Definindo todos os métodos da classe **BaseRepository** como métodos "**virtual**", ou seja, métodos que irão permitir o uso de sobrescrita (OVERRIDE)

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

namespace Projeto.Data.Repositories
{
    public class BaseRepository<T> : IBaseRepository<T>
        where T : class
    {
        //atributo para armazenar o contexto do EF
        private readonly DataContext dataContext;

        //construtor para injeção de dependência
        //(construtor com entrada de argumentos)
        public BaseRepository(DataContext dataContext)
        {
            this.dataContext = dataContext;
        }

         public virtual void Inserir(T entity)
        {
            dataContext.Entry(entity).State = EntityState.Added; //inserção
            dataContext.SaveChanges(); //executando
        }

         public virtual void Alterar(T entity)
        {
            dataContext.Entry(entity).State = EntityState.Modified; //edição
            dataContext.SaveChanges(); //executando
        }

         public virtual void Excluir(T entity)
        {
            dataContext.Entry(entity).State = EntityState.Deleted; //exclusão
            dataContext.SaveChanges(); //executando
        }

         public virtual List<T> Consultar()
        {
            return dataContext.Set<T>().ToList();
        }

         public virtual List<T> Consultar(Func<T, bool> where)
        {
            return dataContext.Set<T>()
                .Where(where)
                .ToList();
        }
```

```
        public virtual T Obter(Func<T, bool> where)
        {
            return dataContext.Set<T>()
                .FirstOrDefault(where);
        }

        public virtual T ObterPorId(int id)
        {
            return dataContext.Set<T>()
                .Find(id); //buscar pelo id..
        }
    }
}
```

/Repositories/ProdutoRepository.cs

Sobrescrevendo os métodos de consulta de forma que façam rotinas de JOIN com as entidades relacionadas.

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


namespace Projeto.Data.Repositories
{
    public class ProdutoRepository : BaseRepository<Produto>, IProdutoRepository
    {
        private readonly DataContext dataContext;

        public ProdutoRepository(DataContext dataContext)
            : base(dataContext)
        {
            this.dataContext = dataContext;
        }

        //sobrescrita de método (OVERRIDE)
        public override List<Produto> Consultar()
        {
            //retornar uma consulta de Produtos
            //fazendo JOIN com a entidade Estoque
            return dataContext.Produto
                    .Include(p => p.Estoque) //JOIN..
                    .ToList();
        }
```

```
        //sobrescrita de método (OVERRIDE)
        public override List<Produto> Consultar(Func<Produto, bool> where)
        {
            //retornar uma consulta de Produtos fazendo
            //JOIN com a entidade Estoque
            return dataContext.Produto
                    .Include(p => p.Estoque) //JOIN..
                    .Where(where)
                    .ToList();
        }

        //sobrescrita de método (OVERRIDE)
        public override Produto Obter(Func<Produto, bool> where)
        {
            //retornar uma consulta de Produtos fazendo
            //JOIN com a entidade Estoque
            return dataContext.Produto
                    .Include(p => p.Estoque) //JOIN..
                    .Where(where)
                    .FirstOrDefault();
        }

        //sobrescrita de método (OVERRIDE)
        public override Produto ObterPorId(int id)
        {
            //retornar uma consulta de Produtos fazendo
            //JOIN com a entidade Estoque
            return dataContext.Produto
                    .Include(p => p.Estoque) //JOIN..
                    .FirstOrDefault(p => p.IdEstoque == id);
        }
    }
}
```

/Repositories/EstoqueRepository.cs

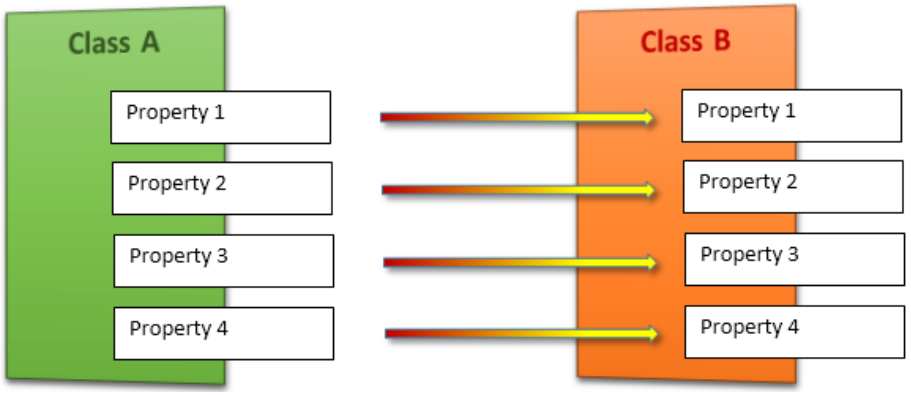
Sobrescrevendo os métodos de consulta de forma que façam rotinas de JOIN com as entidades relacionadas.

