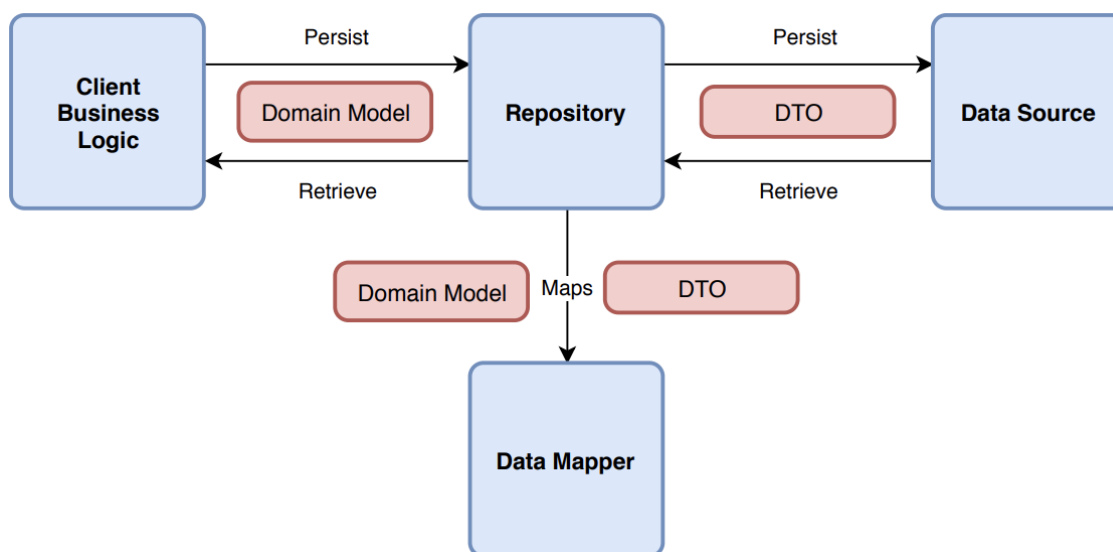


REPOSITORY PATTERN

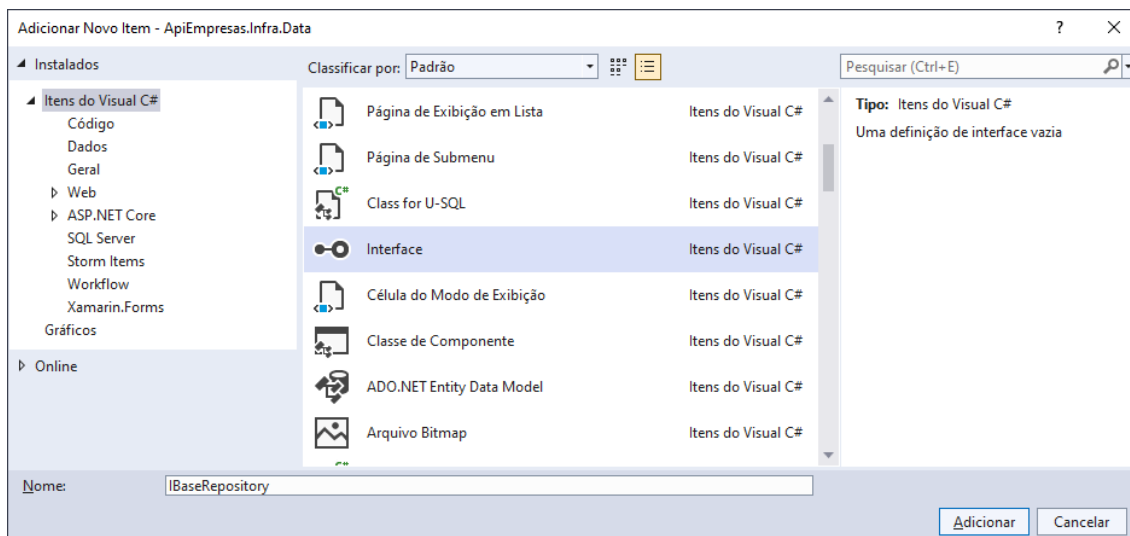
Criando as classes de repositório do projeto **Infra.Data**

Ao utilizar o padrão Repository você pode realizar a persistência e a separação de interesses em seu código de acesso a dados visto que ele encapsula a lógica necessária para persistir os objetos do seu domínio na sua fonte de armazenamento de dados. Em suma, você pode usar o padrão Repository para desacoplar o modelo de domínio do código de acesso a dados.



Primeiro passo:

Criando as interfaces do repositório.

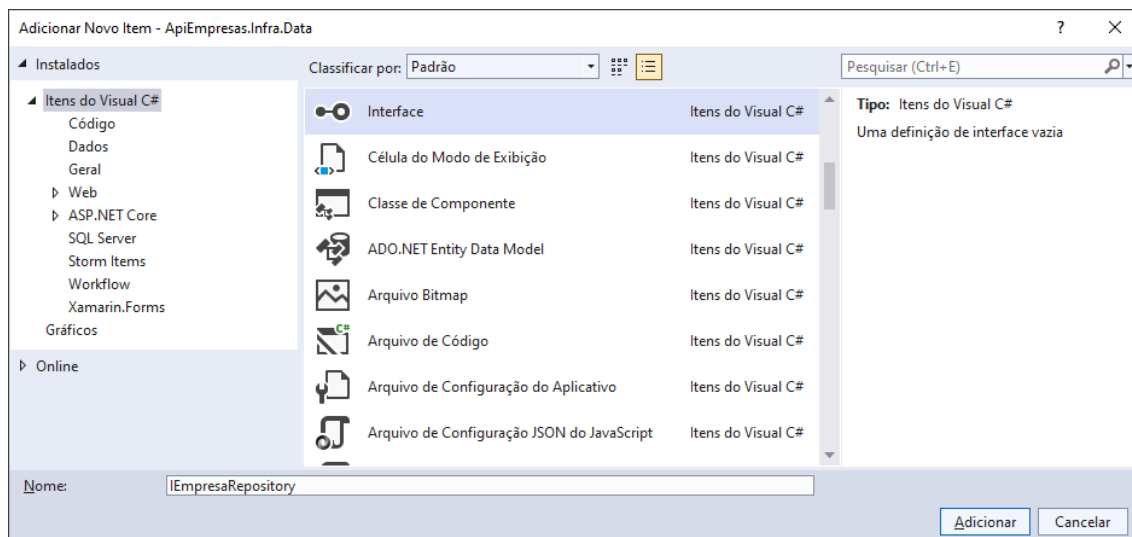


```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
namespace ApiEmpresas.Infra.Data.Interfaces
{
    /// <summary>
    /// Interface genérica para operações de repositório.
    /// </summary>
    /// <typeparam name="TEntity"></typeparam>
    public interface IBaseRepository<TEntity>
        where TEntity : class
    {
        void Inserir(TEntity entity);
        void Alterar(TEntity entity);
        void Excluir(TEntity entity);

        List<TEntity> Consultar();
        TEntity ObterPorId(Guid id);
    }
}
```

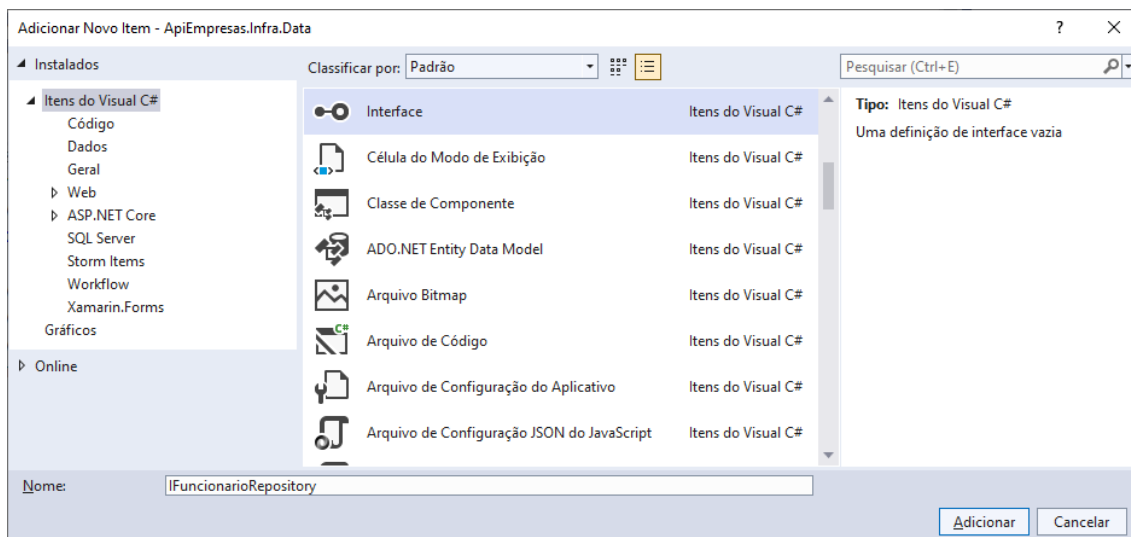
Criando as demais interfaces:



```
using ApiEmpresas.Infra.Data.Entities;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ApiEmpresas.Infra.Data.Interfaces
{
    /// <summary>
    /// Interface de repositório para operações de empresa
    /// </summary>
```

```
public interface IEmpresaRepository : IRepository<Empresa>
{
    /// <summary>
    /// Método para consultar 1 Empresa através do Cnpj
    /// </summary>
    Empresa ObterPorCnpj(string cnj);
}
```



```
using ApiEmpresas.Infra.Data.Entities;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ApiEmpresas.Infra.Data.Interfaces
{
    /// <summary>
    /// Interface de repositório para operações de funcionário
    /// </summary>
    public interface IFuncionarioRepository : IRepository<Funcionario>
    {
        /// <summary>
        /// Método para retornar 1 funcionário baseado no CPF
        /// </summary>
        Funcionario ObterPorCpf(string cpf);

        /// <summary>
        /// Método para retornar 1 funcionário baseado na Matrícula
        /// </summary>
        Funcionario ObterPorMatricula(string matricula);

        /// <summary>
        /// Método para retornar funcionários baseado no nome
        /// </summary>
        List<Funcionario> ObterPorNome(string nome);
    }
}
```

