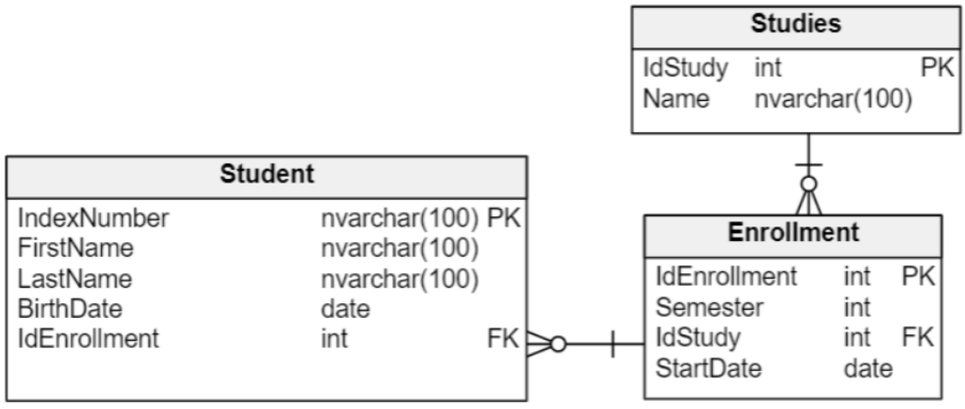
# APBD - Tutorial 5

pgago@pja.edu.pl

April 2, 2020

## 1 Introduction

During this exercise, we will expand project from the previous tutorial. As a reminder, a diagram of the database is presented below.



## Task 1 - enroll student in the existing studies

Due to new requirements, we need to add a new endpoint to our code that will allow us to register a new student and enroll him to the existing studies.

- Add a new controller named EnrollmentsController.
- The controller will handle requests sent to /api/enrollment
- Controller must have a method, that allows to add a new Student and enroll him in the semester.
- Sample request below:

```
POST /api/enrollments HTTP/1.1
Host: localhost:51932
Content-Type: application/json

{
    "IndexNumber": "s12345",
"FirstName": "Jogn",
"LastName": "Doe",
"BirthDate": "30.03.1993"
"Studies": "IT"
}
```

- Endpoint must check if all required data has been delivered. If not, return error 400. Next we have to check if studies from the request exists in the Studies table. If not, return error 400. For existing studies, find the latest entry in the Enrollments table with value Semester = 1 (student enrolls in the first semester). If such record doesn't exist, we must insert it with StartDate set as a current date. In the next step a new Student must be created. Remember about checking if index number provided in the request is not assigned to other student (otherwise return an error).
- All actions described above must happen within single transaction. In case of any error we have to rollback all changes. Use SqlConnection class and method BeginTransaction() (presented on lecture).
- After successfull enrollment of the student return code 201. Include the Enrollment object (with semester to which Student has been enrolled) in body.

## Task 2 - promoting students

In this task we want to add a new endpoint, responsible for promoting students to the next semester.

- Add a new endpoint to the Enrollments controller. The endpoint expects POST requests sent to /api/enrollments/promotions. Sample request presented below:

```
POST /api/enrollments/promotions HTTP/1.1
Host: localhost:51932
Content-Type: application/json

{
"Studies": "IT",
"Semester": 1
}
```

- Check if Enrollment table contains provided Studies and Semester. Otherwise return 404 (Not Found).
- Add a new stored procedure to your database. The procedure shuld expect parameters for Studies and Semester. Our endpoint will execute the procedure to update all Students assigned to a given Semester and Studies. Firstly, we must find a proper record in Enrollment table (given Studies and Semester increased by 1). If such record doesn't exists we must add a new one. To complete the process update IdEnrollment value for all promoted students.
- As we use stored procedure, this time keep all logic on the database side (within the procedure).
- After succesfull promotion return code 201 with the Enrollment object found (or created) in previous steps.

## Task 3 - separated layer responsible for communication with database

In the last step we will move all database logic to the class outside of our controller.

- Add a new folder Services.
- Within the folder create an interface IStudentsDbService. The interface must define all required methods used in our controllers.
- Add an implementation of IStudentsDbService. This is the place where we should keep the code using SqlConnection and SqlCommand.
- Data access layer must be injected by the constructor of our controllers. Remember about registering dependencies in Startup.cs class.
- At the end of this task, all database logic should be in the separated class.