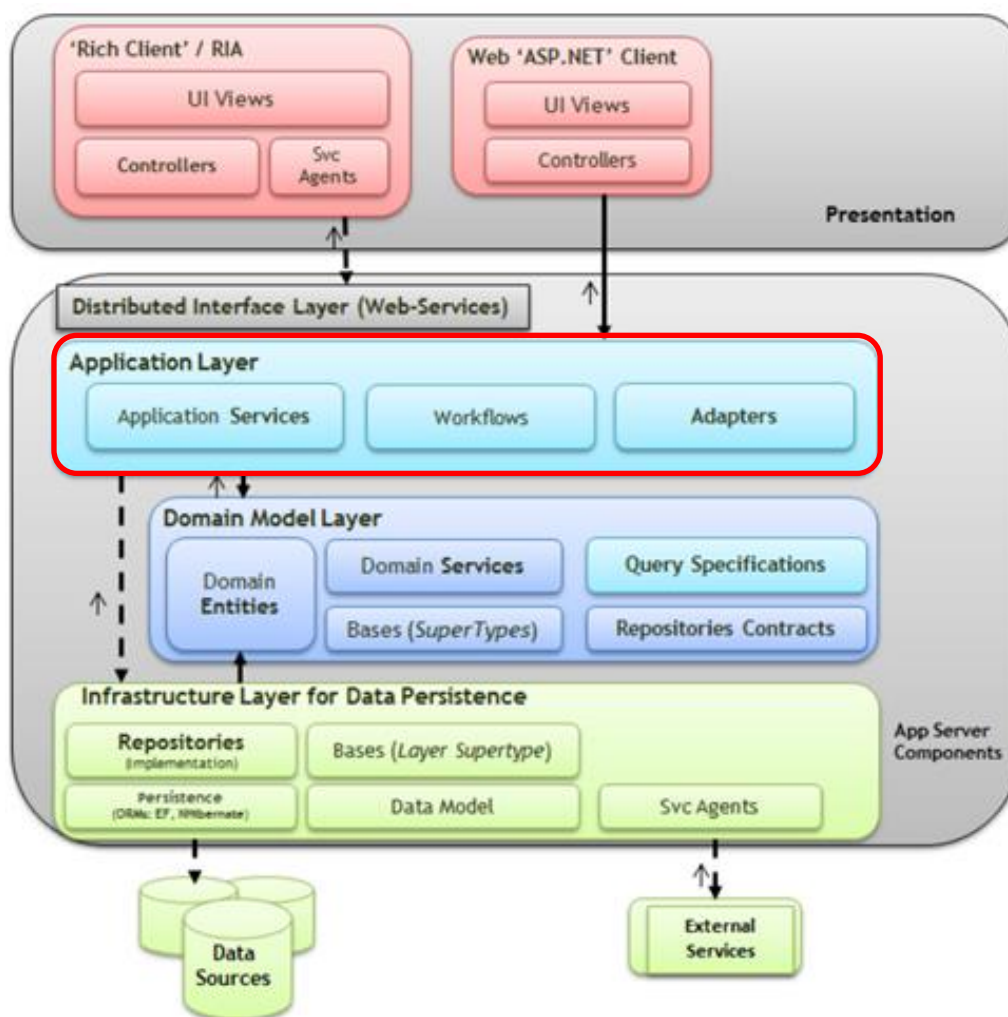


DDD – Domain Driven Design

Arquitetura orientada a domínio

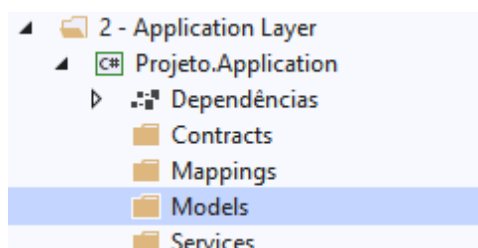
Application Layer

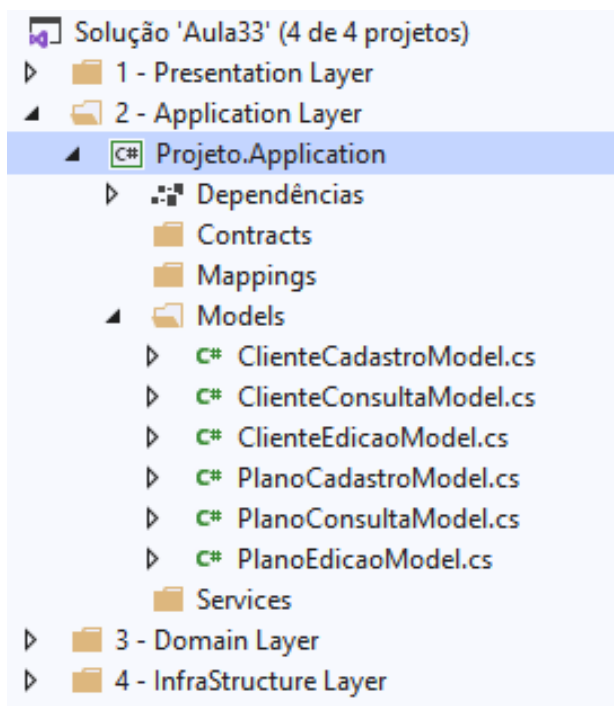
Camada entre o domínio e a apresentação. Tem como objetivo disponibilizar os serviços da camada de domínio para que sejam acessados pela camada de apresentação.



