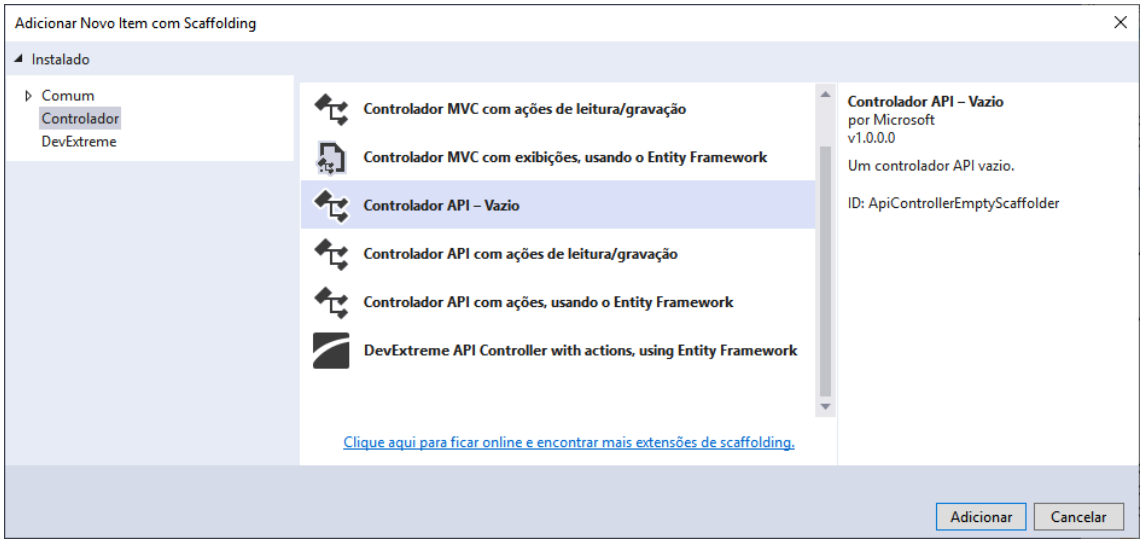
UnitOfWork

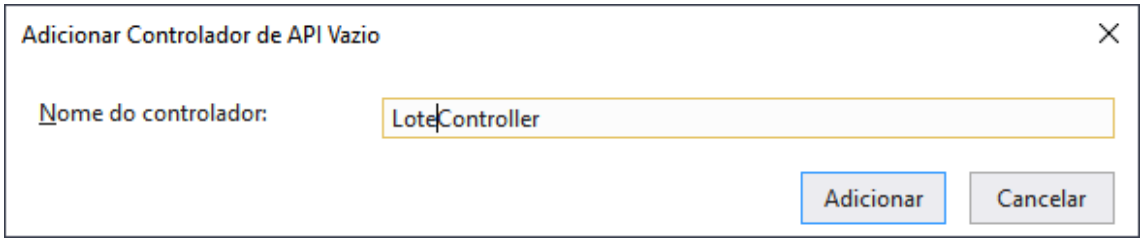
Padrão utilizado para realizar **gerenciamento de transações** no projeto através do EntityFramework (**Commit** e **Rollback**)

Neste projeto, iremos criar um serviço na API para gravação de estoque e produtos em lote, ou seja, será uma requisição para a API onde o cliente deverá enviar os dados de um estoque e junto uma listagem de produtos.

A API deverá receber essas informações e executar uma transação onde será criado o Estoque e adicionado neste estoque todos os produtos enviados.

