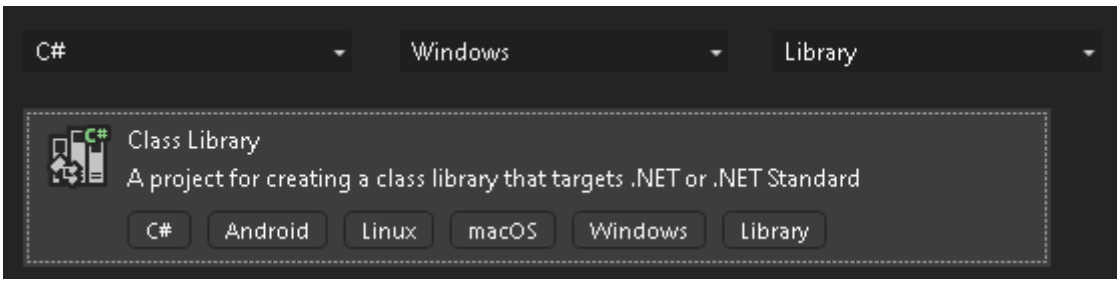
Backend refactoring

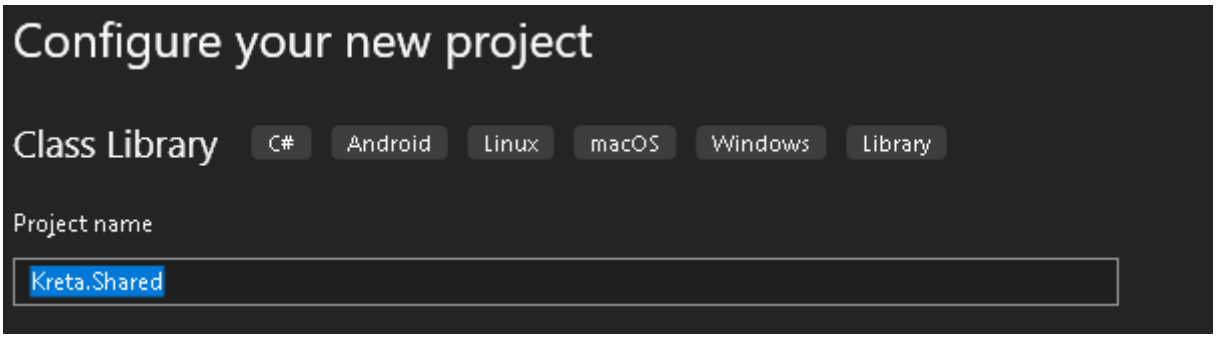
Kódolás a frontend alkalmazásban

A backend oldalon adott egy Student osztály amelyben van Id, a desktop oldali Student osztályban nincs Id Ugyan úgy az EducationLavel osztály mindkét oldalon megtalálható.
De a ControllerResponse osztály csak backend oldalon van, a frontend program így nem ismeri ezt az osztályt, pedig szüksége lesz rá amikor pl. az update híváskor megkapja a választ.

Készítsük egy olyan projektet, melyben az osztályok a Desktop és Backend (Maui, Blazor) projektek számára is elérhető lesz!

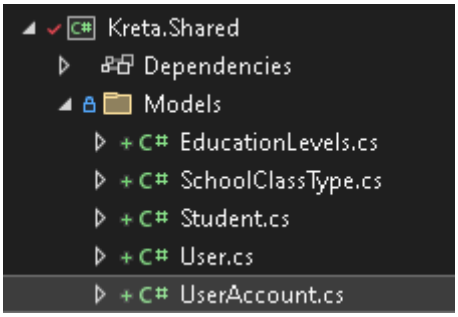


A projekt egy Class Library templete projektből épül fel, ezt válasszuk.
A projekt neve legyen Kreta.Shared.