/Repositories/**EmpresaRepository.cs**

Implementando as interfaces:

```
using ApiEmpresas.Infra.Data.Entities;
using ApiEmpresas.Infra.Data.Interfaces;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ApiEmpresas.Infra.Data.Repositories
{
    public class EmpresaRepository : IEmpresaRepository
    {
        public void Inserir(Empresa entity)
        {
            throw new NotImplementedException();
        }

        public void Alterar(Empresa entity)
        {
            throw new NotImplementedException();
        }

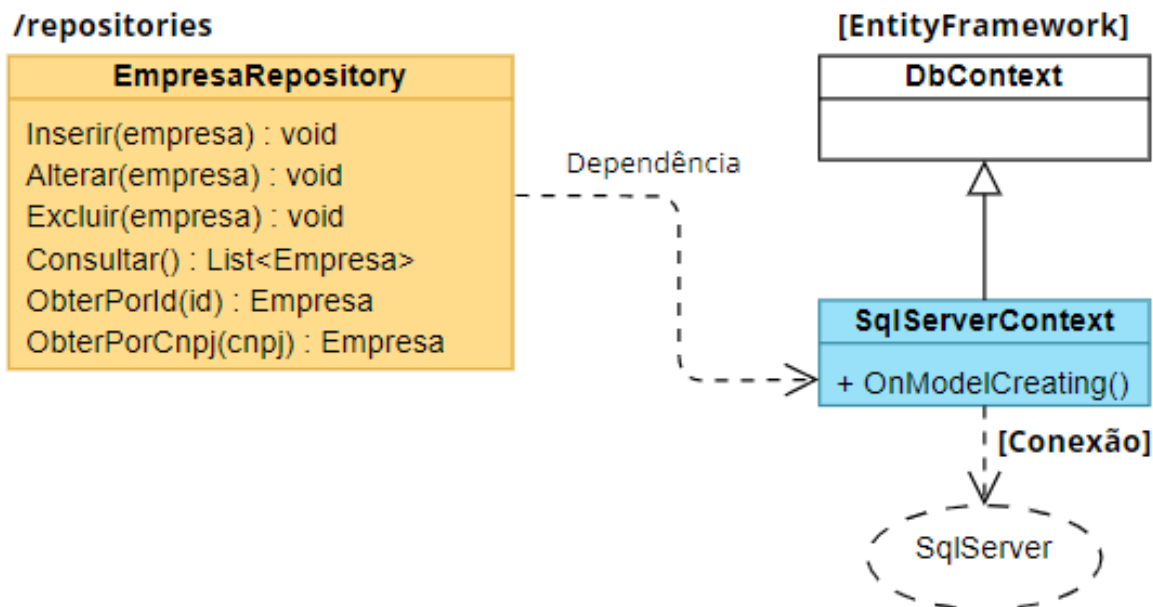
        public void Excluir(Empresa entity)
        {
            throw new NotImplementedException();
        }

        public List<Empresa> Consultar()
        {
            throw new NotImplementedException();
        }

        public Empresa ObterPorId(Guid id)
        {
            throw new NotImplementedException();
        }

        public Empresa ObterPorCnpj(string cnpj)
        {
            throw new NotImplementedException();
        }
    }
}
```

Cada repositório implementado em EntityFramework deverá ter uma dependência da classe **SqlServerContext** (DbContext)



Criando a dependência:

```
using ApiEmpresas.Infra.Data.Contexts;
using ApiEmpresas.Infra.Data.Entities;
using ApiEmpresas.Infra.Data.Interfaces;
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ApiEmpresas.Infra.Data.Repositories
{
    public class EmpresaRepository : IEmpresaRepository
    {
        //atributo
        private readonly SqlServerContext _context;

        //construtor para injeção de dependência
        public EmpresaRepository(SqlServerContext context)
        {
            _context = context;
        }

        (...)
    }
}
```

Implementando o repositório:

```
using ApiEmpresas.Infra.Data.Contexts;
using ApiEmpresas.Infra.Data.Entities;
using ApiEmpresas.Infra.Data.Interfaces;
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ApiEmpresas.Infra.Data.Repositories
{
    public class EmpresaRepository : IEmpresaRepository
    {
        private readonly SqlServerContext _context;

        public EmpresaRepository(SqlServerContext context)
        {
            _context = context;
        }

        public void Inserir(Empresa entity)
        {
            _context.Entry(entity).State = EntityState.Added;
            _context.SaveChanges();
        }

        public void Alterar(Empresa entity)
        {
            _context.Entry(entity).State = EntityState.Modified;
            _context.SaveChanges();
        }

        public void Excluir(Empresa entity)
        {
            _context.Entry(entity).State = EntityState.Deleted;
            _context.SaveChanges();
        }

        public List<Empresa> Consultar()
        {
            return _context.Empresa
                .OrderBy(e => e.NomeFantasia)
                .ToList();
        }

        public Empresa ObterPorId(Guid id)
        {
            return _context.Empresa
                .FirstOrDefault(e => e.IdEmpresa.Equals(id));
        }

        public Empresa ObterPorCnpj(string cnpj)
        {
            return _context.Empresa
                .FirstOrDefault(e => e.Cnpj.Equals(cnpj));
        }
    }
}
```

Implementando a classe FuncionarioRepository.cs

/Repositories/**FuncionarioRepository.cs**

```
using ApiEmpresas.Infra.Data.Contexts;
using ApiEmpresas.Infra.Data.Entities;
using ApiEmpresas.Infra.Data.Interfaces;
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ApiEmpresas.Infra.Data.Repositories
{
    public class FuncionarioRepository : IFuncionarioRepository
    {
        //atributo
        private readonly SqlServerContext _context;

        //construtor para injeção de dependência
        public FuncionarioRepository(SqlServerContext context)
        {
            _context = context;
        }

        public void Inserir(Funcionario entity)
        {
            _context.Entry(entity).State = EntityState.Added;
            _context.SaveChanges();
        }

        public void Alterar(Funcionario entity)
        {
            _context.Entry(entity).State = EntityState.Modified;
            _context.SaveChanges();
        }

        public void Excluir(Funcionario entity)
        {
            _context.Entry(entity).State = EntityState.Deleted;
            _context.SaveChanges();
        }

        public List<Funcionario> Consultar()
        {
            return _context.Funcionario
                .OrderBy(f => f.Nome)
                .ToList();
        }

        public Funcionario ObterPorId(Guid id)
        {
            return _context.Funcionario
                .FirstOrDefault(f => f.IdFuncionario.Equals(id));
        }
    }
}
```

```

public Funcionario ObterPorCpf(string cpf)
{
    return _context.Funcionario
        .FirstOrDefault(f => f.Cpf.Equals(cpf));
}

public Funcionario ObterPorMatricula(string matricula)
{
    return _context.Funcionario
        .FirstOrDefault(f => f.Matricula.Equals(matricula));
}

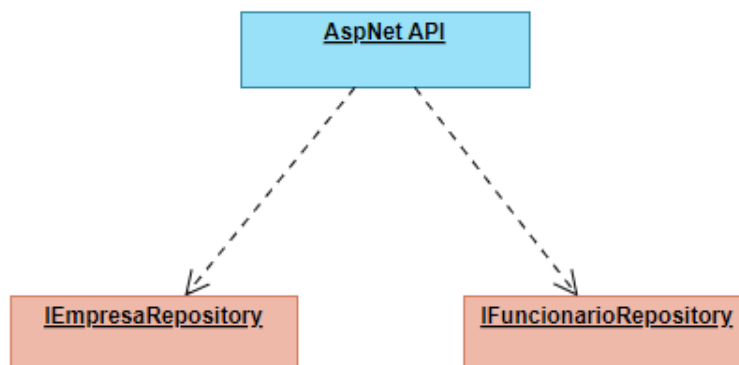
public List<Funcionario> ObterPorNome(string nome)
{
    return _context.Funcionario
        .Where(f => f.Nome.Contains(nome))
        .OrderBy(f => f.Nome)
        .ToList();
}
}
}

```

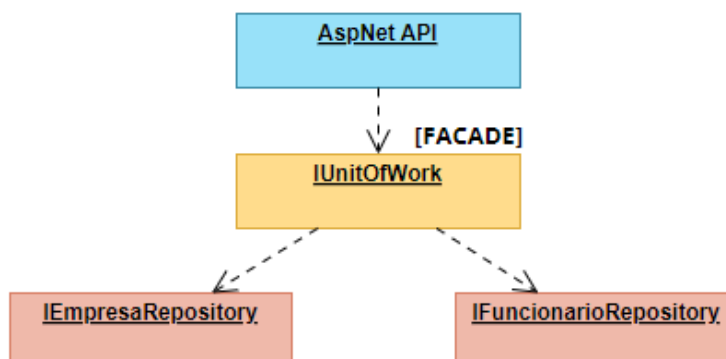
UNIT OF WORK

Padrão utilizado pelo EntityFramework que define um único ponto de acesso para que os projetos de alto nível possam manipular os repositórios.