Models (Classes de modelo de dados)

Classes utilizadas para definir os modelos de entrada / saída de dados do projeto (cadastro, consulta, edição, etc). Outra responsabilidade das Models é a de validar os dados de entrada no projeto.





/Models/ClienteCadastroModel.cs

```
using System;
using System.Collections.Generic;
using System.Text;
using System.ComponentModel.DataAnnotations; //validações

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

        [Required(ErrorMessage = "Por favor, informe o cpf do cliente.")]
        public string Cpf { get; set; }

        [Required(ErrorMessage = "Por favor, informe a data
                                de nascimento do cliente.")]
        public DateTime DataNascimento { get; set; }

        [Required(ErrorMessage = "Por favor, informe o plano do cliente.")]
        public int IdPlano { get; set; }
    }
}
```

/Models/ClienteEdicaoModel.cs

```
using System;
using System.Collections.Generic;
using System.Text;
using System.ComponentModel.DataAnnotations; //validações
```

```
namespace Projeto.Application.Models
{
    public class ClienteEdicaoModel
    {
        [Required(ErrorMessage = "Por favor, informe o id do cliente.")]
        public int IdCliente { get; set; }

        [Required(ErrorMessage = "Por favor, informe o nome do cliente.")]
        public string Nome { get; set; }

        [Required(ErrorMessage = "Por favor, informe o cpf do cliente.")]
        public string Cpf { get; set; }

        [Required(ErrorMessage = "Por favor, informe a data
            de nascimento do cliente.")]
        public DateTime DataNascimento { get; set; }

        [Required(ErrorMessage = "Por favor, informe o plano do cliente.")]
        public int IdPlano { get; set; }
    }
}
```

/Models/PlanoCadastroModel.cs

```
using System;
using System.Collections.Generic;
using System.Text;
using System.ComponentModel.DataAnnotations; //validações

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

        [Required(ErrorMessage = "Por favor, informe a descrição do plano.")]
        public string Descricao { get; set; }
    }
}
```

/Models/PlanoEdicaoModel.cs

```
using System;
using System.Collections.Generic;
using System.Text;
using System.ComponentModel.DataAnnotations; //validações

namespace Projeto.Application.Models
{
    public class PlanoEdicaoModel
    {
        [Required(ErrorMessage = "Por favor, informe o id do plano.")]
        public int IdPlano { get; set; }

        [Required(ErrorMessage = "Por favor, informe o nome do plano.")]
        public string Nome { get; set; }
    }
}
```

```
[Required(ErrorMessage = "Por favor, informe a descrição do plano.")]
public string Descricao { get; set; }
}
}
```

/Models/PlanoConsultaModel.cs

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

namespace Projeto.Application.Models
{
    public class PlanoConsultaModel
    {
        public int IdPlano { get; set; }
        public string Nome { get; set; }
        public string Descricao { get; set; }
    }
}
```

/Models/ClienteConsultaModel.cs

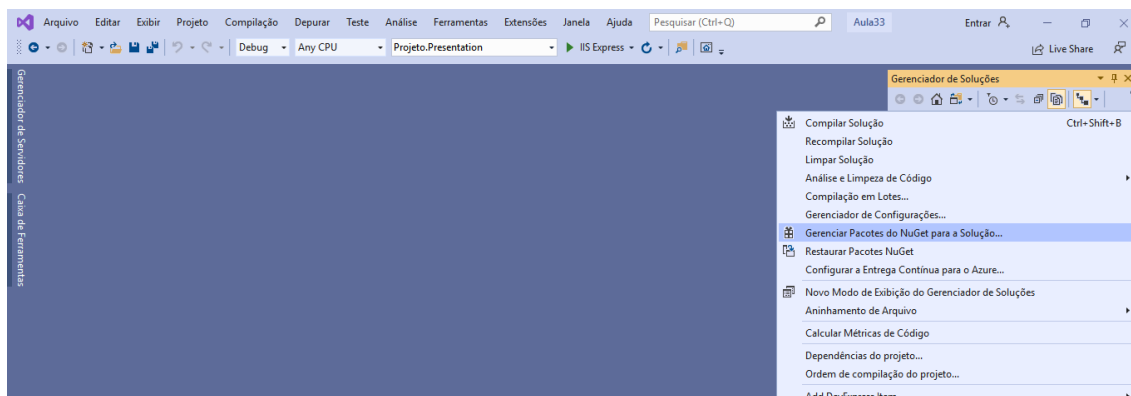
```
using System;
using System.Collections.Generic;
using System.Text;