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

namespace Projeto.Data.Repositories
{
    public class EstoqueRepository : BaseRepository<Estoque>, IEstoqueRepository
    {
        //atributo
        private readonly DataContext dataContext;
```

```
        //construtor para injeção de dependência
        public EstoqueRepository(DataContext dataContext)
            : base(dataContext) //construtor da classe pai..
        {
            this.dataContext = dataContext;
        }

        public override List<Estoque> Consultar()
        {
            return dataContext.Estoque
                .Include(e => e.Produtos)
                .ToList();
        }

        public override List<Estoque> Consultar(Func<Estoque, bool> where)
        {
            return dataContext.Estoque
                .Include(e => e.Produtos)
                .Where(where)
                .ToList();
        }

        public override Estoque Obter(Func<Estoque, bool> where)
        {
            return dataContext.Estoque
                .Include(e => e.Produtos)
                .Where(where)
                .FirstOrDefault();
        }

        public override Estoque ObterPorId(int id)
        {
            return dataContext.Estoque
                .Include(e => e.Produtos)
                .FirstOrDefault(e => e.IdEstoque == id);
        }
    }
}
```

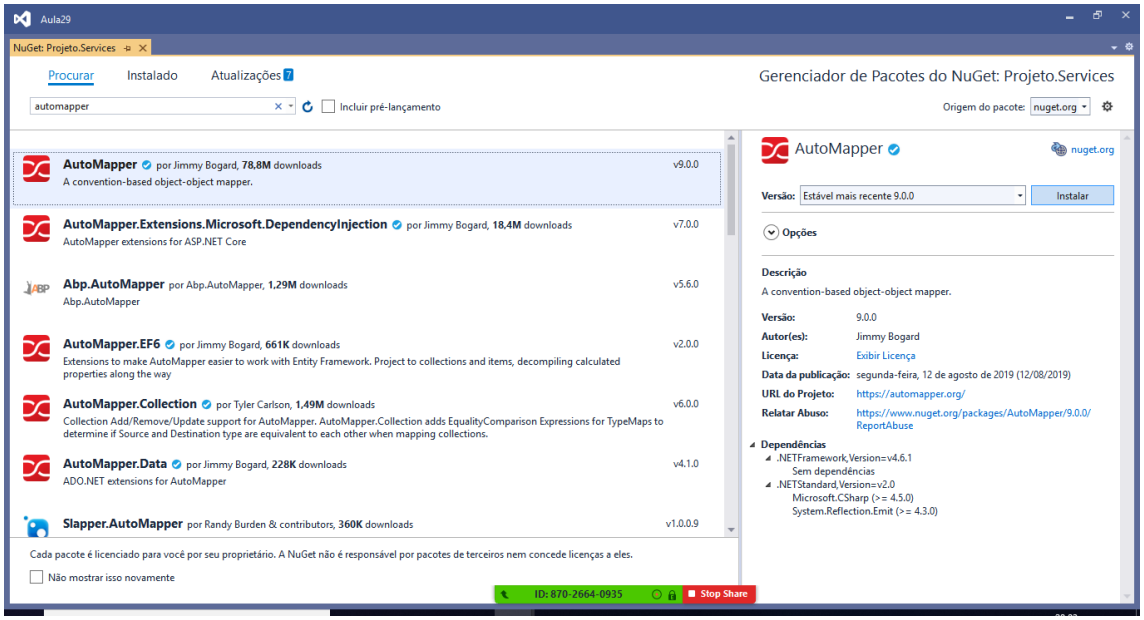
AutoMapper

Framework simples utilizado para mapeamento Objeto / Objeto que visa facilitar o processo de transferência de dados entre objetos (por exemplo a transferência dos dados de uma classe model para uma classe de entidade)

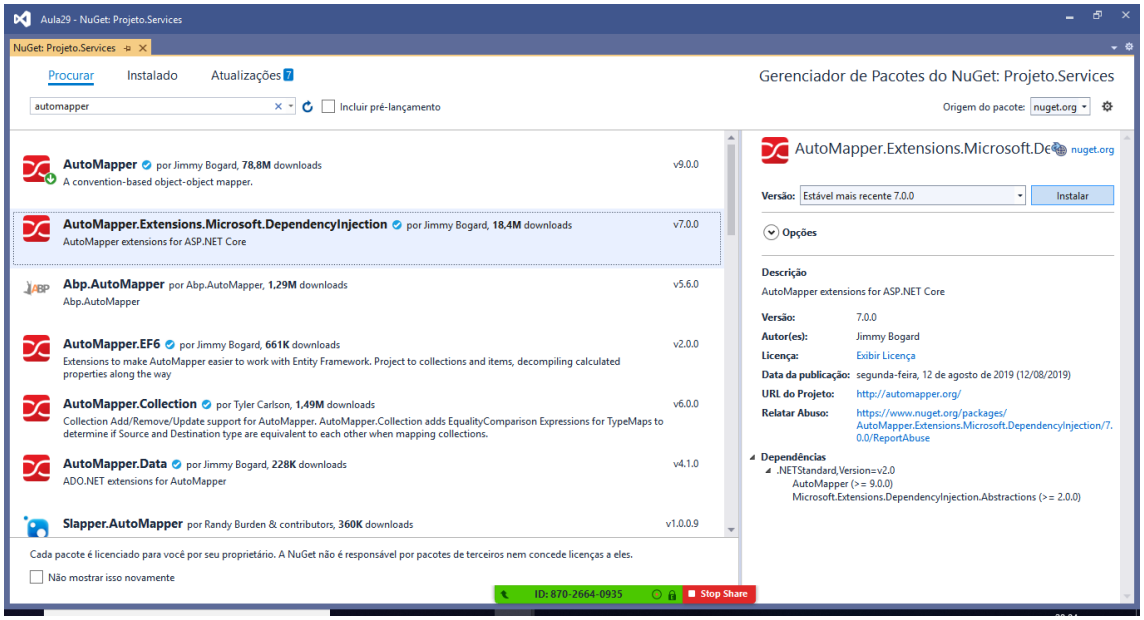


Instalando o **AutoMapper** no projeto **Services**
Gerenciar pacotes do NuGet

AutoMapper

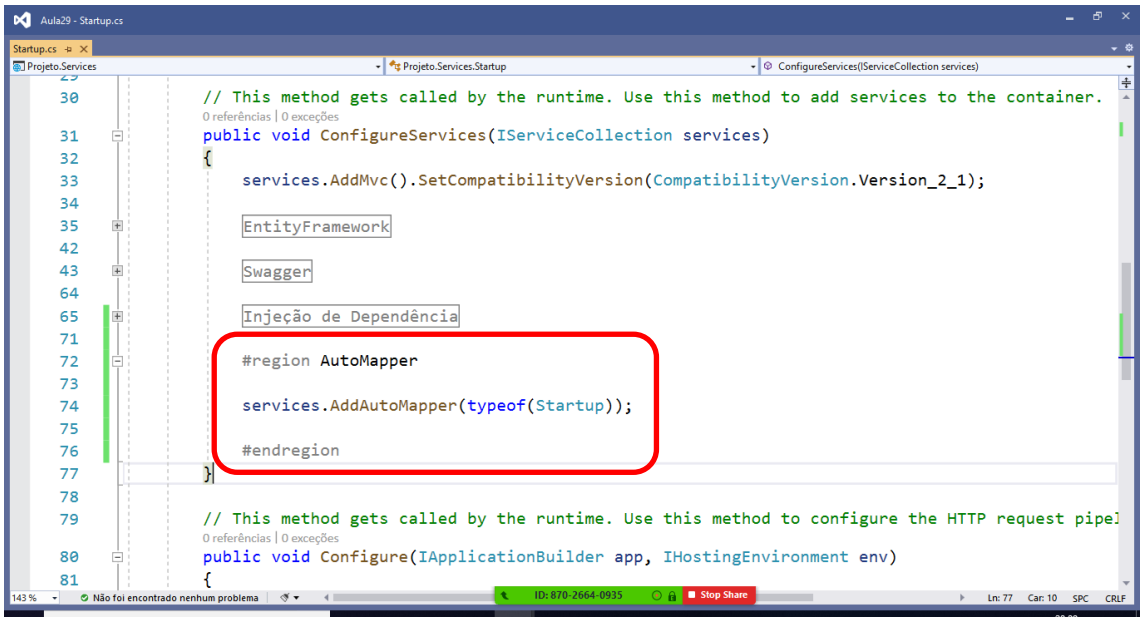


AutoMapper.Extensions.Microsoft.DependencyInjection



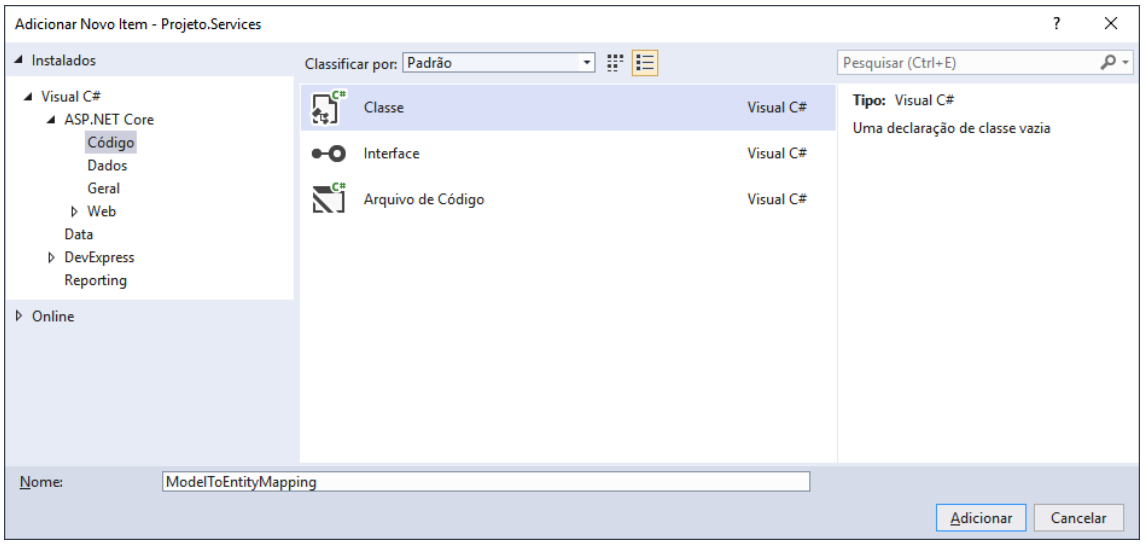
Registrando o AutoMapper no projeto .NET CORE
Startup.cs (Classe de inicialização do projeto)

```
#region AutoMapper
services.AddAutoMapper(typeof(Startup));
#endregion
```



/Mappings

Realizando o mapeamento das classes de Modelo para que possam transferir seus dados para as classes de entidade.