Antes do UnitOfWork:



Com do UnitOfWork:



UNIT OF WORK

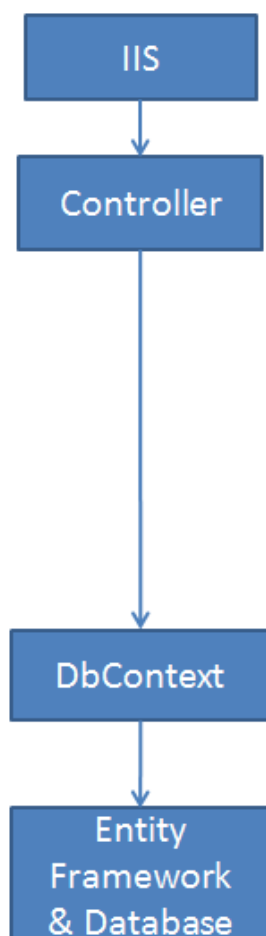
Unit Of Work ou **Unidade de Trabalho** é um padrão de projeto, e, de acordo com Martin Fowler, o padrão de unidade de trabalho "*mantém uma lista de objetos afetados por uma transação e coordena a escrita de mudanças e trata possíveis problemas de concorrência*".

O padrão **Unit of Work** está presente em quase todas as ferramentas OR/M atuais (*digo quase pois não conheço todas*) e geralmente você não terá que criar a sua implementação personalizada a menos que decida realmente fazer isso por uma questão de força maior.

Então o padrão **Unit of Work** pode ser visto como um contexto, sessão ou objeto que acompanha as alterações das entidades de negócio durante uma transação sendo também responsável pelo gerenciamento dos problemas de concorrência que podem ocorrer oriundos dessa transação.

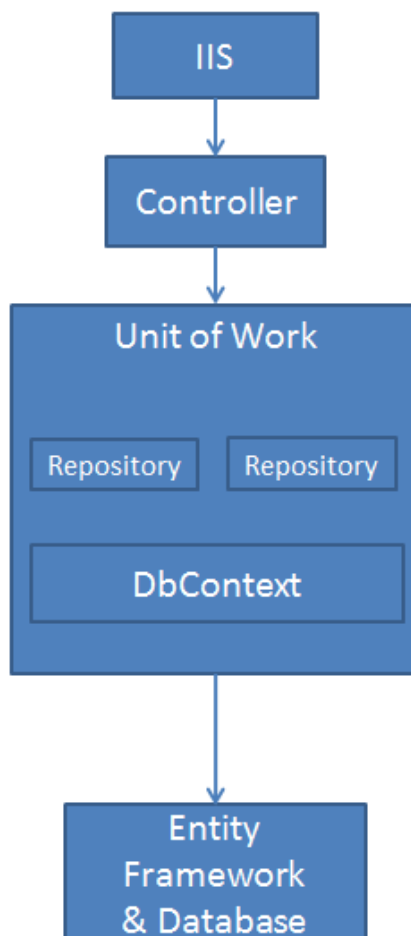
No Repository

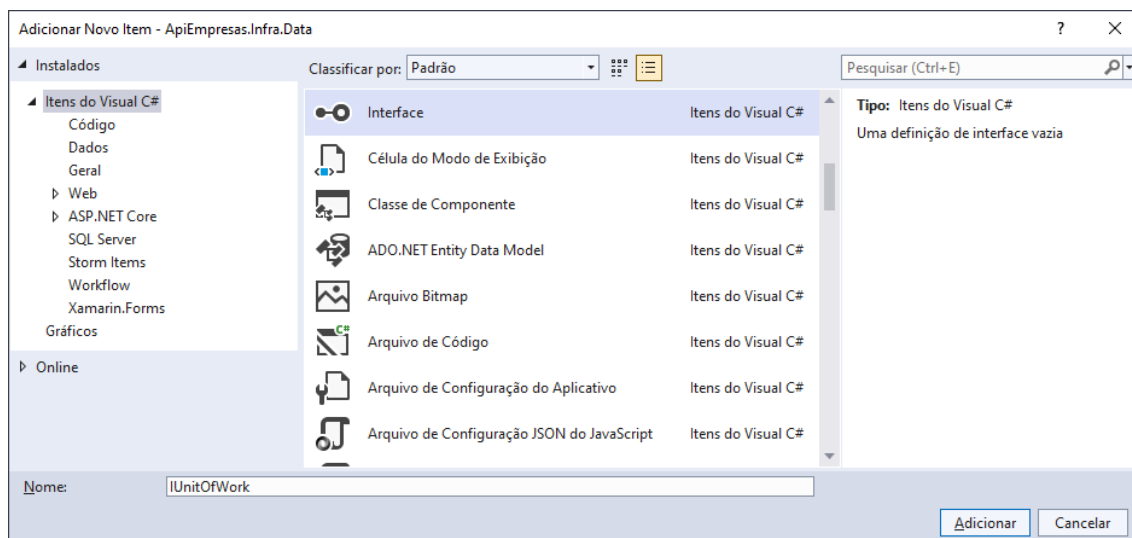
Direct access to database context from controller.



With Repository

Abstraction layer between controller and database context. Unit tests can use a custom persistence layer to facilitate testing.





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

namespace ApiEmpresas.Infra.Data.Interfaces
{
    /// <summary>
    /// Interface de unidade de trabalho do EntityFramework
    /// </summary>
    public interface IUnitOfWork
    {
        #region Métodos para controle de transação

        void BeginTransaction();
        void Commit();
        void Rollback();

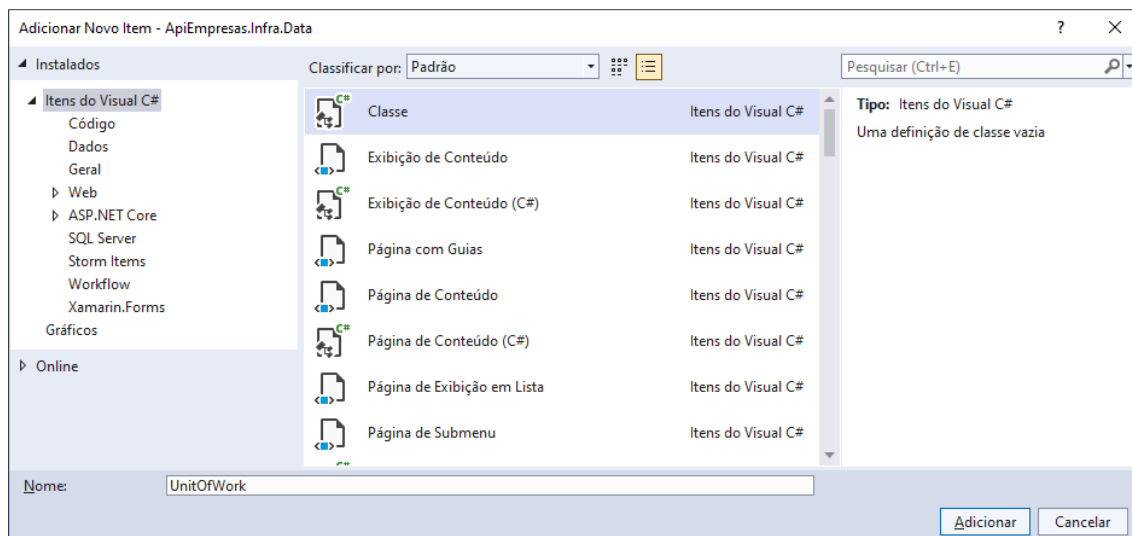
        #endregion

        #region Métodos para acesso aos repositórios

        public IRepository EmpresaRepository { get; }
        public IFuncionarioRepository FuncionarioRepository { get; }

        #endregion
    }
}
```

Implementando a interface:



```
using ApiEmpresas.Infra.Data.Contexts;
using ApiEmpresas.Infra.Data.Interfaces;
using Microsoft.EntityFrameworkCore.Storage;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ApiEmpresas.Infra.Data.Repositories
{
    /// <summary>
    /// Classe para implementar a unidade de trabalho do EntityFramework
    /// </summary>
    public class UnitOfWork : IUnitOfWork
    {
        //atributo
        private readonly SqlServerContext _context;
        private IDbContextTransaction _transaction;

