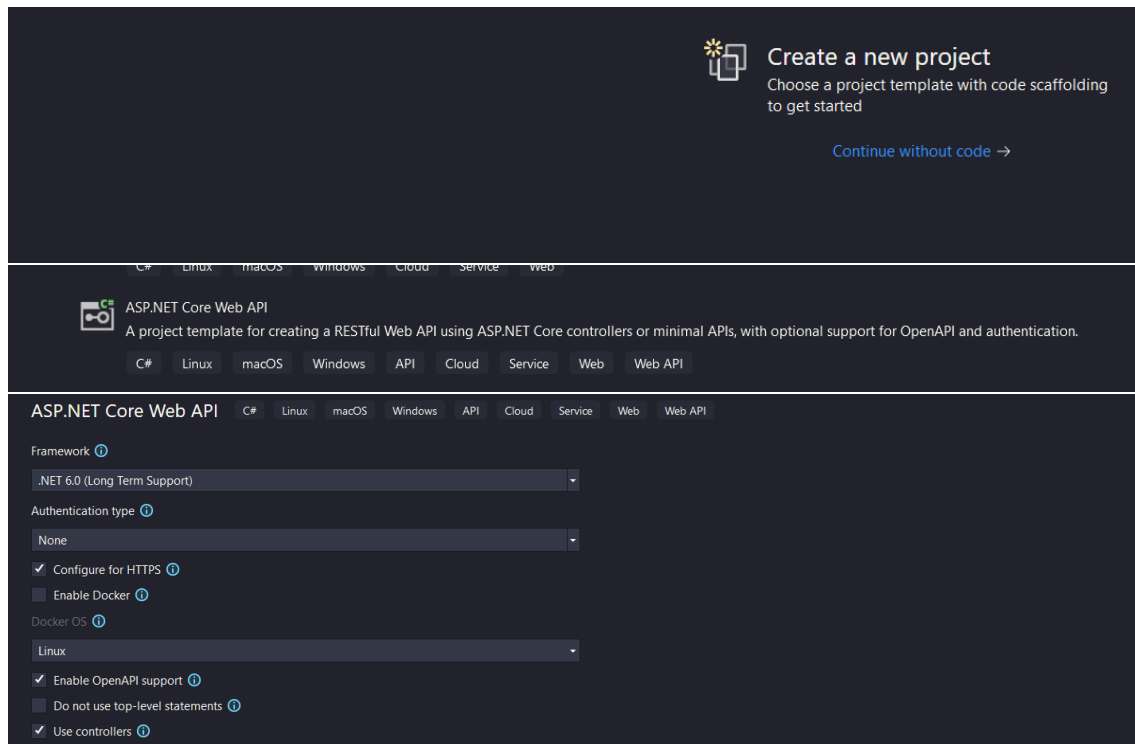
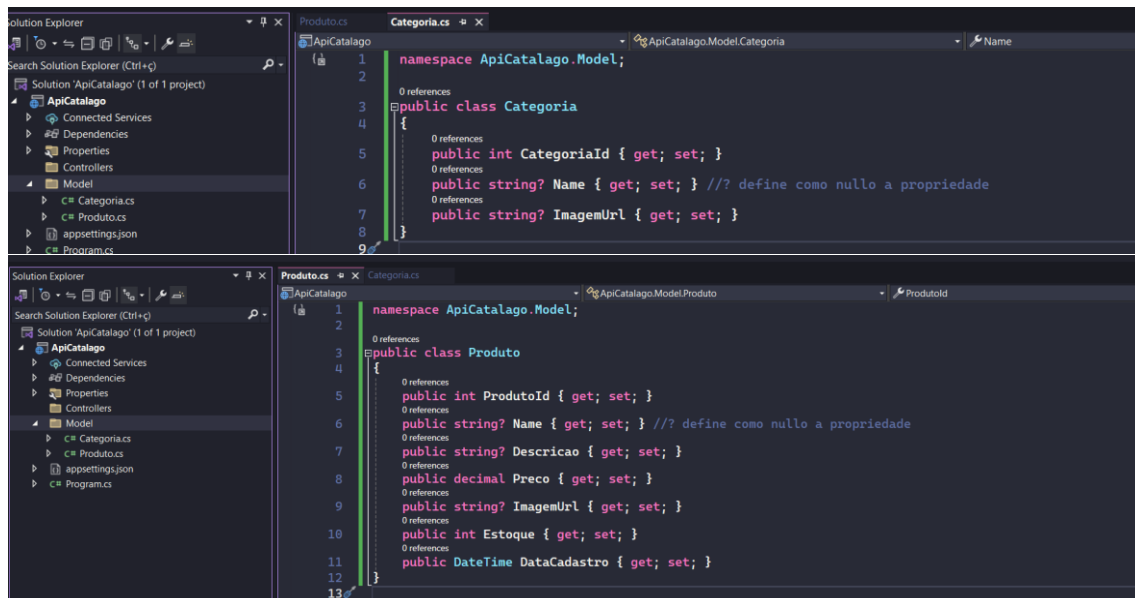