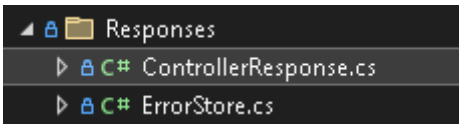


Az eredmény:

Hozzuk át a backendről és a frontendről a szükséges osztályokat:

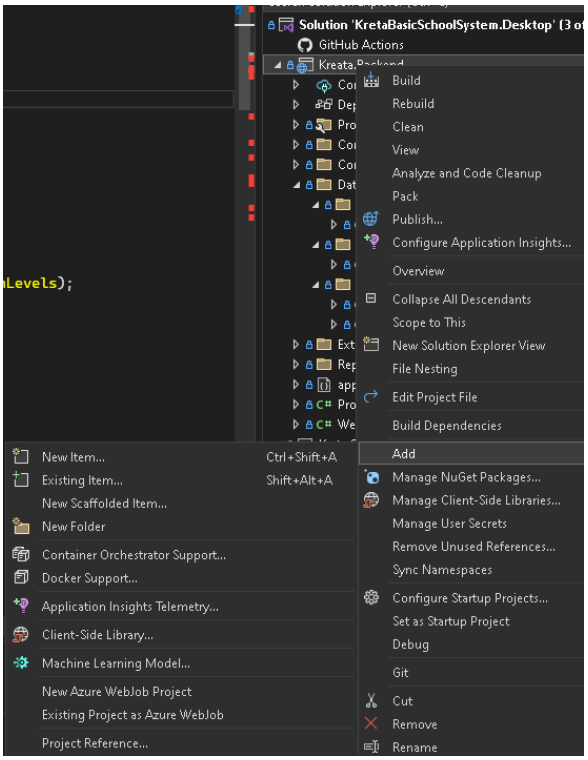


Az osztott mappába létrehozzuk a Responses mappát, abban az ErrorStore.cs és ControllerResponse.cs állományokt. Az állományok kódját a Backend projekt Datas\Responses almappájában lévő fájlok kódját áthelyezzük az osztott mappába:

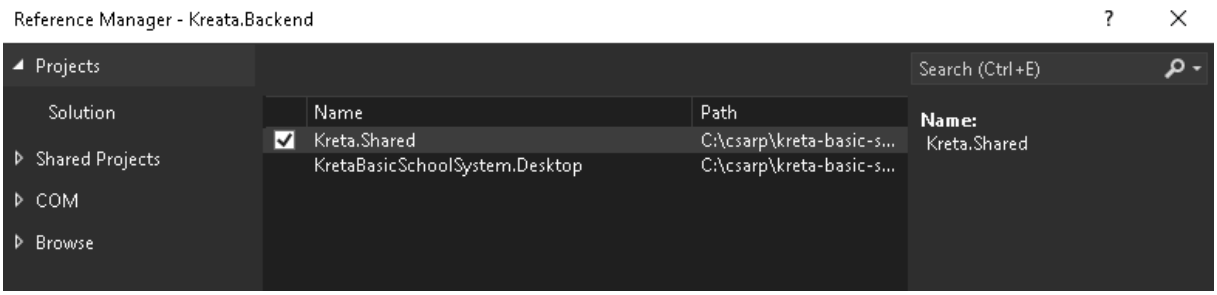


Hogy a két projekt (Desktop, Backend) kapcsolatba legyen az osztott projekttel (Shared) úgynevezett project referenceket készítünk közöttük:

a) A Backend projekten kiválasztjuk a project referenc gyorsmenüt (bal alsó része a képnek)



b) Kiválasztjuk a „Kreta.Shared” projektet.



Ezt megismételjük a frontend (Desktop, Blazor, Maui) projekten is!

DTO osztályok

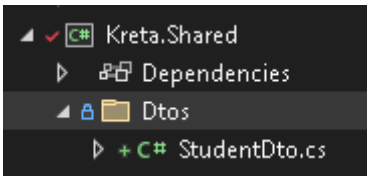
Nem szerencsés, ha a backenden lévő adatok egy-az egyben érkeznek meg a desktopra vagy frontendre.
Pl.

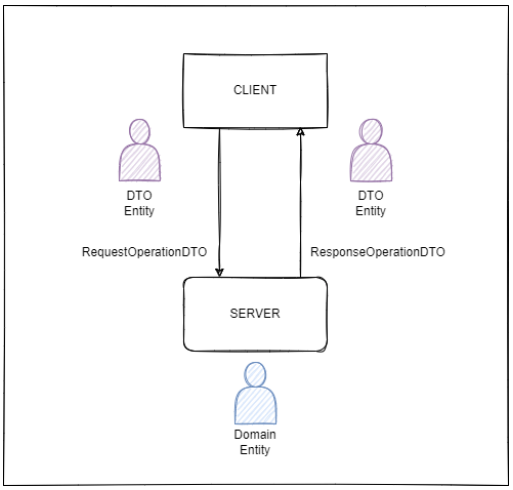
Ha van egy felhasználó, nem célszerű a felhasználói adatok lekérése során a jelszót is elküldeni a backendről, mert valaki megfejtheti.
Készíthetünk külön Dto osztályt a létrehozásra és módosításra (StudentForCreation, StudentForUpdate) ha valami miatt (pl. adatbáziskapcsolatok) más propertyket kell kezelnünk.