        //construtor para injeção de dependência
        public UnitOfWork(SqlServerContext context)
        {
            _context = context;
        }

        public void BeginTransaction()
        {
            _transaction = _context.Database.BeginTransaction();
        }

        public void Commit()
        {
            _transaction.Commit();
        }
    }
}
```

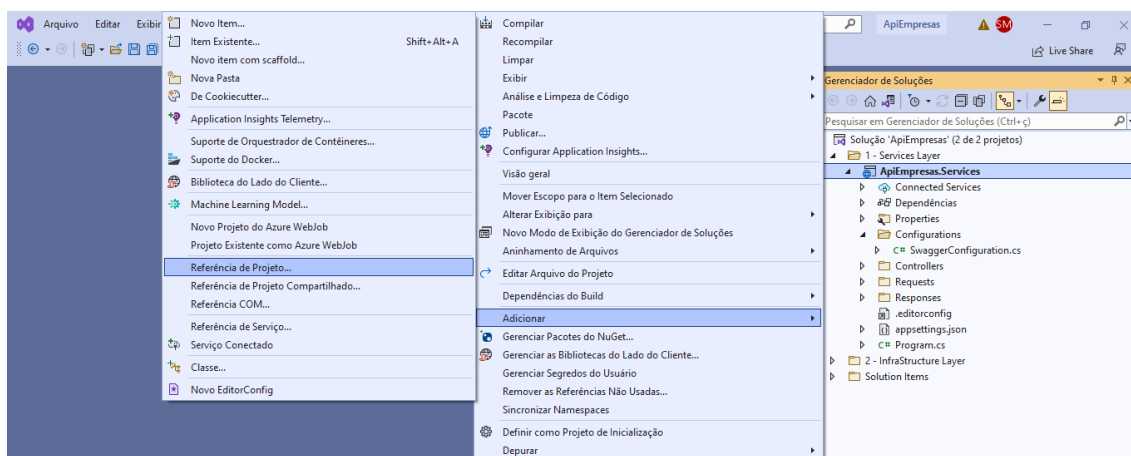
```
public void Rollback()
{
    _transaction.Rollback();
}

public IEmpresaRepository EmpresaRepository
=> new EmpresaRepository(_context);

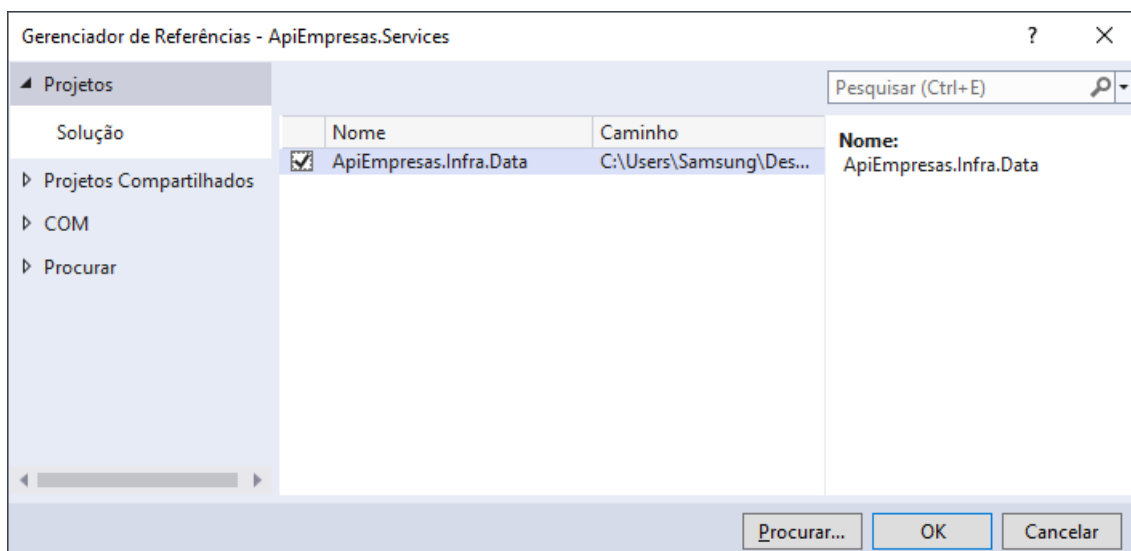
public IFuncionarioRepository FuncionarioRepository
=> new FuncionarioRepository(_context);
}
}
```

Próximo passo: Implementar a API

- Primeiro, precisamos adicionar referência no projeto API para o projeto **Infra.Data**



Adicione referência:



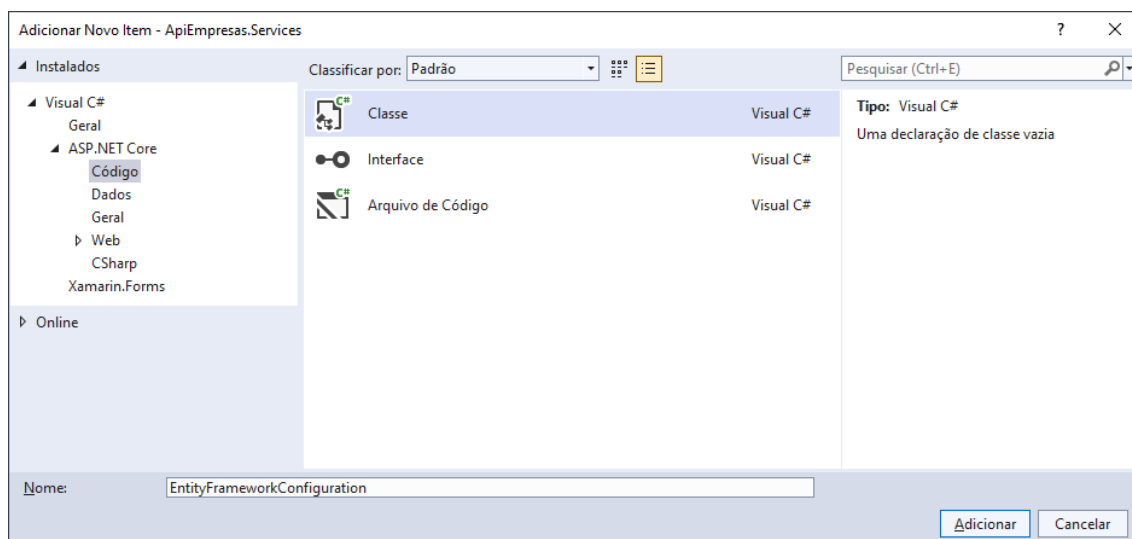
/appsettings.json

Maapeando a connectionstring do banco de dados.

```
{
  /* mapeamento da connectionstring */
  "ConnectionStrings": {
    "BDApiEmpresas": "Data Source=(localdb)\\MSSQLLocalDB;Initial
    Catalog=BDApiEmpresas;Integrated Security=True;Connect
    Timeout=30;Encrypt=False;TrustServerCertificate=False;Applic
    ationIntent=ReadWrite;MultiSubnetFailover=False"
  }
}
```

/Configurations/EntityFrameworkConfiguration.cs

Classe para configuração do EntityFramework.



```
using ApiEmpresas.Infra.Data.Contexts;
using ApiEmpresas.Infra.Data.Interfaces;
using ApiEmpresas.Infra.Data.Repositories;
using Microsoft.EntityFrameworkCore;

namespace ApiEmpresas.Services.Configurations
{
    /// <summary>
    /// Classe para configuração do EntityFramework
    /// </summary>
    public static class EntityFrameworkConfiguration
    {
        /// <summary>
        /// Configurar o entity framework
        /// </summary>
        public static void AddEntityFramework(WebApplicationBuilder builder)
        {
            //capturar a string de conexão do banco de dados
        }
    }
}
```

```
var connectionString = builder.Configuration
    .GetConnectionString("BDApiEmpresas");

//injetar a connectionString
//na classe SqlServerContext do EntityFramework
builder.Services.AddDbContext<SqlServerContext>
    (options => options.UseSqlServer(connectionString));

//injeção de dependência para o UnitOfWork
builder.Services.AddTransient<IUnitOfWork, UnitOfWork>();
    }
}
}
```

/Program.cs

Incluindo a configuração.

```
using ApiEmpresas.Services.Configurations;

var builder = WebApplication.CreateBuilder(args);

// Configurando os controllers da aplicação
builder.Services.AddControllers();

//Adicionando a configuração do Swagger
SwaggerConfiguration.AddSwagger(builder);

//Adicionando a configuração do EntityFramework
EntityFrameworkConfiguration.AddEntityFramework(builder);

// Add services to the container.
var app = builder.Build();

// Habilitar as rotas e endpoints da API
app.UseRouting();

//Configurando o descritor da API
app.UseSwagger();
app.UseSwaggerUI(s =>
{
    s.SwaggerEndpoint("/swagger/v1/swagger.json", "ProjetoAPI");
});

//Executar os serviços
app.UseEndpoints(endpoints =>
{
    endpoints.MapControllers();
});

app.Run();
```

/Controllers/**EmpresasController.cs**

Implementando os serviços da API utilizando os métodos da camada de repositório.