Primeiro passo:
Criando um ENDPOINT na API para executar o processo de gravação de estoque e produtos em lote (**/api/Lote**)

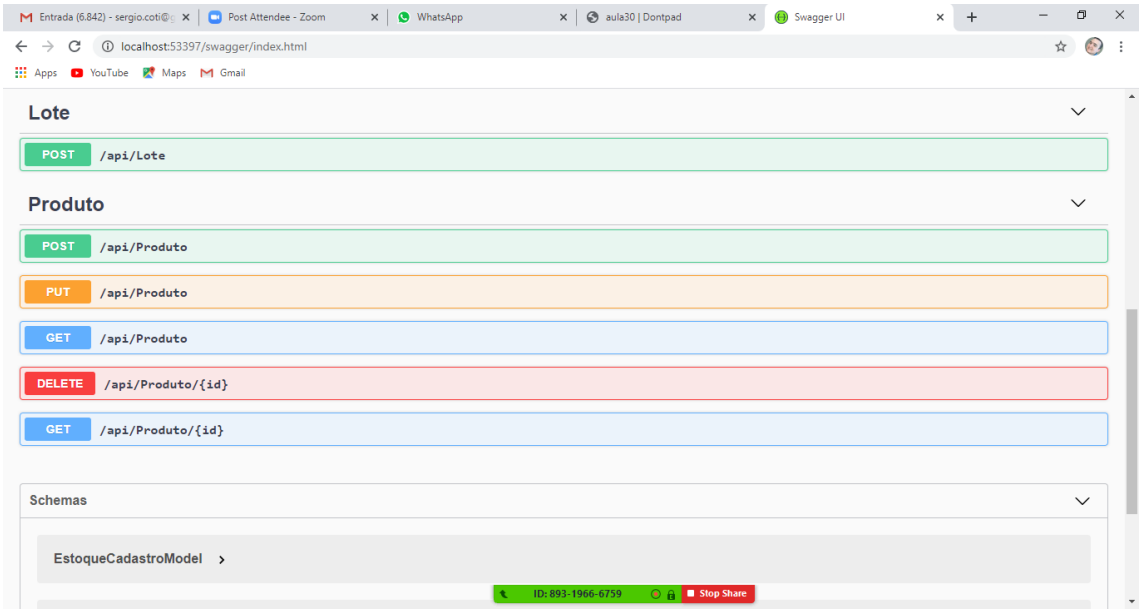





```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;

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