Módulo Criando uma api no visual studio

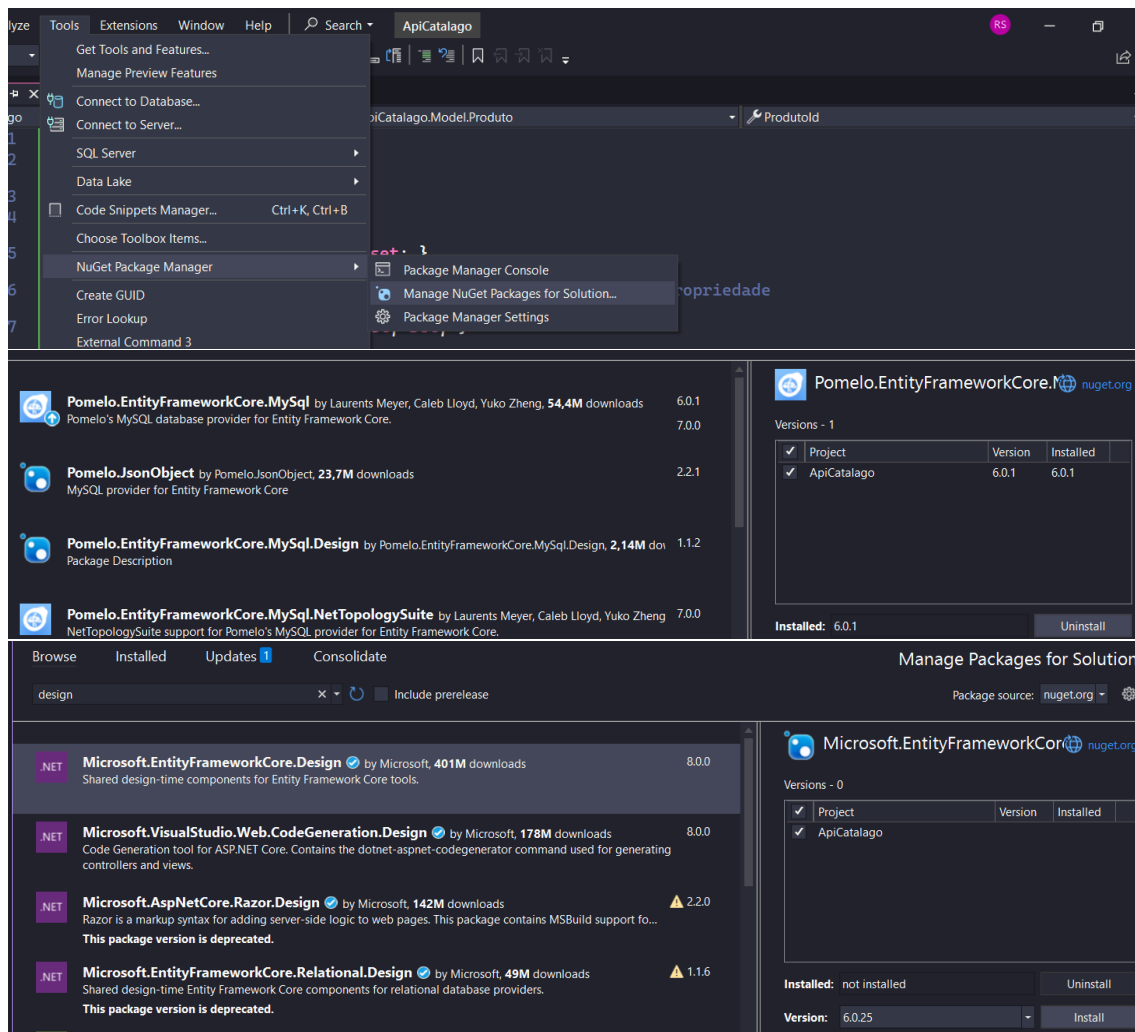
#01 Criar Projeto web api no visual



#02 criando as entidades do modelo de domínio, cria uma pasta model no projeto e define o modelo de domínio

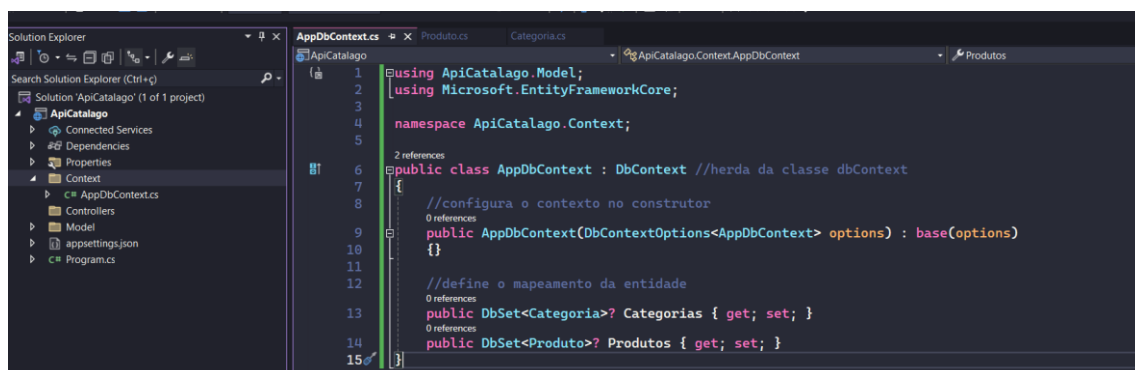


#03 Configurando a webapi, baixar dependências, instala o (pomelo, Microsoft design, tool)

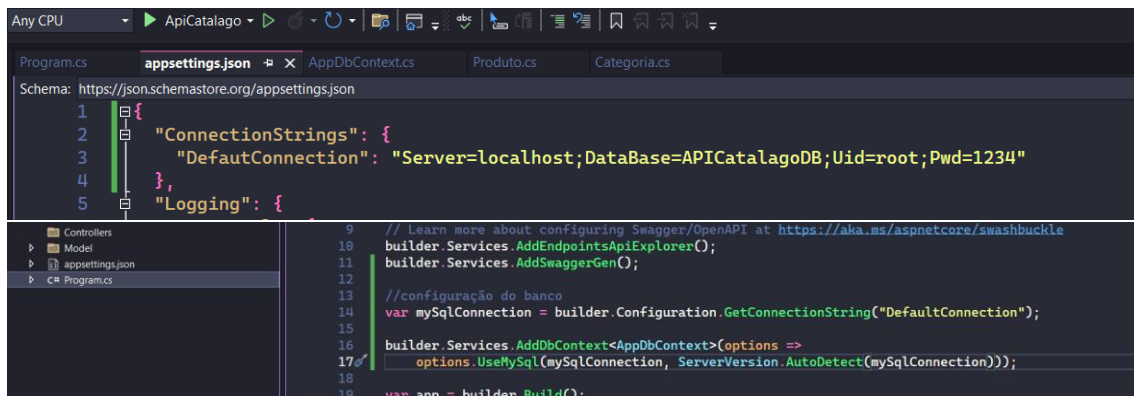


Instalar o dotnet ef tool na linha de comando “dotnet tool install -global dotnet-ef” e fazer update “dotnet tool update -global dotnet-ef”

#04 criando o arquivo de contexto do ef core, cria uma pasta Context e o arquivo AppDbContext.



#05 registrando os serviços na class program, configurando a string de conexão com o banco de dados



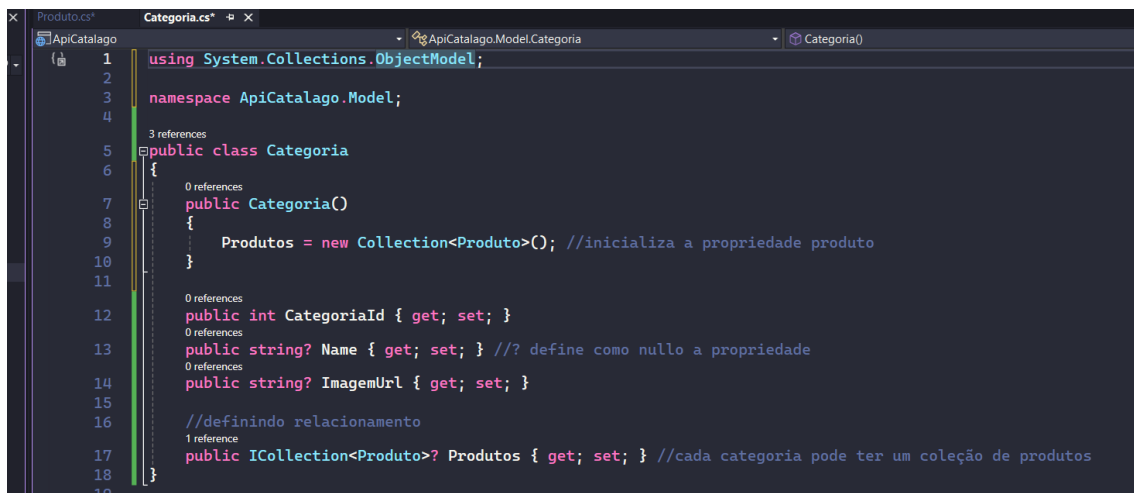
The screenshot shows the Visual Studio IDE with the 'appsettings.json' file open. The JSON content is as follows:

```
{
  "ConnectionStrings": {
    "DefaultConnection": "Server=localhost;DataBase=APICatalagoDB;Uid=root;Pwd=1234"
  },
  "Logging": {
    "LogLevel": {
      "Default": "Information"
    }
  }
}
```

Below the JSON file, the 'Program.cs' file is open, showing the following code:

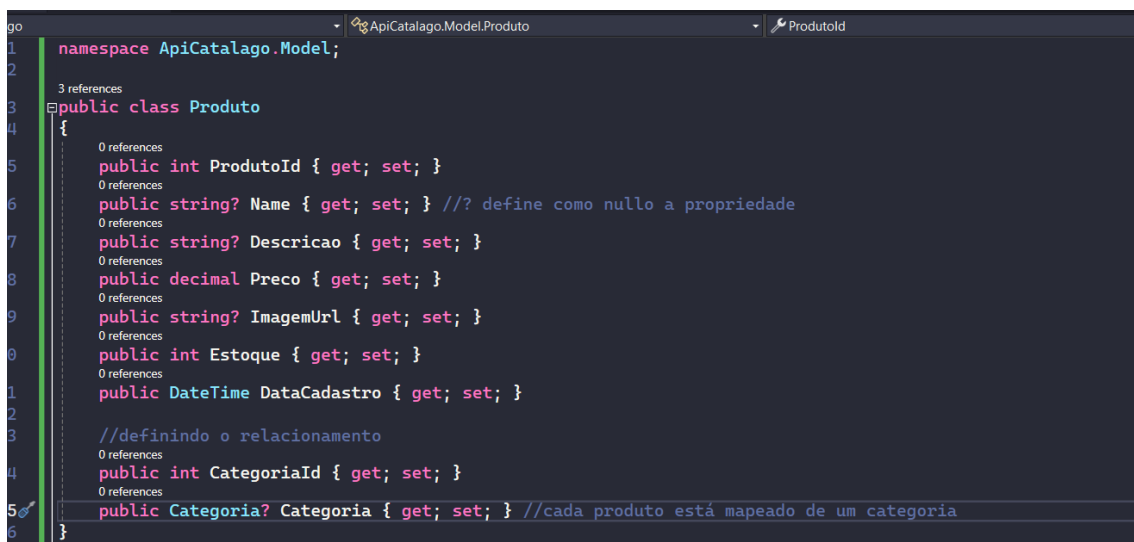
```
1 // Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
2 builder.Services.AddEndpointsApiExplorer();
3 builder.Services.AddSwaggerGen();
4
5 //configuração do banco
6 var mySqlConnection = builder.Configuration.GetConnectionString("DefaultConnection");
7
8 builder.Services.AddDbContext<AppDbContext>(options =>
9 {
10     options.UseMySQL(mySqlConnection, ServerVersion.AutoDetect(mySqlConnection));
11 });
12
13 var app = builder.Build();
```

#06 definindo o relacionamento entre as entidades de domínio



The screenshot shows the 'Categoria.cs' file in the 'ApiCatalago.Model' namespace. The code defines the 'Categoria' class with the following properties and methods:

```
1 using System.Collections.ObjectModel;
2
3 namespace ApiCatalago.Model;
4
5 public class Categoria
6 {
7     public Categoria()
8     {
9         Produtos = new Collection<Produto>(); //inicializa a propriedade produto
10     }
11
12     public int CategoriaId { get; set; }
13     public string? Name { get; set; } //? define como nullo a propriedade
14     public string? ImagemUrl { get; set; }
15
16     //definindo relacionamento
17     public ICollection<Produto>? Produtos { get; set; } //cada categoria pode ter um coleção de produtos
18 }
19
```



The screenshot shows the 'Produto.cs' file in the 'ApiCatalago.Model' namespace. The code defines the 'Produto' class with the following properties and methods:

```
1 namespace ApiCatalago.Model;
2
3 public class Produto
4 {
5     public int ProdutoId { get; set; }
6     public string? Name { get; set; } //? define como nullo a propriedade
7     public string? Descricao { get; set; }
8     public decimal Preco { get; set; }
9     public string? ImagemUrl { get; set; }
10     public int Estoque { get; set; }
11     public DateTime DataCadastro { get; set; }
12
13     //definindo o relacionamento
14     public int CategoriaId { get; set; }
15
16     public Categoria? Categoria { get; set; } //cada produto está mapeado de um categoria
17 }
18
```

#07 aplicando o migrations para gerar o banco e as tabelas, comando

Criar a migrations: "dotnet ef migrations add nome"

Deleta a migrations: "dotnet ef migrations remove nome"

Gerar a tabela no banco: "dotnet ef database update"

#08 aplicando data annotations às propriedades das entidades

```

[Key]
0 references
public int CategoriaId { get; set; }

[Required]
[StringLength(100, ErrorMessage = "Informe o nome.")] //tamanho da string e texto de error
0 references
public string? Name { get; set; } //? define como nullo a propriedade

[Required]
[StringLength(300, ErrorMessage = "Informe a imagem.")]
0 references
public string? ImagemUrl { get; set; }

```

```

[Key]
0 references
public int ProdutoId { get; set; }

[Required] //obrigatório
[StringLength(100, ErrorMessage = "Informe o nome.")]
0 references
public string? Name { get; set; } //? define como nullo a propriedade

[Required]
[StringLength(300, ErrorMessage = "Informe a descrição.")]
0 references
public string? Descricao { get; set; }

[Required]
[Column(TypeName = "decimal(10,2)")]
0 references
public decimal Preco { get; set; }

[Required]
[StringLength(300, ErrorMessage = "Informe a imagem.")]
0 references
public string? ImagemUrl { get; set; }

[Required]
0 references
public int Estoque { get; set; }

```

Depois roda o comando “dotnet ef add nome” e depois “dotnet ef database update”

#09 populando as tabelas do banco de dados mysql faz a migração “dotnet ef migrations add nome”

```

0
1 reference
7 public partial class PopulandoATabelaCategoria : Migration
8 {
9     0 references
10     protected override void Up(MigrationBuilder mb)
11     {
12         mb.Sql("Insert into Categorias(Nome, ImagemUrl) Values('Lanches', 'lanches.jpg')");
13         mb.Sql("Insert into Categorias(Nome, ImagemUrl) Values('Sobremesas', 'sobremesas.jpg')");
14     }
15     0 references
16     protected override void Down(MigrationBuilder mb)
17     {
18         mb.Sql("Delete from Categorias");
19     }
20 }

1 reference
public partial class PopulandoATabelaProdutos : Migration
{
    0 references
    protected override void Up(MigrationBuilder mb)
    {
        mb.Sql("Insert into Produtos(Name,Descricao,Preco,ImagemUrl,Estoque,DataCadastro,CategoriaId) " +
            "Values('Coca-Cola', 'Refrigerante 2 litros',8.99,'bebidas.jpg',100,now(),1)");
        mb.Sql("Insert into Produtos(Name,Descricao,Preco,ImagemUrl,Estoque,DataCadastro,CategoriaId) " +
            "Values('Pudim', 'Pudim 250 gramas',18.89,'pudim.jpg',10,now(),3)");
    }
    0 references
    protected override void Down(MigrationBuilder mb)
    {
        mb.Sql("Delete from Produtos");
    }
}

```

#10 criando o controlador produtoscontroller

```
Properties
  Context
  Controllers
    c= ProdutosController.cs
  Migrations
  Model
  appsettings.json
  Program.cs

7 // rota /produtos
8 [Route("[controller]")]
9 [ApiController]
10 public class ProdutosController : ControllerBase
11 {
12     private readonly ApplicationDbContext _context;
13
14     public ProdutosController(ApplicationDbContext context) // instância o contexto
15     {
16         _context = context;
17     }
18 }
```

#11 produto controller GET todos produtos

```
[HttpGet]
0 references
public ActionResult<IEnumerable<Produto>> Get() // o actionresult permite ter varios tipo de resposta como notfound ok...
{
    var produtos = _context.Produtos.ToList();

    if (produtos is null)
    {
        return NotFound();
    }

    return produtos;
}
```

