6. Salo-godzina 8](#_Toc424064786)

[4.7. Konkretna data 8](#_Toc424064787)

[4.8. Szablon 9](#_Toc424064788)

[4.9. Konkretna Lekcja 9](#_Toc424064789)

[4.10. Jednostka programowa 9](#_Toc424064790)

[4.11. Grupa studencka 10](#_Toc424064791)

[4.12. Wykładowca 10](#_Toc424064792)

[4.13. Użytkownik 11](#_Toc424064793)

[5. Obiekty niewykorzystane 12](#_Toc424064794)

[5.1. Wydział 12](#_Toc424064795)

[5.2. Kierunek 12](#_Toc424064796)

[5.3. Oznaczenie grupy dziekanatowej 12](#_Toc424064797)

[5.4. Typ jednostki programowej 12](#_Toc424064798)

[5.5. Szablon planu lekcji 13](#_Toc424064799)

[5.6. Semestr 13](#_Toc424064800)

[6. Niezrealizowane funkcję, rozszerzenia 14](#_Toc424064801)

[6.1. Raportowanie błędnych akcji 14](#_Toc424064802)

[6.2. Wczytywanie danych 14](#_Toc424064803)

[6.3. Ładowanie danych z pliku 14](#_Toc424064804)

[6.4. Widoki szczegółowe 14](#_Toc424064805)

[6.5. Skrzynka alertów 14](#_Toc424064806)

[7. Problemy 14](#_Toc424064807)

[7.1. Drag and drop w kontekście wielu okien 14](#_Toc424064808)

[8. Propozycje rozwoju 15](#_Toc424064809)

[8.1. Dostęp wykładowców 15](#_Toc424064810)

[8.2. Skrzynka wiadomości 15](#_Toc424064811)

# Wstęp

Dokumentacja ta jest dokumentacją deweloperską przeznaczona dla maintainerów. Powstała ona aby opisać różnice między specyfikacją oraz realizacją projektu

Omówione w tej dokumentacji post implementacyjnej są takie zagadnienia jak: sposób implementacji, używane obiekty, niezrealizowane ulepszenia, rozszerzenia oraz problemy napotkane oraz dopiero przewidywane.

# Baza danych

Baza danych została stworzona przy użyciu systemu bazodanowego PostgreSQL. Baza została zaprojektowana i stworzona zgodnie z pełną specyfikacją wymagań co sprawiło że pojawiły się w niej tabele obsługujące funkcję oraz rozszerzenia, które nie zostały wykorzystane w module frontend w obecnej wersji systemu. W następnych podpunktach przedstawiony jest przegląd poszczególnych tabel przemapowanych na obiekty entity oraz DAO wraz z podziałem ze względu czy zostały wykorzystane w obecnej wersji systemu.

# Moduł Spring

W module Spring został zastosowany plugin Spring Boot pozwalający na uruchomianie systemu bez potrzeby kopiowania folderu war na serwer. Spring Boot ułatwia konfigurowanie modułu Spring, poprzez generowanie automatycznie dispatcher servlet oraz szereg dodatkowych ułatwień. Dokumentacja pluginu dostępna jest pod adresem: <http://projects.spring.io/spring-boot/>.

# Obiekty wykorzystywane w systemie

Poniżej przedstawiony jest spis obiektów wykorzystywanych w systemie wraz z opisem ich zastosowania.

## Agregat

Agregat jest przedstawieniem połączenia wszystkich danych tworzących przesuwalny prostokąt umieszczany na kalendarzu.

**Definicja:**

public class Aggregate {

private Long id;

private Avatar avatar;

private ProgrammeUnit programmeUnit;

private StudentGroup studentGroup;

}

**id:** unikalny klucz agregatu

**avatar:** awatar przypisany do danego agregatu

**programmeUnit:** - jednostka programowa przypisana do danego agregatu

**studentGroup:** grupa studencka przypisana do danego agregatu

## Autoryzacja

Obiekt przygotowany do przechowywania danych autoryzacyjnych.

**Definicja:**

public class Authorization {

private Long id;

private AuthorizationType type;

private User user;

}

**id:** unikalny klucz autoryzacji

**type:** rodzaj dostępu do którego przypisany jest użytkownik

**user:** nazwa użytkownika

## Typ autoryzacji

Obiekt przechowujący rodzaj dostępu dostępny dla użytkownika

**Definicja:**

public class AuthorizationType {

private Long id;

private String name;

}

**id:** unikalny klucz typu autoryzacji

**name:** rodzaj użytkownika

## Awatar

Awatar przedstawia połączenie kilku wykładowców, którzy wspólnie prowadzą daną jednostkę podstawy programowej (przedmiot) dla danej grupy studenckiej.