```
    public class LoteController : ControllerBase
    {
        [HttpPost]
        public IActionResult Post()
        {
            return Ok();
        }
    }
}
```

Executando:
http://localhost:53397/swagger

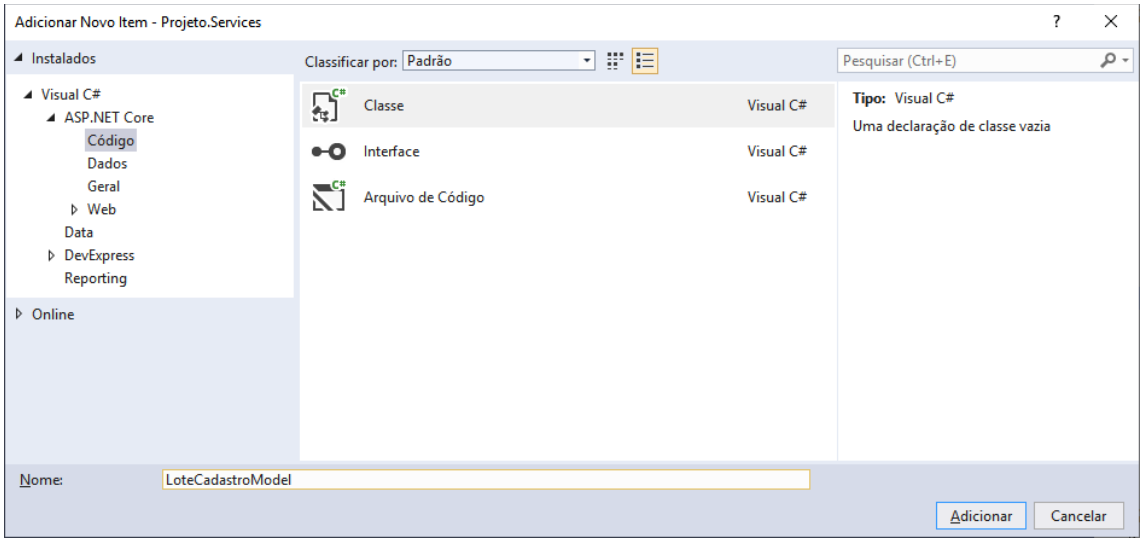


Criando classes de modelo para gravação do lote de produtos:
JSON request

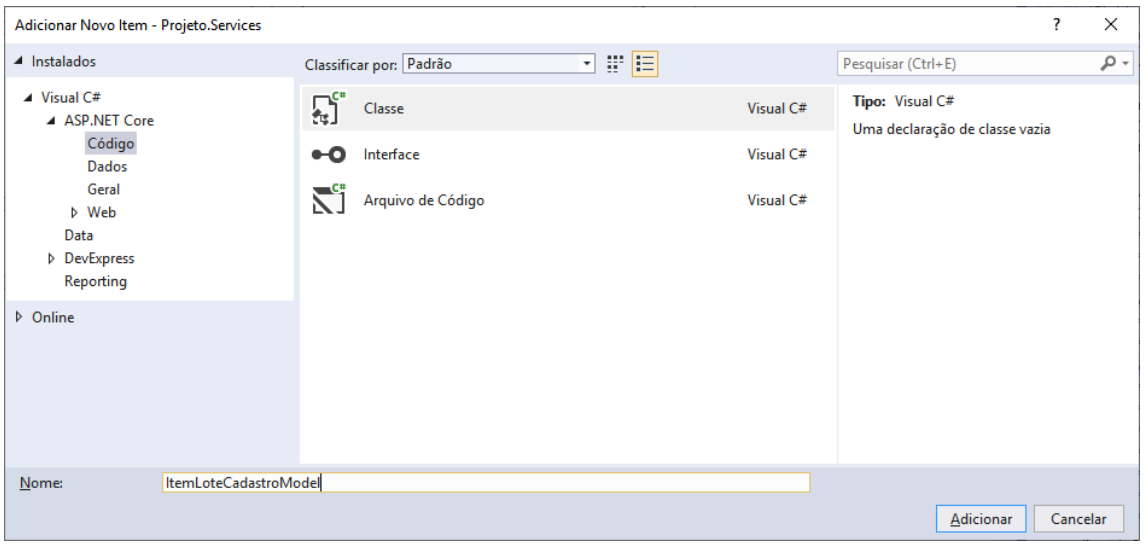
```
{
    "Estoque" : "Produtos de Informática",
    "Produtos" : [
        {
            "Nome" : "Notebook",
            "Preco" : 2000,
            "Quantidade" : 10
        },
        {
            "Nome" : "Mouse",
            "Preco" : 40,
            "Quantidade" : 15
        },
    ]
}
```

Classes de Modelo:

- **LoteCadastroModel.cs**
- **ItemLoteCadastroModel.cs**




```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.ComponentModel.DataAnnotations; //validações

namespace Projeto.Services.Models
{
    public class LoteCadastroModel
    {
        [Required(ErrorMessage = "Por favor, informe o nome do estoque.")]
        public string Estoque { get; set; }
    }
}
```



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.ComponentModel.DataAnnotations; //validações

namespace Projeto.Services.Models
{
    public class ItemLoteCadastroModel
    {
        [Required(ErrorMessage = "Por favor, informe o nome do produto.")]
        public string Nome { get; set; }

        [Required(ErrorMessage = "Por favor, informe o preço do produto.")]
        public decimal Preco { get; set; }

        [Required(ErrorMessage = "Por favor, informe a quantidade do produto.")]
        public int Quantidade { get; set; }
    }
}
```

Relacionando as classes de modelo para gerar o JSON abaixo:

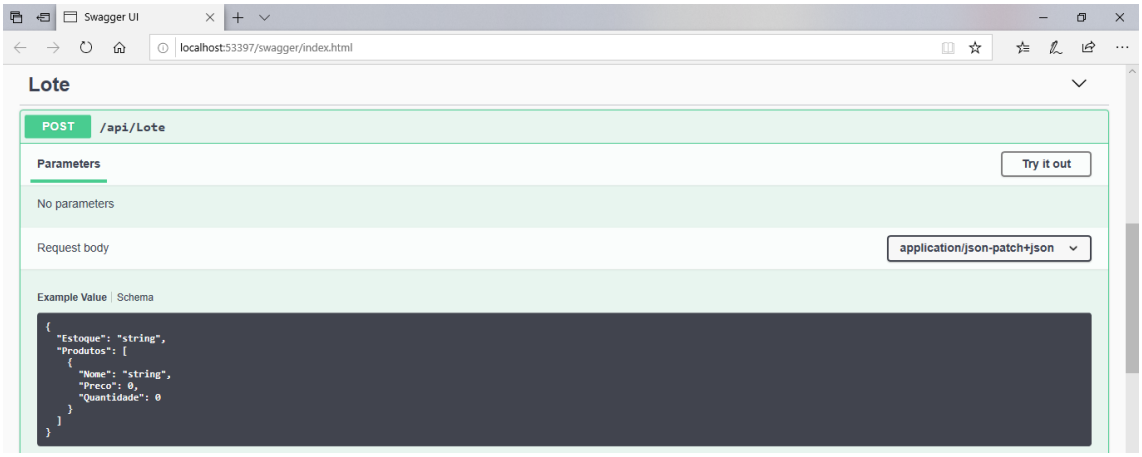
```
{
    "Estoque" : "Produtos de Informática",
    "Produtos" : [
        {
            "Nome" : "Notebook",
            "Preco" : 2000,
            "Quantidade" : 10
        },
        {
            "Nome" : "Mouse",
            "Preco" : 40,
            "Quantidade" : 15
        },
    ]
}
```

/Models/LoteCadastroModel.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.ComponentModel.DataAnnotations; //validações

namespace Projeto.Services.Models
{
    public class LoteCadastroModel
    {
        [Required(ErrorMessage = "Por favor, informe o nome do estoque.")]
        public string Estoque { get; set; }

        public List<ItemLoteCadastroModel> Produtos { get; set; }
    }
}
```

Voltando na classe LoteController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Projeto.Services.Models;

namespace Projeto.Services.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class LoteController : ControllerBase
    {
        [HttpPost]
        public IActionResult Post(LoteCadastroModel model)
        {
            return Ok();
        }
    }
}
```



AutoMapper

Fazendo o mapeamento para obter os dados das Models de Lote e passá-los para as entidades Estoque e Produto.

ModelToEntityMapping

```
using AutoMapper;
using Projeto.Data.Entities;
using Projeto.Services.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace Projeto.Services.Mappings
{
    //classe de mapeamento do AutoMapper de forma a permitir
    //que classes de modelo (Models) possam transferir seus dados
    //para classes de entidade (Entities)
    public class ModelToEntityMapping : Profile
    {
        //construtor -> ctor + 2x[tab]
        public ModelToEntityMapping()
        {
            CreateMap<EstoqueCadastroModel, Estoque>();
            CreateMap<ProdutoCadastroModel, Produto>();

            CreateMap<EstoqueEdicaoModel, Estoque>();
            CreateMap<ProdutoEdicaoModel, Produto>();

            CreateMap<LoteCadastroModel, Estoque>()
                .AfterMap((src, dest) => dest.Nome = src.Estoque);

            CreateMap<ItemLoteCadastroModel, Produto>();
        }
    }
}
```

Desenvolvendo o UnitOfWork

/Contracts/IUnitOfWork

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Projeto.Data.Contracts
{
    public interface IUnitOfWork
    {
        void BeginTransaction();
        void Commit();
        void Rollback();

        IEstoqueRepository EstoqueRepository { get; }
        IProdutoRepository ProdutoRepository { get; }
    }
}
```

IDbContextTransaction

Classe do EntityFramework utilizada para realizar o gerenciamento de transações (COMMIT / ROLLBACK, abertura de Transações)

