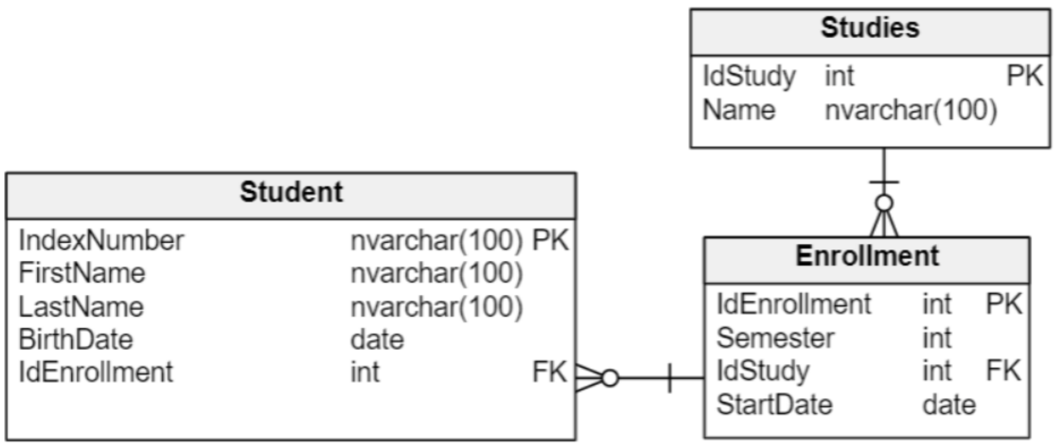
APBD - tutorial 6

pgago@pja.edu.pl

April 10, 2020

1 Introduction

In this tutorial we are going to improve our API by adding a documentation and two middlewares. Work on your projects from previous tutorials.



1. Adding middleware - checking the index number

In this task we will add our own middleware. Remember, that adding middleware and connecting it to the so-called "pipeline" (related to the processing of HTTP requests) takes place in the Startup.cs class and the Configure method.
Our task is to add the middleware that will check if all requests to our API come from students. For this reason, we expect thatevery request to have a student number in the header of the request stored under the key "Index".
In addition, our middleware should check if such a student actually exists in the database.
Below we have a piece of code to complete. Remember that the order of adding middleware to the pipeline is important. Authentication should occur at the beginning.
If the "Index" header is not present in the request - return a 401 error (Unathorized).

```
app.Use(async (context, next) =>
{
    //Custom code
    await next();
});
```

2. Adding middleware - logging requests to a file

In this case, we would like to create an additional middleware that will log basic information about all incoming requests. This time we will create middleware in a separate class. To do this, we should add a new folder called Middlewares. Within the folder we should add a class similar to the one presented below. Our code will be placed in method InvokeAsync. The goal is to write all this information to file requestsLog.txt:

1. HTTP Method (GET, POST, etc.)
2. Endpoint path (/api/students)
3. Body of each request (e.g. JSON sent)
4. Query strings (?name=Kowalski)

```
public class LoggingMiddleware
{
    private readonly RequestDelegate _next;

    public LoggingMiddleware(RequestDelegate next)
    {
        _next = next;
    }

    public async Task InvokeAsync(HttpContext httpContext)
    {
        //Our code
        await _next(httpContext);
    }
}
```

In the Startup.cs class we must run method UseMiddleware. To register the middleware, add the line in the Startup.cs -> Configure class:

```
app.UseMiddleware<LoggingMiddleware>();
```