**Definicja:**

public class Avatar {

private Long id;

private String name;

private Set<Aggregate> aggregates;

private Set<Teacher> teachers;

}

**id:** unikalny klucz awatara

**name:** unikalna nazwa awatara

**aggregates:** agregaty do których przypisany jest dany awatar

**teachers:** wykładowcy przypisani do danego awataru

## Sala

Obiekt sala jak sama nazwa wskazuje przechowuje informacje na temat danej Sali.

**Definicja:**

public class Classroom{

private Long id;

private String number;

private int buildingId;

private int numberOfSeats;

private Set<ClassroomHour> classroomHours;

}

**id:** unikalny klucz sali

**number:** numer sali

**buildingId:** numer budynku w którym znajduje się sala

**numberOfSeats:** ilość miejsc dostępnych w Sali

## Salo-godzina

Obiekt jest połączeniem Sali oraz czasu w którym odbywa się lekcja.

**Definicja:**

public class ClassroomHour{

private Long id;

private int lessonBeginId;

private int lessonEndId;

private Classroom classroom;

}

**id:** unikalny klucz salo-sodziny

**lessonBeginId**: id bloku w którym zaczyna się lekcja

**lessonEndId**: id bloku w którym kończy się lekcja

**classroom:** przypisana sala

## Konkretna data

Konkretna data odzwierciedla konkretny punkt w czasie (rok, miesiąc, dzień)

**Definicja:**

public class ConcreteDate{

private Long id;

private Date realDate;

private ConcreteDateTemplate concreteDateTemplate;

}

**id:** unikalny klucz konkretnej daty

**realDate:** unikalna data konkretne daty

**concreteDateTemplate:** Szablon do którego konkretna data należy

## Szablon

Obiekt służy do łączenie poszczególnych dni których plan zajęć jest identyczny w szablony

**Definicja:**

public class ConcreteDateTemplate{

private Long id;

private String name;

private Set<ConcreteDate> concreteDates;

private TermsSet termsSet;

}

**id:** unikalny klucz szablonu

**name:** unikalna nazwa szablonu

**concreteDates:** dni należące do szablonu

**termsSet:** semestr do którego szablon jest podpięty

## Konkretna Lekcja

Obiekt ten przedstawia pojedynczą kratkę na kalendarzu opisaną datą i godziną, do której może być dołączony agregat.

**Definicja:**

public class ConcreteLesson {

private Long id;

private Aggregate aggregate;

private Schedule schedule;

private ClassroomHour classroomHour;

private ConcreteDate concreteDate;

}

**id:** unikalny klucz konkretnej lekcji

**aggregate:** agregatprzypisany do konkretnej lekcji

**schedule:** Szablon do którego został przypisana konkretna lekcja

**classromHour:** godzina odbywania się lekcji w danej sali

**concreteDate:** data odbywania się lekcji

## Jednostka programowa

Obiekt przechowuje informacje na temat przedmiotu lekcyjnego

**Definicja:**

public class ProgrammeUnit{

private Long id;

private String name;

private ProgrammeUnitType type;

private Set<Aggregate> aggregates;

}

**id:** unikalny klucz jednostki programowej

**name:** nazwa jednostki programowej

**type:** typprowadzonych zajęć (wykład/ćwiczenia/projekt)

**aggregates:** agregaty stworzone do danego przedmiotu

## Grupa studencka

Obiekt opisuje dane grupy studenckiej

**Definicja:**

public class StudentGroup{

private Long id;

private Date startDate;

private Date endDate;

private Department department;

private LetterDesignation letterDesignation;

private Faculty faculty;

private Set<Aggregate> aggregates;

}

**id:** unikalny klucz grupy studenckiej

**startDate:** data rozpoczęcia studiów

**endDate:** data zakończenia studiów

**department:** wydział na którym grupa studiuje

**letterDesignation:** znak grupy studenckiej

**faculty:** kierunek na którym studiuje grupa studencka

**aggregates:** agregaty podpięte do grupy studenckiej

## Wykładowca

Obiekt opisujący wykładowcę

**Definicja:**

public class Teacher{

private Long id;

private String name;

private String lastName;

private Avatar avatar;

}

**id:** unikalny klucz wykładowcy

**name:** imię wykładowcy

**lastName**: nazwisko wykładowcy

**avatar:** awatar do którego podpięty jest wykładowca

## Użytkownik

Obiekt przechowujący dane na temat użytkownika

**Definicja:**

public class User {

private Long id;

private String login;

private String password;

private String passwordMd5;

private Set<Authorization> authorizations;

private boolean isLoggedIn;

}

**id:** unikalny klucz użytkownika

**login:** login użytkownika

**password:** hasło użytkownika

**passwordMd5:** zaszyfrowane hasło użytkownika

**authorizations:** autoryzacje przypisane do użytkownika

**isLoggedIn:** flaga zalogowania

# Obiekty niewykorzystane

## Wydział

Obiekt przechowuje wydział którego dotyczy plan. Nie jest używany w obecnej wersji systemu

**Definicja:**

public class Department{

private Long id;

private String name;