```
using Microsoft.EntityFrameworkCore.Storage;
using Projeto.Data.Contexts;
using Projeto.Data.Contracts;
using System;
using System.Collections.Generic;
using System.Text;

namespace Projeto.Data.Repositories
{
    public class UnitOfWork : IUnitOfWork
    {
        private readonly DataContext dataContext;
        private IDbContextTransaction transaction;

        public UnitOfWork(DataContext dataContext)
        {
            this.dataContext = dataContext;
        }

        public void BeginTransaction()
        {
            //abrir uma transação no banco de dados através do EF
            transaction = dataContext.Database.BeginTransaction();
        }

        public void Commit()
        {
            //executando e finalizando a transação
            transaction.Commit();
        }

        public void Rollback()
        {
            //desfazer a transação
            transaction.Rollback();
        }

        public IEstoqueRepository EstoqueRepository
        {
            get { return new EstoqueRepository(dataContext); }
        }

        public IProdutoRepository ProdutoRepository
        {
            get { return new ProdutoRepository(dataContext); }
        }
    }
}
```

Startup.cs

Mapear a injeção de dependência do **UnitOfWork**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using AutoMapper;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Options;
using Microsoft.OpenApi.Models;
using Projeto.Data.Contexts;
using Projeto.Data.Contracts;
using Projeto.Data.Repositories;

namespace Projeto.Services
{
    public class Startup
    {
        public Startup(IConfiguration configuration)
        {
            Configuration = configuration;
        }

        public IConfiguration Configuration { get; }

        // This method gets called by the runtime.
        // Use this method to add services to the container.
        public void ConfigureServices(IServiceCollection services)
        {
            services.AddMvc().SetCompatibilityVersion
                    (CompatibilityVersion.Version_2_1);

            #region EntityFramework

            services.AddDbContext<DataContext>
                (options => options.UseSqlServer
                (Configuration.GetConnectionString("Aula")));

            #endregion

            #region Swagger

            services.AddSwaggerGen(
               c =>
               {
                   c.SwaggerDoc("v1", new OpenApiInfo
                   {
                       Title = "Sistema de Controle de Produtos",
                       Description = "API REST para integração
                                      com serviços de produtos",
                       Version = "v1",
                       Contact = new OpenApiContact
                       {
```

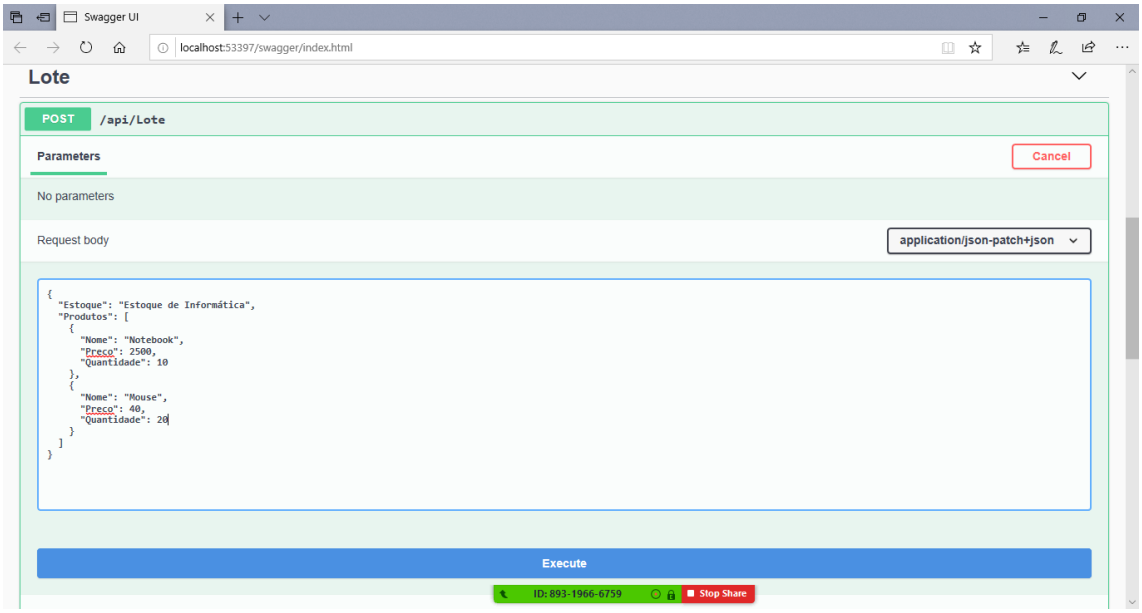
```
                    Name = "COTI Informática",
                    Url = new Uri("http://www.cotiinformatica.com.br/"),
                    Email = "contato@cotiinformatica.com.br"
                }
            });
        }
        );

        #endregion

        #region Injeção de Dependência


        services.AddTransient<IEstoqueRepository, EstoqueRepository>();
        services.AddTransient<IProdutoRepository, ProdutoRepository>();
        services.AddTransient<IUnitOfWork, UnitOfWork>();


        #endregion


        #region AutoMapper

        services.AddAutoMapper(typeof(Startup));

        #endregion
    }

    // This method gets called by the runtime.
    // Use this method to configure the HTTP request pipeline.
    public void Configure(IApplicationBuilder app, IHostingEnvironment env)
    {
        if (env.IsDevelopment())
        {
            app.UseDeveloperExceptionPage();
        }

        #region Swagger

        app.UseSwagger();
        app.UseSwaggerUI(c =>
        {
            c.SwaggerEndpoint("/swagger/v1/swagger.json", "Projeto API");
        });

        #endregion

        app.UseMvc();


    }

}
```

