Nowakowski Michał

Sprawozdanie potwierdzające działanie

Spis treści

1. Utworzyć repozytorium git o nazwie np. PU_KOL1..... 2

2. W EntityFramework należy zaimplementować następującą strukturę bazy danych: a. Student – ID, imię, nazwisko, IDGrupy (opcjonalne). b. Grupa – ID, nazwa. c. Historia – ID, imię, nazwisko, idGrupy, typ_akcji (usuwanie lub edycja), data. 2

3. Należy zaimplementować operacje CRUD dla tabeli Student i wystawić je w WebAPI – w implementacji mają się znaleźć następujące warstwy BLL, BLL_EF, DAL oraz WebAPI, które ma operować tylko na klasach DTO. 4

4. Należy dodatkowo wystawić stronnicowane wyświetlanie tabeli Historia. 19

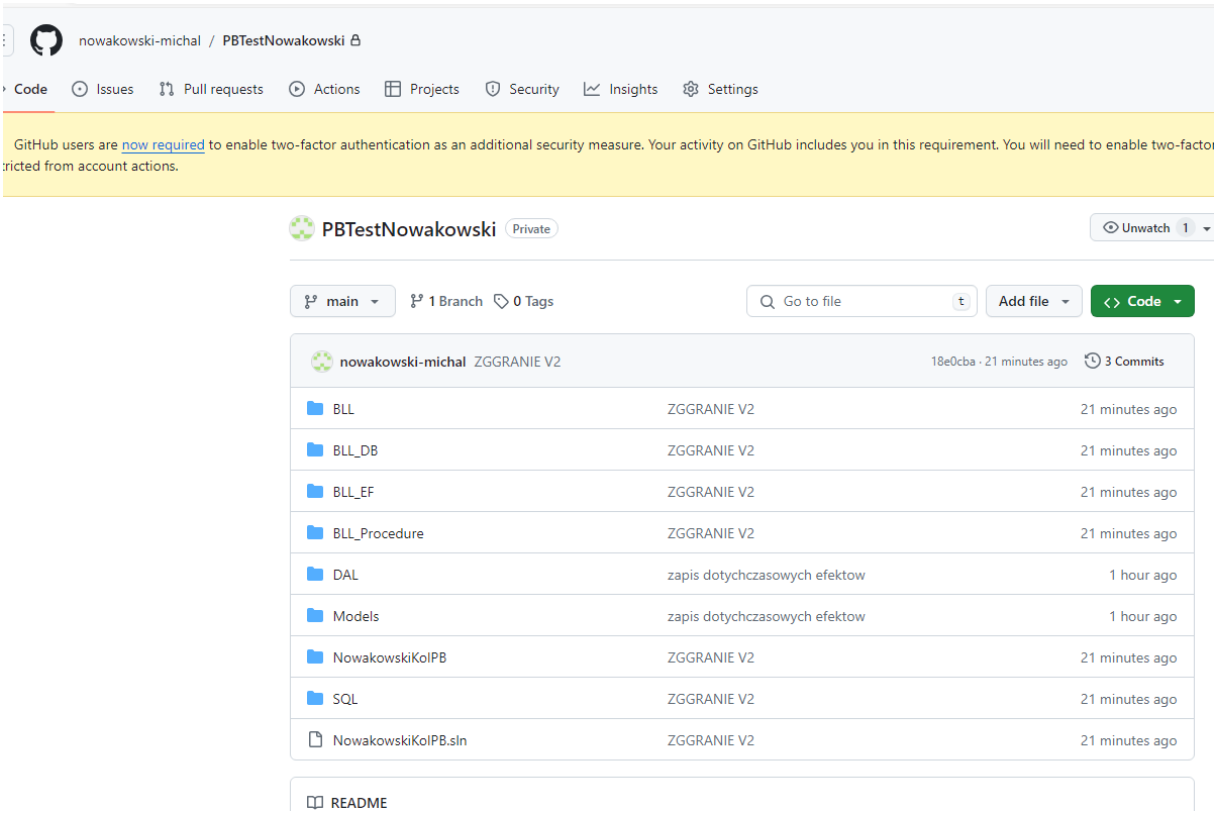
5. Przerób implementację dodawania studenta na procedurę składowaną. 22

6. Przerób stronnicowane wyświetlanie tabeli Historia na procedurę składowaną. 25

7. Dodaj triggera, który po każdej modyfikacji wpisu w tabeli Student doda wpis do tabeli Historia ze imieniem, nazwiskiem i idGrupy sprzed edycji, typem akcji edycja oraz aktualną datą. Przy usuwaniu trigger powinien wstawić imię, nazwisko i idGrupy usuwanego rekordu, typ akcji usuwanie oraz aktualną datę..... 28

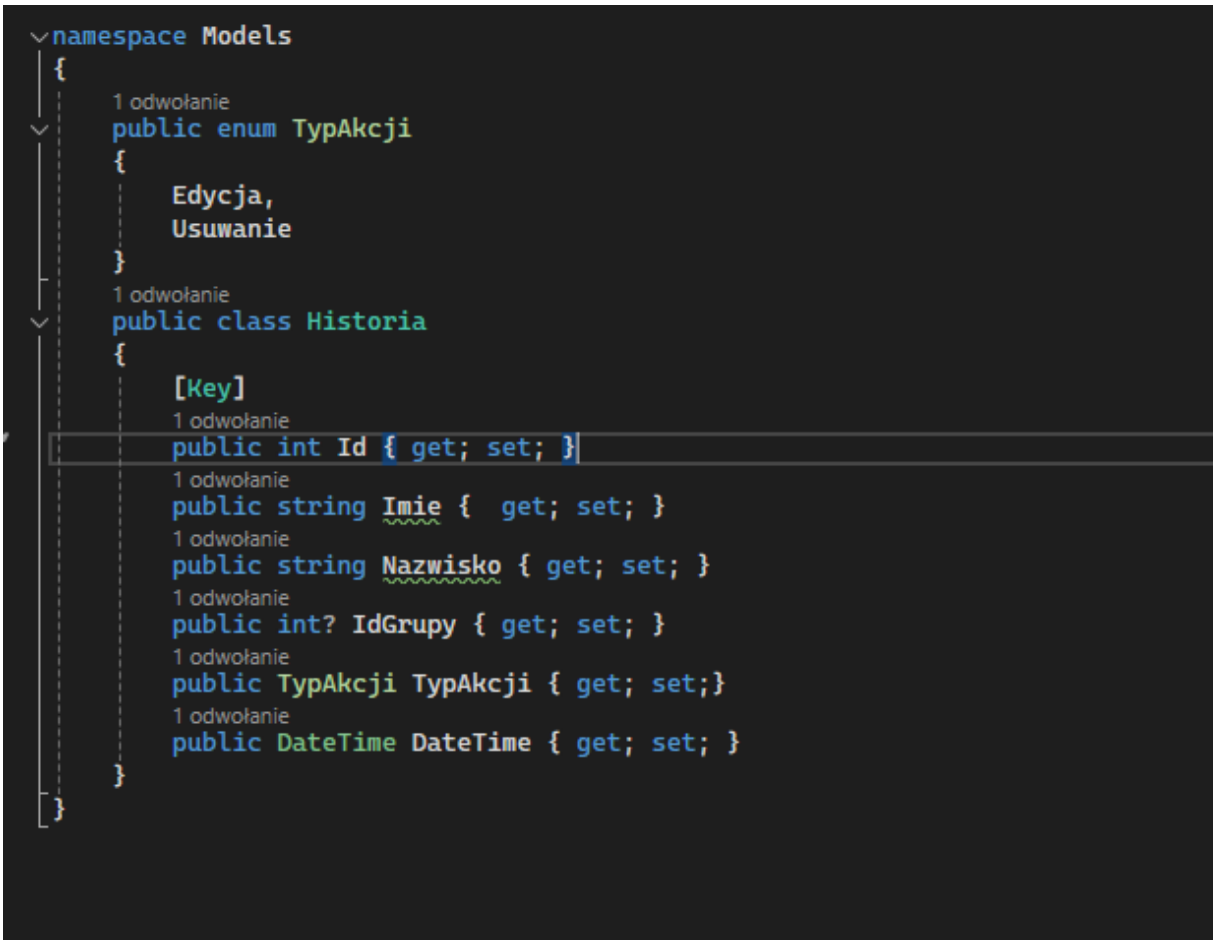
8. Rozbuduj encję grupa o relacje rekurencyjną. Zaimplementuj w SQL pobieranie pełnej nazwy grupy dla danego ID. Przykładowo jeśli w bazie posiadamy grupę o nazwie „trzy”, której rodzicem jest grupa o nazwie „dwa”, której rodzicem jest grupa o nazwie „jeden” to pełna nazwa dla grupy „trzy” będzie wyglądać następująco „jeden / dwa / trzy”. Wystaw tą funkcjonalność w WebAPI 28

1. Utworzyć repozytorium git o nazwie np. PU_KOL1.

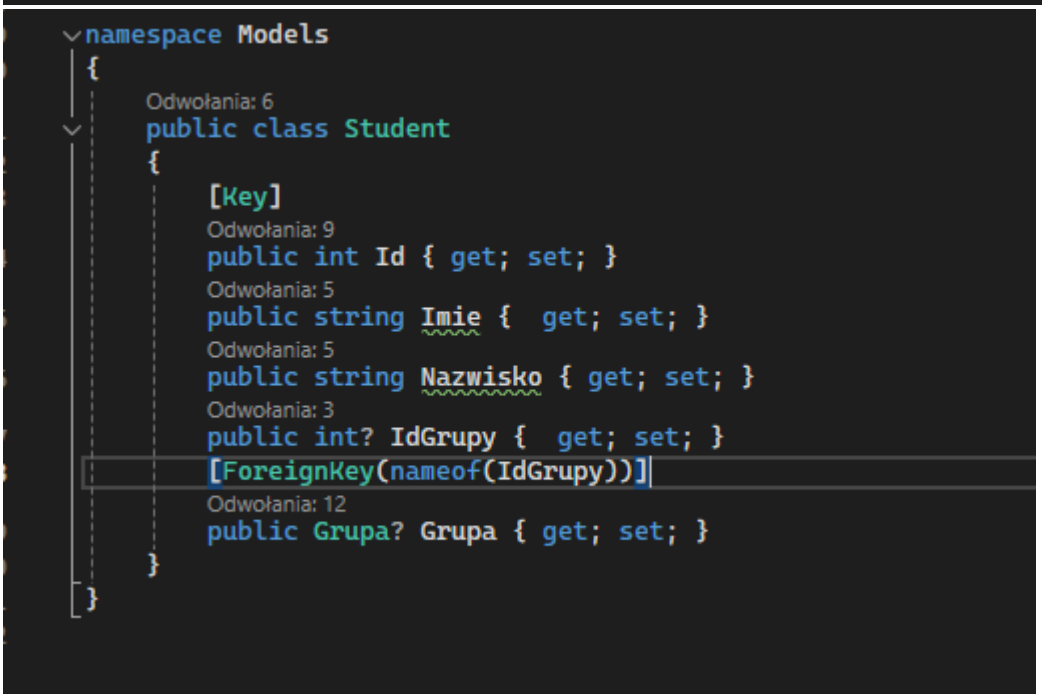


2. W EntityFramework należy zaimplementować następującą strukturę bazy danych: a. Student – ID, imię, nazwisko, IDGrupy (opcjonalne). b. Grupa – ID, nazwa. c. Historia – ID, imię, nazwisko, idGrupy, typ_akcji (usuwanie lub edycja), data.

```
namespace Models
{
    Odwołania: 7
    public class Grupa
    {
        [Key]
        Odwołania: 2
        public int Id { get; set; }
        Odwołania: 9
        public string Nazwa { get; set; }
        1 odwołanie
        public List<Student>? Students { get; set; }
        //RELACJA REKURENCYJNA
        Odwołania: 3
        public int? ParentId { get; set; }
        [ForeignKey(nameof(ParentId))]
        1 odwołanie
        public Grupa? Parent { get; set; }
        1 odwołanie
        public List<Grupa>? Children { get; set; }


        //między historia a grupa nie widzę sensu relacji wiec jej nie tworzę
        //tabela ma przechowywać tylko id jak było powiedziane, a nazwę grupy moge pobrać z relacji student-grupa
    }

}
```