```
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>();
        }
    }
}
```

Refatorando o código dos controllers

Utilizando o AutoMapper

/Controller/EstoqueController.cs

```
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 EstoqueController : ControllerBase
    {
        //atributo
        private readonly IEstoqueRepository estoqueRepository;
        private readonly IMapper mapper;

        //construtor para injeção de dependência
        public EstoqueController(IEstoqueRepository estoqueRepository,
                                 IMapper mapper)
        {
            this.estoqueRepository = estoqueRepository;
            this.mapper = mapper;
        }

        [HttpPost]
        public IActionResult Post(EstoqueCadastroModel model)
        {
            //verificando se os campos da model passaram nas validações
            if(ModelState.IsValid)
            {
```

```
            try
            {
                var estoque = mapper.Map<Estoque>(model);
                estoqueRepository.Inserir(estoque);

                var result = new
                {
                    message = "Estoque cadastrado com sucesso",
                    estoque
                };

                return Ok(result); //HTTP 200 (SUCESSO!)
            }
            catch(Exception e)
            {
                return StatusCode(500, "Erro: " + e.Message);
            }
        }
        else
        {
            //Erro HTTP 400 (BAD REQUEST)
            return BadRequest("Ocorreram erros de validação.");
        }
    }

    [HttpPut]
    public IActionResult Put(EstoqueEdicaoModel model)
    {
        //verificando se os campos da model passaram nas validações
        if (ModelState.IsValid)
        {
            try
            {
                var estoque = mapper.Map<Estoque>(model);
                estoqueRepository.Alterar(estoque);

                var result = new
                {
                    message = "Estoque atualizado com sucesso",
                    estoque
                };

                return Ok(result); //HTTP 200 (SUCESSO!)
            }
            catch (Exception e)
            {
                return StatusCode(500, "Erro: " + e.Message);
            }
        }
        else
        {
            //Erro HTTP 400 (BAD REQUEST)
            return BadRequest("Ocorreram erros de validação.");
        }
    }

    [HttpDelete("{id}")]
    public IActionResult Delete(int id)
    {
```

```
        try
        {
            //buscar o estoque referente ao id informado..
            var estoque = estoqueRepository.ObterPorId(id);

            //verificar se o estoque foi encontrado..
            if(estoque != null)
            {
                //excluindo o estoque
                estoqueRepository.Excluir(estoque);

                var result = new
                {
                    message = "Estoque excluído com sucesso.",
                    estoque
                };

                return Ok(result);
            }
            else
            {
                return BadRequest("Estoque não encontrado.");
            }
        }
        catch(Exception e)
        {
            return StatusCode(500, "Erro: " + e.Message);
        }
    }

    [HttpGet]
    public IActionResult GetAll()
    {
        try
        {
            var result = estoqueRepository.Consultar();
            return Ok(result);
        }
        catch(Exception e)
        {
            return StatusCode(500, "Erro: " + e.Message);
        }
    }

    [HttpGet("{id}")]
    public IActionResult GetById(int id)
    {
        try
        {
            var result = estoqueRepository.ObterPorId(id);

            if(result != null) //se o estoque foi encontrado..
            {
                return Ok(result);
            }
            else
            {
                return NoContent(); //HTTP 204 (SUCESSO -> Vazio)
            }
        }
```

```
            catch (Exception e)
            {
                return StatusCode(500, "Erro: " + e.Message);
            }
        }
    }
}
```

/Controller/ProdutoController.cs

```
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 ProdutoController : ControllerBase
    {
        //atributo
        private readonly IProdutoRepository produtoRepository;
        private readonly IMapper mapper;

        //construtor para injeção de dependência
        public ProdutoController(IProdutoRepository produtoRepository,
                                 IMapper mapper)
        {
            this.produtoRepository = produtoRepository;
            this.mapper = mapper;
        }

        [HttpPost]
        public IActionResult Post(ProdutoCadastroModel model)
        {
            if(ModelState.IsValid)
            {
                try
                {
                    var produto = mapper.Map<Produto>(model);
                    produtoRepository.Inserir(produto);

                    var result = new
                    {
                        message = "Produto cadastrado com sucesso.",
                        produto
                    };

                    return Ok(result);
                }
```

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

    [HttpPut]
    public IActionResult Put(ProdutoEdicaoModel model)
    {
        if (ModelState.IsValid)
        {
            try
            {
                var produto = mapper.Map<Produto>(model);
                produtoRepository.Alterar(produto);

                var result = new
                {
                    message = "Produto atualizado com sucesso.",
                    produto
                };

                return Ok(result);
            }
            catch (Exception e)
            {
                return StatusCode(500, "Erro: " + e.Message);
            }
        }
        else
        {
            return BadRequest("Ocorreram erros de validação.");
        }
    }

    [HttpDelete("{id}")]
    public IActionResult Delete(int id)
    {
        try
        {
            var produto = produtoRepository.ObterPorId(id);

            if(produto != null)
            {
                produtoRepository.Excluir(produto);

                var result = new
                {
                    message = "Produto excluído com sucesso.",
                    produto
                };

                return Ok(result);
            }
```