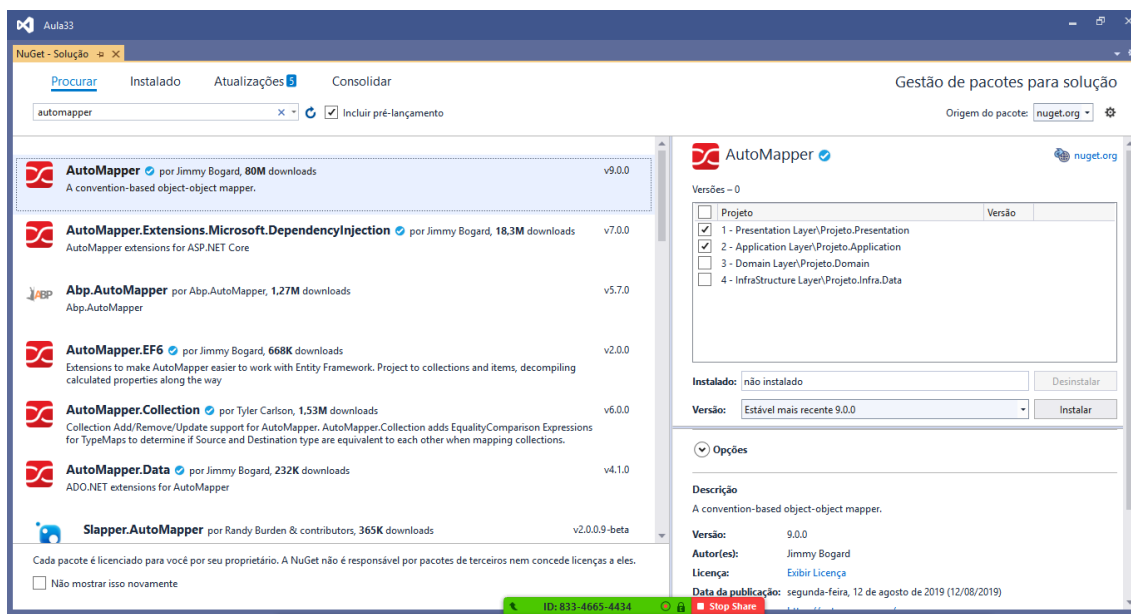
namespace Projeto.Application.Models
{
    public class ClienteConsultaModel
    {
        public int IdCliente { get; set; }
        public string Nome { get; set; }
        public string Cpf { get; set; }
        public DateTime DataNascimento { get; set; }

        public PlanoConsultaModel Plano { get; set; }
    }
}
```

Instalando o AutoMapper (v.8.1.0)

Gerenciador de pacotes do NuGet





/Application/Mappings

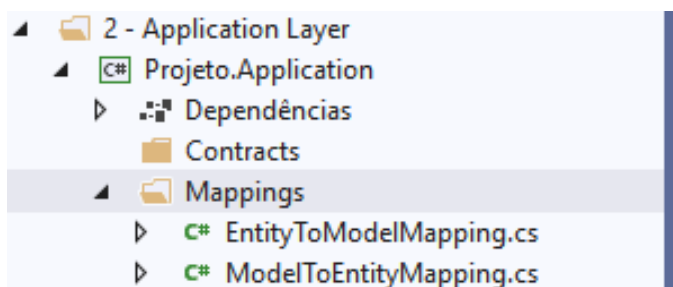
Criar os mapeamentos do AutoMapper

EntityToModelMapping (Consultas / Saída de dados)

- Plano → PlanoConsultaModel
- Cliente → ClienteConsultaModel

ModelToEntityMapping (Cadastro, Edição / Entrada de dados)

- PlanoCadastroModel → Plano
- PlanoEdicaoModel → Plano
- ClienteCadastroModel → Cliente
- ClienteEdicaoModel → Cliente



/Application/Mappings/EntityToModelMapping.cs

```
using System;
using System.Collections.Generic;
using System.Text;
using AutoMapper;
using Projeto.Application.Models;
using Projeto.Domain.Entities;
```

```
namespace Projeto.Application.Mappings
{
    public class EntityToModelMapping : Profile
    {
        //construtor -> ctor + 2x[tab]
        public EntityToModelMapping()
        {
            //consultas (saída de dados)
            CreateMap<PlanoEntity, PlanoConsultaModel>();
            CreateMap<ClienteEntity, ClienteConsultaModel>();
        }
    }
}
```

/Application/Mappings/ModelToEntityMapping.cs

```
using AutoMapper;
using Projeto.Application.Models;
using Projeto.Domain.Entities;
using System;
using System.Collections.Generic;
using System.Text;

namespace Projeto.Application.Mappings
{
    public class ModelToEntityMapping : Profile
    {
        //construtor -> ctor + 2x[tab]
        public ModelToEntityMapping()
        {
            //entradas de dados (cadastro, edição, etc)
            CreateMap<PlanoCadastroModel, PlanoEntity>();
            CreateMap<ClienteCadastroModel, ClienteEntity>();