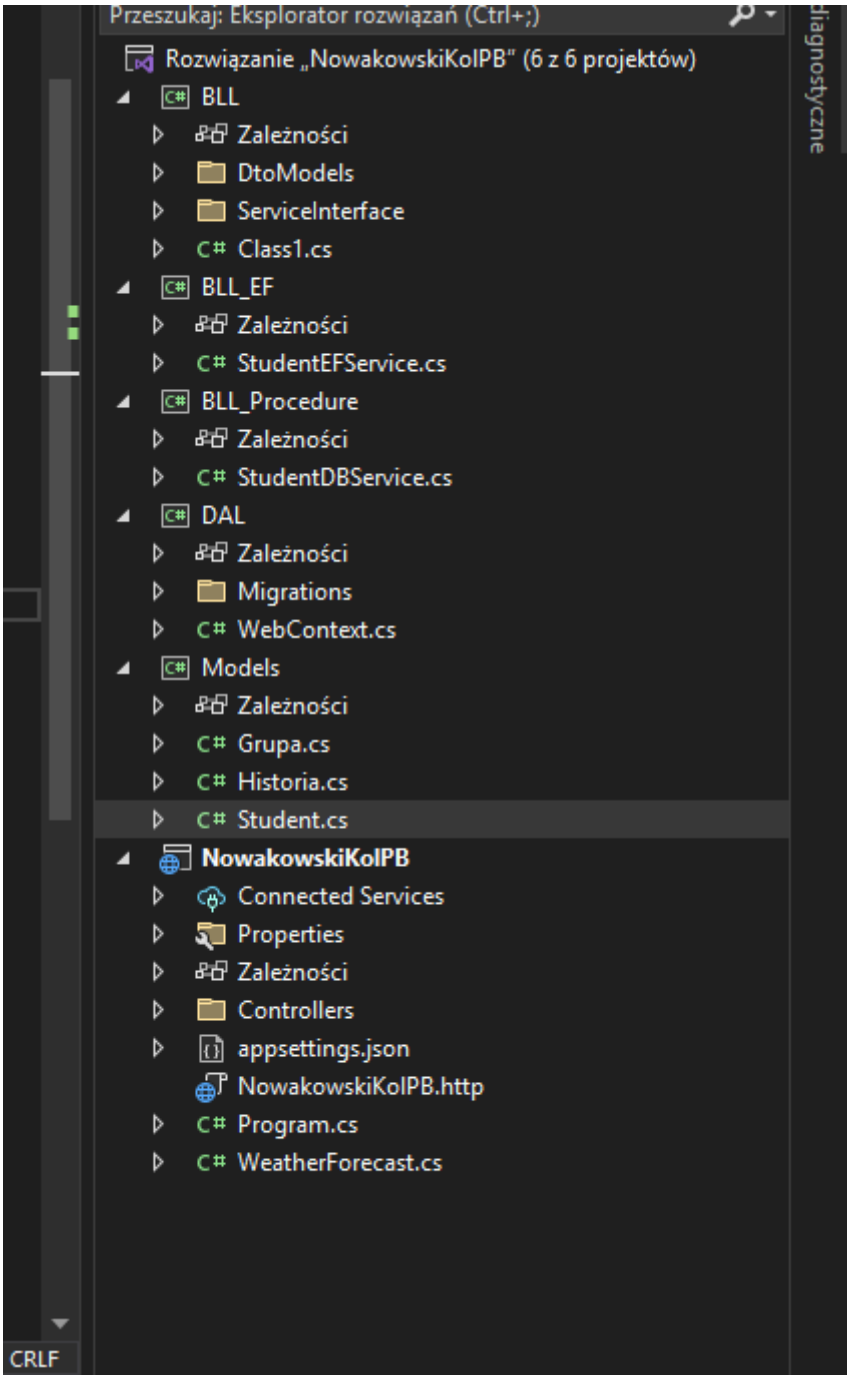


```
namespace Models
{
    1 odwołanie
    public enum TypAkcji
    {
        Edycja,
        Usuwanie
    }
    1 odwołanie
    public class Historia
    {
        [Key]
        1 odwołanie
        public int Id { get; set; }
        1 odwołanie
        public string Imie {  get; set; }
        1 odwołanie
        public string Nazwisko { get; set; }
        1 odwołanie
        public int? IdGrupy { get; set; }
        1 odwołanie
        public TypAkcji TypAkcji { get; set;}
        1 odwołanie
        public DateTime DateTime { get; set; }
    }
}
```



```
namespace Models
{
    Odwołania: 6
    public class Student
    {
        [Key]
        Odwołania: 9
        public int Id { get; set; }
        Odwołania: 5
        public string Imie {  get; set; }
        Odwołania: 5
        public string Nazwisko { get; set; }
        Odwołania: 3
        public int? IdGrupy {  get; set; }
        [ForeignKey(nameof(IdGrupy))]
        Odwołania: 12
        public Grupa? Grupa { get; set; }
    }
}
```

3. Należy zaimplementować operacje CRUD dla tabeli Student i wystawić je w WebAPI – w implementacji mają się znaleźć następujące warstwy BLL, BLL_EF, DAL oraz WebAPI, które ma operować tylko na klasach DTO.



Klasy DTO

Do testu paginacji:

```
namespace BLL.DtoModels
{
    Odwołania: 0
    public class HistoriaRequestDTO
    {
        Odwołania: 0
        public string Imie { get; init; }
        Odwołania: 0
        public string Nazwisko { get; init; }
        Odwołania: 0
        public int? IdGrupy { get; init; }
        Odwołania: 0
        public int TypAkcji { get; init; }
        Odwołania: 0
        public DateTime DateTime { get; init; }
    }
}
```

```
namespace BLL.DtoModels
{
    Odwołania: 9
    public class HistoriaResponseDTO
    {
        Odwołania: 3
        public int Id { get; init; }
        Odwołania: 2
        public string Imie { get; init; }
        Odwołania: 2
        public string Nazwisko { get; init; }
        Odwołania: 2
        public string NazwaGrupy { get; init; }
        Odwołania: 2
        public string TypAkcji { get; init; }
        Odwołania: 2
        public DateTime DateTime { get; init; }
    }
}
```

```
Odwołania: 6
public class StudentRequestDTO
{
    Odwołania: 2
    public string Imie { get; init; }
    Odwołania: 2
    public string Nazwisko { get; init; }
    Odwołania: 2
    public int? IdGrupy { get; init; }

}
```

```
namespace BLL.DtoModels
{
    Odwołania: 15
    public class StudentResponseDTO
    {
        Odwołania: 5
        public int Id { get; init; }
        Odwołania: 5
        public string Imie { get; init; }
        Odwołania: 5
        public string Nazwisko { get; init; }
        Odwołania: 5
        public string NazwaGrupy { get; init; }
    }
}
```

Interfejs

Procedury

```
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace BLL.ServiceInterface
{
    Odwołania: 4
    public interface IStudentDB
    {
        Odwołania: 2
        StudentResponseDTO AddStudent(StudentRequestDTO student); //po wstawieniu zwroci nowego studenta PROCEDURA
        Odwołania: 2
        List<HistoriaResponseDTO> GetHistoryPaginated(int strona, int ilosc);
    }
}
```

EF

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace BLL.ServiceInterface
{
    Odwołania: 4
    public interface IStudentEF
    {
        Odwołania: 2
        StudentResponseDTO AddStudent(StudentRequestDTO student); //po wstawieniu zwroci nowego studenta
        Odwołania: 2
        bool StudentUpdateGroup(int id, int groupId);
        Odwołania: 2
        bool DeleteStudent(int id);
        Odwołania: 2
        StudentResponseDTO GetStudentById(int id);
        Odwołania: 2
        List<StudentResponseDTO> GetAllStudents();
        Odwołania: 2
        List<StudentResponseDTO> GetAllStudentsPaginated(int strona, int ilosc);
        Odwołania: 2
        List<HistoriaResponseDTO> GetHistoryPaginated(int strona, int ilosc);

    }
}
```

Serwisy:

```
namespace BLL_EF
{
    public class StudentEFService : IStudentEF
    {
        private readonly WebContext _dbContext;
        public StudentEFService(WebContext dbContext)
        {
            _dbContext = dbContext;
        }
        // POLECENIE 8  RELACJA W MODELU DZIAŁA
        public string GetFullGroupName(Grupa grupa)
        {
            if (grupa == null)
            {
                return "brak";
            }

            var groupName = grupa.Nazwa != null ? grupa.Nazwa : "brak";

            if (grupa.ParentId != null)
            {
                var parentGroup = _dbContext.Grupy.Find(grupa.ParentId);
                var parentGroupName = GetFullGroupName(parentGroup);
                groupName = $"{parentGroupName} > {groupName}";
            }

            return groupName;
        }

        public StudentResponseDTO AddStudent(StudentRequestDTO student)
        {
            Student newStudent = new Student
            {
                Imie = student.Imie,
                Nazwisko = student.Nazwisko,
                IdGrupy = student.IdGrupy
            };

            _dbContext.Studenci.Add(newStudent);
            _dbContext.SaveChanges();

            var result = _dbContext.Studenci.Include(x => x.Grupa).OrderBy(i =>
i.Id).LastOrDefault();
            if (result == null)
            {
                throw new Exception("Wyjatek");
            }
            return new StudentResponseDTO
            {
                Id = result.Id,
                Imie = result.Imie,
                Nazwisko = result.Nazwisko,
                NazwaGrupy = GetFullGroupName(result.Grupa)
            };
        }

        public bool DeleteStudent(int id)
        {
            Student user = _dbContext.Studenci.Where(x => x.Id == id).OrderBy(i
=> i.Id).FirstOrDefault();
            if (user == null)
```

```
            {
                return false;
            }
            else
            {
                /* do testu
                //wg mnie to dto nie ma tu sensu ale bylo w poleceniu aby
stworzyć to jest :)
                HistoriaRequestDTO historia = new HistoriaRequestDTO
                {
                    Imie = user.Imie,
                    Nazwisko = user.Nazwisko,
                    IdGrupy = user.IdGrupy,
                    DateTime = DateTime.Now,
                    TypAkcji = 1 //dla ułątwienia w dto nie mam enum tylko 0 to
edycja a 1 usuwanie
                };
                _dbContext.Historia.Add(new Historia
                {
                    Imie = historia.Imie,
                    Nazwisko = historia.Nazwisko,
                    DateTime = historia.DateTime,
                    IdGrupy = historia.IdGrupy,
                    TypAkcji = (TypAkcji)Enum.Parse(typeof(TypAkcji),
historia.TypAkcji.ToString())
                });
                */
                _dbContext.Studenci.Remove(user);
                _dbContext.SaveChanges();
                return true;
            }
        }

        public List<StudentResponseDTO> GetAllStudents()
        {
            var users = _dbContext.Studenci.Include(u => u.Grupa).AsQueryable();

            return users.Select(p => new StudentResponseDTO
            {
                Id = p.Id,
                Imie = p.Imie,
                Nazwisko = p.Nazwisko,
                NazwaGrupy = p.Grupa.Nazwa != null ? p.Grupa.Nazwa : "brak"

            }).ToList();

        }

        public List<StudentResponseDTO> GetAllStudentsPagined(int strona, int
ilosc)
        {
            return _dbContext.Studenci.Include(g=>g.Grupa).Skip(strona * ilosc)
                   .Take(ilosc)
                   .Select(o => new StudentResponseDTO
                   {
                       Id = o.Id,
                       Imie = o.Imie,
                       Nazwisko = o.Nazwisko,
                       NazwaGrupy = o.Grupa.Nazwa != null ? o.Grupa.Nazwa :
"brak"

                   }).ToList();
        }
```

```
        public List<HistoriaResponseDTO> GetHistoryPagined(int strona, int ilosc)
        {
            //strony numerowane od 0 bo nic nie było w poleceniu jak ma być od 1
to --strona;
            return _dbContext.Historia.Skip(strona * ilosc).Take(ilosc).Select(o
=> new HistoriaResponseDTO
            {
                Id = o.Id,
                Imie = o.Imie,
                Nazwisko = o.Nazwisko,
                NazwaGrupy = _dbContext.Grupy.Where(i => i.Id ==
o.IdGrupy).FirstOrDefault().Nazwa,
                DateTime = o.DateTime,
                TypAkcji = o.TypAkcji.ToString()

            }).ToList();
        }

        public StudentResponseDTO GetStudentById(int id)
        {
            Student user = _dbContext.Studenci.Include(u => u.Grupa).Where(x =>
x.Id == id).FirstOrDefault();
            if (user == null)
            {
                throw new Exception("Wyjatek");
            }
            else
            {
                return new StudentResponseDTO
                {
                    Id = user.Id,
                    Imie = user.Imie,
                    Nazwisko = user.Nazwisko,
                    NazwaGrupy = user.Grupa.Nazwa != null ? user.Grupa.Nazwa :
"brak"
                };
            }
        }

        public bool StudentUpdateGroup(int id, int groupId)
        {
            var result = _dbContext.Studenci.Where(i => i.Id ==
id).FirstOrDefault();

            var checkGroup = _dbContext.Grupy.Where(i=>i.Id ==
groupId).FirstOrDefault();
            if (result == null || checkGroup == null)
            {
                return false;
            }
            else
            {
                /*
                 //do testu
                //wg mnie to dto nie ma tu sensu ale bylo w poleceniu aby
stworzyć to jest :)
                HistoriaRequestDTO historia = new HistoriaRequestDTO
                {
                    Imie = result.Imie,
                    Nazwisko = result.Nazwisko,
                    IdGrupy = result.IdGrupy,
                    DateTime = DateTime.Now,
```