}

**id:** unikalny klucz wydziału

**name:** unikalna nazwa wydziału

## Kierunek

Obiekt przechowuje kierunek którego dotyczy plan. Nie jest używany w obecnej wersji systemu

**Definicja:**

public class Faculty {

private Long id;

private String name;

}

**id:** unikalny klucz kierunku

**name:** nazwa kierunku

## Oznaczenie grupy dziekanatowej

Obiekt przechowuje znak grupy dziekanatowej

**Definicja:**

public class LetterDesignation{

private int id;

private String name;

}

**id:** unikalny klucz znaku grupy dziekanatowej

**name:** znak grupy dziekanatowej

## Typ jednostki programowej

Obiekt przechowuje rodzaj zajęć

**Definicja:**

public class ProgrammeUnitType{

private Long id;

private String name;

}

**id:** unikalny klucz typu jednostki programowej

**name:** nazwa typu jednostki programowej

## Szablon planu lekcji

Definiuje główny szablon planu lekcji stworzony przez użytkownika

**Definicja:**

public class Schedule{

private Long id;

private Set<ConcreteLesson> concreteLessons;

private String name;

private Date date;

private User user;

}

**id:** unikalny klucz szablonu lekcji

**concreteLessons:** konkretne lekcje należące do szablonu

**name:** unikalnanazwa szablonu

**date:** data stworzenia

**user:** właściciel szablonu

## Semestr

Obiekt opisuje semester

**Definicja:**

public class TermsSet {

private Long id;

private Set<ConcreteDateTemplate> concreteDateTemplates;

}

**id:** unikalny klucz semestru

**concreteDateTemplates:** szablony podpięte do semestru

# Niezrealizowane funkcję, rozszerzenia

W tym podpunkcie omówione zostaną wszystkie funkcję oraz rozszerzenia, które były planowane ale nie zostały zrealizowane.

## Raportowanie błędnych akcji

W obecnej wersji systemu przy używaniu funkcji *drag and drop* pomiędzy przeglądarkami jeżeli nad dany obiekt zostanie przeciągnięty obiekt, może z nim wchodzić w interakcję, zostaje wywołana zaprogramowana funkcja i baza jest odpowiednio zmieniana.

W przypadku gdy obiekt zostanie błędnie przeciągnięty nie następuje żadna akcja ponieważ przeglądarki w takim przypadku nie widzą siebie nawzajem. Aby zakomunikować błędną akcję należy rozwiązać ten problem. Może to być osiągnięte za pomocą web socketów, po stronie backendu mogą być to websockety Java.

## Wczytywanie danych

W obecnej wersji systemu wszystkie dane wczytywane są metodą *eager loading*. Zaimplementowany jest obiekt Dozer, który mapuje encje na obiekty DTO. Ta metoda wczytywania jest wygodna przy małej ilości danych, jednak wraz z ich wzrostem znacząco spowalnia system. Należy zaimplementować wczytywanie danych poprzez metodę *lazy-loading.*

## Ładowanie danych z pliku

W obecnej wersji nie została zaimplementowany mechanizm wczytywania danych niezmieniających się w planie: wykładowców, grup studenckich, przedmiotów z spreparowanego pliku. Wczytywanie z pliku wymaga wypracowania sposobu organizacji pliku oraz odpowiedniego sposobu parsowania pliku.

## Widoki szczegółowe

W systemie nie została zaimplementowana opcja wyboru i otworzenia widoku szczegółowego dla danego podmiotu, np. dla danego wykładowcy widok ze wszystkimi przedmiotami które prowadzi na siatce kalendarza.

## Skrzynka alertów

W systemie nie została stworzona skrzynka wyświetlająca widomości alertowe związane z konfliktami agregatów na kalendarzu

# Problemy

W tym rozdziale opisane zostały problemy na jakie natknął się zespól w czasie tworzenia systemu.

## Drag and drop w kontekście wielu okien

W czasie tworzenia systemu głównym problemem na jaki natknął się zespól było użycie metodu *drag and drop* w wielu oknach. W czasie rozwiązywania tego problemu zostały przetestowanych kilka technologii tworzenia stron internetowych jak GWT czy Jquery.

Większość testowanych technologii pozwalało na używanie *drag and drop* tylko w ramach jednego okna. Zespól doszedł do wniosku że jedynym dostępnym rozwiązaniem tego problemu jest zastosowania technologii HTML5.

# Propozycje rozwoju

Przedstawiamy tutaj dodatkowe rozszerzenia które mogą być stworzone w celu usprawnienia lub rozbudowania systemu.

## Dostęp wykładowców

Dostęp dla wykładowców w celu zapoznania się z widokiem szczegółowym dostępnym dla danego wykładowcy

## Skrzynka wiadomości

Udostępnienie wykładowcą komunikacji z planistą w ramach systemu w celu wyrażenia opinii na temat planu zajęć.