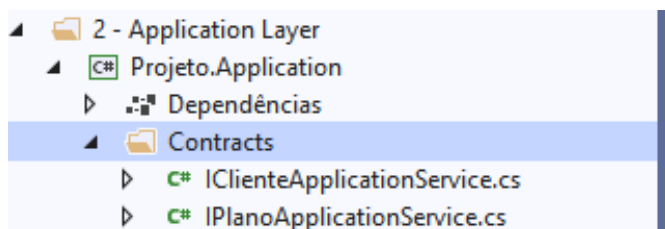
            CreateMap<PlanoEdicaoModel, PlanoEntity>();
            CreateMap<ClienteEdicaoModel, ClienteEntity>();
        }
    }
}
```

Application Services

Serviços da camada de aplicação.

Classes que irão disponibilizar as regras de negócio do Domínio para a camada de apresentação, por meio de Serviços de aplicação.

Primeiro, iremos criar os contratos (interfaces):



```
using Projeto.Application.Models;
using System;
using System.Collections.Generic;
using System.Text;

namespace Projeto.Application.Contracts
{
    public interface IClienteApplicationService
    {
        void Cadastrar(ClienteCadastroModel model);
        void Atualizar(ClienteEdicaoModel model);
        void Excluir(int idCliente);
        List<ClienteConsultaModel> Consultar();
        ClienteConsultaModel ObterPorId(int idCliente);
    }
}
```

```
using Projeto.Application.Models;
using System;
using System.Collections.Generic;
using System.Text;

namespace Projeto.Application.Contracts
{
    public interface IP planoApplicationService
    {
        void Cadastrar(PlanoCadastroModel model);
        void Atualizar(PlanoEdicaoModel model);
        void Excluir(int idPlano);
        List<PlanoConsultaModel> Consultar();
        PlanoConsultaModel ObterPorId(int idPlano);
    }
}
```

Implementando as interfaces:

/Application/Services/ClienteApplicationService.cs

```
using AutoMapper;
using Projeto.Application.Contracts;
using Projeto.Application.Models;
using Projeto.Domain.Contracts.Services;
using Projeto.Domain.Entities;
using System;
using System.Collections.Generic;
using System.Text;

namespace Projeto.Application.Services
{
    public class ClienteApplicationService : IClienteApplicationService
    {
        //atributo
        private readonly IClienteDomainService domainService;

        //construtor para injeção de dependência (inicialização)
        public ClienteApplicationService(IClienteDomainService domainService)
        {
            this.domainService = domainService;
        }
    }
}
```

```

public void Cadastrar(ClienteCadastroModel model)
{
    var cliente = Mapper.Map<ClienteEntity>(model);
    domainService.Cadastrar(cliente);
}

public void Atualizar(ClienteEdicaoModel model)
{
    var cliente = Mapper.Map<ClienteEntity>(model);
    domainService.Atualizar(cliente);
}

public void Excluir(int idCliente)
{
    var cliente = domainService.ObterPorId(idCliente);
    domainService.Excluir(cliente);
}

public List<ClienteConsultaModel> Consultar()
{
    var lista = domainService.Consultar();
    return Mapper.Map<List<ClienteConsultaModel>>(lista);
}

public ClienteConsultaModel ObterPorId(int idCliente)
{
    var cliente = domainService.ObterPorId(idCliente);
    return Mapper.Map<ClienteConsultaModel>(cliente);
}
}
}

```

/Application/Services/**PlanoApplicationService.cs**

```

using AutoMapper;
using Projeto.Application.Contracts;
using Projeto.Application.Models;
using Projeto.Domain.Contracts.Services;
using Projeto.Domain.Entities;
using System;
using System.Collections.Generic;
using System.Text;

namespace Projeto.Application.Services
{
    public class PlanoApplicationService : IPlanoApplicationService
    {
        //atributo
        private readonly IPlanoDomainService domainService;

        //construtor para injeção de dependência
        public PlanoApplicationService(IPlanoDomainService domainService)
        {
            this.domainService = domainService;
        }

        public void Cadastrar(PlanoCadastroModel model)
        {
            var plano = Mapper.Map<PlanoEntity>(model);

```



```
        domainService.Cadastrar(plano);
    }

    public void Atualizar(PlanoEdicaoModel model)
    {
        var plano = Mapper.Map<PlanoEntity>(model);
        domainService.Atualizar(plano);
    }

    public void Excluir(int idPlano)
    {
        var plano = domainService.ObterPorId(idPlano);
        domainService.Excluir(plano);
    }

    public List<PlanoConsultaModel> Consultar()
    {
        var lista = domainService.Consultar();
        return Mapper.Map<List<PlanoConsultaModel>>(lista);
    }