```
                    TypAkcji = 0 //dla ułątwienia w dto nie mam enum tylko 0 to
edycja a 1 usuwanie
                };

                result.IdGrupy = groupId;
                //narazie bez triggera dodawanie historii przy modyfikacji grupy
                _dbContext.Historia.Add(new Historia
                {
                    Imie = historia.Imie,
                    Nazwisko = historia.Nazwisko,
                    DateTime = historia.DateTime,
                    IdGrupy = historia.IdGrupy,
                    TypAkcji = (TypAkcji)Enum.Parse(typeof(TypAkcji),
historia.TypAkcji.ToString())
                });
                */
                result.IdGrupy = groupId;
                _dbContext.SaveChanges();

                return true;
            }
        }
    }
```

DLA PROCEDUR:

```
namespace BLL_Procedure
{
    public class StudentDBService : IStudentDB
    {
        public string connectionString = "Data
Source=(localdb)\\MSSQLLocalDB;Initial Catalog=NowakowskiMPB;Integrated
Security=True;Connect Timeout=30;Encrypt=False;Trust Server
Certificate=False;Application Intent=ReadWrite;Multi Subnet Failover=False";
        public StudentResponseDTO AddStudent(StudentRequestDTO student)
        {
            using (var connection = new SqlConnection(connectionString))
            {
                connection.Open();



                var sql = @"exec dbo.AddStudent @Imie,@Nazwisko,@IdGrupy";


                using (var command = new SqlCommand(sql, connection))
                {

                    command.Parameters.AddWithValue("@Imie", student.Imie);
                    command.Parameters.AddWithValue("@Nazwisko",
student.Nazwisko);
                    command.Parameters.AddWithValue("@IdGrupy", student.IdGrupy);


                    using (var reader = command.ExecuteReader())
                    {

                        if (reader.HasRows)
                        {

                            reader.Read();
```

```
                            return new StudentResponseDTO
                            {
                                Id = reader.GetInt32(0),
                                Imie = reader.GetString(1),
                                Nazwisko = reader.GetString(2),
                                NazwaGrupy = reader.GetString(3)
                            };
                        }
                        else
                        {
                            throw new Exception("Brak wyników.");
                        }
                    }
                }
            }
        }

        public List<HistoriaResponseDTO> GetHistoryPagined(int strona, int ilosc)
        {
            List<HistoriaResponseDTO> history = new List<HistoriaResponseDTO>();
            using (SqlConnection connection = new
SqlConnection(connectionString))
            {
                connection.Open();
                string query = "dbo.GetHistoryPaginated";

                using (SqlCommand command = new SqlCommand(query, connection))
                {
                    command.Parameters.AddWithValue("@Offset",strona);
                    command.Parameters.AddWithValue("@PageSize", ilosc);
                    command.CommandType = CommandType.StoredProcedure;
                    SqlDataReader reader = command.ExecuteReader();

                    while (reader.Read())
                    {
                        int historyId = reader.GetInt32(reader.GetOrdinal("Id"));

                        // Sprawdź, czy użytkownik już istnieje na liście
                        HistoriaResponseDTO historia = history.FirstOrDefault(u
=> u.Id == historyId);
                        int typAkcjiInt = reader.GetInt32(5);
                        string typAkcjiString;

                        if (typAkcjiInt == 0)
                        {
                            typAkcjiString = "Edycja";
                        }
                        else if (typAkcjiInt == 1)
                        {
                            typAkcjiString = "Usuwanie";
                        }
                        else
                        {
                            typAkcjiString = "Nieznany";
                        }
                        if (historia == null)
                        {
                            historia = new HistoriaResponseDTO
                            {
                                Id = reader.GetInt32(0),
                                Imie = reader.GetString(1),
                                Nazwisko = reader.GetString(2),
```

```
                                NazwaGrupy = reader.GetString(3),
                                DateTime = reader.GetDateTime(4),
                                TypAkcji = typAkcjiString,
                            };
                            history.Add(historia);
                        }
                    }

                    reader.Close();
                }
            }
            return history;
        }
    }
```

Kontrolery:

```
namespace NowakowskiKolPB.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    1 odwołanie
    public class StudentDBProcedureController : ControllerBase
    {
        private readonly IStudentDB _studentService;  //interfejs dla procedur

        Odwołania: 0
        public StudentDBProcedureController(IStudentDB studentService)
        {
            _studentService = studentService;
        }
        [HttpPost]
        Odwołania: 0
        public IActionResult AddStudent(StudentRequestDTO student)
        {
            try
            {
                var response = _studentService.AddStudent(student);
                return Ok(response);
            }
            catch (Exception ex)
            {
                return StatusCode(500, ex.Message);
            }
        }
        [HttpGet("history/paginated{strona}/{ilosc}")]
        Odwołania: 0
        public IActionResult GetHistoryPaginated(int strona, int ilosc)
        {
            try
            {
                var response = _studentService.GetHistoryPagined(strona, ilosc);
                return Ok(response);
            }
            catch (Exception ex)
            {
                return NotFound(ex.Message);
            }
        }
    }
}
```

EF:

```
namespace NowakowskiKolPB.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
```

```
    public class StudentEFController : ControllerBase
    {
        private readonly IStudentEF _studentService;  //interfejs dla EF

        public StudentEFController(IStudentEF studentService)
        {
            _studentService = studentService;
        }

        [HttpPost]
        public IActionResult AddStudent(StudentRequestDTO student)
        {
            try
            {
                var response = _studentService.AddStudent(student);
                return Ok(response);
            }
            catch (Exception ex)
            {
                return StatusCode(500, ex.Message);
            }
        }

        [HttpGet("{id}")]
        public IActionResult GetStudentById(int id)
        {
            try
            {
                var response = _studentService.GetStudentById(id);
                return Ok(response);
            }
            catch (Exception ex)
            {
                return NotFound(ex.Message);
            }
        }

        [HttpGet("history/paginated{strona}/{ilosc}")]
        public IActionResult GetHistoryPaginated(int strona,int ilosc)
        {
            try
            {
                var response = _studentService.GetHistoryPagined(strona,ilosc);
                return Ok(response);
            }
            catch (Exception ex)
            {
                return NotFound(ex.Message);
            }
        }

        [HttpGet("students/paginated{strona}/{ilosc}")]
        public IActionResult GetAllStudentsPaginated(int strona, int ilosc)
        {
            try
            {
                var response =
_studentService.GetAllStudentsPagined(strona,ilosc);
                return Ok(response);
            }
            catch (Exception ex)
            {
                return NotFound(ex.Message);
```

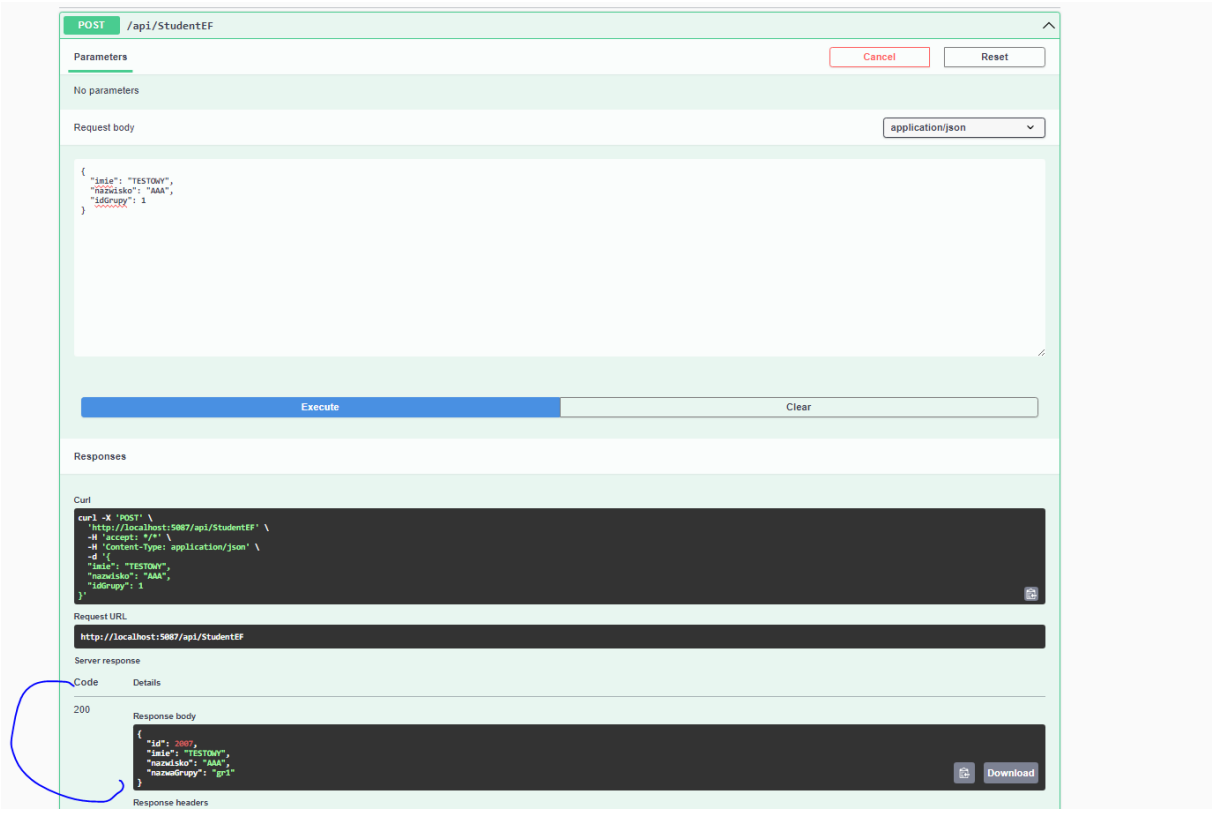
```
            }
        }
        [HttpGet("AllStudents")]
        public IActionResult GetAllStudent()
        {

            try
            {
                var response = _studentService.GetAllStudents();
                return Ok(response);
            }
            catch (Exception ex)
            {
                return NotFound(ex.Message);
            }

        }


        [HttpDelete("{id}")]
        public IActionResult DeleteStudent(int id)
        {
            try
            {
                var response = _studentService.DeleteStudent(id);
                return Ok(response);
            }
            catch (Exception ex)
            {
                return NotFound(ex.Message);
            }
        }

