Sprawozdanie 1

Baza Danych

Paweł Gacek i Aleksander Pytel, 16.04.2022

1. Temat Projektu

Aplikacja okienkowa umożliwiająca studentom zapisy na zajęcia. (Technologie: PostgreSql i Hibernate)

2. Schemat bazy i opis tabel

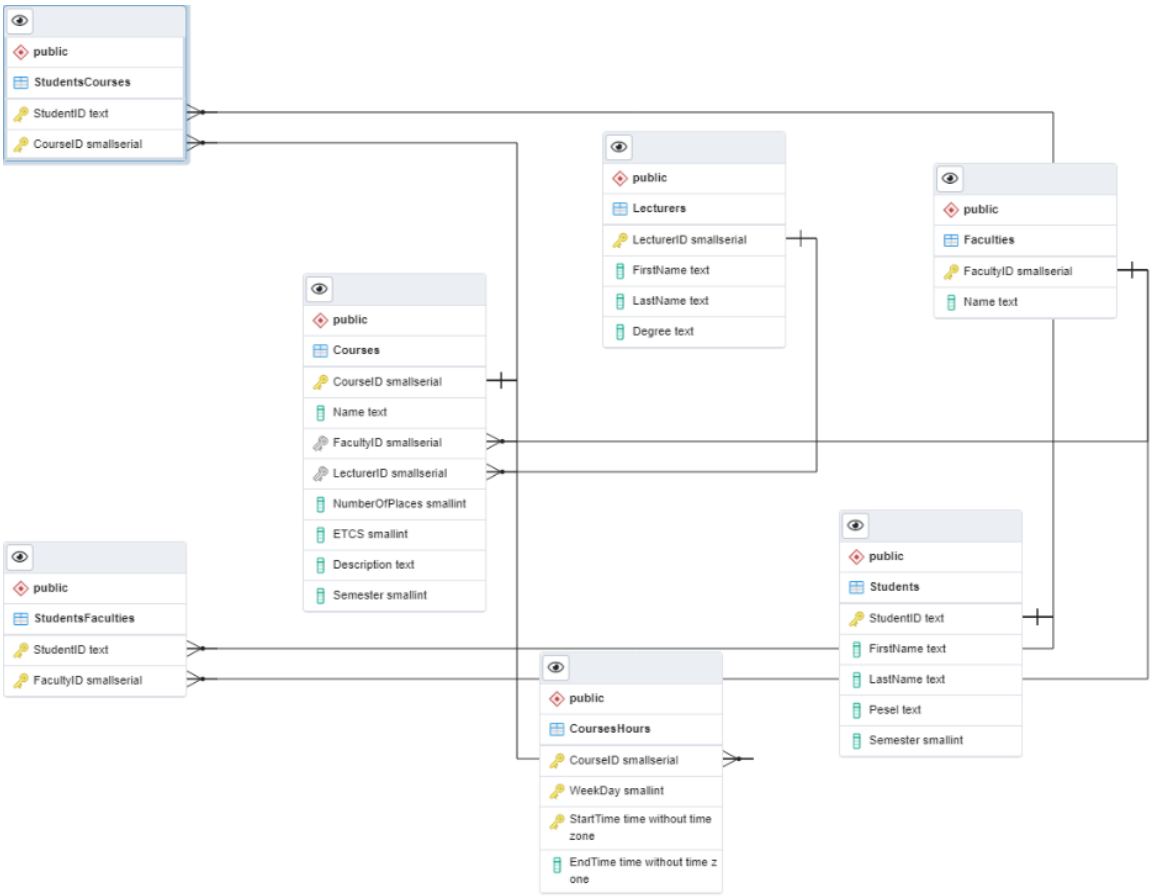
a) Idea

Student po zalogowaniu się do aplikacji będzie mógł zapisać się na zajęcia przypisane do swojego wydziału i semestru oraz zobaczyć zajęcia, na które już się zapisał. Aplikacja uniemożliwi zapis na zajęcia, których termin będzie kolidował z innymi zajęciami, na które student już się zapisał oraz na zajęcia, które nie mają wolnych miejsc. Student może filtrować zajęcia po dacie, dostępności, punktach ETCS. Po kliknięciu na konkretne zajęcia ukaże mu się ich opis i wykładowca prowadzący zajęcia. Student nie może przekroczyć maksymalnej ilości punktów ETCS.