```
using ApiEmpresas.Infra.Data.Entities;
using ApiEmpresas.Infra.Data.Interfaces;
using ApiEmpresas.Services.Requests;
using ApiEmpresas.Services.Responses;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;

namespace ApiEmpresas.Services.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class EmpresasController : ControllerBase
    {
        //atributo
        private readonly IUnitOfWork _unitOfWork;

        //construtor para injeção de dependência
        public EmpresasController(IUnitOfWork unitOfWork)
        {
            _unitOfWork = unitOfWork;
        }

        [HttpPost]
        public IActionResult Post(EmpresaPostRequest request)
        {
            try
            {
                //verificar se o CNPJ informado já está cadastrado..
                if (_unitOfWork.EmpresaRepository.ObterPorCnpj
                    (request.Cnpj) != null)
                {
                    //HTTP 422 -> UNPROCESSABLE ENTITY
                    return StatusCode(422,
                        new { message = "O CNPJ informado
                                já está cadastrado." });
                }

                var empresa = new Empresa
                {
                    IdEmpresa = Guid.NewGuid(),
                    NomeFantasia = request.NomeFantasia,
                    RazaoSocial = request.RazaoSocial,
                    Cnpj = request.Cnpj,
                };

                //gravar no banco de dados
                _unitOfWork.EmpresaRepository.Inserir(empresa);
            }
        }
    }
}
```

```
var response = new EmpresaResponse
{
    Id = empresa.IdEmpresa,
    NomeFantasia = empresa.NomeFantasia,
    RazaoSocial = empresa.RazaoSocial,
    Cnpj = empresa.Cnpj
};

//HTTP 201 -> SUCCESS CREATED
return StatusCode(201, response);
}
catch(Exception e)
{
    //retornando status e mensagem de erro
    //HTTP 500 -> ERRO INTERNO DE SERVIDOR
    return StatusCode(500, e.Message);
}
}

[HttpPut]
public IActionResult Put(EmpresaPutRequest request)
{
    var response = new EmpresaResponse
    {
        Id = request.IdEmpresa,
        NomeFantasia = request.NomeFantasia,
        RazaoSocial = request.RazaoSocial,
        Cnpj = request.Cnpj,
        DataInclusao = DateTime.Now,
        DataUltimaAlteracao = DateTime.Now,
    };

    return StatusCode(200, response);
}

[HttpDelete("{idEmpresa}")]
public IActionResult Delete(Guid idEmpresa)
{
    var response = new EmpresaResponse
    {
        Id = idEmpresa,
        NomeFantasia = "Empresa Teste",
        RazaoSocial = "Empresa Teste LTDA",
        Cnpj = "44.424.467/0001-34",
        DataInclusao = DateTime.Now,
        DataUltimaAlteracao = DateTime.Now,
    };

    return StatusCode(200, response);
}
```



```
[HttpGet]
public IActionResult GetAll()
{
    var lista = new List<EmpresaResponse>();

    lista.Add(new EmpresaResponse
    {
        Id = Guid.NewGuid(),
        NomeFantasia = "Empresa Teste 01",
        RazaoSocial = "Empresa Teste 01 LTDA",
        Cnpj = "44.424.467/0001-34",
        DataInclusao = DateTime.Now,
        DataUltimaAlteracao = DateTime.Now,
    });

    lista.Add(new EmpresaResponse
    {
        Id = Guid.NewGuid(),
        NomeFantasia = "Empresa Teste 02",
        RazaoSocial = "Empresa Teste 02 LTDA",
        Cnpj = "70.614.891/0001-51",
        DataInclusao = DateTime.Now,
        DataUltimaAlteracao = DateTime.Now,
    });

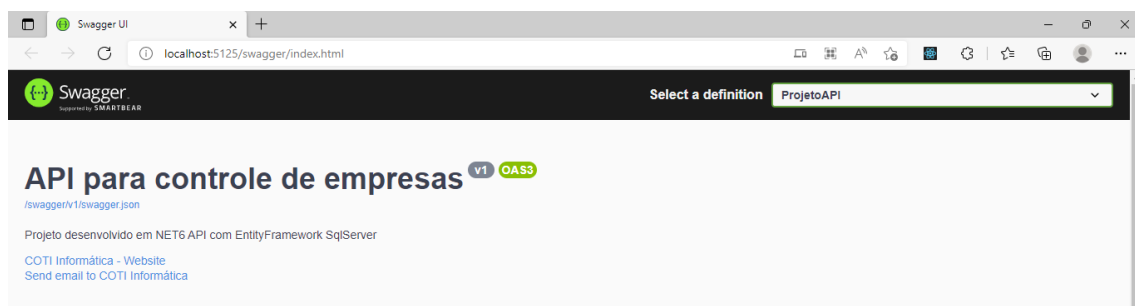
    return StatusCode(200, lista);
}

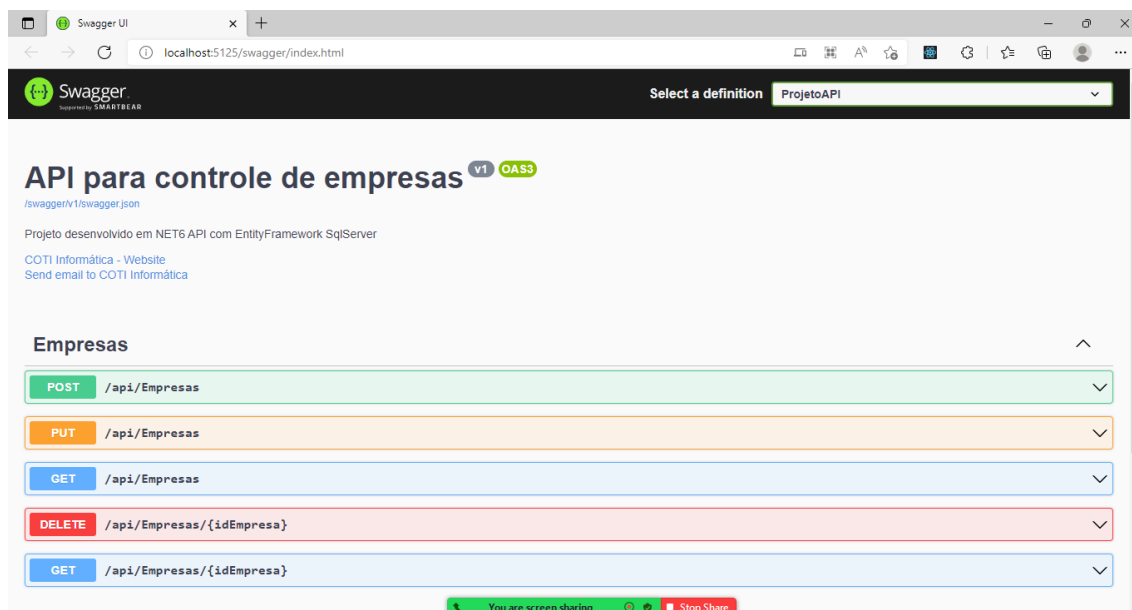
[HttpGet("{idEmpresa}")]
public IActionResult GetById(Guid idEmpresa)
{
    var response = new EmpresaResponse
    {
        Id = Guid.NewGuid(),
        NomeFantasia = "Empresa Teste",
        RazaoSocial = "Empresa Teste LTDA",
        Cnpj = "44.424.467/0001-34",
        DataInclusao = DateTime.Now,
        DataUltimaAlteracao = DateTime.Now,
    };