Ezekben a DTO osztályokban általában csak tulajdonságok vannak:

- Data transfer object - Wikipedia
- model view controller - What is a Data Transfer Object (DTO)? - Stack Overflow
- What is the Difference Between a DTO and a POCO? - Code Maze (code-maze.com)
- .NET Core Web API Best Practices - Code Maze Blog (code-maze.com)

Készítsünk az osztott mappában egy StudentDto-t, amely most nagyban hasonlítani fog a Student osztályra, de a jövőbe a Student osztályhoz adhatunk hozzá tulajdonságokat, a StudentDto osztályban csak az adatok lesznek.





A Student osztályba viszont vegyük fel az Id-t. A StudentDto osztályban is legyen Id. A StudentDto csak property-ket és konstruktort tartalmazzon!

Refaktorálás

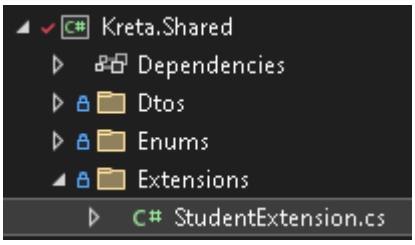
Refaktorálásnak hívjuk azt a folyamatot, amikor a kód adatait vagy annak algoritmusát újra végig gondoljuk és újratervezzük:
1. refaktorálás
A backend (Cotnrollerek) DTO adatokat szolgáltatnak:

DM (DataModel) to DTO (DataTransfer Model)

Student->StudentDto
Készítünk metódusokat a DTO és DM osztályok közötti konvertálásra. Ehhez felhasználjuk a statikus osztályokat és statikus metósusokat.

Statikus osztályok és metódusok

- C# Intermediate - Static Members in C# - Code Maze (code-maze.com)



Az osztott (Shared) projektben az Extension mappába elkészítjük a StudentExtension statikus osztályt, abba a ToStudentDto metódust, amely a Student osztályt StudentDto osztállyá konvertálja. És elkészítjük ennek inverz műveletét is (itt az egyik metódust látjuk példaként).

```
public static StudentDto ToStudentDto(this Student student)
{
    return new StudentDto
    {
        Id=student.Id,
        FirstName = student.FirstName,
        LastName = student.LastName,
```

```
        BirthsDay = student.BirthsDay,
        SchoolYear = student.SchoolYear,
        SchoolClass = student.SchoolClass,
        EducationLevel = student.EducationLevel
    };
}
```

Ez egy kiterjesztett metódus, amely lehetővé teszi, hogy egy meglévő típust (this Student) további statikus metódusokkal bővítsünk. Az ilyen metódusokat egy statikus osztályon belül kell létrehoznunk, és az első paraméterük elé a this kulcsszó kerül.
De miért kell az első paraméter elé előtagot tennünk?
Mert ez a paraméter egy jelző, amely megmondja a fordítónak, hogy melyik típust bővítjük.
Tehát használhatjuk a következő képpen (ez csak példa, nem kell sehova begépelni):

```
Student student=new Student
{
    Id=Guid.NewGuid(),
    FirstName="János",
    LastName="Jegy",
    BirthsDay=new DateTime(2022,10,10),
    SchoolYear=9,
    SchoolClass = SchoolClassType.ClassA,
    EducationLevel="érettségi"
}
StudentDto studentDto=student.ToStudentDto();
```

Lássuk mit kell tenni ezért a backenden:

- Backend oldalon a model mappák törlése
- Context mappában a namespcaek egyeztetése
- A repó rétegben a namespcaek egyeztetése
- A Controller rétegben a namespcaek egyeztetése
- Megmaradt a StudentDto osztályban egy namespace egyeztetés