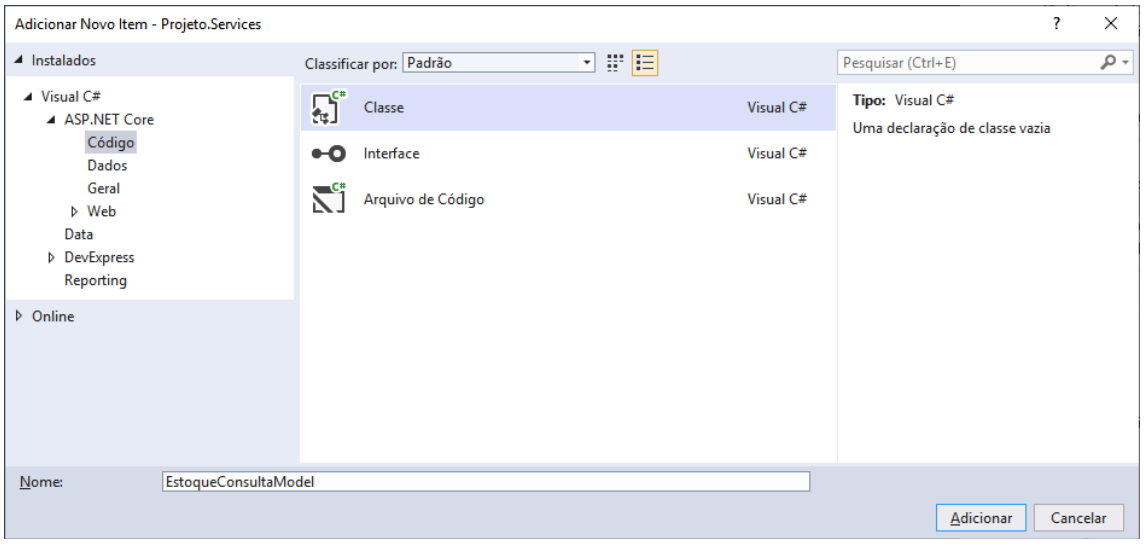
```
                else
                {
                    return BadRequest("Produto não encontrado.");
                }
            }
            catch(Exception e)
            {
                return StatusCode(500, "Erro: " + e.Message);
            }
        }

        [HttpGet]
        public IActionResult GetAll()
        {
            try
            {
                var result = produtoRepository.Consultar();
                return Ok(result);
            }
            catch(Exception e)
            {
                return StatusCode(500, "Erro: " + e.Message);
            }
        }

        [HttpGet("{id}")]
        public IActionResult GetById(int id)
        {
            try
            {
                var result = produtoRepository.ObterPorId(id);

                if(result != null)
                {
                    return Ok(result);
                }
                else
                {
                    return NoContent();
                }
            }
            catch (Exception e)
            {
                return StatusCode(500, "Erro: " + e.Message);
            }
        }
    }
}
```

Criando classes de Modelo para retornar dados de consultas:
Neste caso, iremos criar 2 classes novas:

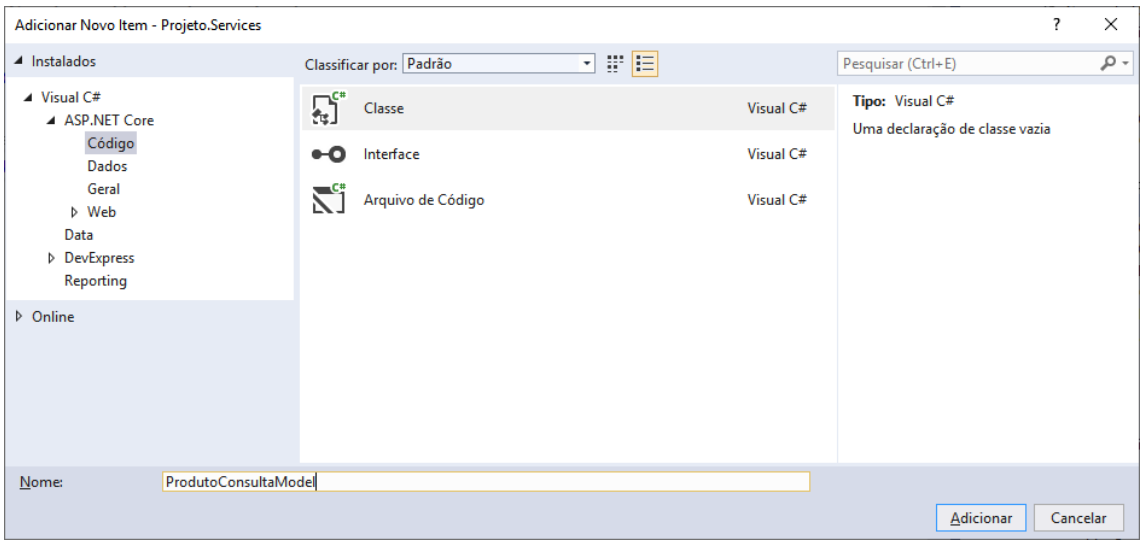
- EstoqueConsultaModel
- ProdutoConsultaModel

/Models/EstoqueConsultaModel.cs

Dados que serão retornados na consulta de Estoques da API.