    return StatusCode(200, response);
}
}
```

Testando:

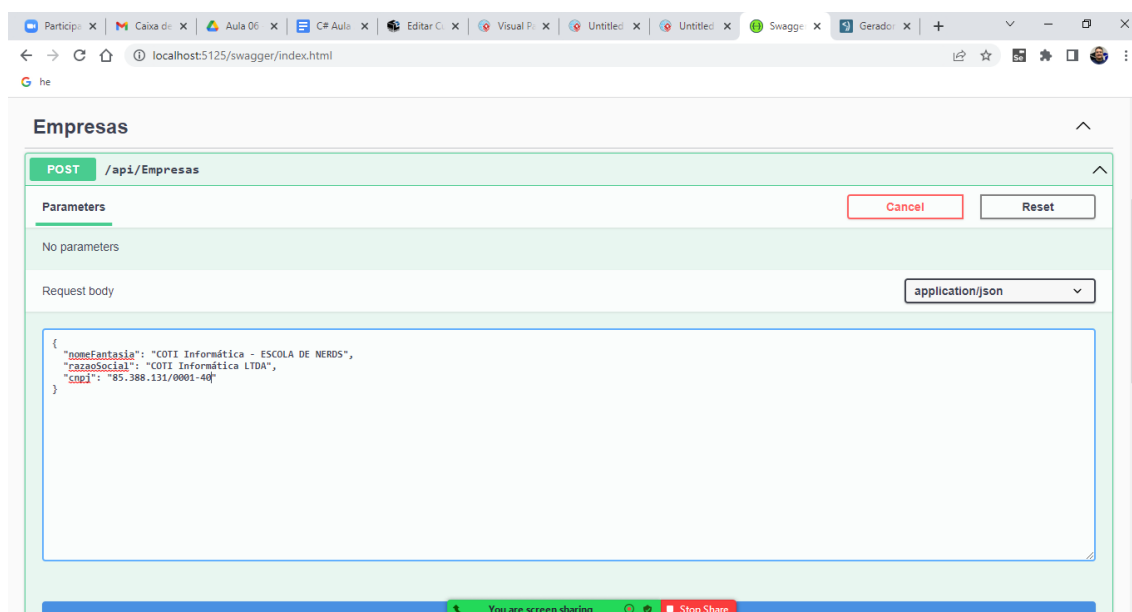
<http://localhost:5125/swagger/index.html>





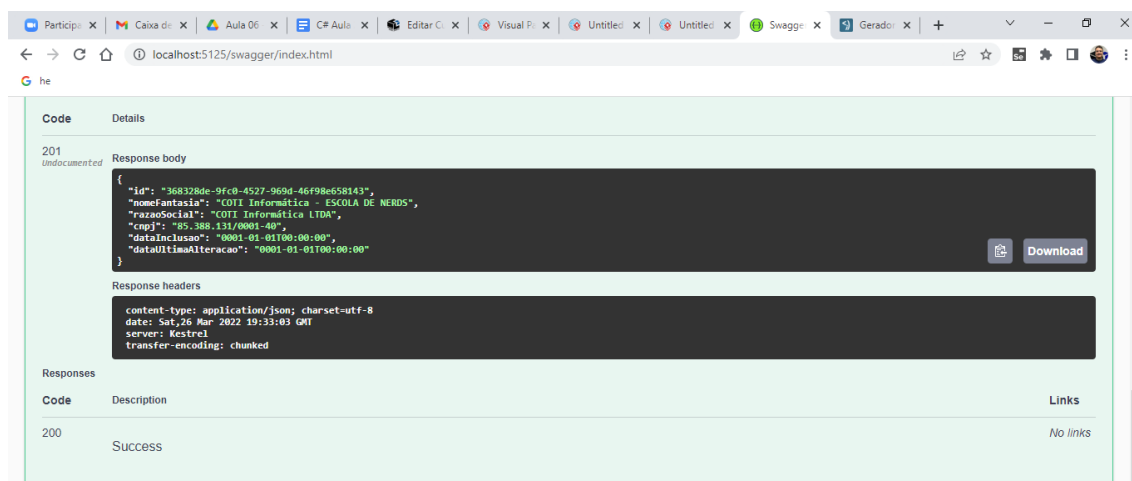
Swagger UI interface showing the API definition for "API para controle de empresas" (v1 OAS3). The API is developed in .NET 6 API with EntityFramework SqlServer. The interface lists several endpoints under the "Empresas" group:

- POST** /api/Empresas
- PUT** /api/Empresas
- GET** /api/Empresas
- DELETE** /api/Empresas/{idEmpresa}
- GET** /api/Empresas/{idEmpresa}



Swagger UI interface showing the details for the **POST** endpoint /api/Empresas. The interface includes a "Parameters" section (No parameters) and a "Request body" section (application/json). The request body is a JSON object:

```
{
  "nomeFantasia": "COTI Informática - ESCOLA DE NERDS",
  "razaoSocial": "COTI Informática LTDA",
  "cnpj": "85.388.131/0001-40"
}
```



Swagger UI interface showing the response details for the **POST** endpoint. The response is a 201 status code (undocumented). The response body is a JSON object:

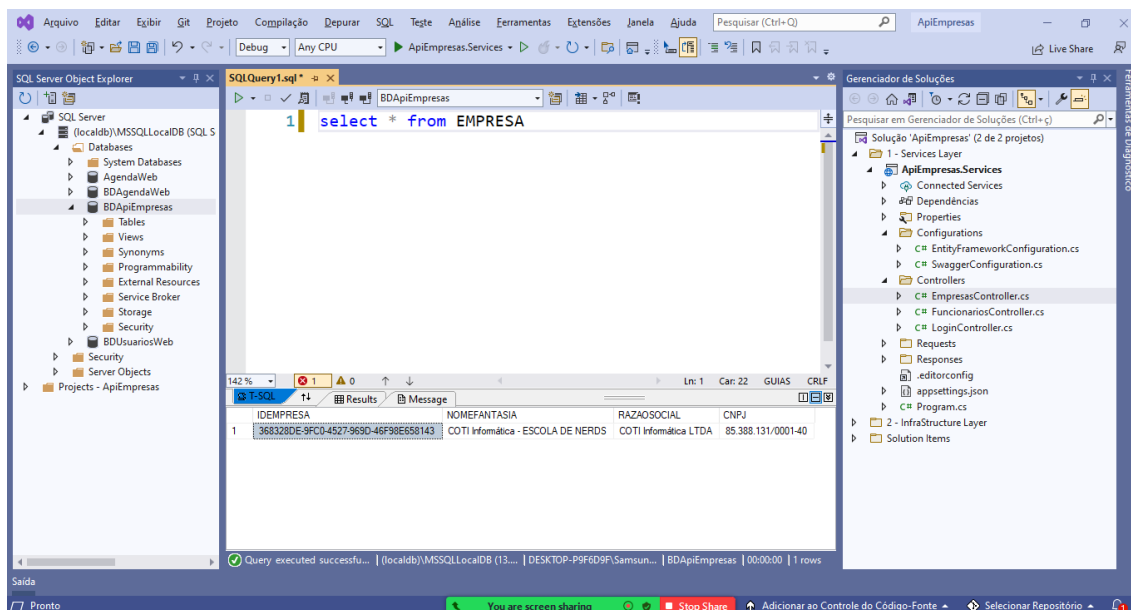
```
{
  "id": "368328de-9fc0-4527-969d-46f98e58143",
  "nomeFantasia": "COTI Informática - ESCOLA DE NERDS",
  "razaoSocial": "COTI Informática LTDA",
  "cnpj": "85.388.131/0001-40",
  "dataInclusao": "0001-01-01T00:00:00",
  "dataUltimaAlteracao": "0001-01-01T00:00:00"
}
```

The response headers are:

```
content-type: application/json; charset=utf-8
date: Sat, 26 Mar 2022 19:33:03 GMT
server: Kestrel
transfer-encoding: chunked
```

The response is successful (200 Success).

No banco de dados:



Voltando no controller:

```
using ApiEmpresas.Infra.Data.Entities;
using ApiEmpresas.Infra.Data.Interfaces;
using ApiEmpresas.Services.Requests;
using ApiEmpresas.Services.Responses;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;

namespace ApiEmpresas.Services.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class EmpresasController : ControllerBase
    {
        //atributo
        private readonly IUnitOfWork _unitOfWork;

        //construtor para injeção de dependência
        public EmpresasController(IUnitOfWork unitOfWork)
        {
            _unitOfWork = unitOfWork;
        }

        [HttpPost]
        public IActionResult Post(EmpresaPostRequest request)
        {
            try
            {
                //verificar se o CNPJ informado já está cadastrado..
                if (_unitOfWork.EmpresaRepository.ObterPorCnpj
                    (request.Cnpj) != null)
                {
                    //HTTP 422 -> UNPROCESSABLE ENTITY
                    return StatusCode(422,
                        new { message = "O CNPJ informado
                            já está cadastrado." });
                }
            }
        }
    }
}
```

```
var empresa = new Empresa
{
    IdEmpresa = Guid.NewGuid(),
    NomeFantasia = request.NomeFantasia,
    RazaoSocial = request.RazaoSocial,
    Cnpj = request.Cnpj,
};

//gravar no banco de dados
_unitOfWork.EmpresaRepository.Inserir(empresa);

var response = new EmpresaResponse
{
    Id = empresa.IdEmpresa,
    NomeFantasia = empresa.NomeFantasia,
    RazaoSocial = empresa.RazaoSocial,
    Cnpj = empresa.Cnpj
};

//HTTP 201 -> SUCCESS CREATED
return StatusCode(201, response);
}
catch(Exception e)
{
    //retornando status e mensagem de erro
    //HTTP 500 -> ERRO INTERNO DE SERVIDOR
    return StatusCode(500, e.Message);
}
}

[HttpPut]
public IActionResult Put(EmpresaPutRequest request)
{
    try
    {
        //pesquisando a empresa através do id..
        var empresa = _unitOfWork.EmpresaRepository
            .ObterPorId(request.IdEmpresa);

        //verificando se a empresa não foi encontrada
        if(empresa == null)
        {
            //HTTP 422 -> UNPROCESSABLE ENTITY
            return StatusCode(422, new
            { message = "Empresa não encontrada,
            verifique o ID informado." });
        }

        //verificando se o cnpj informado
        //já está cadastrado para outra empresa
        var registro = _unitOfWork.EmpresaRepository
            .ObterPorCnpj(request.Cnpj);
        if(registro != null && registro.IdEmpresa
            != empresa.IdEmpresa)
        {
            //HTTP 422 -> UNPROCESSABLE ENTITY
            return StatusCode(422, new
            { message = "O CNPJ informado já está
            cadastrado para outra empresa." });
        }