A Controller rétegben a `GetBy(Guid Id)` StudentDto-t ad vissza

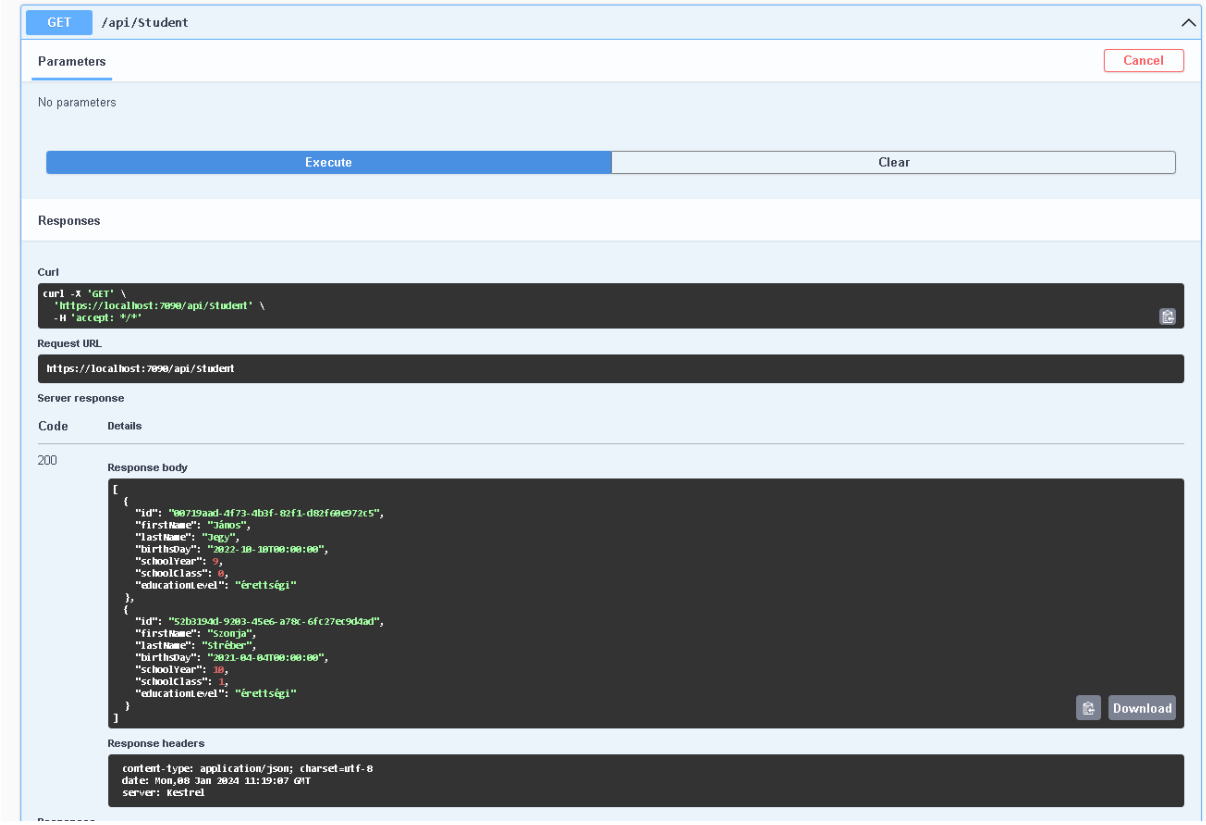
```
        [HttpGet("{id}")]
        public async Task<IActionResult> GetBy(Guid id)
        {
            Student? entity = new();
            if (_studentRepo is not null)
            {
                entity = await _studentRepo.GetBy(id);
                if (entity!=null)
                    return Ok(entity.ToStudentDto());
            }
            return BadRequest("Az adatok elérhetetlenek!");
        }
```

d) A `SelectAllRecordToListAsync` metódus StudentDto-t ad vissza
Lásd Select metódust: [LINQ Select (csharptutorial.net)]