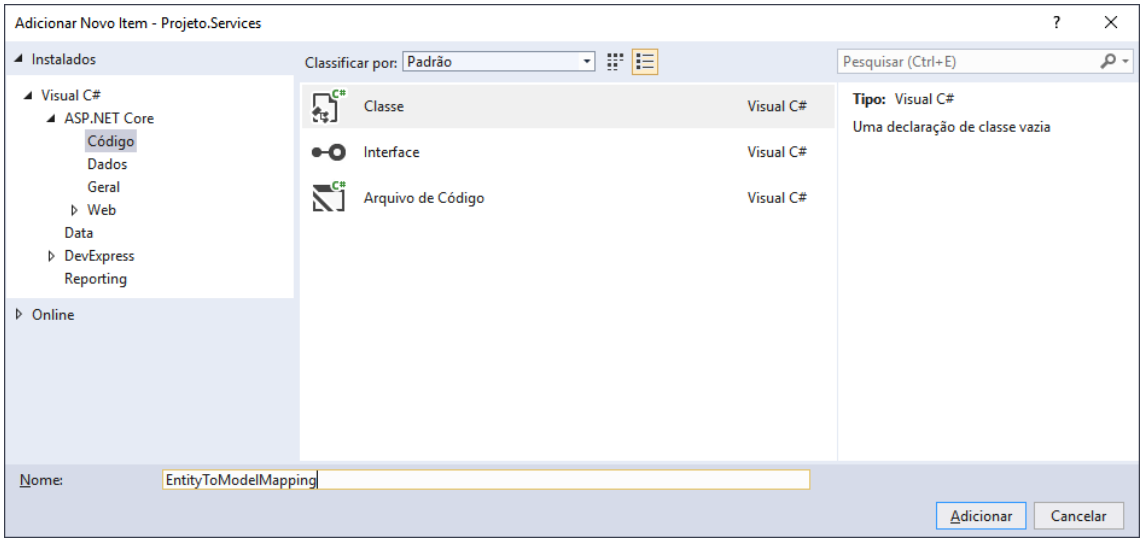

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace Projeto.Services.Models
{
    public class EstoqueConsultaModel
    {
        public int IdEstoque { get; set; }
        public string Nome { get; set; }
        public int QuantidadeProdutos { get; set; }
    }
}
```




```
using System;
using System.Collections.Generic;
using System.Linq;
```

```
using System.Threading.Tasks;

namespace Projeto.Services.Models
{
    public class ProdutoConsultaModel
    {
        public int IdProduto { get; set; }
        public string Nome { get; set; }
        public decimal Preco { get; set; }
        public int Quantidade { get; set; }
        public decimal Total { get; set; }
        public EstoqueConsultaModel Estoque { get; set; }
    }
}
```

Criando uma classe de mapeamento do **AutoMapper** para mapear a transferência de dados das entidades para as models de consulta.




```
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
{
    public class EntityToModelMapping : Profile
    {
        //ctor + 2x[tab] --> construtor default
        public EntityToModelMapping()
        {
            CreateMap<Estoque, EstoqueConsultaModel>()
                .AfterMap((src, dest)
                  => dest.QuantidadeProdutos = src.Produtos.Count);
```

```
            CreateMap<Produto, ProdutoConsultaModel>()
                .AfterMap((src, dest)
                  => dest.Total = (src.Preco * src.Quantidade));
        }
    }
}
```

Refatorando os métodos de consulta nas classes de controle:

/Controllers/EstoqueController.cs

```
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 EstoqueController : ControllerBase
    {
        //atributo
        private readonly IEstoqueRepository estoqueRepository;
        private readonly IMapper mapper;

        //construtor para injeção de dependência
        public EstoqueController(IEstoqueRepository estoqueRepository,
                                 IMapper mapper)
        {
            this.estoqueRepository = estoqueRepository;
            this.mapper = mapper;
        }

        [HttpPost]
        public IActionResult Post(EstoqueCadastroModel model)
        {
            //verificando se os campos da model passaram nas validações
            if(ModelState.IsValid)
            {
                try
                {
                    var estoque = mapper.Map<Estoque>(model);
                    estoqueRepository.Inserir(estoque);

                    var result = new
                    {
                        message = "Estoque cadastrado com sucesso",
                        estoque
                    };
```

```
                return Ok(result); //HTTP 200 (SUCESSO!)
            }
            catch(Exception e)
            {
                return StatusCode(500, "Erro: " + e.Message);
            }
        }
        else
        {
            //Erro HTTP 400 (BAD REQUEST)
            return BadRequest("Ocorreram erros de validação.");
        }
    }