    public PlanoConsultaModel ObterPorId(int idPlano)
    {
        var plano = domainService.ObterPorId(idPlano);
        return Mapper.Map<PlanoConsultaModel>(plano);
    }
}
}
```

Para que possamos fazer com que o AutoMapper funcione na camada de apresentação, precisamos criar a classe abaixo:

Esta classe será registrada no Startup.cs

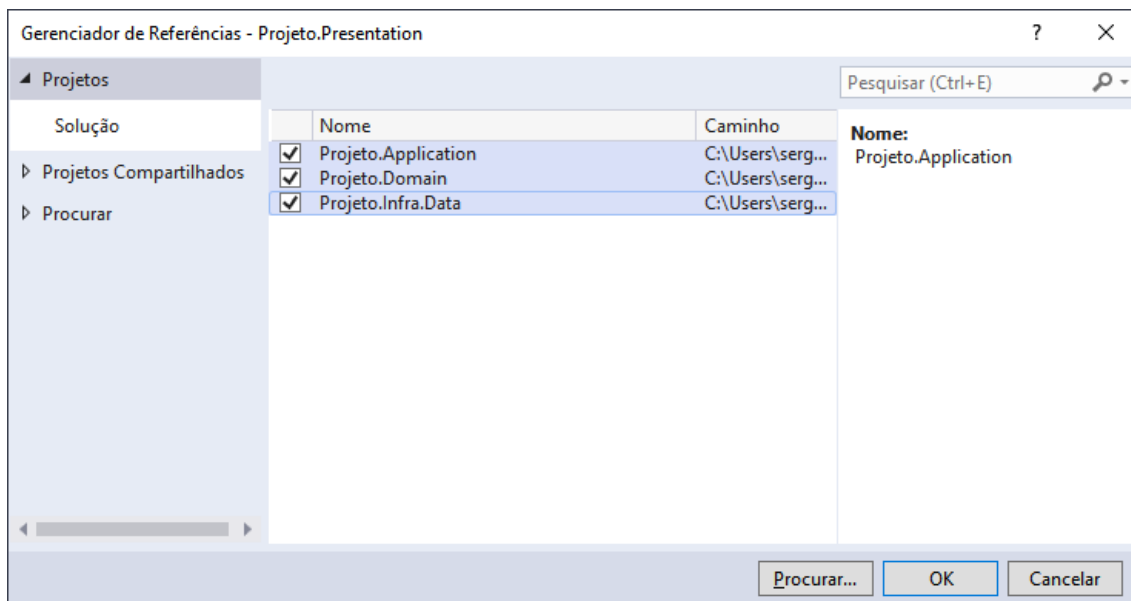
Classe de configuração do AutoMapper

/Mappings/AutoMapperConfig.cs

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

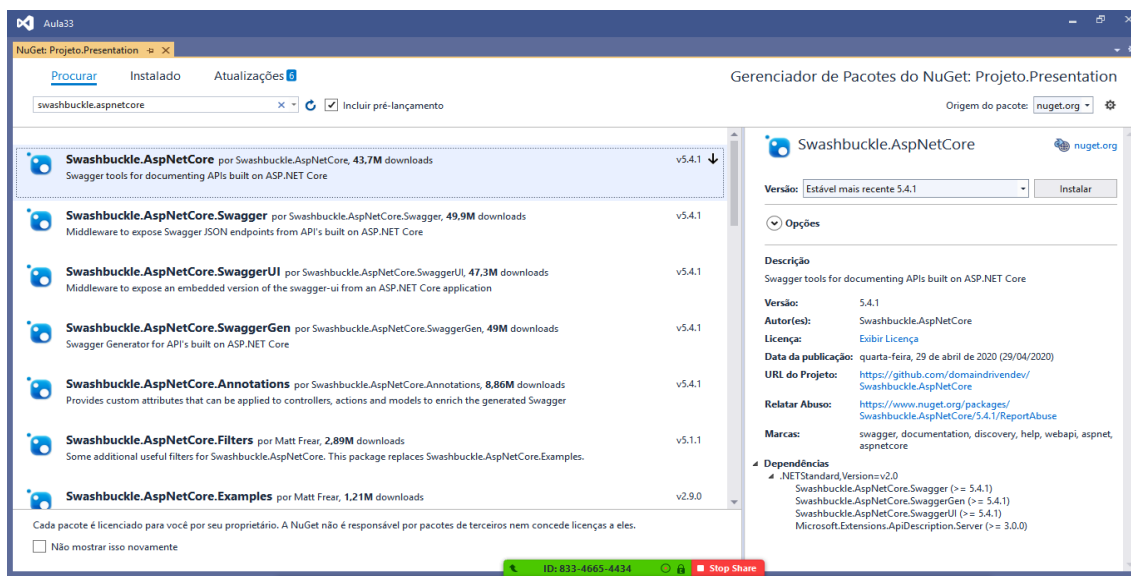
namespace Projeto.Application.Mappings
{
    public class AutoMapperConfig
    {
        public static void Register()
        {
            Mapper.Initialize(
                map =>
                {
                    map.AddProfile<EntityToModelMapping>();
                    map.AddProfile<ModelToEntityMapping>();
                });
        }
    }
}
```

Adicionando referência no projeto Presentation para os demais:

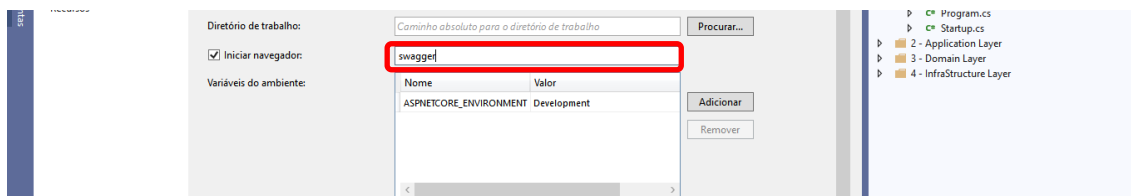


Instalando o Swagger: Documentação da API

Swashbuckle.aspnetcore



Modificando a página inicial do projeto para: swagger



Startup.cs

- Configuração do EntityFramework
- Configuração do AutoMapper
- Configuração do Swagger
- Mapeamento da injeção de dependência