        [HttpPut("{id}/updategroup/{groupId}")]
        public IActionResult StudentUpdateGroup(int id, int groupId)
        {
            try
            {
                var response = _studentService.StudentUpdateGroup(id, groupId);
                return Ok(response);
            }
            catch (Exception ex)
            {
                return NotFound(ex.Message);
            }
        }
    }
```

Modyfikacja program.cs

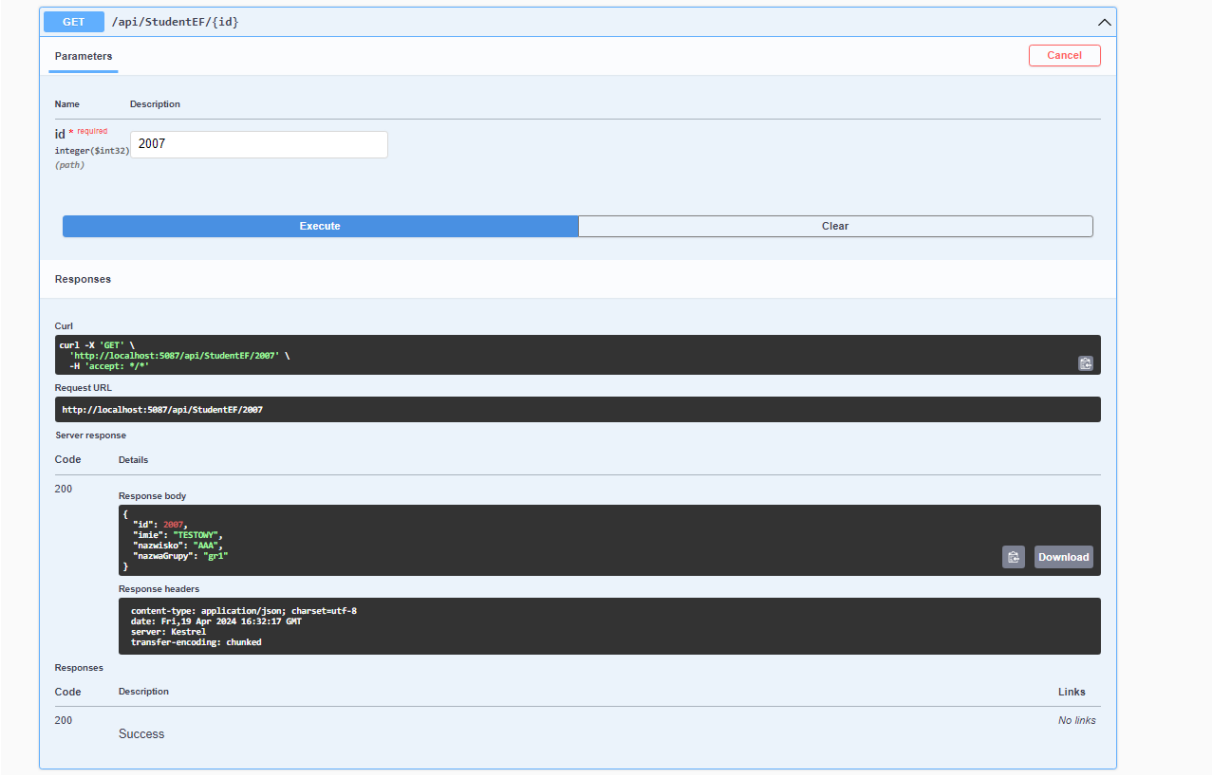
```
using BLL_Procedure;
using DAL;

namespace NowakowskiKolPB
{
    Odwołania: 0
    public class Program
    {
        Odwołania: 0
        public static void Main(string[] args)
        {
            var builder = WebApplication.CreateBuilder(args);

            // Add services to the container.

            builder.Services.AddControllers();
            // Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
            builder.Services.AddDbContext<WebContext>();
            builder.Services.AddScoped<IStudentEF,StudentEFService>();
            builder.Services.AddScoped<IStudentDB,StudentDBService>();
            builder.Services.AddEndpointsApiExplorer();
            builder.Services.AddSwaggerGen();

            var app = builder.Build();

            // Configure the HTTP request pipeline.
            if (app.Environment.IsDevelopment())
            {
                app.UseSwagger();
                app.UseSwaggerUI();
            }

            app.UseAuthorization();


            app.MapControllers();

            app.Run();
        }
    }
}
```

TESTOWANIE CRUD:

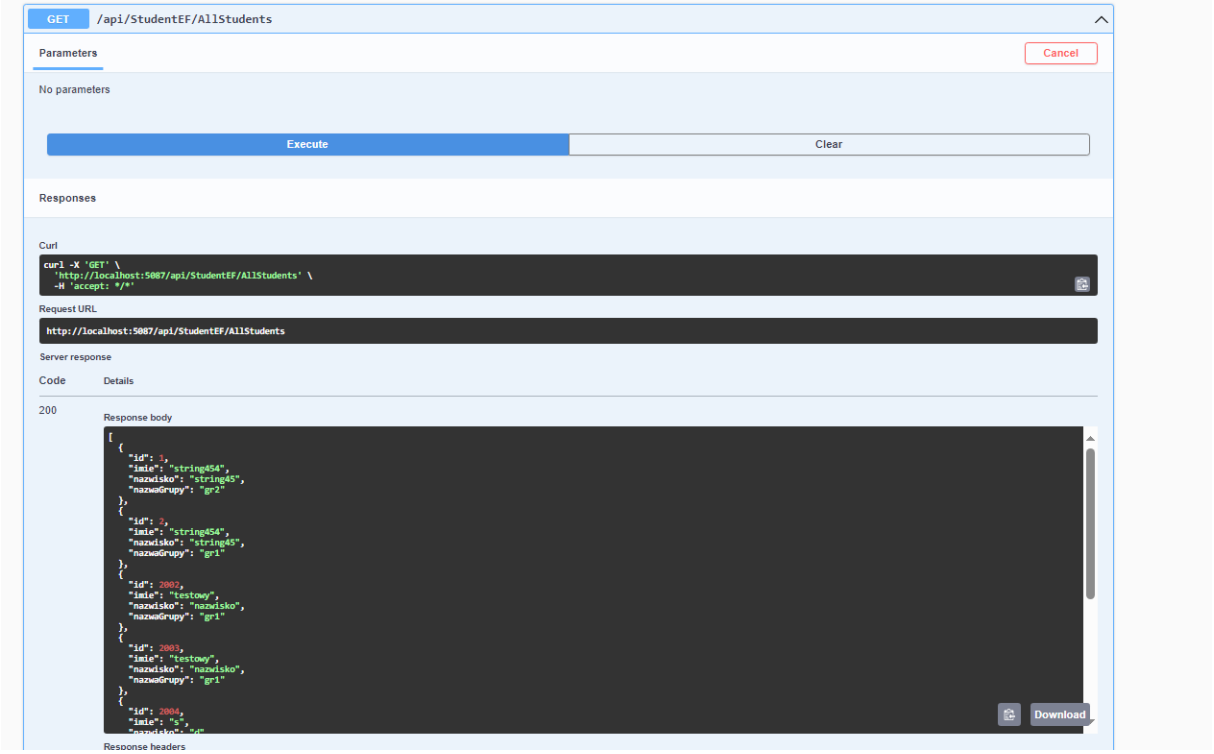
POST:



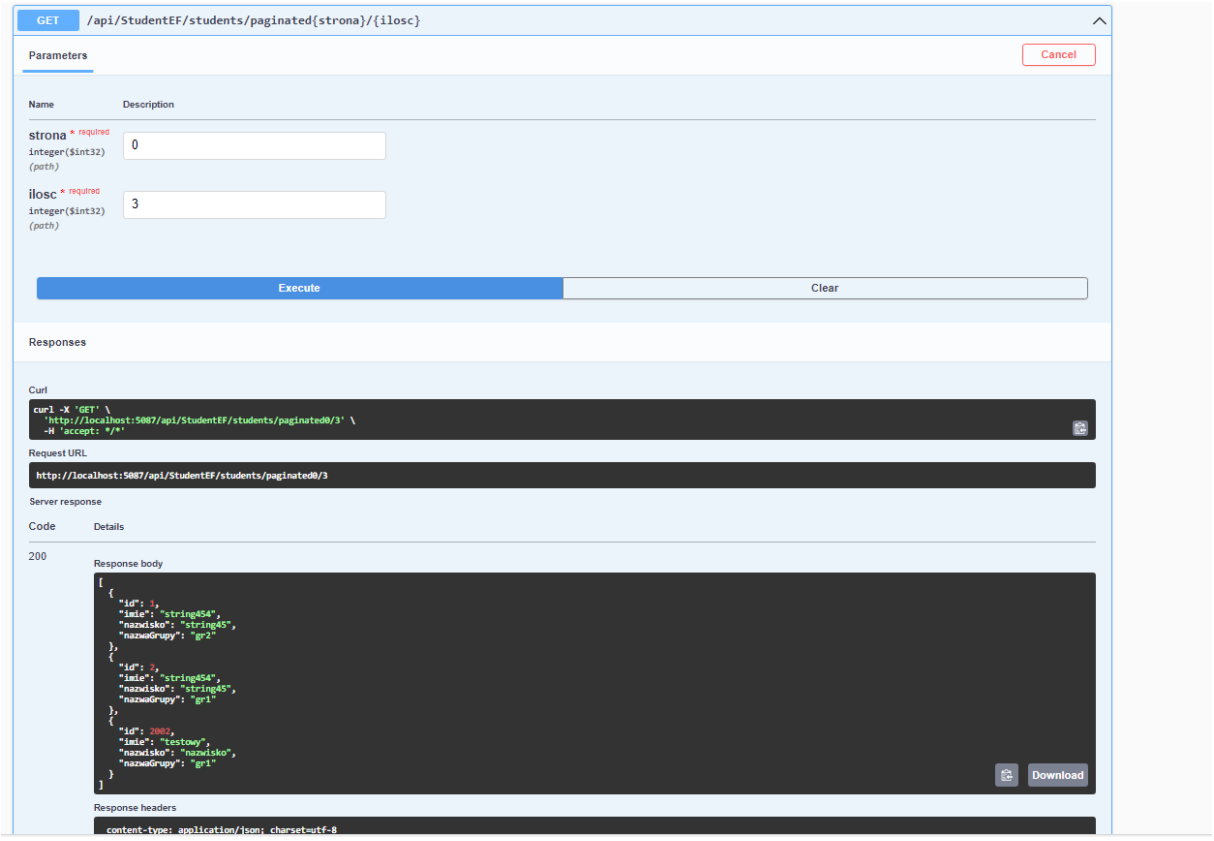
READ BY ID:



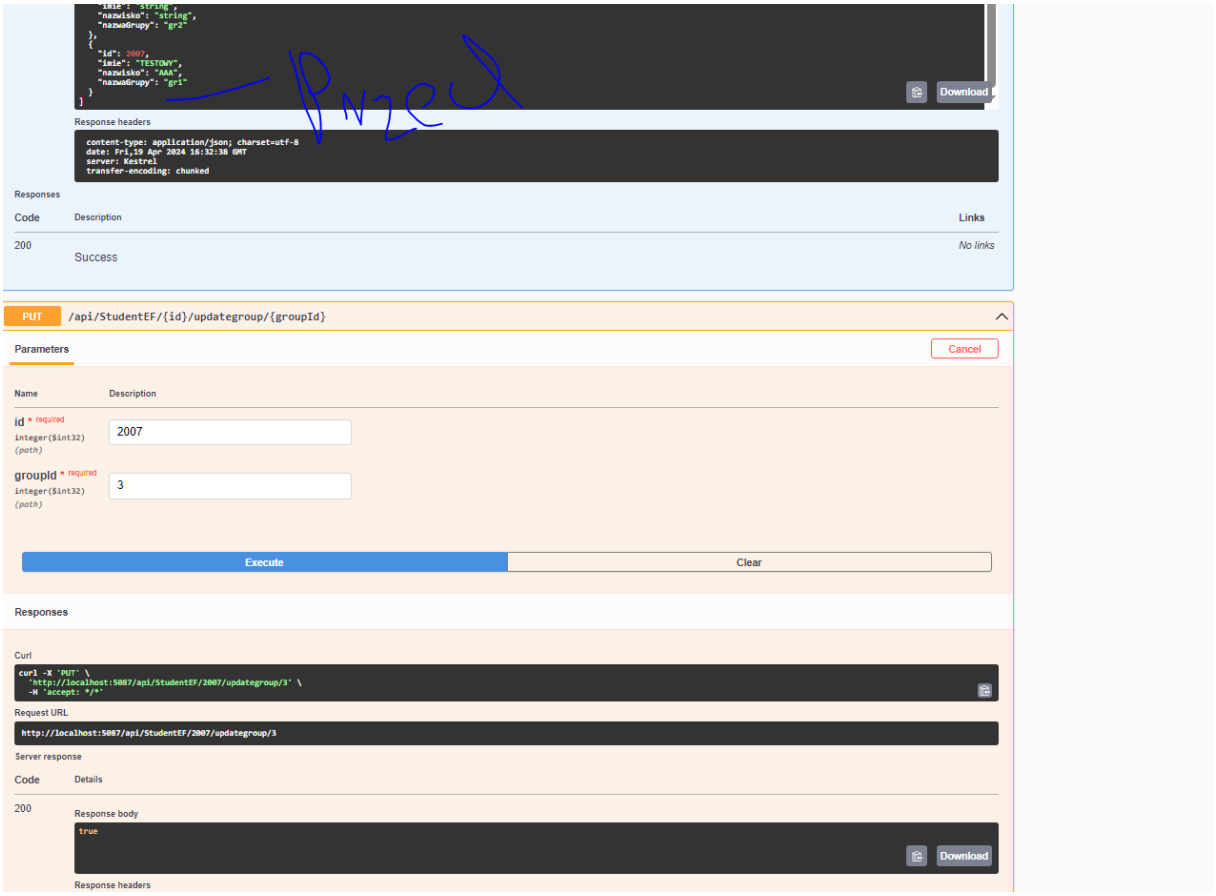
Read all:

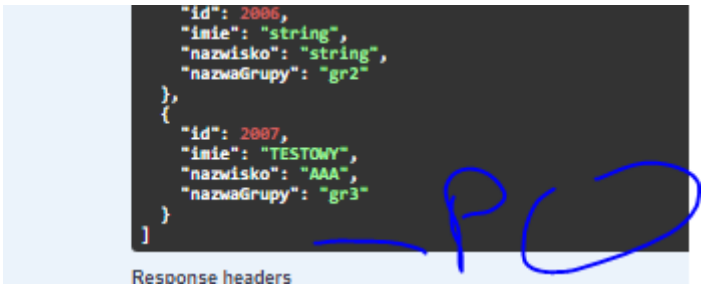


Z paginacją dodatkowo:

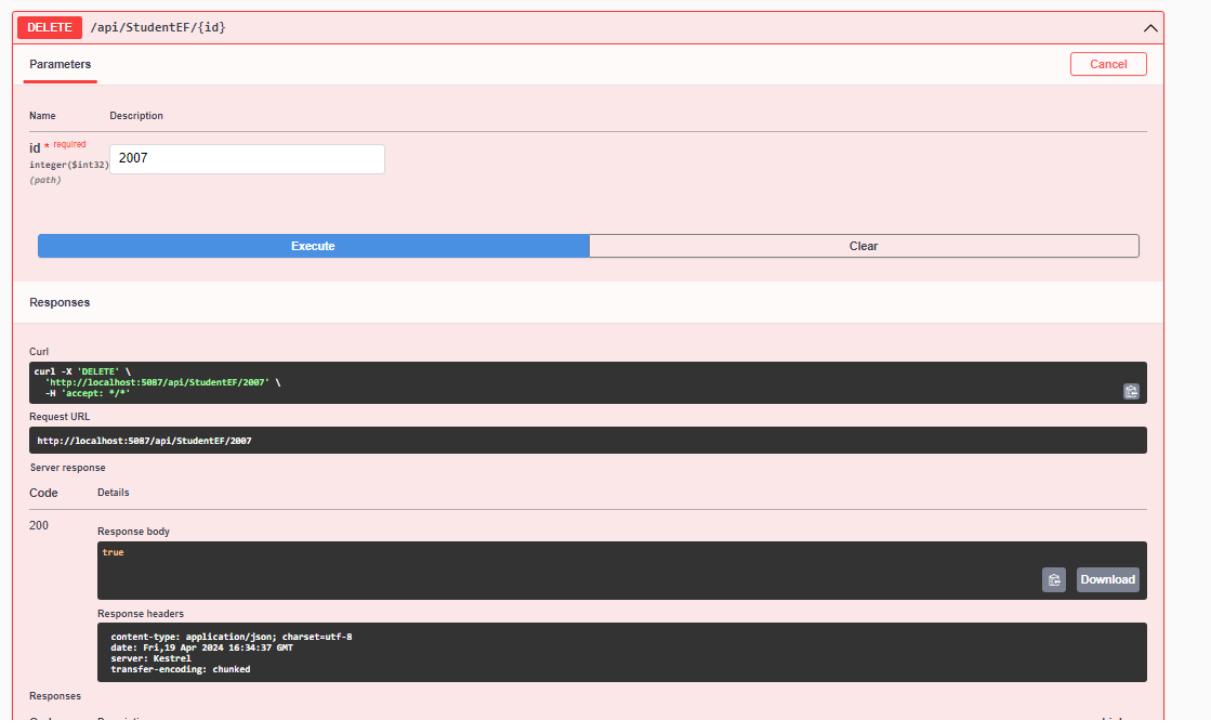


UPDATE GRUPY:

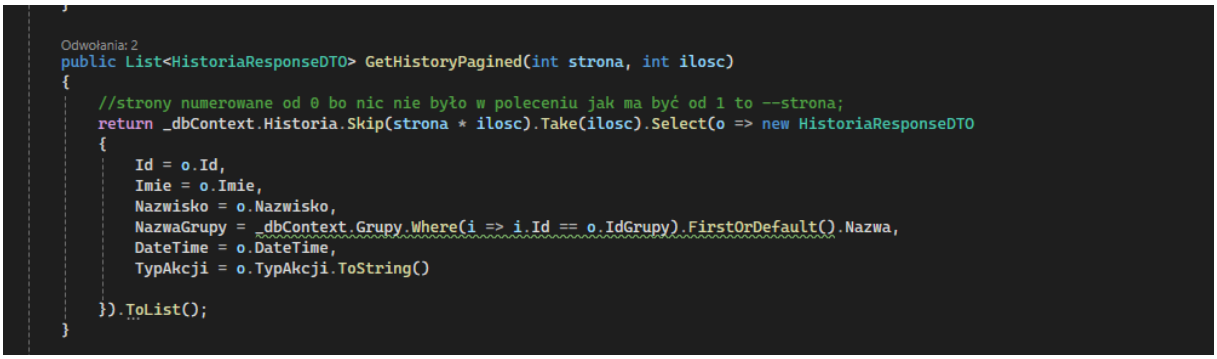




DELETE:

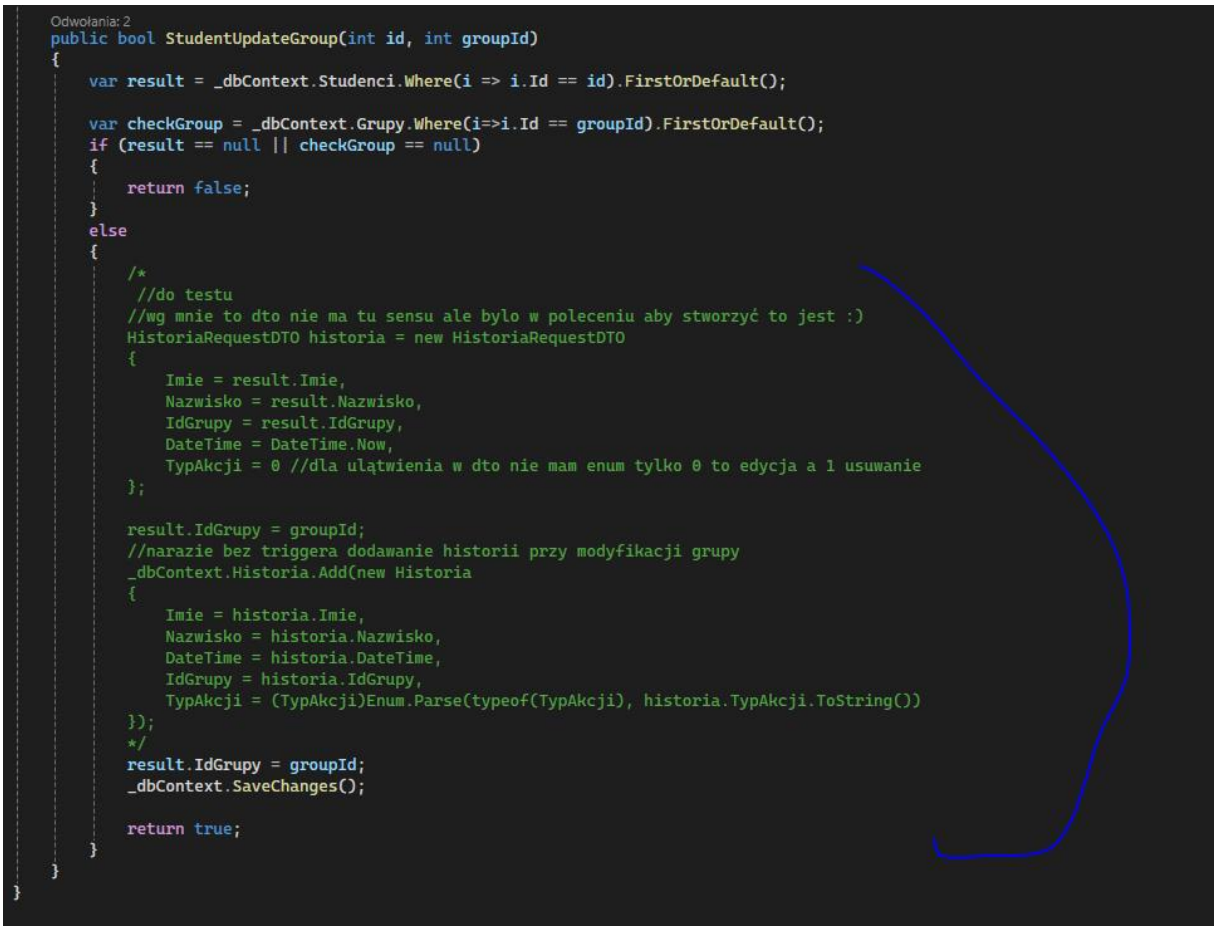


4. Należy dodatkowo wystawić stronnicowane wyświetlanie tabeli Historia.