    [HttpPut]
    public IActionResult Put(EstoqueEdicaoModel model)
    {
        //verificando se os campos da model passaram nas validações
        if (ModelState.IsValid)
        {
            try
            {
                var estoque = mapper.Map<Estoque>(model);
                estoqueRepository.Alterar(estoque);

                var result = new
                {
                    message = "Estoque atualizado com sucesso",
                    estoque
                };

                return Ok(result); //HTTP 200 (SUCESSO!)
            }
            catch (Exception e)
            {
                return StatusCode(500, "Erro: " + e.Message);
            }
        }
        else
        {
            //Erro HTTP 400 (BAD REQUEST)
            return BadRequest("Ocorreram erros de validação.");
        }
    }

    [HttpDelete("{id}")]
    public IActionResult Delete(int id)
    {
        try
        {
            //buscar o estoque referente ao id informado..
            var estoque = estoqueRepository.ObterPorId(id);

            //verificar se o estoque foi encontrado..
            if(estoque != null)
            {
                //excluindo o estoque
                estoqueRepository.Excluir(estoque);

                var result = new
                {
```

```
                        message = "Estoque excluído com sucesso.",
                        estoque
                    };

                    return Ok(result);
                }
                else
                {
                    return BadRequest("Estoque não encontrado.");
                }
            }
            catch(Exception e)
            {
                return StatusCode(500, "Erro: " + e.Message);
            }
        }

        [HttpGet]
        public IActionResult GetAll()
        {
            try
            {
               var result = mapper.Map<List<EstoqueConsultaModel>>
                      (estoqueRepository.Consultar());

                return Ok(result);
            }
            catch(Exception e)
            {
                return StatusCode(500, "Erro: " + e.Message);
            }
        }

        [HttpGet("{id}")]
        public IActionResult GetById(int id)
        {
            try
            {
                var result = mapper.Map<EstoqueConsultaModel>
                      (estoqueRepository.ObterPorId(id));

                if(result != null) //se o estoque foi encontrado..
                {
                    return Ok(result);
                }
                else
                {
                    return NoContent(); //HTTP 204 (SUCESSO -> Vazio)
                }
            }
            catch (Exception e)
            {
                return StatusCode(500, "Erro: " + e.Message);
            }
        }
    }
}
```

/Controllers/ProdutoController.cs

```
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 ProdutoController : ControllerBase
    {
        //atributo
        private readonly IProdutoRepository produtoRepository;
        private readonly IMapper mapper;

        //construtor para injeção de dependência
        public ProdutoController(IProdutoRepository produtoRepository,
                                 IMapper mapper)
        {
            this.produtoRepository = produtoRepository;
            this.mapper = mapper;
        }

        [HttpPost]
        public IActionResult Post(ProdutoCadastroModel model)
        {
            if(ModelState.IsValid)
            {
                try
                {
                    var produto = mapper.Map<Produto>(model);
                    produtoRepository.Inserir(produto);

                    var result = new
                    {
                        message = "Produto cadastrado com sucesso.",
                        produto
                    };

                    return Ok(result);
                }
                catch(Exception e)
                {
                    return StatusCode(500, "Erro: " + e.Message);
                }
            }
            else
            {
                return BadRequest("Ocorreram erros de validação.");
            }
        }
```

```
[HttpPut]
public IActionResult Put(ProdutoEdicaoModel model)
{
    if (ModelState.IsValid)
    {
        try
        {
            var produto = mapper.Map<Produto>(model);
            produtoRepository.Alterar(produto);

            var result = new
            {
                message = "Produto atualizado com sucesso.",
                produto
            };

            return Ok(result);
        }
        catch (Exception e)
        {
            return StatusCode(500, "Erro: " + e.Message);
        }
    }
    else
    {
        return BadRequest("Ocorreram erros de validação.");
    }
}

[HttpDelete("{id}")]
public IActionResult Delete(int id)
{
    try
    {
        var produto = produtoRepository.ObterPorId(id);

        if(produto != null)
        {
            produtoRepository.Excluir(produto);

            var result = new
            {
                message = "Produto excluído com sucesso.",
                produto
            };

            return Ok(result);
        }
        else
        {
            return BadRequest("Produto não encontrado.");
        }
    }
    catch(Exception e)
    {
        return StatusCode(500, "Erro: " + e.Message);
    }
}
```

```
        [HttpGet]
        public IActionResult GetAll()
        {
            try
            {
               var result = mapper.Map<List<ProdutoConsultaModel>>
                        (produtoRepository.Consultar());

                return Ok(result);
            }
            catch(Exception e)
            {
                return StatusCode(500, "Erro: " + e.Message);
            }
        }