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

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

        public IConfiguration Configuration { get; }

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

            #region EntityFramework

            //configurar o uso do EntityFramework na aplicação
            //injeção de dependencia na classe DataContext de forma a enviar
            //o caminho da string de conexão do banco de dados
            services.AddDbContext<DataContext>(
                options => options.UseSqlServer(
                    Configuration.GetConnectionString("ProjetoDDD")));

            #endregion
        }
    }
}
```

```
#region AutoMapper

AutoMapperConfig.Register();

#endregion

#region Swagger

services.AddSwaggerGen(
    c =>
    {
        c.SwaggerDoc("v1", new OpenApiInfo
        {
            Title = "Sistema de Controle de Clientes e Planos",
            Description = "Projeto desenvolvido  
em DDD com API e EntityFramework",
            Version = "v1",
            Contact = new OpenApiContact
            {
                Name = "COTI Informática",
                Url = new Uri("http://www.cotiinformatica.com.br/"),
                Email = "contato@cotiinformatica.com.br"
            }
        });
    });

#endregion

#region Injeção de dependência

services.AddTransient<IPlanoApplicationService,
    PlanoApplicationService>();
services.AddTransient<IClienteApplicationService,
    ClienteApplicationService>();

services.AddTransient<IPlanoDomainService, PlanoDomainService>();
services.AddTransient<IClienteDomainService, ClienteDomainService>();

services.AddTransient<IPlanoRepository, PlanoRepository>();
services.AddTransient<IClienteRepository, ClienteRepository>();
services.AddTransient<IUnitOfWork, UnitOfWork>();

#endregion

}

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

    #region Swagger

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

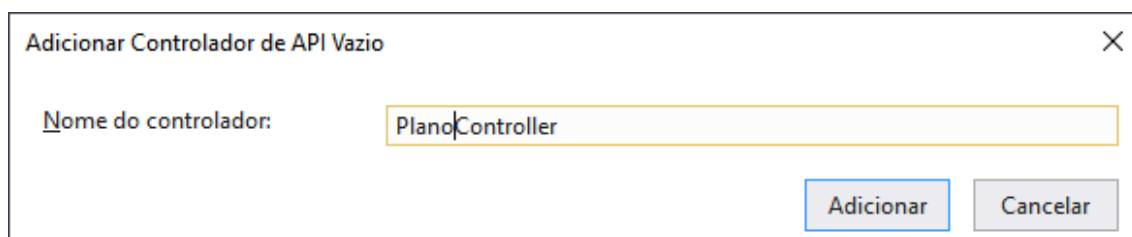
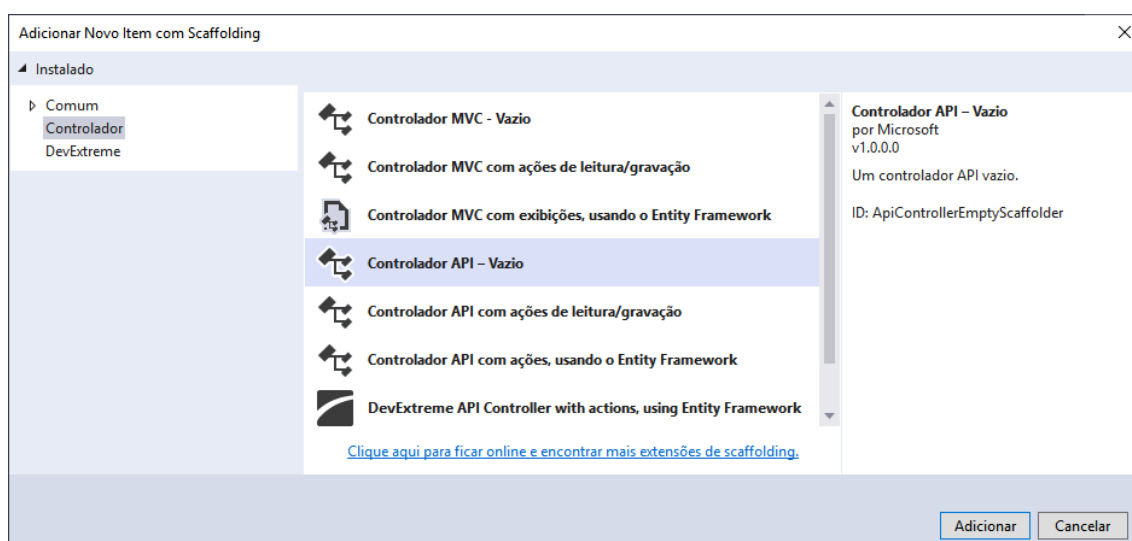
```
});

#endregion

app.UseMvc();
}
}
}
```

- Criando os serviços da API:

/Controllers/PlanoController.cs



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

namespace Projeto.Presentation.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class PlanoController : ControllerBase
    {
        //atributo
        private readonly IPlanoApplicationService service;
```