#12 produto controller GET pelo id

```
// /produtos/id
[HttpGet("{id:int}")]
0 references
public ActionResult<Produto> Get(int id)
{
    var produto = _context.Produtos.FirstOrDefault(p => p.ProdutoId == id);

    if (produto is null)
    {
        return NotFound("Produto não encontrado.");
    }

    return produto;
}
```

#13 produto controller POST criar um produto

```
[HttpPost]
0 references
public ActionResult Post(Produto produto)
{
    if (produto is null)
    {
        return BadRequest("Dados inválido.");
    }

    _context.Produtos.Add(produto); // chama o contexto a tabela produtos e adiciona
    _context.SaveChanges(); // persiste os dados na DB

    // chama a rota e com o produto criado
    return new CreatedAtRouteResult("obter-produto", new { id = produto.ProdutoId }, produto);
}

return produtos;
}

// /produtos/id
[HttpGet("{id:int}", Name= "obter-produto")] // rota nomeada /produtos/obter-produto
0 references
public ActionResult<Produto> Get(int id)
{
    var produto = _context.Produtos.FirstOrDefault(p => p.ProdutoId == id);
}
```

#14 produto controller PUT atualiza o produto

```
[HttpPut("{id:int}")]
0 references
public ActionResult Put(int id, Produto produto)
{
    if (id != produto.ProdutoId)
    {
        return BadRequest("Produto inválido.");
    }

    // usa o entry e define o estado para modificado
    _context.Entry(produto).State = EntityState.Modified;
    _context.SaveChanges();

    return Ok(produto);
}
```

#15 produto controller DELETE deleta o produto

```
[HttpDelete("{id:int}")]
0 references
public ActionResult Delete(int id)
{
    // busca no banco o id
    var produto = _context.Produtos.FirstOrDefault(p => p.ProdutoId == id);

    if (produto is null)
    {
        return NotFound("Produto não encontrado.");
    }

    _context.Produtos.Remove(produto);
    _context.SaveChanges();

    return Ok(produto);
}
```

#16 criando o controller de categorias

```
6
7 [Route("[controller]")]
8 [ApiController]
9 public class CategoriasController : ControllerBase
10 {
11     public readonly ApplicationDbContext _context;
12
13     0 references
14     public CategoriasController(ApplicationDbContext context)
15     {
16         _context = context;
17     }
18
19     [HttpGet]
20     0 references
21     public ActionResult<IEnumerable<Categoria>> Get()
22     {
23         return _context.Categorias.ToList();
24     }
25
26     [HttpGet("{id:int}", Name = "obter-categoria")]
27     0 references
28     public ActionResult<Categoria> Get(int id)
29     {
30         var categoria = _context.Categorias.FirstOrDefault(c => c.CategoriaId == id);
31
32         if (categoria is null)
33         {
34             return NotFound("Categoria não encontrada.");
35         }
36
37         return categoria;
38     }
39 }
```

```
[HttpPost]
0 references
public ActionResult Post(Categoria categoria)
{
    if (categoria is null)
    {
        return BadRequest("Dados inválido.");
    }

    _context.Categorias.Add(categoria);
    _context.SaveChanges();

    return new CreatedAtRouteResult("obter-categoria", new { id = categoria.CategoriaId }, categoria);
}
```

```

[HttpPut("{id:int}")]
0 references
public ActionResult Put(int id, Categoria categoria)
{
    if (id != categoria.CategoriaId)
    {
        return BadRequest("Categoria inválida.");
    }

    _context.Entry(categoria).State = EntityState.Modified;
    _context.SaveChanges();

    return Ok();
}

[HttpDelete("{id:int}")]
0 references
public ActionResult<Categoria> Delete(int id)
{
    var categoria = _context.Categorias.FirstOrDefault(c => c.CategoriaId == id);

    if (categoria is null)
    {
        return NotFound("Categoria não encontrada.");
    }

    _context.Categorias.Remove(categoria);
    _context.SaveChanges();

    return Ok(categoria);
}

```

```

[HttpGet("produtos")] // /categorias/produtos
0 references
public ActionResult<IEnumerable<Categoria>> GetCategoriasEProdutos()
{
    return _context.Categorias.Include(p => p.Produtos).ToList(); // incluido os produtos relacionado
}

```

Cai no error 500 erro de serialização de JSON

#17 Serialização e Desserialização JSON para corrigir vá na classe program

```

7 // Add services to the container.
8 // arrumando a serialização JSON
9 builder.Services.AddControllers().AddJsonOptions(options =>
10     options.JsonSerializerOptions.ReferenceHandler = ReferenceHandler.IgnoreCycles);
11

```

A serialização das propriedades pública, serializar é trazer os produtos relacionado junto no retorno igual aí

```

{
  "categoriaId": 0,
  "name": "string",
  "imageUrl": "string",
  "produtos": [
    {
      "produtoId": 0,
      "name": "string",
      "descricao": "string",
      "preco": 0,
      "imageUrl": "string",
      "estoque": 0,
      "dataCadastro": "2024-01-08T21:34:53.277Z",
      "categoriaId": 0,
      "categoria": "string"
    }
  ]
}

```

Assim ele ignora individual

```

35
36 [Required]
37 [JsonIgnore]
38 public Categoria? Categoria { get; set; } //cada produto está mapeado de um categoria
39

```

#18 API otimizando as consultas GET as entidade fica em cache no rastreador e ela perde desempenho

```

    }

    [HttpGet]
    0 references
    public ActionResult<IEnumerable<Categoria>> Get()
    {
        return _context.Categorias.AsNoTracking().ToList(); // não rastreia
    }

    [HttpGet("produtos")] // /categorias/produtos
    0 references
    public ActionResult<IEnumerable<Categoria>> GetCategoriasEProdutos()
    {
        //return _context.Categorias.Include(p => p.Produtos).ToList(); // incluído os produtos relacionado
        return _context.Categorias.Include(p => p.Produtos).Where(c => c.CategoriaId <= 5).ToList();
    }
}

```

#19 fazendo o tratamento de error com bloco try-catch

```

[HttpGet("{id:int}", Name = "obter-categoria")]
0 references
public ActionResult<Categoria> Get(int id)
{
    try
    {
        var categoria = _context.Categorias.FirstOrDefault(c => c.CategoriaId == id);

        if (categoria is null)
        {
            return NotFound("Categoria não encontrada.");
        }

        return categoria;
    }
    catch (Exception)
    {
        return StatusCode(StatusCodes.Status500InternalServerError, "Error no servidor.");
    }
}

```

Módulo Fundamento

#1

#2

#3

#4

#5

#6

#7

#8

#9

#10

Módulo Repositorio

#01 padrão Repositório



Definição de **Martin Fowler** :

"O padrão **Repository** faz a mediação entre o domínio e as camadas de mapeamento de dados, agindo como uma coleção de **objetos de domínio em memória....**"

#02 Repositório implementação

```
using System.Linq.Expressions;

namespace ApiCatalago.Repository;

public interface IRepository<T> // o tipo T é qualquer tipo
{
    IQueryable<T> Get(); // método para consultar
    T GetById(Expression<Func<T, bool>> predicate); // consultar por id
    void Add (T entity); // adicionar
    void Update (T entity);
    void Delete (T entity);
}

using ApiCatalago.Context;
using Microsoft.EntityFrameworkCore;
using System.Linq.Expressions;

namespace ApiCatalago.Repository;

public class Repository<T> : IRepository<T> where T : class // classe genérica, o só poder ser classe
{
    protected AppDbContext _context;

    public Repository(AppDbContext context) // injeta o contexto
    {
        _context = context;
    }

    public IQueryable<T> Get() // lista de entidade
    {
        // o set do contexto retorna uma instância do dbset para acesso a entidades de determinado tipo no contexto
        return _context.Set<T>().AsNoTracking();
    }

    public T GetById(Expression<Func<T, bool>> predicate)
    {
        return _context.Set<T>().SingleOrDefault(predicate);
    }

    public void Add(T entity)
    {
        _context.Set<T>().Add(entity);
    }
}
```

IQueryable faz o filtro no banco de dados e manda só os dados filtrado.