        //atualizando os dados da empresa
        empresa.NomeFantasia = request.NomeFantasia;
        empresa.RazaoSocial = request.RazaoSocial;
    }
}
```

```

        empresa.Cnpj = request.Cnpj;

        _unitOfWork.EmpresaRepository.Alterar(empresa);

        var response = new EmpresaResponse
        {
            Id = empresa.IdEmpresa,
            NomeFantasia = empresa.NomeFantasia,
            RazaoSocial = empresa.RazaoSocial,
            Cnpj = empresa.Cnpj
        };

        return StatusCode(200, response);
    }
    catch (Exception e)
    {
        //retornando status e mensagem de erro
        //HTTP 500 -> ERRO INTERNO DE SERVIDOR
        return StatusCode(500, e.Message);
    }
}

[HttpDelete("{idEmpresa}")]
public IActionResult Delete(Guid idEmpresa)
{
    try
    {
        //pesquisando a empresa atraves do id..
        var empresa = _unitOfWork.EmpresaRepository
            .ObterPorId(idEmpresa);

        //verificando se a empresa não foi encontrada
        if (empresa == null)
        {
            //HTTP 422 -> UNPROCESSABLE ENTITY
            return StatusCode(422, new
            {
                message = "Empresa não encontrada, verifique o ID informado."
            });
        }

        //excluindo a empresa
        _unitOfWork.EmpresaRepository.Excluir(empresa);

        var response = new EmpresaResponse
        {
            Id = empresa.IdEmpresa,
            NomeFantasia = empresa.NomeFantasia,
            RazaoSocial = empresa.RazaoSocial,
            Cnpj = empresa.Cnpj
        };

        return StatusCode(200, response);
    }
    catch (Exception e)
    {
        //retornando status e mensagem de erro
        //HTTP 500 -> ERRO INTERNO DE SERVIDOR
        return StatusCode(500, e.Message);
    }
}
}

```

```
[HttpGet]
public IActionResult GetAll()
{
    try
    {
        var lista = new List<EmpresaResponse>();

        //consultar as empresas no repositório
        foreach (var item in _unitOfWork
            .EmpresaRepository.Consultar())
        {
            lista.Add(new EmpresaResponse
            {
                Id = item.IdEmpresa,
                NomeFantasia = item.NomeFantasia,
                RazaoSocial = item.RazaoSocial,
                Cnpj = item.Cnpj
            });
        }

        if (lista.Count > 0)
            return StatusCode(200, lista);
        else
            return StatusCode(204);
    }
    catch (Exception e)
    {
        //retornando status e mensagem de erro
        //HTTP 500 -> ERRO INTERNO DE SERVIDOR
        return StatusCode(500, e.Message);
    }
}

[HttpGet("{idEmpresa}")]
public IActionResult GetById(Guid idEmpresa)
{
    try
    {
        //buscar a empresa no repositório através do id
        var empresa = _unitOfWork
            .EmpresaRepository.ObterPorId(idEmpresa);

        //verificar se a empresa foi encontrada
        if (empresa != null)
        {
            var response = new EmpresaResponse
            {
                Id = empresa.IdEmpresa,
                NomeFantasia = empresa.NomeFantasia,
                RazaoSocial = empresa.RazaoSocial,
                Cnpj = empresa.Cnpj
            };

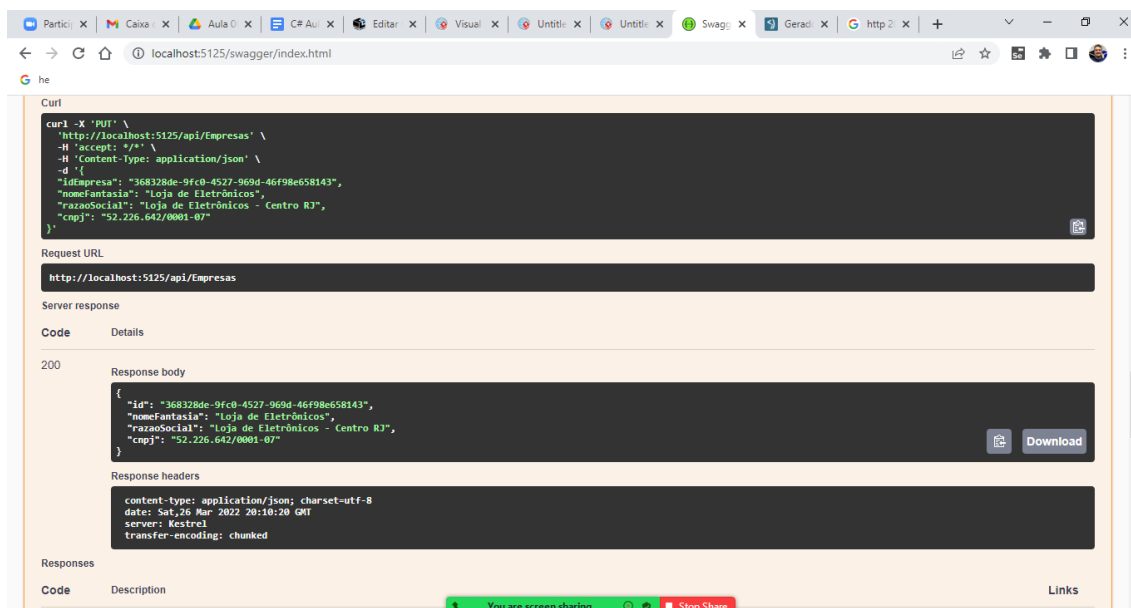
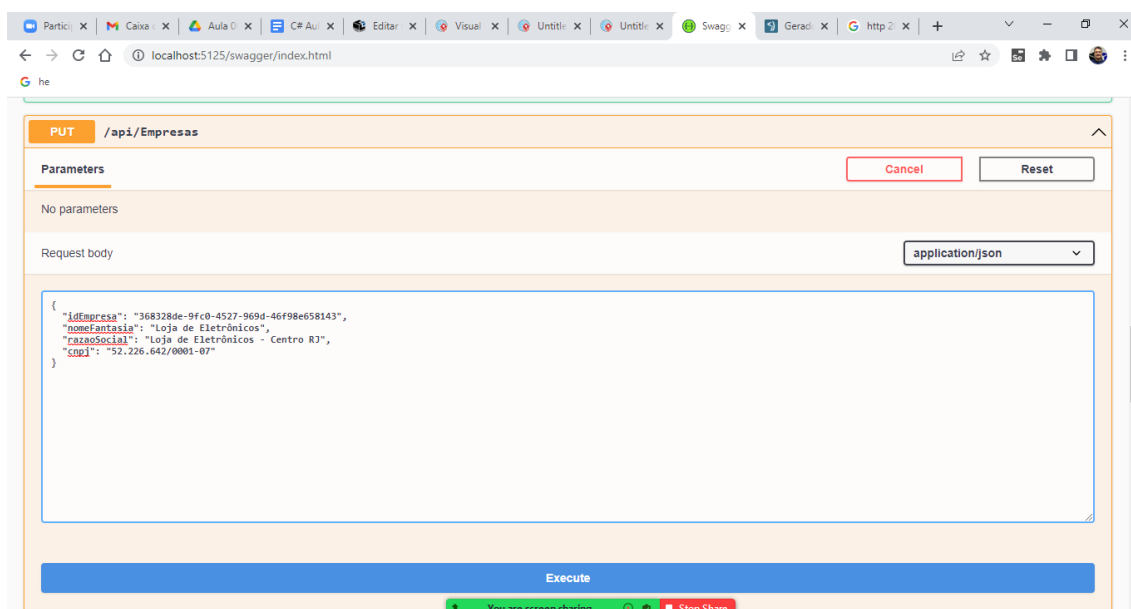
            return StatusCode(200, response);
        }
        else
        {
            return StatusCode(204);
        }
    }
}
```

```

        catch(Exception e)
        {
            //retornando status e mensagem de erro
            //HTTP 500 -> ERRO INTERNO DE SERVIDOR
            return StatusCode(500, e.Message);
        }
    }
}

```

Testando:



Swagger UI interface for the DELETE endpoint: `/api/Empresas/{idEmpresa}`.

Parameters

Name	Description
idEmpresa <small>* required</small>	
string(suuid)	
(path)	

Value:

Execute **Clear**

Responses

Curl

```
curl -X 'DELETE' \
  'http://localhost:5125/api/Empresas/368328de-9fc0-4527-969d-46f98e658143' \
  -H 'accept: */*'
```

Request URL

```
http://localhost:5125/api/Empresas/368328de-9fc0-4527-969d-46f98e658143
```

Server response

Code **Details** You are screen sharing Stop Share

Swagger UI interface showing the response for the DELETE endpoint.

Curl

```
curl -X 'DELETE' \
  'http://localhost:5125/api/Empresas/368328de-9fc0-4527-969d-46f98e658143' \
  -H 'accept: */*'
```

Request URL

```
http://localhost:5125/api/Empresas/368328de-9fc0-4527-969d-46f98e658143
```

Server response

Code **Details**

200

Response body