        [HttpGet("{id}")]
        public IActionResult GetById(int id)
        {
            try
            {
               var result = mapper.Map<ProdutoConsultaModel>
                        (produtoRepository.ObterPorId(id));

                if(result != null)
                {
                    return Ok(result);
                }
                else
                {
                    return NoContent();
                }
            }
            catch (Exception e)
            {
                return StatusCode(500, "Erro: " + e.Message);
            }
        }
    }
}
```

Testando:

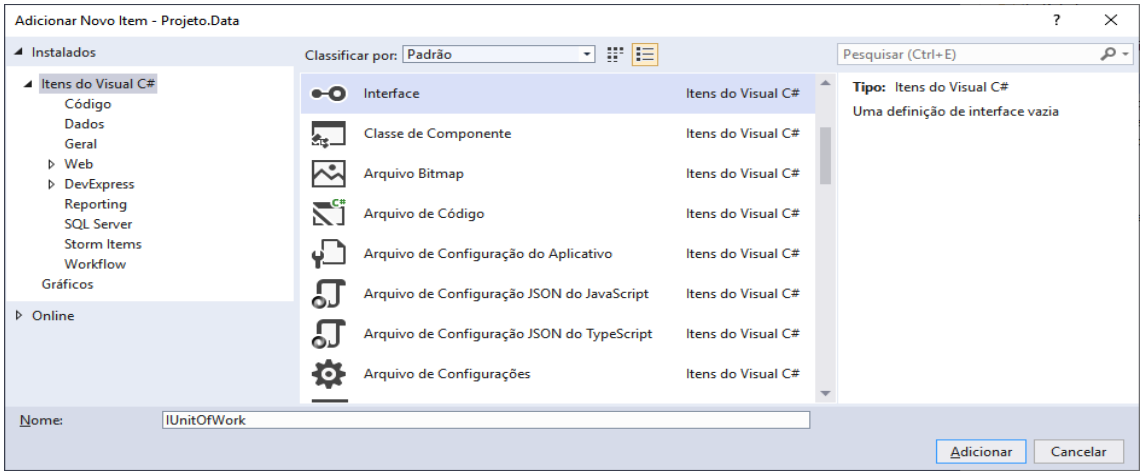
Consulta de produtos:


```
[
  {
    "idProduto": 2,
    "nome": "Mouse",
    "preco": 50,
    "quantidade": 15,
    "total": 750,
    "estoque": {
      "idEstoque": 2,
      "nome": "Informática",
      "quantidadeProdutos": 3
    }
  },
  {
    "idProduto": 1003,
    "nome": "Mouse",
    "preco": 30,
    "quantidade": 10,
    "total": 300,
    "estoque": {
      "idEstoque": 2,
      "nome": "Informática",
      "quantidadeProdutos": 3
    }
  },
  {
    "idProduto": 1004,
    "nome": "Computador",
    "preco": 3000,
    "quantidade": 15,
    "total": 45000,
    "estoque": {
      "idEstoque": 2,
      "nome": "Informática",
      "quantidadeProdutos": 3
    }
  }
```

UnitOfWork

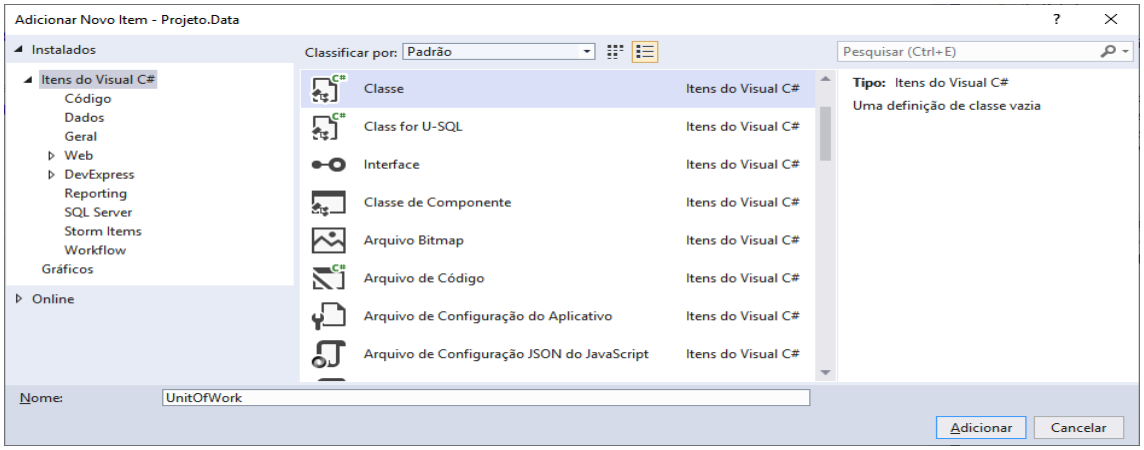
Padrão utilizado em projetos que trabalham com ORM para acesso a banco de dados. A ideia o UnitOfWork é criar uma classe onde possamos executar qualquer operação envolvendo os repositórios por meio de gerenciamento de **transações** (**Commit** e **Rollback**)

Primeiro, iremos criar uma interface chamada IUnitOfWork




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

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

Implementando a interface:



```
using Projeto.Data.Contracts;
using System;
using System.Collections.Generic;
using System.Text;

namespace Projeto.Data.Repositories
{
    public class UnitOfWork : IUnitOfWork
    {
        public void BeginTransaction()
        {
            throw new NotImplementedException();
        }

        public void Commit()
        {
            throw new NotImplementedException();
        }

        public void Rollback()
        {
            throw new NotImplementedException();
        }
    }
}
```

Continua...