IEnumerable manda todos os dados, e faz o filtro na memória.

```
1 reference
public void Update(T entity)
{
    // define o estado como modificado, e informe ao contexto que ele foi modificado
    _context.Entry(entity).State = EntityState.Modified;
    _context.Set<T>().Update(entity);
}

1 reference
public void Delete(T entity)
{
    _context.Set<T>().Remove(entity);
}
```

Definindo os repositórios do produto e da categoria

```
ApiCatalogo
1 using ApiCatalogo.Model;
2
3 namespace ApiCatalogo.Repository;
4
5 1 reference
6 public interface IProdutoRepository : IRepository<Produto> // herda a classe padrão e define o tipo do produto
7 {
8     1 reference
9     IEnumerable<Produto> GetProdutosPeloPreco();
}
```

```
ProdutoRepository.cs
1 using ApiCatalogo.Context;
2 using ApiCatalogo.Model;
3
4 namespace ApiCatalogo.Repository;
5
6 // herda da classe repository por que assim tem acesso a toda implementação que foi feita na classe
7 1 reference
8 public class ProdutoRepository : Repository<Produto>, IProdutoRepository
9 {
10     0 references
11     public ProdutoRepository(AppDbContext contexto) : base(contexto) // receba o dbcontexto e passe para classe base
12     {}
13
14     1 reference
15     public IEnumerable<Produto> GetProdutosPeloPreco()
16     {
17         return Get().OrderBy(c => c.Preco).ToList(); //ordena pelo preço
18     }
19 }
```

```
ICategoriaRepository.cs
1 using ApiCatalogo.Model;
2
3 namespace ApiCatalogo.Repository;
4
5 0 references
6 public interface ICategoriaRepository : IRepository<Categoria>
7 {
8     0 references
9     IEnumerable<Categoria> GetCategoriasProdutos();
10 }
```

```
CategoriaRepository.cs
1 using ApiCatalogo.Context;
2 using ApiCatalogo.Model;
3 using Microsoft.EntityFrameworkCore;
4
5 namespace ApiCatalogo.Repository;
6
7 1 reference
8 public class CategoriaRepository : Repository<Categoria>, ICategoriaRepository
9 {
10     0 references
11     public CategoriaRepository(AppDbContext contexto) : base(contexto)
12     {}
13
14     1 reference
15     public IEnumerable<Categoria> GetCategoriasProdutos()
16     {
17         return Get().Include(x => x.Produtos);
18     }
19 }
```

agora para usar esse repositório você precisa implementar o padrão unity of work

#03 padrão unit of work, não tem sentido o repositório salvar dados na memória, isso é papel do unit of work

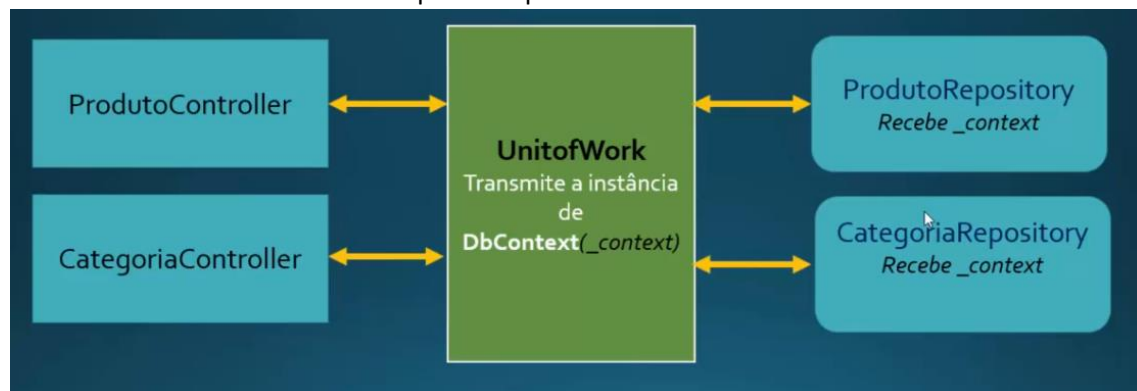
"Uma **unidade de trabalho** rastreia tudo o que é feito durante a transação de negócios que pode afetar a **camada de persistência**.
Ao concluir, ela descobre tudo o que precisa ser feito para alterar a camada de persistência como resultado do trabalho.



O **Unit of Work** é um padrão de projeto que ajuda a gerenciar transações em bancos de dados em aplicações **ASP.NET Core Web API**. Ele garante que todas as alterações feitas em vários objetos na aplicação sejam confirmadas no banco de dados ou revertidas, mantendo a consistência dos dados e a precisão das modificações ¹.

Em outras palavras, o **Unit of Work** é uma camada de abstração que gerencia as transações do banco de dados e garante que todas as operações sejam executadas com sucesso ou falhem juntas, mantendo a integridade dos dados ¹.

#04 implementando o unit of work, uma camada entre o controller e o repositório que recebe a instância do contexto e transmite para o repositório.



```

1 namespace ApiCatalogo.Repository;
2
3 public interface IUnitOfWork
4 {
5     // implementa as 2 classes
6     IProdutoRepository ProdutoRepository { get; }
7     ICategoriaRepository CategoriaRepository { get; }
8     // método para salvar
9     void Commit();
10 }
11

```

```

1 using ApiCatalogo.Context;
2
3 namespace ApiCatalogo.Repository;
4
5 public class UnitOfWork : IUnitOfWork
6 {
7     private ProdutoRepository _produtoRepository;
8     private CategoriaRepository _categoriaRepository;
9     private ApplicationDbContext _context;
10
11     public UnitOfWork(ApplicationDbContext context) // injeta o dbcontext
12     {
13         _context = context;
14     }
15
16     public IProdutoRepository ProdutoRepository
17     {
18         get { return _produtoRepository ?? new ProdutoRepository(_context); }
19     }
20
21     public ICategoriaRepository CategoriaRepository
22     {
23         get { return _categoriaRepository ?? new CategoriaRepository(_context); }
24     }
25
26     public void Commit() // salva
27     {
28         _context.SaveChanges();
29     }
30
31     public void Dispose() // libera os recursos usado
32     {
33         _context.Dispose();
34     }
35 }
36

```

#05 registrar serviço no net 6

```

20 builder.Services.AddDbContext<AppDbContext>(options =>
21     options.UseMySQL(mysqlConnection, ServerVersion.AutoDetect(mysqlConnection)));
22
23 // registrando o serviço do unit of work
24 builder.Services.AddScoped<IUnitOfWork, UnitOfWork>();
25
26 var app = builder.Build();
27

```

#06 ajustando os controller com unit of work

```

7 // rota /produtos
8 [Route("[controller]")]
9 [ApiController]
10
11 public class ProdutosController : ControllerBase
12 {
13     private readonly IUnitOfWork _uow; // implementa o unit of work
14
15     public ProdutosController(IUnitOfWork context) // instancia o contexto
16     {
17         _uow = context;
18     }
19
20     [HttpGet]
21     public ActionResult<IEnumerable<Produto>> Get() // o actionresult permite ter varios tipo de resposta como notfound ok...
22     {
23         try
24         {
25             var produtos = _uow.ProdutoRepository.Get().ToList();
26
27             if (produtos is null)
28             {
29                 return NotFound();
30             }
31
32             return produtos;
33         }
34         catch (Exception)
35         {
36             return StatusCode(StatusCode.Status500InternalServerError, "Error no servidor.");
37         }
38     }
39 }