```
{
  "id": "368328de-9fc0-4527-969d-46f98e658143",
  "nomeFantasia": "Loja de Eletrônicos",
  "razaoSocial": "Loja de Eletrônicos - Centro RJ",
  "cnpj": "52.226.642/0001-07"
}
```

Response headers

```
content-type: application/json; charset=utf-8
date: Sat, 26 Mar 2022 20:11:54 GMT
server: Kestrel
transfer-encoding: chunked
```

Responses

Code	Description	Links
200	Success	No links

You are screen sharing Stop Share

Swagger UI interface for the GET endpoint: `/api/Empresas`.

Parameters

No parameters

Execute **Clear**

Responses

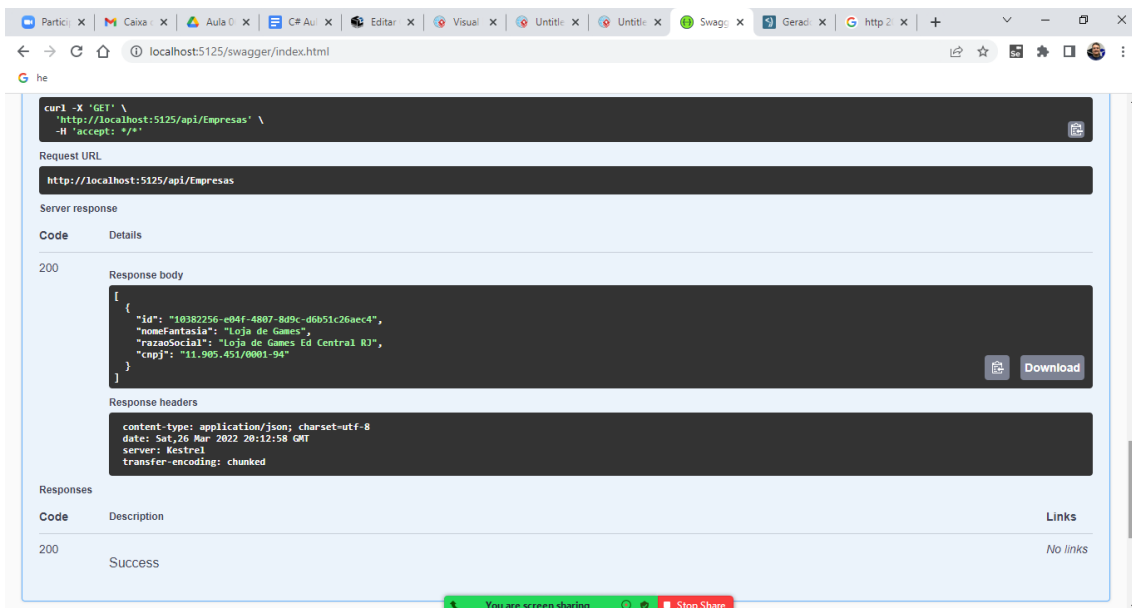
Curl

```
curl -X 'GET' \
  'http://localhost:5125/api/Empresas' \
  -H 'accept: */*'
```

Request URL

```
http://localhost:5125/api/Empresas
```

Server response



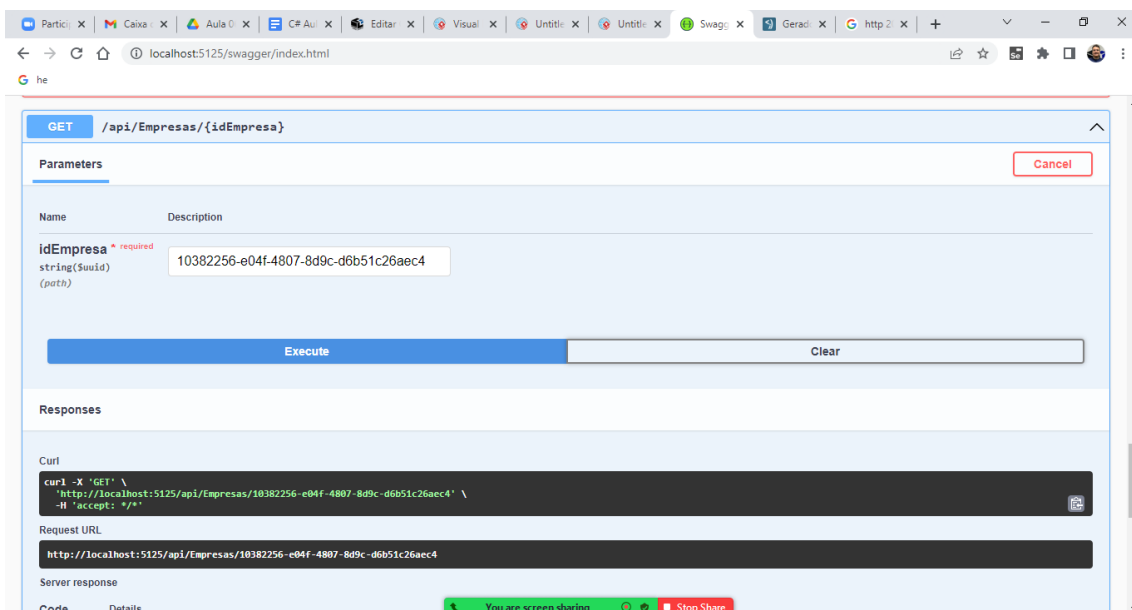
Swagger UI interface showing a successful GET request to `/api/Empresas`. The request URL is `http://localhost:5125/api/Empresas`. The server response is a 200 status code with a JSON body:

```
{
  "id": "10382256-e04f-4807-8d9c-d6b51c26aec4",
  "nomeFantasia": "Loja de Games",
  "razaoSocial": "Loja de Games Ed Central RJ",
  "cnpj": "11.905.451/0001-94"
}
```

The response headers are:

```
content-type: application/json; charset=utf-8
date: Sat, 26 Mar 2022 20:12:58 GMT
server: Kestrel
transfer-encoding: chunked
```

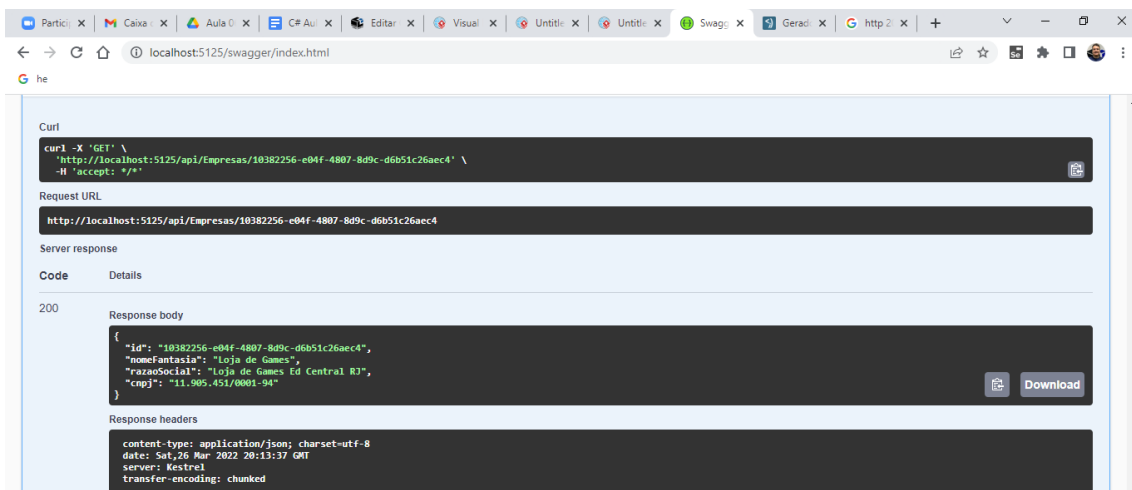
The response status is 200, and the description is "Success".



Swagger UI interface showing the GET endpoint `/api/Empresas/{idEmpresa}`. The parameter `idEmpresa` is required and has a value of `10382256-e04f-4807-8d9c-d6b51c26aec4`. The "Execute" button is visible.

Below the "Execute" button, the "Responses" section shows the curl command and the request URL:

```
curl -X 'GET' \
  "http://localhost:5125/api/Empresas/10382256-e04f-4807-8d9c-d6b51c26aec4" \
  -H 'accept: */*'
http://localhost:5125/api/Empresas/10382256-e04f-4807-8d9c-d6b51c26aec4
```



Swagger UI interface showing the successful GET response for the endpoint `/api/Empresas/10382256-e04f-4807-8d9c-d6b51c26aec4`. The response is a 200 status code with a JSON body:

```
{
  "id": "10382256-e04f-4807-8d9c-d6b51c26aec4",
  "nomeFantasia": "Loja de Games",
  "razaoSocial": "Loja de Games Ed Central RJ",
  "cnpj": "11.905.451/0001-94"
}
```

The response headers are:

```
content-type: application/json; charset=utf-8
date: Sat, 26 Mar 2022 20:13:37 GMT
server: Kestrel
transfer-encoding: chunked
```



Treinamento em C# WebDeveloper

Sábado, 26 de Março de 2022

Desenvolvimento ASP.NET WebAPI com EntityFramework.

Aula

14