```
Odwołania: 2
public List<HistoriaResponseDTO> GetHistoryPagined(int strona, int ilosc)
{
    //strony numerowane od 0 bo nic nie było w poleceniu jak ma być od 1 to --strona;
    return _dbContext.Historia.Skip(strona * ilosc).Take(ilosc).Select(o => new HistoriaResponseDTO
    {
        Id = o.Id,
        Imie = o.Imie,
        Nazwisko = o.Nazwisko,
        NazwaGrupy = _dbContext.Grupy.Where(i => i.Id == o.IdGrupy).FirstOrDefault().Nazwa,
        DateTime = o.DateTime,
        TypAkcji = o.TypAkcji.ToString()

    }).ToList();
}
```

Funkcje testowe:



```
Odwołania: 2
public bool StudentUpdateGroup(int id, int groupId)
{
    var result = _dbContext.Studenci.Where(i => i.Id == id).FirstOrDefault();

    var checkGroup = _dbContext.Grupy.Where(i=>i.Id == groupId).FirstOrDefault();
    if (result == null || checkGroup == null)
    {
        return false;
    }
    else
    {
        /*
         //do testu
        //wg mnie to dto nie ma tu sensu ale bylo w poleceniu aby stworzyć to jest :)
        HistoriaRequestDTO historia = new HistoriaRequestDTO
        {
            Imie = result.Imie,
            Nazwisko = result.Nazwisko,
            IdGrupy = result.IdGrupy,
            DateTime = DateTime.Now,
            TypAkcji = 0 //dla ułątwienia w dto nie mam enum tylko 0 to edycja a 1 usuwanie
        };

        result.IdGrupy = groupId;
        //narazie bez triggera dodawanie historii przy modyfikacji grupy
        _dbContext.Historia.Add(new Historia
        {
            Imie = historia.Imie,
            Nazwisko = historia.Nazwisko,
            DateTime = historia.DateTime,
            IdGrupy = historia.IdGrupy,
            TypAkcji = (TypAkcji)Enum.Parse(typeof(TypAkcji), historia.TypAkcji.ToString())
        });
        */
        result.IdGrupy = groupId;
        _dbContext.SaveChanges();

        return true;
    }
}
}
```

NA SWAGGERZE:

GET /api/StudentEF/history/paginated{strona}/{ilosc}

Parameters Cancel

Name	Description
strona * required integer($int32) (path)	1
ilosc * required integer($int32) (path)	2

Execute Clear

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:5087/api/StudentEF/history/paginated1/2' \
  -H 'accept: */*'
```

Request URL

```
http://localhost:5087/api/StudentEF/history/paginated1/2
```

Server response

Code	Details
200	Response body

```
[
  {
    "id": 4,
    "imie": "string454",
    "nazwisko": "string45",
    "nazwaGrupy": "gr1",
    "typAkcji": "Usuwanie",
    "dateTime": "2024-04-19T18:17:19.44"
  },
  {
    "id": 5,
    "imie": "s",
    "nazwisko": "d",
    "nazwaGrupy": "gr1",
    "typAkcji": "Usuwanie",
    "dateTime": "2024-04-19T18:17:26.1733333"
  }
]
```