```
//construtor para injeção de dependência
public PlanoController(IPlanoApplicationService service)
{
    this.service = service;
}

[HttpPost]
public IActionResult Post(PlanoCadastroModel model)
{
    if(ModelState.IsValid)
    {
        try
        {
            service.Cadastrar(model);
            var result = new { message = "Plano cadastrado  
com sucesso" };
            return Ok(result);
        }
        catch(Exception e)
        {
            return StatusCode(500, e.Message);
        }
    }
    else
    {
        return BadRequest(); //HTTP 400
    }
}

[HttpPut]
public IActionResult Put(PlanoEdicaoModel model)
{
    if (ModelState.IsValid)
    {
        try
        {
            service.Atualizar(model);
            var result = new { message = "Plano atualizado  
com sucesso" };
            return Ok(result);
        }
        catch (Exception e)
        {
            return StatusCode(500, e.Message);
        }
    }
    else
    {
        return BadRequest(); //HTTP 400
    }
}

[HttpDelete("{id}")]
public IActionResult Delete(int id)
{
    try
    {
        service.Excluir(id);
        var result = new { message = "Plano excluído com sucesso" };
        return Ok(result);
    }
}
```

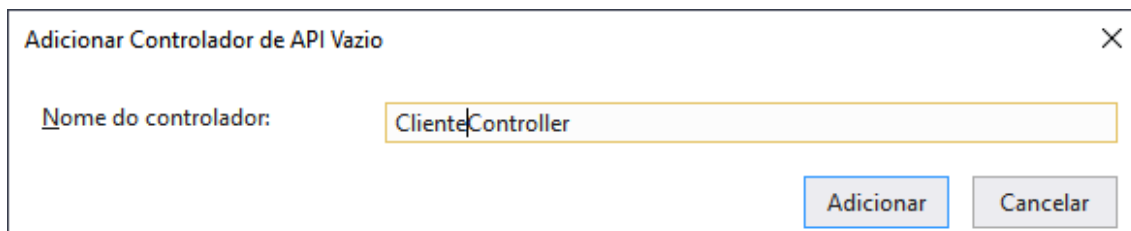
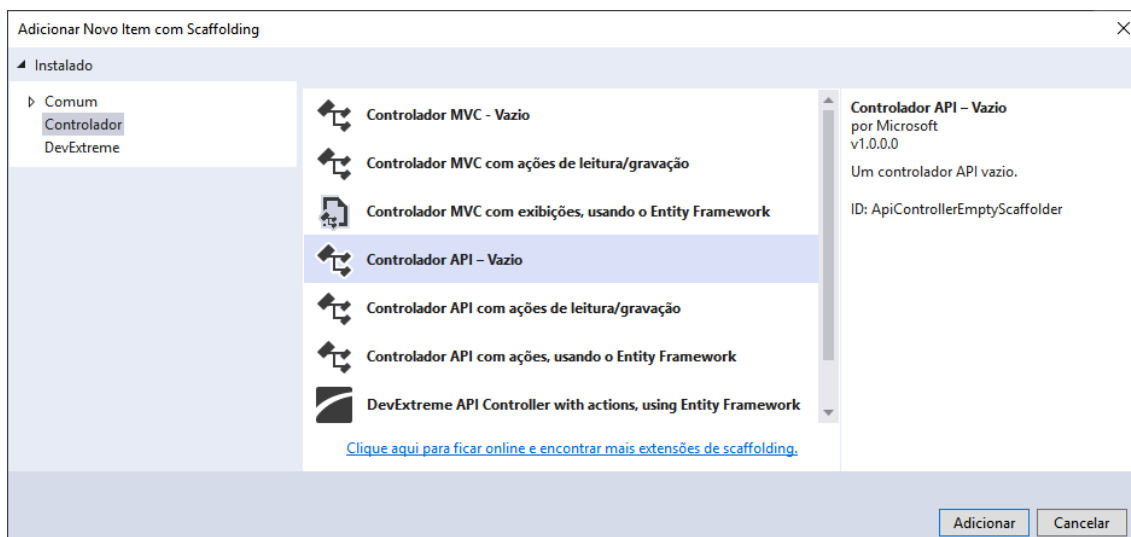
```

        catch (Exception e)
        {
            return StatusCode(500, e.Message);
        }
    }

    [HttpGet]
    public IActionResult GetAll()
    {
        try
        {
            return Ok(service.Consultar());
        }
        catch (Exception e)
        {
            return StatusCode(500, e.Message);
        }
    }

    [HttpGet("{id}")]
    public IActionResult GetById(int id)
    {
        try
        {
            return Ok(service.ObterPorId(id));
        }
        catch (Exception e)
        {
            return StatusCode(500, e.Message);
        }
    }
}

```



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

namespace Projeto.Presentation.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class ClienteController : ControllerBase
    {
        //atributo
        private readonly IClienteApplicationService service;

        //construtor para injeção de dependência
        public ClienteController(IClienteApplicationService service)
        {
            this.service = service;
        }

        [HttpPost]
        public IActionResult Post(ClienteCadastroModel model)
        {
            if (ModelState.IsValid)
            {
                try
                {
                    service.Cadastrar(model);
                    var result = new { message = "Cliente cadastrado  
com sucesso" };
                    return Ok(result);
                }
                catch (Exception e)
                {
                    return StatusCode(500, e.Message);
                }
            }
            else
            {
                return BadRequest(); //HTTP 400
            }
        }

        [HttpPut]
        public IActionResult Put(ClienteEdicaoModel model)
        {
            if (ModelState.IsValid)
            {