Voltando na classe LoteController.cs
Implementando o cadastro do Estoque em Lote

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using AutoMapper;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Projeto.Data.Contracts;
using Projeto.Data.Entities;
using Projeto.Services.Models;

namespace Projeto.Services.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class LoteController : ControllerBase
    {
        //atributos..
        private readonly IUnitOfWork unitOfWork;
        private readonly IMapper mapper;

        //construtor para injeção de dependência
        public LoteController(IUnitOfWork unitOfWork, IMapper mapper)
        {
            this.unitOfWork = unitOfWork;
            this.mapper = mapper;
        }

        [HttpPost]
        public IActionResult Post(LoteCadastroModel model)
        {
            //verificar se todos os campos passaram nas regras de validação
            if(ModelState.IsValid)
            {
                unitOfWork.BeginTransaction(); //abrindo transação

                try
                {
                    //gravar estoque..
                    var estoque = mapper.Map<Estoque>(model);
                    unitOfWork.EstoqueRepository.Inserir(estoque);

                    //gravar os produtos..
                    foreach (var item in model.Produtos)
                    {
                        //gravar o produto
                        var produto = mapper.Map<Produto>(item);
                        produto.IdEstoque = estoque.IdEstoque;
                       //chave estrangeira

                        unitOfWork.ProdutoRepository.Inserir(produto);
                    }

                    unitOfWork.Commit(); //finalizar a transação

                    var result = new {message = "Lote cadastrado com sucesso."};
                    return Ok(result);
```

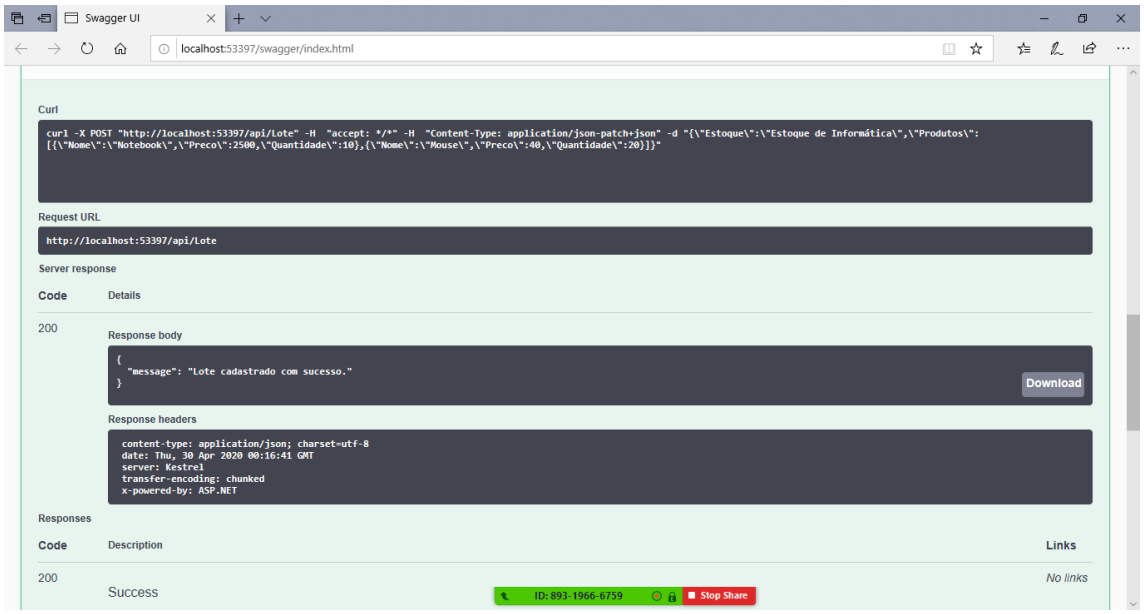
```
                }
                catch(Exception e)
                {
                    unitOfWork.Rollback(); //desfazer a transação
                    return StatusCode(500, "Erro: " + e.Message);
                }
            }
            else
            {
                return BadRequest("Ocorreram erros de validação.");
            }
        }
    }
}
```

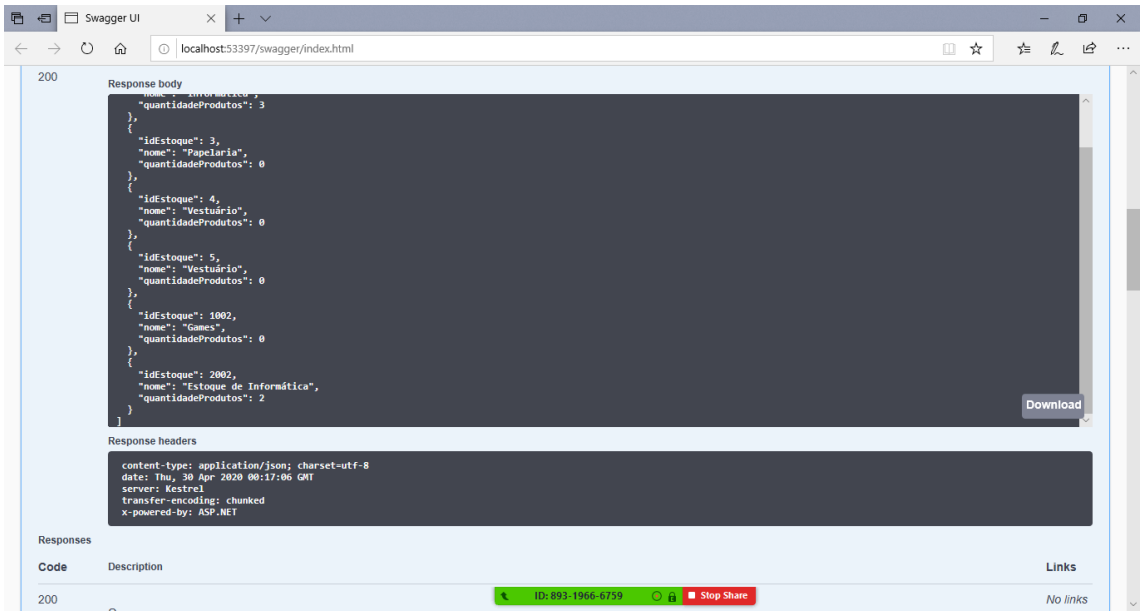
Testando:

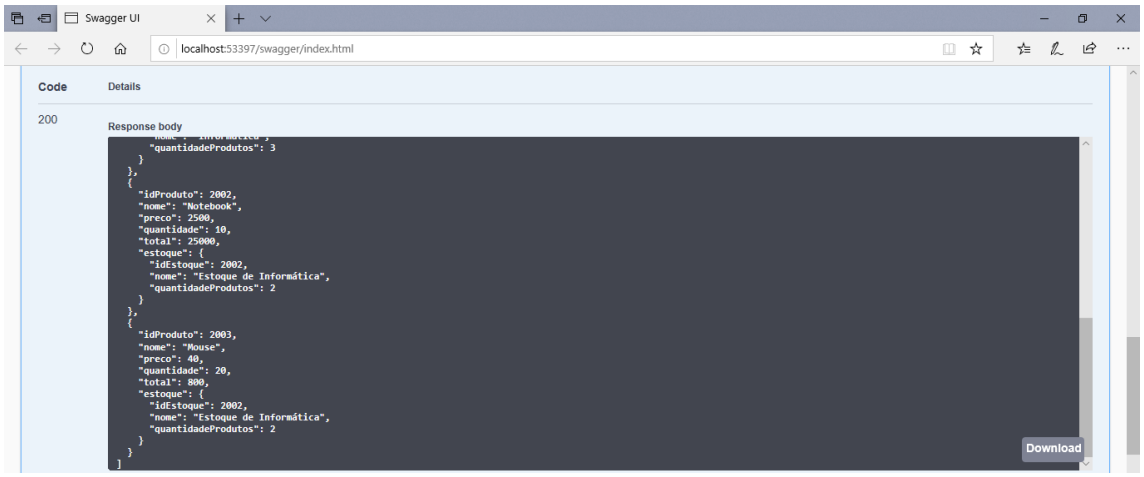


JSON Request:

```
{
  "Estoque": "Estoque de Informática",
  "Produtos": [
    {
      "Nome": "Notebook",
      "Preco": 2500,
      "Quantidade": 10
    },
    {
      "Nome": "Mouse",
      "Preco": 40,
      "Quantidade": 20
    }
  ]
}
```


Swagger UI
localhost:53397/swagger/index.html
Curl
curl -X POST "http://localhost:53397/api/Lote" -H "accept: */*" -H "Content-Type: application/json-patch+json" -d "{\"Estoque\":\"Estoque de Informática\",\"Produtos\":[{\"Nome\":\"Notebook\",\"Preco\":2500,\"Quantidade\":10},{\"Nome\":\"Mouse\",\"Preco\":40,\"Quantidade\":20}]}"
Request URL
http://localhost:53397/api/Lote
Server response
Code
Details
200
Response body
{
"message": "Lote cadastrado com sucesso."
}
Download
Response headers
content-type: application/json; charset=utf-8
date: Thu, 30 Apr 2020 00:16:41 GMT
server: Kestrel
transfer-encoding: chunked
x-powered-by: ASP.NET
Responses
Code
Description
Links
200
Success
No links
ID: 893-1966-6759
Stop Share


Swagger UI
localhost:53397/swagger/index.html
200
Response body
"quantidadeProdutos": 3
},
{
"idEstoque": 3,
"nome": "Papelaria",
"quantidadeProdutos": 0
},
{
"idEstoque": 4,
"nome": "Vestuário",
"quantidadeProdutos": 0
},
{
"idEstoque": 5,
"nome": "Vestuário",
"quantidadeProdutos": 0
},
{
"idEstoque": 1002,
"nome": "Games",
"quantidadeProdutos": 0
},
{
"idEstoque": 2002,
"nome": "Estoque de Informática",
"quantidadeProdutos": 2
}
]
Download
Response headers
content-type: application/json; charset=utf-8
date: Thu, 30 Apr 2020 00:17:06 GMT
server: Kestrel
transfer-encoding: chunked
x-powered-by: ASP.NET
Responses
Code
Description
Links
200
No links
ID: 893-1966-6759
Stop Share


Swagger UI
localhost:53397/swagger/index.html
Code
Details
200
Response body
"quantidadeProdutos": 3
}
},
{
"idProduto": 2002,
"nome": "Notebook",
"preco": 2500,
"quantidade": 10,
"total": 25000,
"estoque": {
"idEstoque": 2002,
"nome": "Estoque de Informática",
"quantidadeProdutos": 2
}
},
{
"idProduto": 2003,
"nome": "Mouse",
"preco": 40,
"quantidade": 20,
"total": 800,
"estoque": {
"idEstoque": 2002,
"nome": "Estoque de Informática",
"quantidadeProdutos": 2
}
}
]
Download