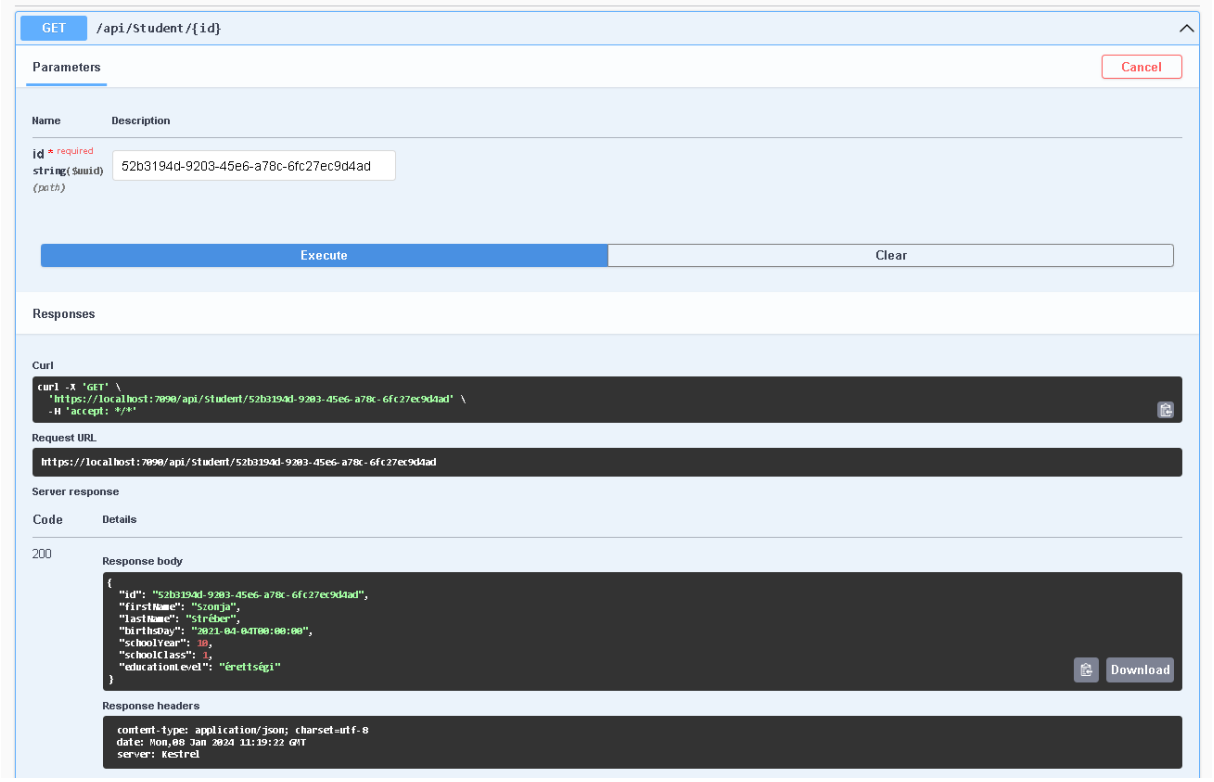
```
[HttpGet]
public async Task<IActionResult> SelectAllRecordToListAsync()
{
    List<Student>? users = new();

    if (_studentRepo != null)
    {
        users = await _studentRepo.GetAll();
        return Ok(users.Select(student => student.ToStudentDto()));
    }
    return BadRequest("Az adatok elérhetetlenek!");
}
```

e) Az `UpdateStudentAsync` metódus StudentDto osztályt vár. Ennek megfelelően írjuk át a controller működését!
f) ModelBuilderExtension osztályban is módosítani kell a usingokat