```

```
// /produtos/id
[HttpGet("{id:int}", Name= "obter-produto")] // rota nomeada /produtos/obter-produto
0 references
public ActionResult<Produto> Get(int id)
{
    try
    {
        var produto = _uow.ProdutoRepository.GetById(p => p.ProdutoId == id);

        if (produto is null)
        {
            return NotFound("Produto não encontrado.");
        }

        return produto;
    }
    catch (Exception)
    {
        return StatusCode(StatusCodes.Status500InternalServerError, "Error no servidor.");
    }
}
```

```
[HttpPost]
0 references
public ActionResult Post(Produto produto)
{
    try
    {
        if (produto is null)
        {
            return BadRequest("Dados inválido.");
        }

        _uow.ProdutoRepository.Add(produto); // chama o contexto a tabela produtos e adiciona
        _uow.Commit(); // persiste os dados na DB

        // chama a rota e com o produto criado
        return new CreatedAtRouteResult("obter-produto",
            new { id = produto.ProdutoId }, produto);
    }
    catch (Exception)
    {
        return StatusCode(StatusCodes.Status500InternalServerError, "Error no servidor.");
    }
}
```

```

[HttpPut("{id:int}")]
0 references
public ActionResult Put(int id, Produto produto)
{
    try
    {
        if (id != produto.ProdutoId)
        {
            return BadRequest("Produto inválido.");
        }

        // usa o entry e define o estado para modificado
        _uow.ProdutoRepository.Update(produto);
        _uow.Commit();

        return Ok(produto);
    }
    catch (Exception)
    {
        return StatusCode(StatusCodes.Status500InternalServerError, "Error no servidor.");
    }
}

```

```

[HttpDelete("{id:int}")]
0 references
public ActionResult Delete(int id)
{
    try
    {
        // busca no banco o id
        var produto = _uow.ProdutoRepository.GetById(p => p.ProdutoId == id);

        if (produto is null)
        {
            return NotFound("Produto não encontrado.");
        }

        _uow.ProdutoRepository.Delete(produto);
        _uow.Commit();

        return Ok(produto);
    }
    catch (Exception)
    {
        return StatusCode(StatusCodes.Status500InternalServerError, "Error no servidor.");
    }
}

```

```

[HttpGet("menor-preco")]
0 references
public ActionResult<IEnumerable<Produto>> GetProdutosPeloPreco()
{
    try
    {
        return _uow.ProdutoRepository.GetProdutosPeloPreco().ToList();
    }
    catch (Exception)
    {
        return StatusCode(StatusCodes.Status500InternalServerError, "Error no servidor.");
    }
}

```

da categoria a mesma coisa

```

8 // rota /produtos
9 [Route("[controller]")]
10 [ApiController]
11 public class CategoriasController : ControllerBase
12 {
13     private readonly IUnitOfWork _uow;
14
15     0 references
16     public CategoriasController(IUnitOfWork context) // instância o contexto
17     {
18         _uow = context;
19     }
20 }

```



```
[HttpGet]
0 references
public ActionResult<IEnumerable<Categoria>> Get() // o actionresult permite ter varios tipo de reposta como notfound ok...
{
    try
    {
        var categorias = _uow.CategoriaRepository.Get().ToList();

        if (categorias is null)
        {
            return NotFound();
        }

        return categorias;
    }
    catch (Exception)
    {
        return StatusCode(StatusCodes.Status500InternalServerError, "Error no servidor.");
    }
}
```

```
[HttpGet("produtos")] // /categorias/produtos
0 references
public ActionResult<IEnumerable<Categoria>> GetCategoriasEProdutos()
{
    try
    {
        return _uow.CategoriaRepository.GetCategoriasProdutos().ToList(); // inclui os produtos relacionado
    }
    catch (Exception)
    {
        return StatusCode(StatusCodes.Status500InternalServerError, "Error no servidor.");
    }
}
```

```
// /produtos/id
[HttpGet("{id:int}", Name = "obter-produto")] // rota nomeada /produtos/obter-produto
0 references
public ActionResult<Categoria> Get(int id)
{
    try
    {
        var categoria = _uow.CategoriaRepository.GetById(p => p.CategoriaId == id);

        if (categoria is null)
        {
            return NotFound("Categoria não encontrado.");
        }

        return categoria;
    }
    catch (Exception)
    {
        return StatusCode(StatusCodes.Status500InternalServerError, "Error no servidor.");
    }
}
```

```
[HttpPost]
0 references
public ActionResult Post(Categoria categoria)
{
    try
    {
        if (categoria is null)
        {
            return BadRequest("Dados inválido.");
        }

        _uow.CategoriaRepository.Add(categoria); // chama o contexto a tabela produtos e adiciona
        _uow.Commit(); // persiste os dado na DB

        // chama a rota e com o produto criado
        return new CreatedAtRouteResult("obter-produto",
            new { id = categoria.CategoriaId }, categoria);
    }
    catch (Exception)
    {
        return StatusCode(StatusCodes.Status500InternalServerError, "Error no servidor.");
    }
}
```

```

[HttpPut("{id:int}")]
0 references
public ActionResult Put(int id, Categoria categoria)
{
    try
    {
        if (id != categoria.CategoriaId)
        {
            return BadRequest("Produto inválido.");
        }

        // usa o entry e define o estado para modificado
        _uow.CategoriaRepository.Update(categoria);
        _uow.Commit();

        return Ok(categoria);
    }
    catch (Exception)
    {
        return StatusCode(StatusCodes.Status500InternalServerError, "Error no servidor.");
    }
}