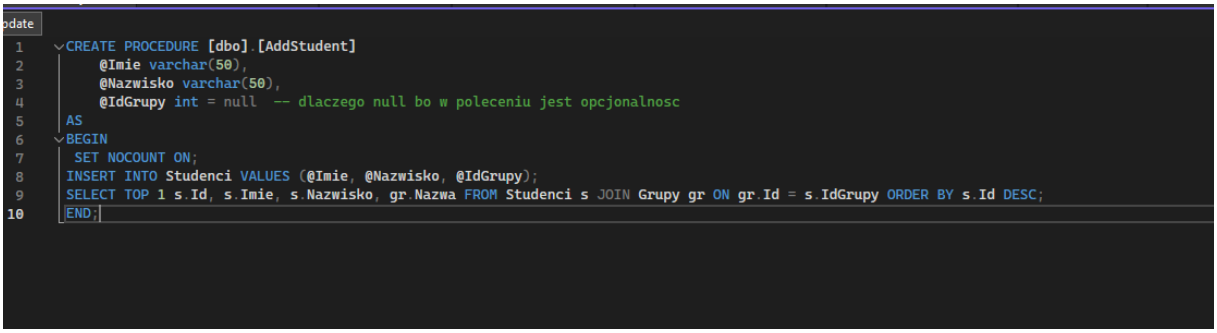
Download

Response headers

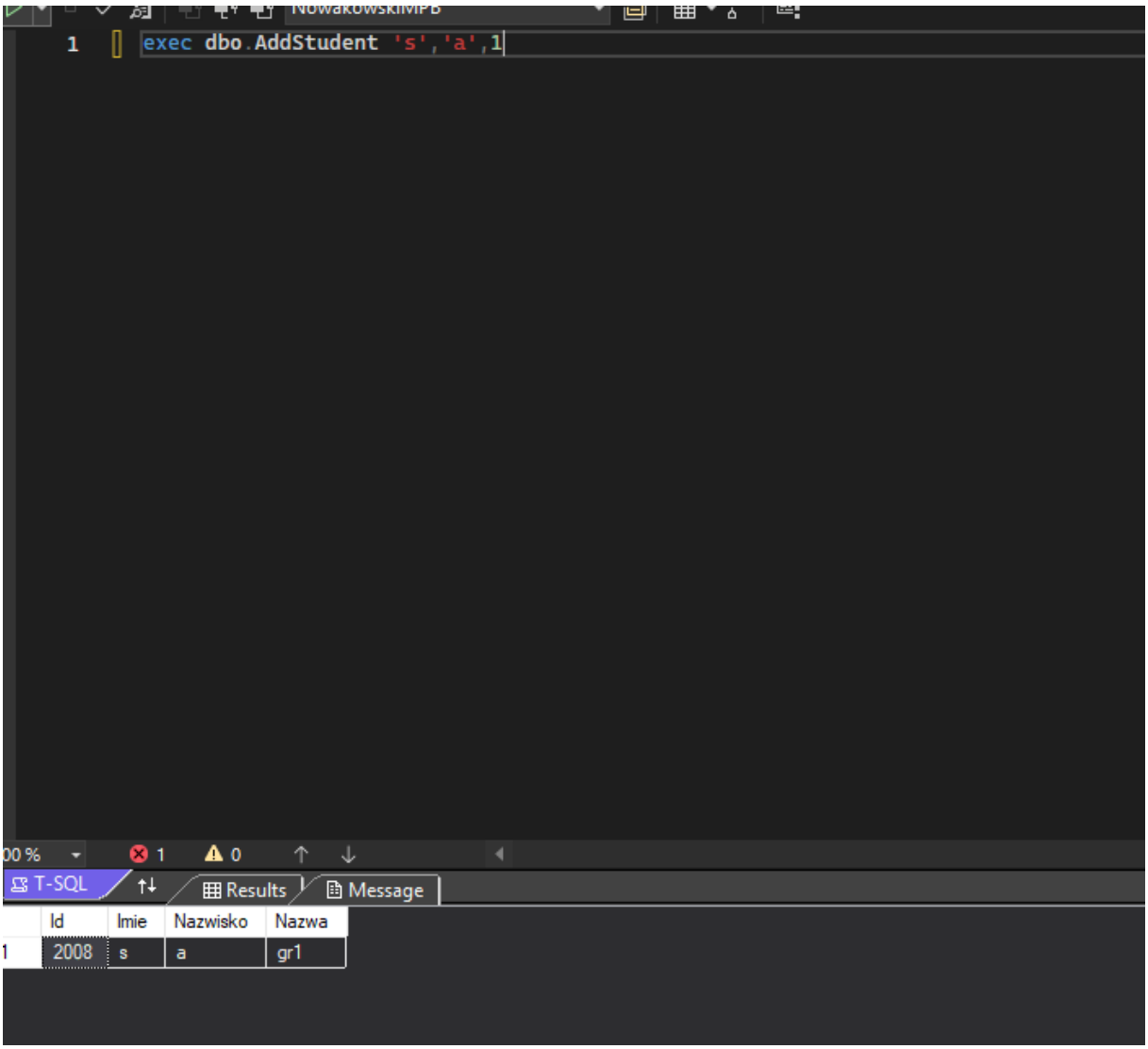
```
content-type: application/json; charset=utf-8
date: Fri,19 Apr 2024 16:35:56 GMT
server: Kestrel
transfer-encoding: chunked
```

Responses

5. Przerób implementację dodawania studenta na procedurę składowaną.



```
pdate
1   CREATE PROCEDURE [dbo].[AddStudent]
2       @Imie varchar(50),
3       @Nazwisko varchar(50),
4       @IdGrupy int = null  -- dlaczego null bo w poleceniu jest opcjonalnosc
5   AS
6   BEGIN
7     SET NOCOUNT ON;
8     INSERT INTO Studenci VALUES (@Imie, @Nazwisko, @IdGrupy);
9     SELECT TOP 1 s.Id, s.Imie, s.Nazwisko, gr.Nazwa FROM Studenci s JOIN Grupy gr ON gr.Id = s.IdGrupy ORDER BY s.Id DESC;
10    END;
```



```
1   exec dbo.AddStudent 's','a',1
```

00 % 1 0

T-SQL | Results | Message

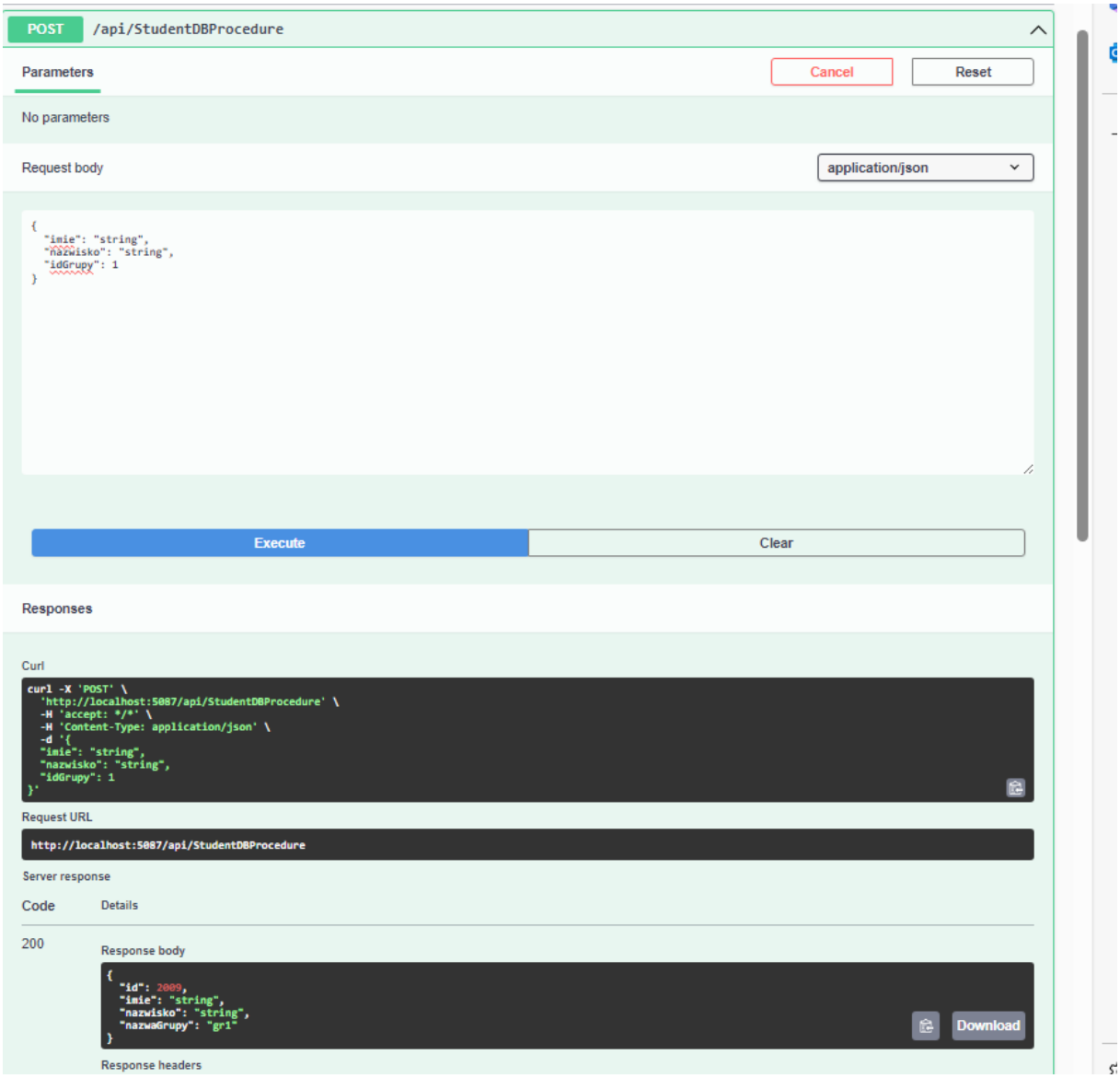
	Id	Imie	Nazwisko	Nazwa
1	2008	s	a	gr1

Wstawia i zwraca

A w serwisie:

```
{
    public string connectionString = "Data Source=(localdb)\\MSSQLLocalDB;Initial Catalog=NowakowskiMPB;Integrated Security=True;Connect Timeout=30;Encrypt=False;Trust Serve
    Odwołania: 2
    public StudentResponseDTO AddStudent(StudentRequestDTO student)
    {
        using (var connection = new SqlConnection(connectionString))
        {
            connection.Open();



            var sql = @"exec dbo.AddStudent @Imie,@Nazwisko,@IdGrupy";


            using (var command = new SqlCommand(sql, connection))
            {

                command.Parameters.AddWithValue("@Imie", student.Imie);
                command.Parameters.AddWithValue("@Nazwisko", student.Nazwisko);
                command.Parameters.AddWithValue("@IdGrupy", student.IdGrupy);


                using (var reader = command.ExecuteReader())
                {

                    if (reader.HasRows)
                    {

                        reader.Read();

                        return new StudentResponseDTO
                        {
                            Id = reader.GetInt32(0),
                            Imie = reader.GetString(1),
                            Nazwisko = reader.GetString(2),
                            NazwaGrupy = reader.GetString(3)
                        };
                    }
                    else
                    {
                        throw new Exception("Brak wyników.");
                    }
                }
            }
        }
    }
```

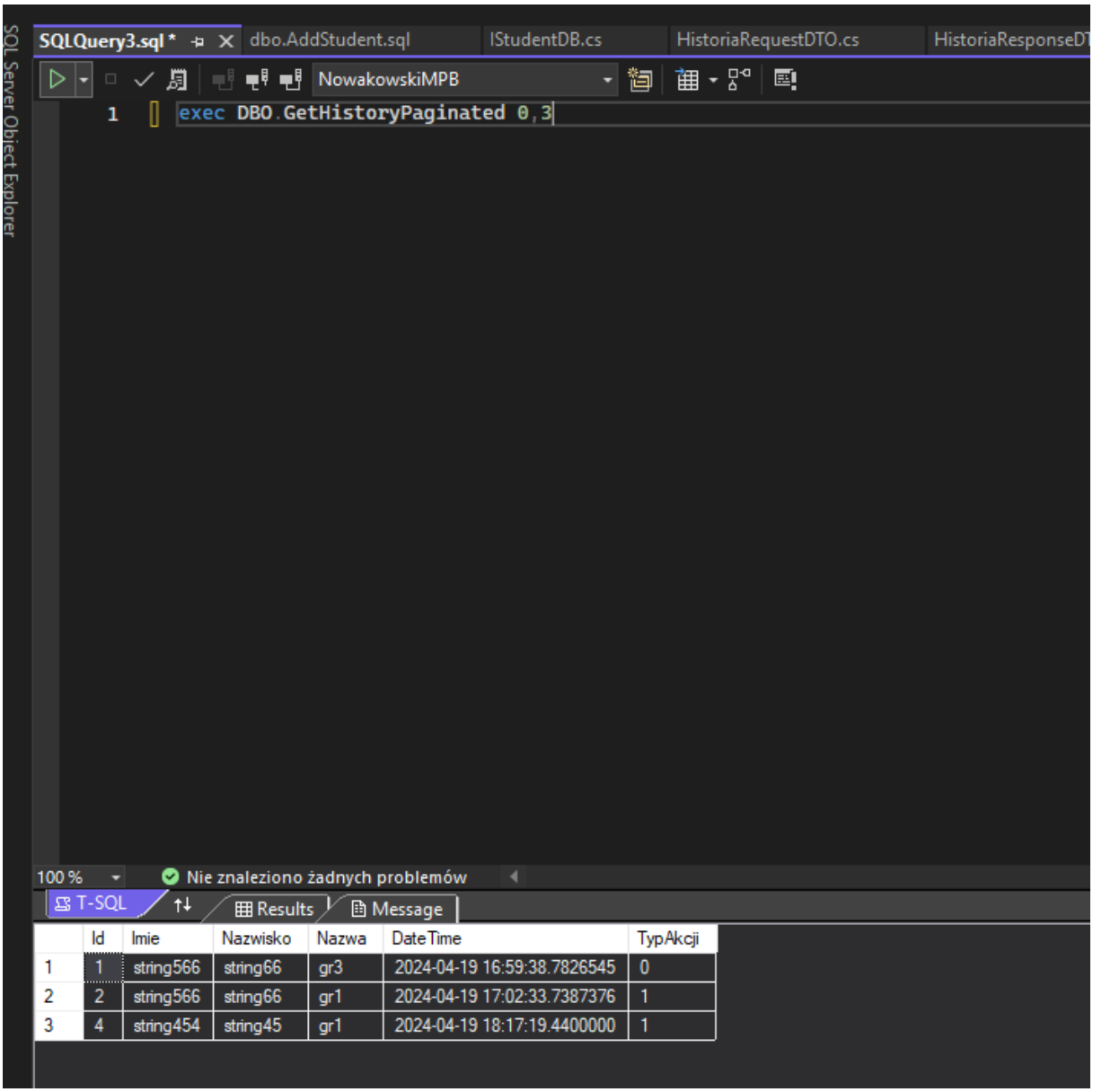
W swaggerze:


POST
/api/StudentDBProcedure
Parameters
Cancel
Reset
No parameters
Request body
application/json
{
"imie": "string",
"nazwisko": "string",
"idGrupy": 1
}
Execute
Clear
Responses
Curl
curl -X 'POST' \
'http://localhost:5087/api/StudentDBProcedure' \
-H 'accept: */*' \
-H 'Content-Type: application/json' \
-d '{
"imie": "string",
"nazwisko": "string",
"idGrupy": 1
}'
Request URL
http://localhost:5087/api/StudentDBProcedure
Server response
Code
Details
200
Response body
{
"id": 2009,
"imie": "string",
"nazwisko": "string",
"nazwaGrupy": "gr1"
}
Download
Response headers

6. Przerób stronnicowane wyświetlanie tabeli Historia na procedurę składowaną.

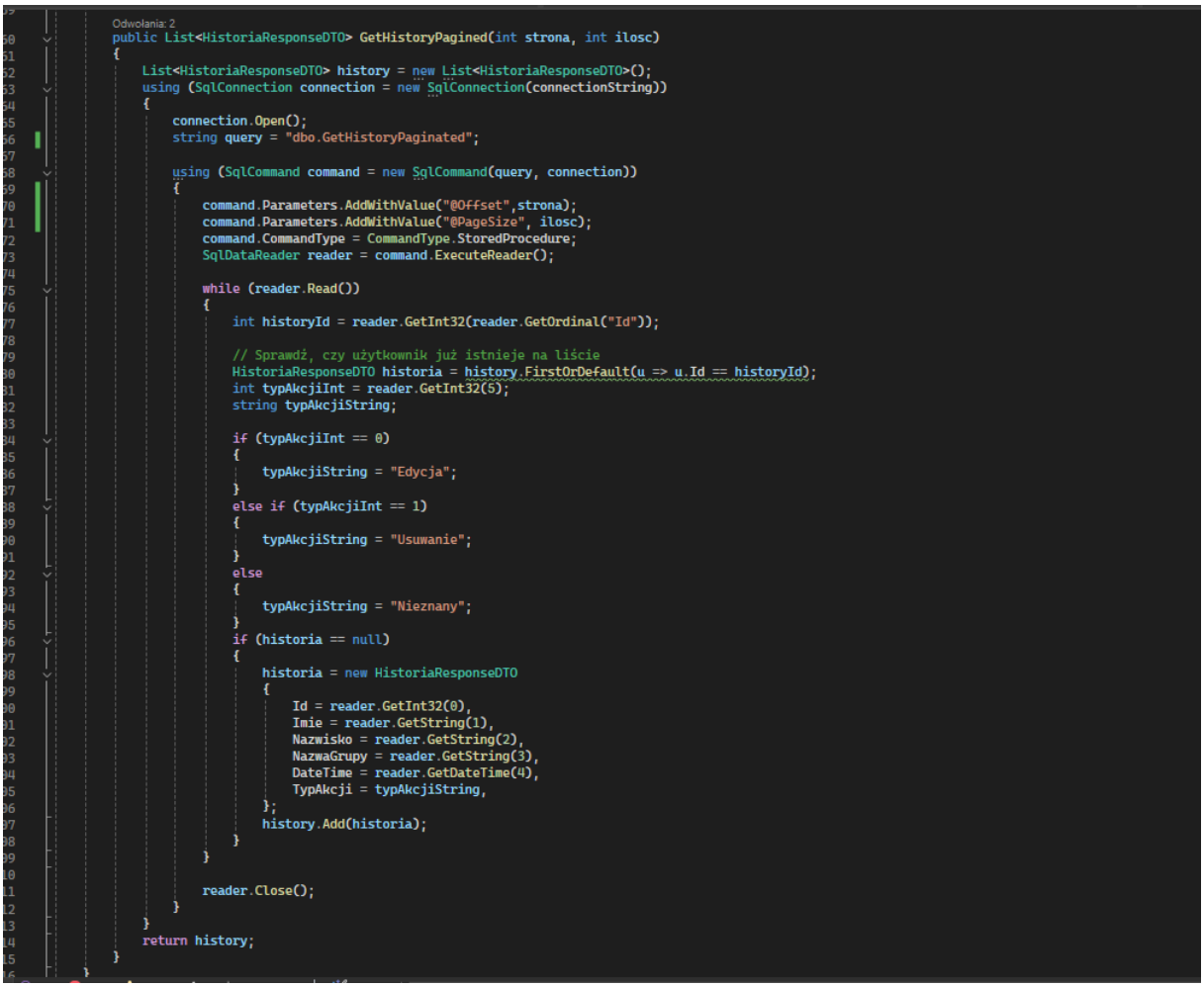
```
CREATE PROCEDURE [dbo].[GetHistoryPaginated]
    @Offset int ,
    @PageSize int
AS
BEGIN
SELECT h.Id, h.Imie, h.Nazwisko, g.Nazwa, h.DateTime, h.TypAkcji FROM Historia h LEFT JOIN Grupy g ON h.IdGrupy = g.Id ORDER BY h.Id OFFSET @Offset ROWS FETCH NEXT @PageSize ROWS ONLY;
END;
```

Test :



	Id	Imie	Nazwisko	Nazwa	DateTime	TypAkcji
1	1	string566	string66	gr3	2024-04-19 16:59:38.7826545	0
2	2	string566	string66	gr1	2024-04-19 17:02:33.7387376	1
3	4	string454	string45	gr1	2024-04-19 18:17:19.4400000	1

W SERWISIE:



```
Odwołania: 2
public List<HistoriaResponseDTO> GetHistoryPagined(int strona, int ilosc)
{
    List<HistoriaResponseDTO> history = new List<HistoriaResponseDTO>();
    using (SqlConnection connection = new SqlConnection(connectionString))
    {
        connection.Open();
        string query = "dbo.GetHistoryPaginated";

        using (SqlCommand command = new SqlCommand(query, connection))
        {
            command.Parameters.AddWithValue("@Offset",strona);
            command.Parameters.AddWithValue("@PageSize", ilosc);
            command.CommandType = CommandType.StoredProcedure;
            SqlDataReader reader = command.ExecuteReader();

            while (reader.Read())
            {
                int historyId = reader.GetInt32(reader.GetOrdinal("Id"));

                // Spramdź, czy użytkownik już istnieje na liście
                HistoriaResponseDTO historia = history.FirstOrDefault(u => u.Id == historyId);
                int typAkcjiInt = reader.GetInt32(5);
                string typAkcjiString;

                if (typAkcjiInt == 0)
                {
                    typAkcjiString = "Edycja";
                }
                else if (typAkcjiInt == 1)
                {
                    typAkcjiString = "Usuwanie";
                }
                else
                {
                    typAkcjiString = "Nieznany";
                }
                if (historia == null)
                {
                    historia = new HistoriaResponseDTO
                    {
                        Id = reader.GetInt32(0),
                        Imie = reader.GetString(1),
                        Nazwisko = reader.GetString(2),
                        NazwaGrupy = reader.GetString(3),
                        DateTime = reader.GetDateTime(4),
                        TypAkcji = typAkcjiString,
                    };
                    history.Add(historia);
                }
            }

            reader.Close();
        }
    }
    return history;
}
```

W SWAGERZE:

StudentDBProcedure

POST /api/StudentDBProcedure

GET /api/StudentDBProcedure/history/paginated{strona}/{ilosc}

Parameters | Cancel

Name	Description
strona * required integer($int32) (path)	0
ilosc * required integer($int32) (path)	2

Execute | Clear

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:5087/api/StudentDBProcedure/history/paginated0/2' \
  -H 'accept: */*'
```

Request URL

```
http://localhost:5087/api/StudentDBProcedure/history/paginated0/2
```

Server response

Code	Details
200	Response body


```
[
  {
    "id": 1,
    "imie": "string566",
    "nazwisko": "string66",
    "nazwaGrupy": "gr3",
    "typAkcji": "Edycja",
    "dateTime": "2024-04-19T16:59:38.7826545"
  },
  {
    "id": 2,
    "imie": "string566",
    "nazwisko": "string66",
    "nazwaGrupy": "gr1",
    "typAkcji": "Usuwanie",
    "dateTime": "2024-04-19T17:02:33.7387376"
  }
]
```


Download

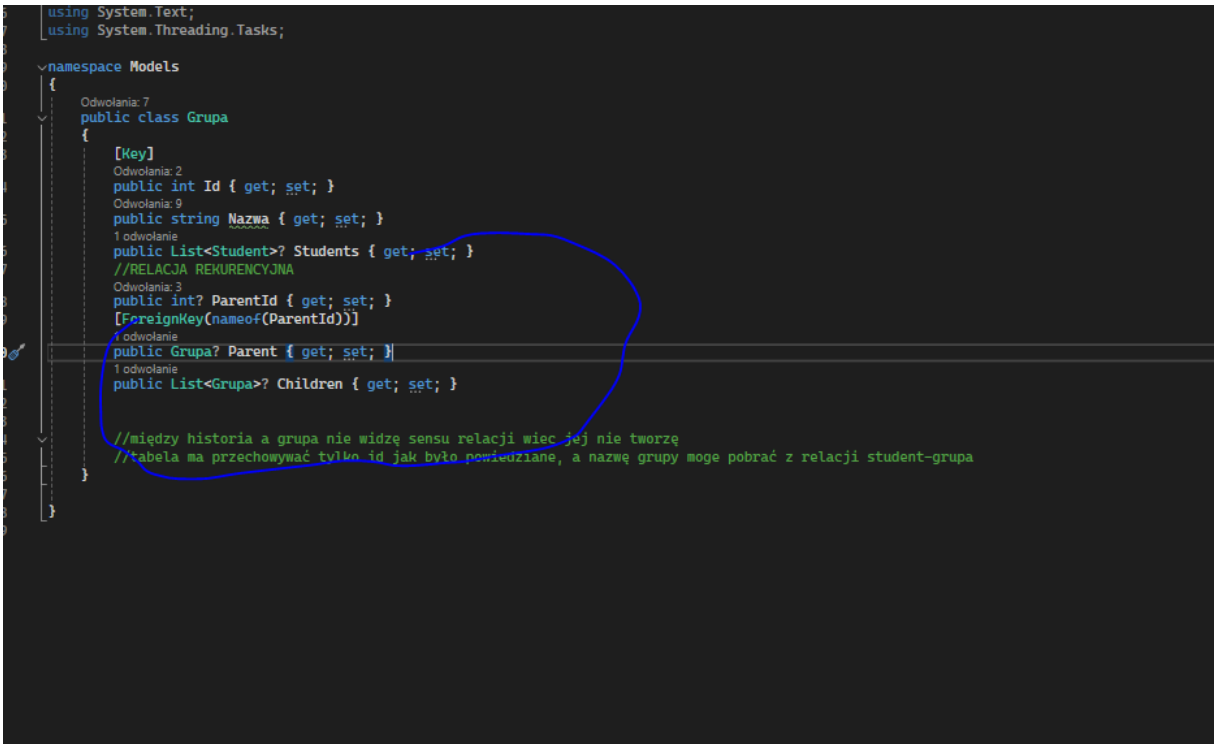
7. Dodaj triggera, który po każdej modyfikacji wpisu w tabeli Student doda wpis do tabeli Historia ze imieniem, nazwiskiem i idGrupy sprzed edycji, typem akcji edycja oraz aktualną datą. Przy usuwaniu trigger powinien wstawić imię, nazwisko i idGrupy usuwanego rekordu, typ akcji usuwanie oraz aktualną datę.

Update