b) Baza składa się 7 tabel:

- **Students** – zawiera podstawowe informacje o studencie, kluczem głównym jest indeks
- **Lecturers** – zawiera informacje o wykładowcach prowadzących zajęcia. Informacje te pokazywane będą po wejściu przez studenta w dany kurs.
- **Courses** – zawiera informacje o zajęciach takie jak nazwa, ilość wolnych miejsc, liczba punktów ETCS, opis, semestr
- **Faculties** – zawiera nazwy wydziałów
- **StudentsCurses** – zawiera informację o tym, jaki student jest zapisany na jakie zajęcia
- **StudentsFaculties** – zawiera informację na jakich wydziałach są studenci (jeden student może studiować na wielu wydziałach)
- **CoursesHours** – zawiera terminy zajęć (zajęcia mogą odbywać się częściej niż raz w tygodniu)

c) Schemat



3. Funkcje, procedury, triggery

a) Funkcje

- availablePlaces(courseID) – zwraca ilość wolnych miejsc na zajęciach

```
create or replace function availablePlaces (courseID smallint)
returns smallint
language plpgsql
as $$
declare
    takenPlaces integer;
    totalPlaces integer;

begin
    if not exists(select * from "Courses" where "CourseID" = courseid) then
        raise exception 'Course not found';
    end if;
    select c."NumberOfPlaces" into totalPlaces from "Courses" as c
       where c."CourseID" = courseID;

    select COUNT(*) into takenPlaces from "StudentsCourses" as sc
       where sc."CourseID" = CourseID;

    return totalPlaces - takenPlaces;
end
$$;
```

- getCourses(studentID) – zwraca zajęcia odbywające się na semestrze i wydziałach studenta

```
create or replace function getCourses(StudentID text)
returns table (
    "CourseID" smallint,
    "Name" text,
    "FacultyID" smallint,
    "LecturerID" smallint,
    "NumberOfPlaces" smallint,
    "ETCS" smallint,
    "Description" text
) as $$
declare
       ids smallint[];
       student "Students"%rowtype;
begin
       select  * into student from "Students" as s where StudentID = s."StudentID";
       if student is null then
              raise exception 'Student not found';
       end if;
       select ARRAY(select sf."FacultyID" from "StudentsFaculties" as sf where
       sf."StudentID" = StudentID) into ids;

       return query
              select c."CourseID",c."Name",c."FacultyID",c."LecturerID",
              c."NumberOfPlaces",c."ETCS",c."Description" from "Courses" as c
              where c."Semester" = student."Semester" and c."FacultyID" = any(ids);
end
$$ Language plpgsql;
```

b) Procedury

c) Triggery

- enrollTrigger – sprawdza czy student może się zapisać na zajęcia

```
CREATE or replace FUNCTION canEnroll()
    returns TRIGGER
    language plpgsql
as
$$
declare
    course "Courses"%ROWTYPE;
    student "Students"%ROWTYPE;
begin
    if availableplaces(new."CourseID") <= 0 then
        raise exception 'no places available';
    end if;

    select * into course from "Courses" c where c."CourseID" = new."CourseID";
    select * into student from "Students" s where s."StudentID" = new."StudentID";

    if course."Semester" != student."Semester" then
        raise exception 'Wrong semester';
    end if;

    if not exists(select * from "StudentsFaculties" sf where sf."StudentID" =
    student."StudentID" and sf."FacultyID" = course."FacultyID") then
        raise exception 'Wrong faculty';
    end if;

    return new;
end;
$$;

CREATE TRIGGER enrollTrigger
    before INSERT on "StudentsCourses"
    for each row execute procedure canEnroll();
```