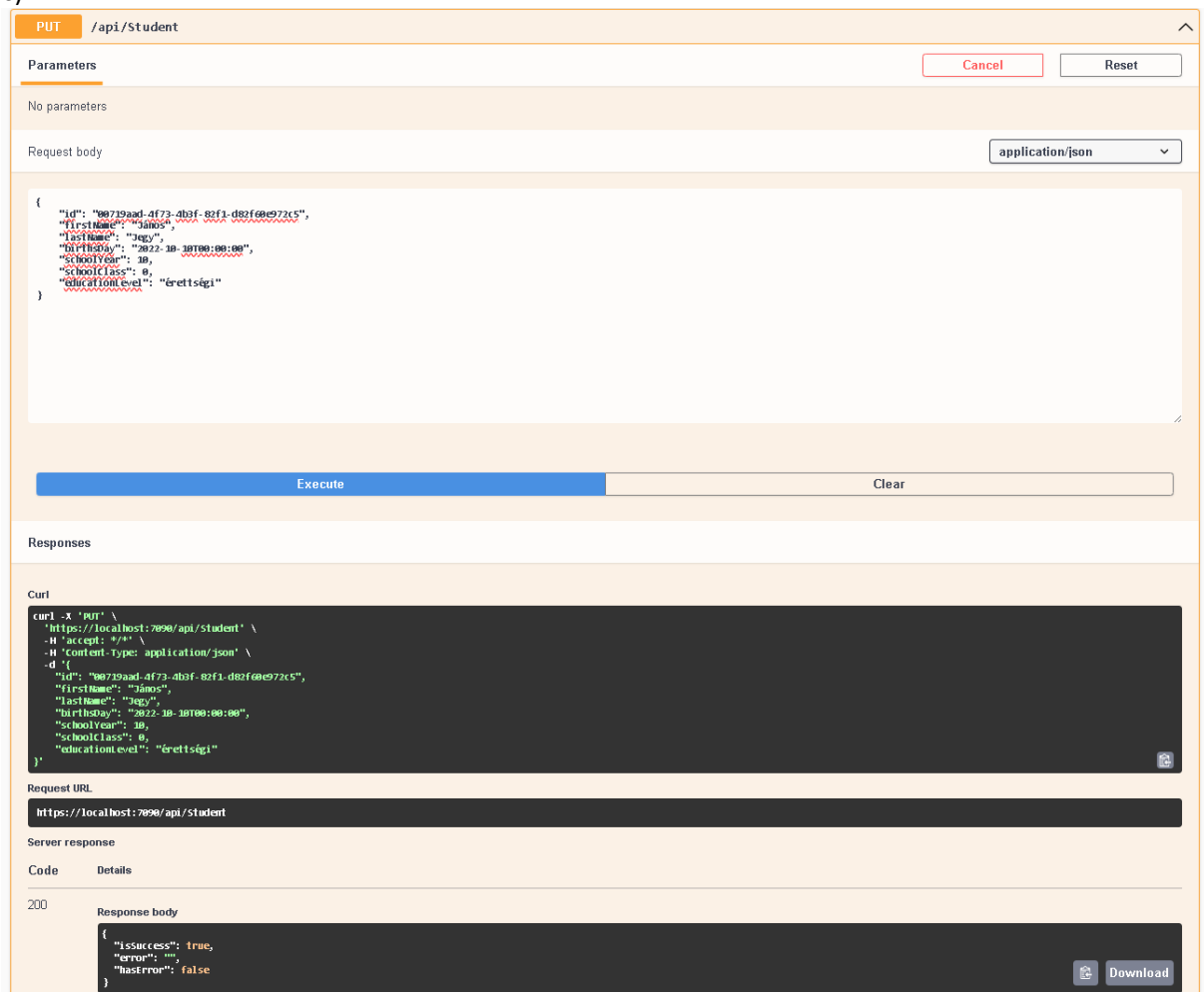
Tesztelhetjük a backend-et!
a)



b)


GET
/api/Student/{id}
Parameters
Cancel
Name
Description
id * required
string($uuid)
(path)
52b3194d-9203-45e6-a78c-6fc27ec9d4ad
Execute
Clear
Responses
Curl
curl -X 'GET' \
'https://localhost:7090/api/Student/52b3194d-9203-45e6-a78c-6fc27ec9d4ad' \
-H 'accept: */*'
Request URL
https://localhost:7090/api/Student/52b3194d-9203-45e6-a78c-6fc27ec9d4ad
Server response
Code
Details
200
Response body
{
"id": "52b3194d-9203-45e6-a78c-6fc27ec9d4ad",
"firstName": "Sonja",
"lastName": "Strèber",
"birthsDay": "2021-04-04T00:00:00",
"schoolYear": 10,
"schoolClass": 1,
"educationLevel": "érettségi"
}
Download
Response headers
content-type: application/json; charset=utf-8
date: Mon,08 Jan 2024 11:19:22 GMT
server: Kestrel

c)


PUT
/api/Student
Parameters
Cancel
Reset
No parameters
Request body
application/json
{
"id": "00719aad-4f73-4b3f-82f1-d82f60e972c5",
"firstName": "János",
"lastName": "Jegy",
"birthsDay": "2022-10-10T00:00:00",
"schoolYear": 10,
"schoolClass": 0,
"educationLevel": "érettségi"
}
Execute
Clear
Responses
Curl
curl -X 'PUT' \
'https://localhost:7090/api/Student' \
-H 'accept: */*' \
-H 'Content-Type: application/json' \
-d '{
"id": "00719aad-4f73-4b3f-82f1-d82f60e972c5",
"firstName": "János",
"lastName": "Jegy",
"birthsDay": "2022-10-10T00:00:00",
"schoolYear": 10,
"schoolClass": 0,
"educationLevel": "érettségi"
}'
Request URL
https://localhost:7090/api/Student
Server response
Code
Details
200
Response body
{
"issuccess": true,
"error": "",
"hasError": false
}
Download