```sql
CREATE TRIGGER Trigger_Historia_Studenci
ON [dbo].[Studenci]
FOR DELETE, UPDATE
AS
BEGIN
    SET NOCOUNT ON;

    IF EXISTS(SELECT * FROM deleted)
    BEGIN
        -- INSERT INTO Historia  DELETE
        INSERT INTO Historia (Imie, Nazwisko, IdGrupy, TypAkcji, DateTime)
        SELECT d.Imie, d.Nazwisko, d.IdGrupy, 1 , GETDATE()
        FROM deleted AS d;
    END;

    IF EXISTS(SELECT * FROM inserted)
    BEGIN
        -- INSERT INTO Historia UPDATE
        INSERT INTO Historia (Imie, Nazwisko, IdGrupy, TypAkcji, DateTime)
        SELECT i.Imie, i.Nazwisko, i.IdGrupy,0, GETDATE()
        FROM inserted AS i
        WHERE NOT EXISTS(SELECT 1 FROM deleted AS d WHERE d.Id = i.Id);
    END;
END;
```

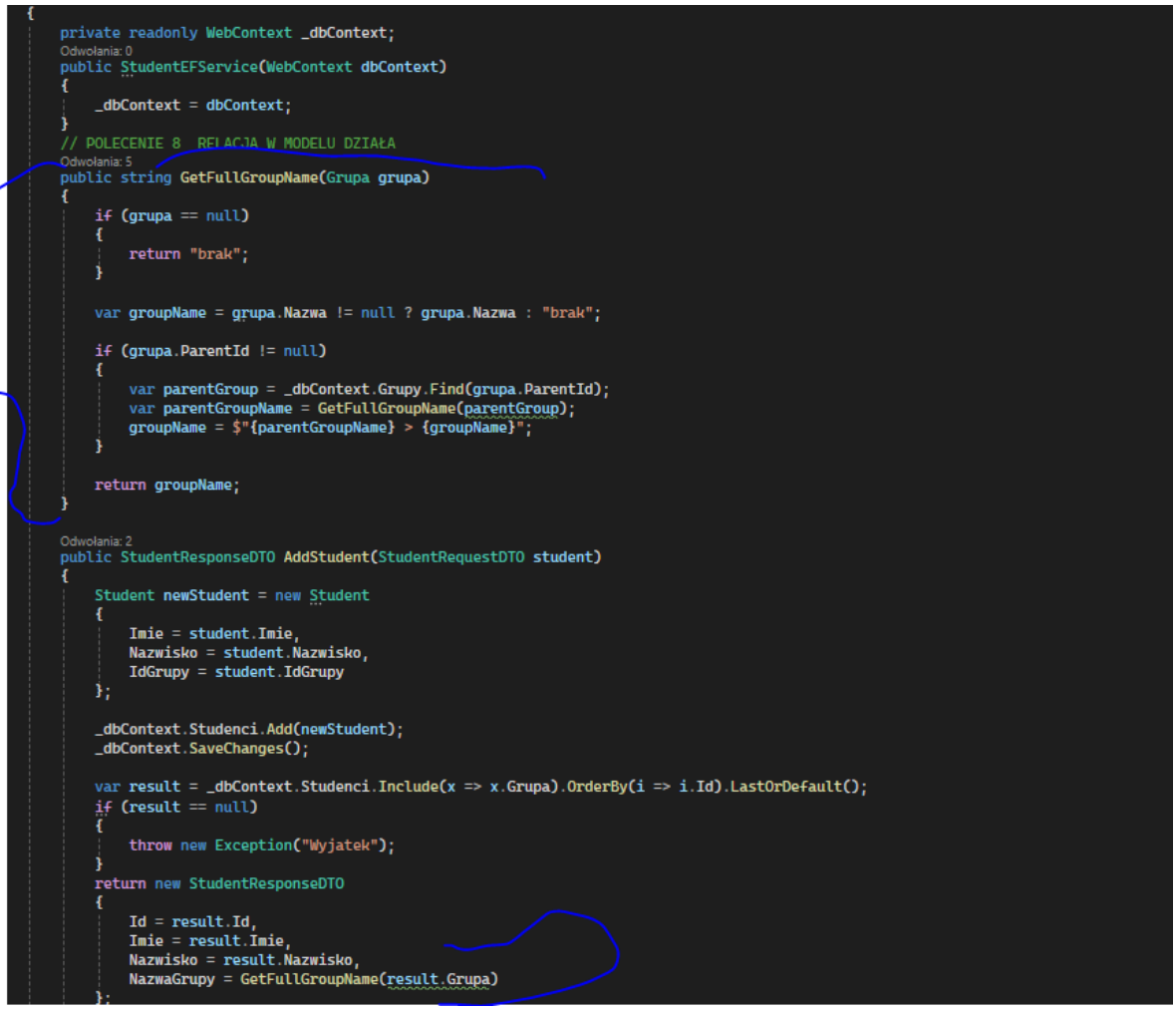
Przetestowany, działa

8. Rozbuduj encję grupa o relacje rekurencyjną. Zaimplementuj w SQL pobieranie pełnej nazwy grupy dla danego ID. Przykładowo jeśli w bazie posiadamy grupę o nazwie „trzy”, której rodzicem jest grupa o nazwie „dwa”, której rodzicem jest grupa o nazwie „jeden” to pełna nazwa dla grupy „trzy” będzie wyglądać następująco „jeden / dwa / trzy”. Wystaw tą funkcjonalność w WebAPI



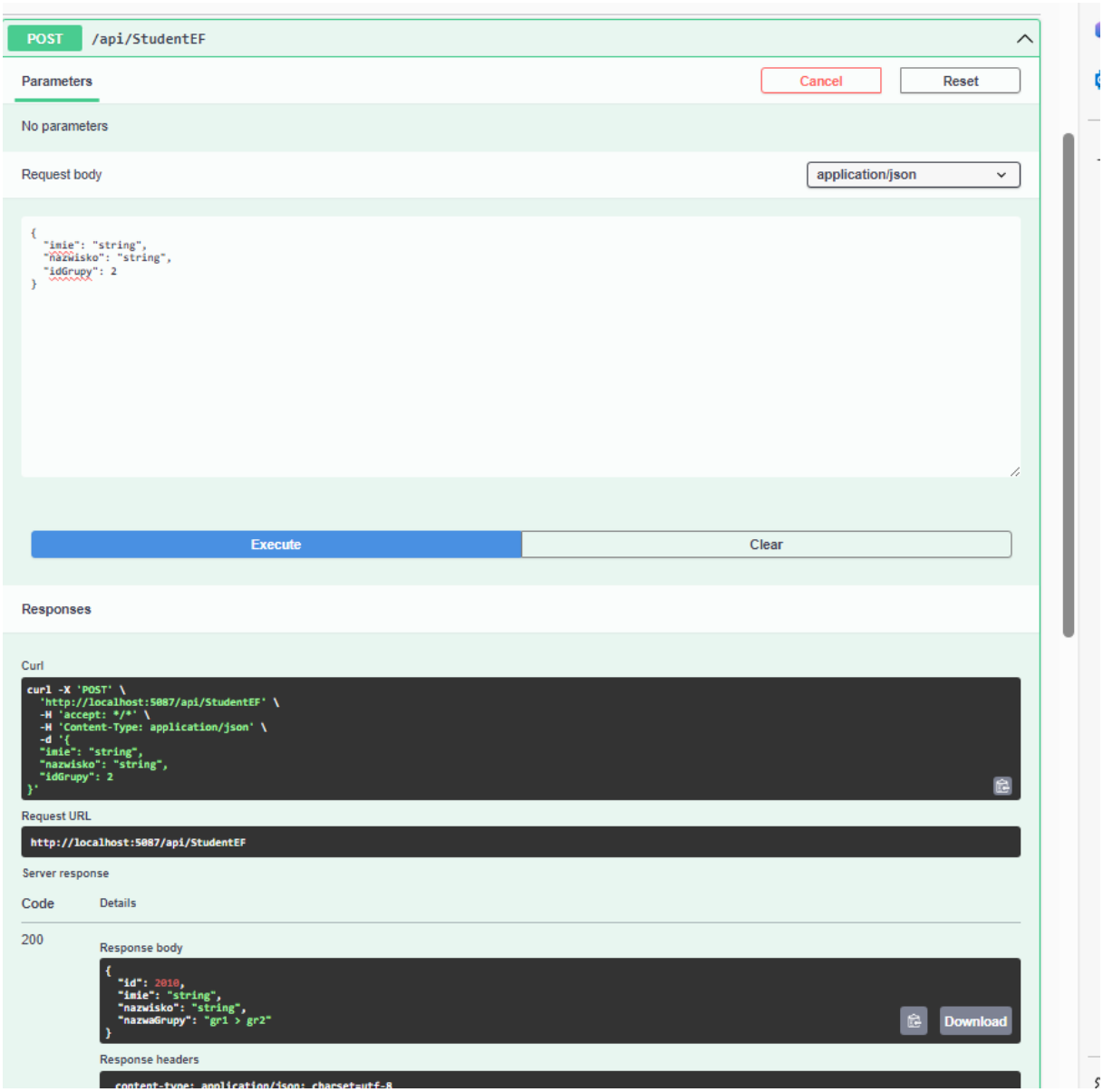
```
using System.Text;
using System.Threading.Tasks;

namespace Models
{
    Odwołania: 7
    public class Grupa
    {
        [Key]
        Odwołania: 2
        public int Id { get; set; }
        Odwołania: 9
        public string Nazwa { get; set; }
        1 odwołanie
        public List<Student>? Students { get; set; }
        //RELACJA REKURENCYJNA
        Odwołania: 3
        public int? ParentId { get; set; }
        [ForeignKey(nameof(ParentId))]
        1 odwołanie
        public Grupa? Parent { get; set; }
        1 odwołanie
        public List<Grupa>? Children { get; set; }


        //między historia a grupa nie widzę sensu relacji więc jej nie tworzę
        //tabela ma przechowywać tylko id jak było powiedziane, a nazwę grupy mogę pobrać z relacji student-grupa
    }

}
```

Funkcja:



```
{
    private readonly WebContext _dbContext;
    Odwołania: 0
    public StudentEFService(WebContext dbContext)
    {
        _dbContext = dbContext;
    }
    // POLECENIE 8  RELACJA W MODELU DZIAŁA
    Odwołania: 5
    public string GetFullGroupName(Grupa grupa)
    {
        if (grupa == null)
        {
            return "brak";
        }

        var groupName = grupa.Nazwa != null ? grupa.Nazwa : "brak";

        if (grupa.ParentId != null)
        {
            var parentGroup = _dbContext.Grupy.Find(grupa.ParentId);
            var parentGroupName = GetFullGroupName(parentGroup);
            groupName = $"{parentGroupName} > {groupName}";
        }

        return groupName;
    }

    Odwołania: 2
    public StudentResponseDTO AddStudent(StudentRequestDTO student)
    {
        Student newStudent = new Student
        {
            Imie = student.Imie,
            Nazwisko = student.Nazwisko,
            IdGrupy = student.IdGrupy
        };

        _dbContext.Studenci.Add(newStudent);
        _dbContext.SaveChanges();

        var result = _dbContext.Studenci.Include(x => x.Grupa).OrderBy(i => i.Id).LastOrDefault();
        if (result == null)
        {
            throw new Exception("Wyjatek");
        }
        return new StudentResponseDTO
        {
            Id = result.Id,
            Imie = result.Imie,
            Nazwisko = result.Nazwisko,
            NazwaGrupy = GetFullGroupName(result.Grupa)
        };
```

Może nie do końca rekurencyjnie ale coś jest na 0,25 punkta.


POST
/api/StudentEF
Parameters
Cancel
Reset
No parameters
Request body
application/json
{
"imie": "string",
"nazwisko": "string",
"idGrupy": 2
}
Execute
Clear
Responses
Curl
curl -X 'POST' \
'http://localhost:5087/api/StudentEF' \
-H 'accept: */*' \
-H 'Content-Type: application/json' \
-d '{
"imie": "string",
"nazwisko": "string",
"idGrupy": 2
}'
Request URL
http://localhost:5087/api/StudentEF
Server response
Code
Details
200
Response body
{
"id": 2010,
"imie": "string",
"nazwisko": "string",
"nazwaGrupy": "gr1 > gr2"
}
Download
Response headers