```

```

[HttpDelete("{id:int}")]
0 references
public ActionResult Delete(int id)
{
    try
    {
        // busca no banco o id
        var categoria = _uow.CategoriaRepository.GetById(c => c.CategoriaId == id);

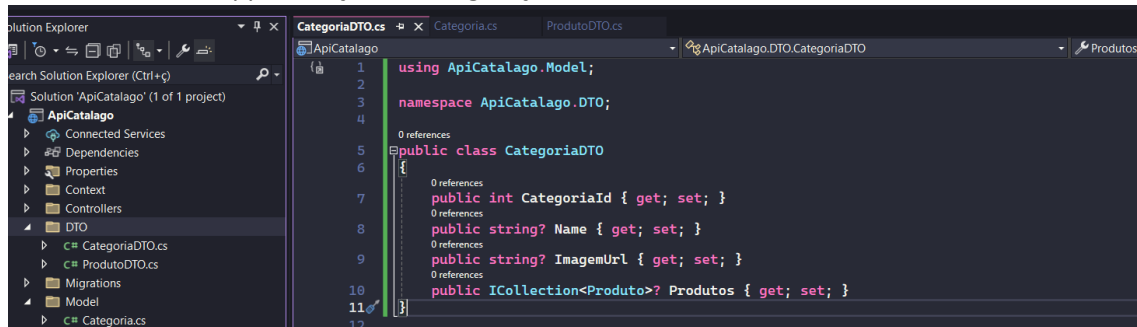
        if (categoria is null)
        {
            return NotFound("Produto não encontrado.");
        }

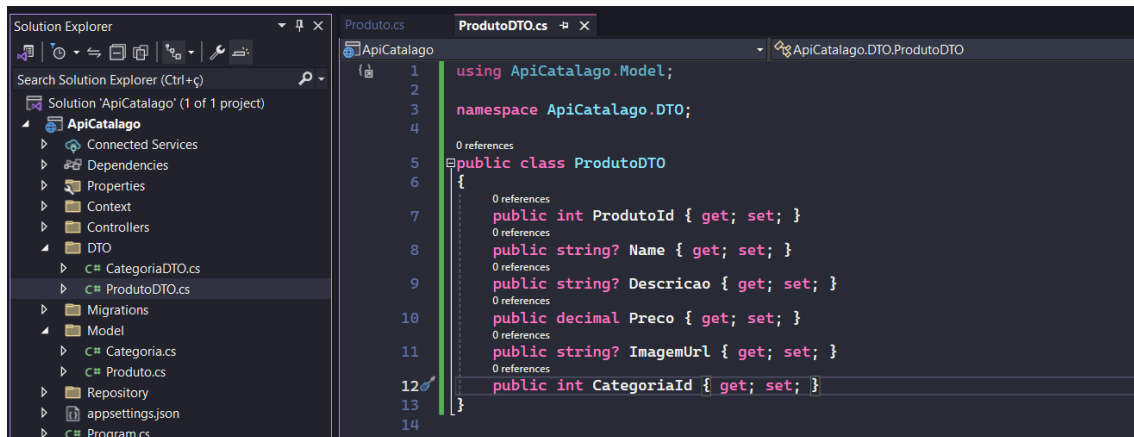
        _uow.CategoriaRepository.Delete(categoria);
        _uow.Commit();

        return Ok(categoria);
    }
    catch (Exception)
    {
        return StatusCode(StatusCodes.Status500InternalServerError, "Error no servidor.");
    }
}

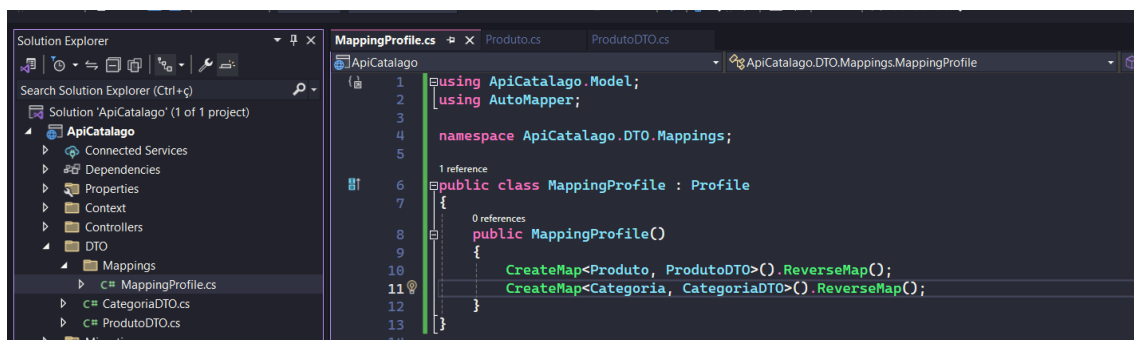
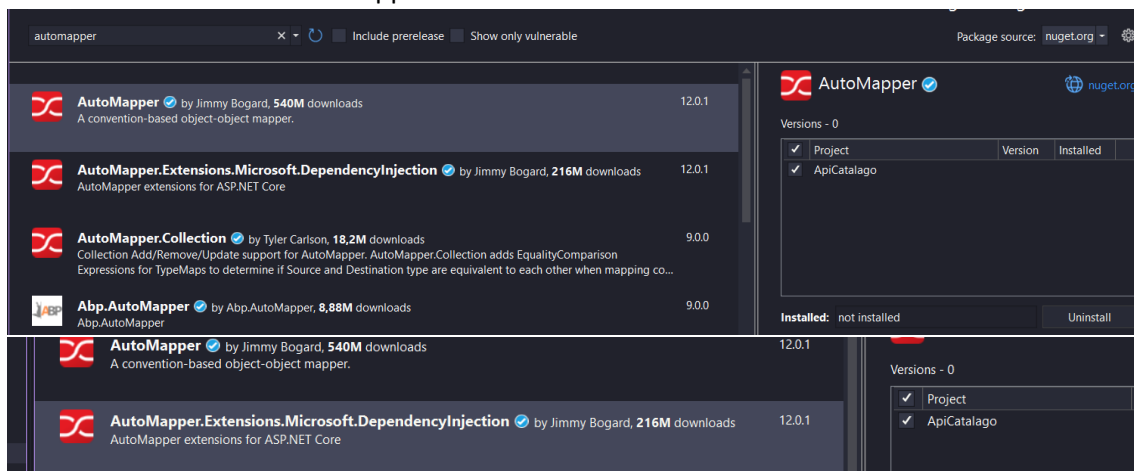
```

#07 DTO e automapper criação e configuração

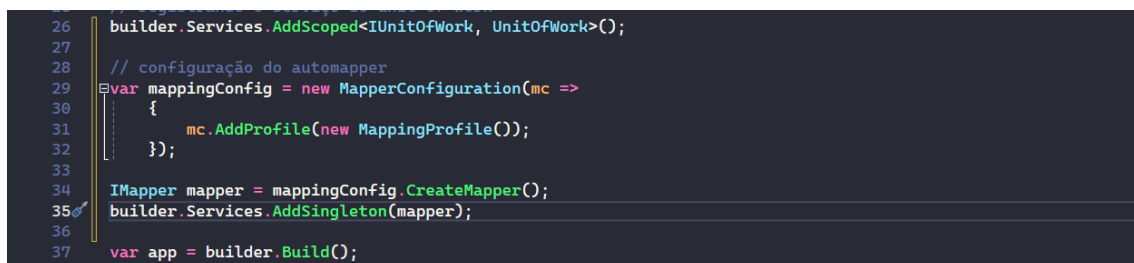




Incluir as biblioteca do automapper



#08 registrar o serviço do automapper net 6



#09 controllers ajustes e mapeamento via automapper

```
ApiCatalogo
  Controllers
    ProdutosController
      ProdutosController(IUnitOfWork context, IMapper mapper)
        namespace ApiCatalogo.Controllers;
        // rota /produtos
        [Route("[controller]*")]
        [ApiController]
        public class ProdutosController : ControllerBase
        {
            private readonly IUnitOfWork _uow; // implementa o unit of work
            private readonly IMapper _mapper; // implementa o mapper

            public ProdutosController(IUnitOfWork context, IMapper mapper) // instancia o contexto
            {
                _uow = context;
                _mapper = mapper;
            }
        }
    
```

```
[HttpGet("menor-preco")]
0 references
public IActionResult IEnumerable<ProdutoDTO> GetProdutosPeloPreco()
{
    try
    {
        var produtos = _uow.ProdutoRepository.GetProdutosPeloPreco().ToList();
        var produtosDTO = _mapper.Map<List<ProdutoDTO>>(produtos); // mapea os produtosdto

        return produtosDTO;
    }
    catch (Exception)
    {
        return StatusCode(StatusCodes.Status500InternalServerError, "Error no servidor.");
    }
}

```

```
[HttpGet]
0 references
public IActionResult IEnumerable<ProdutoDTO> Get() // o IActionResult permite ter varios tipo de resposta como notfound ok...
{
    try
    {
        var produtos = _uow.ProdutoRepository.Get().ToList();
        var produtosDTO = _mapper.Map<List<ProdutoDTO>>(produtos); // mapea os produtosdto

        return produtosDTO;
    }
    catch (Exception)
    {
        return StatusCode(StatusCodes.Status500InternalServerError, "Error no servidor.");
    }
}

```

```
// /produtos/id
[HttpGet("{id:int}", Name = "obter-produto")] // rota nomeada /produtos/obter-produto
0 references
public IActionResult ProdutoDTO Get(int id)
{
    try
    {
        var produto = _uow.ProdutoRepository.GetById(p => p.ProdutoId == id);

        if (produto is null)
        {
            return NotFound("Produto não encontrado.");
        }

        var produtoDTO = _mapper.Map<ProdutoDTO>(produto);

        return produtoDTO;
    }
    catch (Exception)
    {
        return StatusCode(StatusCodes.Status500InternalServerError, "Error no servidor.");
    }
}

```

```

[HttpPost]
0 references
public ActionResult Post(ProdutoDTO produtoDto)
{
    try
    {
        if (produtoDto is null)
        {
            return BadRequest("Dados inválido.");
        }

        var produto = _mapper.Map<Produto>(produtoDto); // mapea para produto

        _uow.ProdutoRepository.Add(produto); // chama o contexto a tabela produtos e adiciona
        _uow.Commit(); // persiste os dados na DB

        var produtoDTO = _mapper.Map<ProdutoDTO>(produto);

        // chama a rota e com o produto
        return new CreatedAtRouteResult(
            new { id = produtoDTO.ProdutoId }, produtoDTO);
    }
    catch (Exception)
    {
        return StatusCode(StatusCode.Status500InternalServerError, "Error no servidor.");
    }
}

```

```

[HttpPut("{id:int}")]
0 references
public ActionResult Put(int id, ProdutoDTO produtoDto)
{
    try
    {
        if (id != produtoDto.ProdutoId)
        {
            return BadRequest("Produto inválido.");
        }

        var produto = _mapper.Map<Produto>(produtoDto);

        // usa o entry e define o estado para modificado
        _uow.ProdutoRepository.Update(produto);
        _uow.Commit();

        return Ok(produto);
    }
    catch (Exception)
    {
        return StatusCode(StatusCode.Status500InternalServerError, "Error no servidor.");
    }
}

```

```

[HttpDelete("{id:int}")]
0 references
public ActionResult<ProdutoDTO> Delete(int id)
{
    try
    {
        // busca no banco o id
        var produto = _uow.ProdutoRepository.GetById(p => p.ProdutoId == id);

        if (produto is null)
        {
            return NotFound("Produto não encontrado.");
        }

        _uow.ProdutoRepository.Delete(produto);
        _uow.Commit();

        var produtoDTO = _mapper.Map<ProdutoDTO>(produto);

        return Ok(produtoDTO);
    }
    catch (Exception)
    {
        return StatusCode(StatusCode.Status500InternalServerError, "Error no servidor.");
    }
}

```

```

10
11 // rota /produtos
12 [Route("[controller]")]
13 [ApiController]
14 public class CategoriasController : ControllerBase
15 {
16     private readonly IUnitOfWork _uow;
17     private readonly IMapper _mapper;
18
19     0 references
20     public CategoriasController(IUnitOfWork context, IMapper mapper) // instância o contexto
21     {
22         _uow = context;
23         _mapper = mapper;
24     }
25
26 }

```

```

27
28 [HttpGet("produtos")] // /categorias/produtos
29 0 references
30 public ActionResult<IEnumerable<CategoriaDTO>> GetCategoriasEProdutos()
31 {
32     try
33     {
34         var categorias = _uow.CategoriaRepository.GetCategoriasProdutos().ToList(); // inclui os produtos relacionado
35         var categoriasDTO = _mapper.Map<List<CategoriaDTO>>(categorias);
36
37         return categoriasDTO;
38     }
39     catch (Exception)
40     {
41         return StatusCode(StatusCodes.Status500InternalServerError, "Error no servidor.");
42     }
43 }
44
45 }

```

```

46
47 [HttpGet]
48 0 references
49 public ActionResult<IEnumerable<CategoriaDTO>> Get() // o actionresult permite ter varios tipo de resposta como notfound o
50 {
51     try
52     {
53         var categorias = _uow.CategoriaRepository.Get().ToList();
54         var categoriasDTO = _mapper.Map<List<CategoriaDTO>>(categorias);
55
56         return categoriasDTO;
57     }
58     catch (Exception)
59     {
60         return StatusCode(StatusCodes.Status500InternalServerError, "Error no servidor.");
61     }
62 }
63
64 }

```

```

65 // /produtos/id
66 [HttpGet("{id:int}", Name = "obter-categoria")] // rota nomeada /produtos/obter-produto
67 0 references
68 public ActionResult<CategoriaDTO> Get(int id)
69 {
70     try
71     {
72         var categoria = _uow.CategoriaRepository.GetById(p => p.CategoriaId == id);
73
74         if (categoria is null)
75         {
76             return NotFound("Categoria não encontrado.");
77         }
78
79         var categoriaDTO = _mapper.Map<CategoriaDTO>(categoria);
80
81         return categoriaDTO;
82     }
83     catch (Exception)
84     {
85         return StatusCode(StatusCodes.Status500InternalServerError, "Error no servidor.");
86     }
87 }
88
89 }

```

```

[HttpPost]
0 references
public ActionResult Post(CategoriaDTO categoriaDto)
{
    try
    {
        if (categoriaDto is null)
        {
            return BadRequest("Dados inválido.");
        }

        var categoria = _mapper.Map<Categoria>(categoriaDto);

        _uow.CategoriaRepository.Add(categoria); // chama o contexto a tabela produtos e adiciona
        _uow.Commit(); // persiste os dado na DB

        var categoriaDTO = _mapper.Map<CategoriaDTO>(categoria);

        // chama a rota e com o produto criado
        return new CreatedAtRouteResult("obter-categoria",
            new { id = categoriaDTO.CategoriaId }, categoriaDTO);
    }
    catch (Exception)
    {
        return StatusCode(StatusCodes.Status500InternalServerError, "Error no servidor.");
    }
}

```

```

[HttpPut("{id:int}")]
0 references
public ActionResult Put(int id, CategoriaDTO categoriaDto)
{
    try
    {
        if (id != categoriaDto.CategoriaId)
        {
            return BadRequest("Produto inválido.");
        }

        var categoria = _mapper.Map<Categoria>(categoriaDto);

        // usa o entry e define o estado para modificado
        _uow.CategoriaRepository.Update(categoria);
        _uow.Commit();

        return Ok(categoria);
    }
    catch (Exception)
    {
        return StatusCode(StatusCodes.Status500InternalServerError, "Error no servidor.");
    }
}

```

```

[HttpDelete("{id:int}")]
0 references
public ActionResult<CategoriaDTO> Delete(int id)
{
    try
    {
        // busca no banco o id
        var categoria = _uow.CategoriaRepository.GetById(c => c.CategoriaId == id);

        if (categoria is null)
        {
            return NotFound("Produto não encontrado.");
        }

        _uow.CategoriaRepository.Delete(categoria);
        _uow.Commit();

        var categoriaDTO = _mapper.Map<CategoriaDTO>(categoria);

        return Ok(categoriaDTO);
    }
    catch (Exception)
    {
        return StatusCode(StatusCodes.Status500InternalServerError, "Error no servidor.");
    }
}

```

Módulo Paginação de dados na API

Módulo Programação assíncrona repositório e paginação