```



```
        try
        {
            service.Atualizar(model);
            var result = new { message = "Cliente atualizado  
com sucesso" };
            return Ok(result);
        }
        catch (Exception e)
        {
            return StatusCode(500, e.Message);
        }
    }
    else
    {
        return BadRequest(); //HTTP 400
    }
}

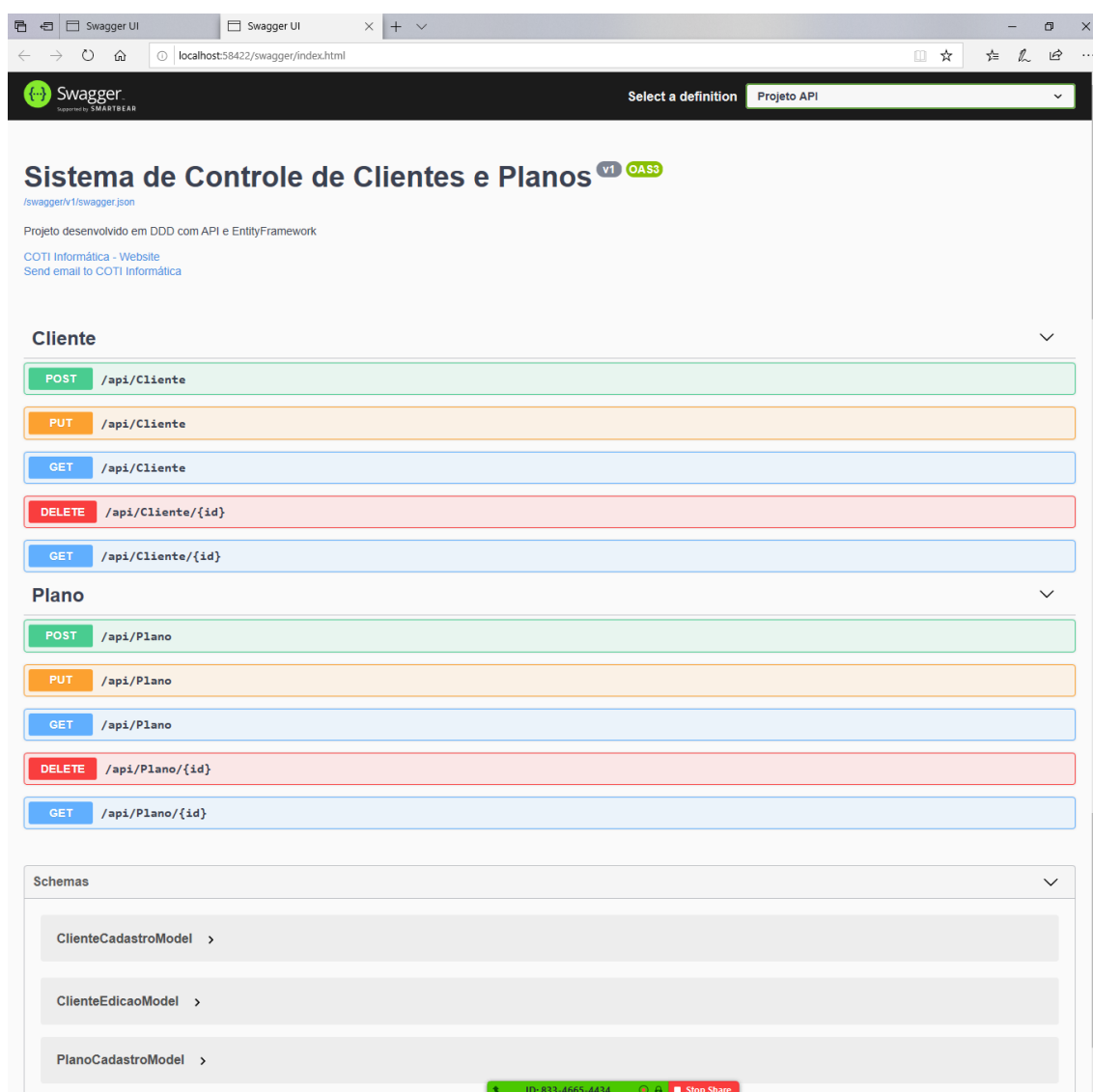
[HttpDelete("{id}")]
public IActionResult Delete(int id)
{
    try
    {
        {
            service.Excluir(id);
            var result = new { message = "Cliente excluído com sucesso" };
            return Ok(result);
        }
        catch (Exception e)
        {
            return StatusCode(500, e.Message);
        }
    }
}

[HttpGet]
public IActionResult GetAll()
{
    try
    {
        {
            return Ok(service.Consultar());
        }
        catch (Exception e)
        {
            return StatusCode(500, e.Message);
        }
    }
}
```

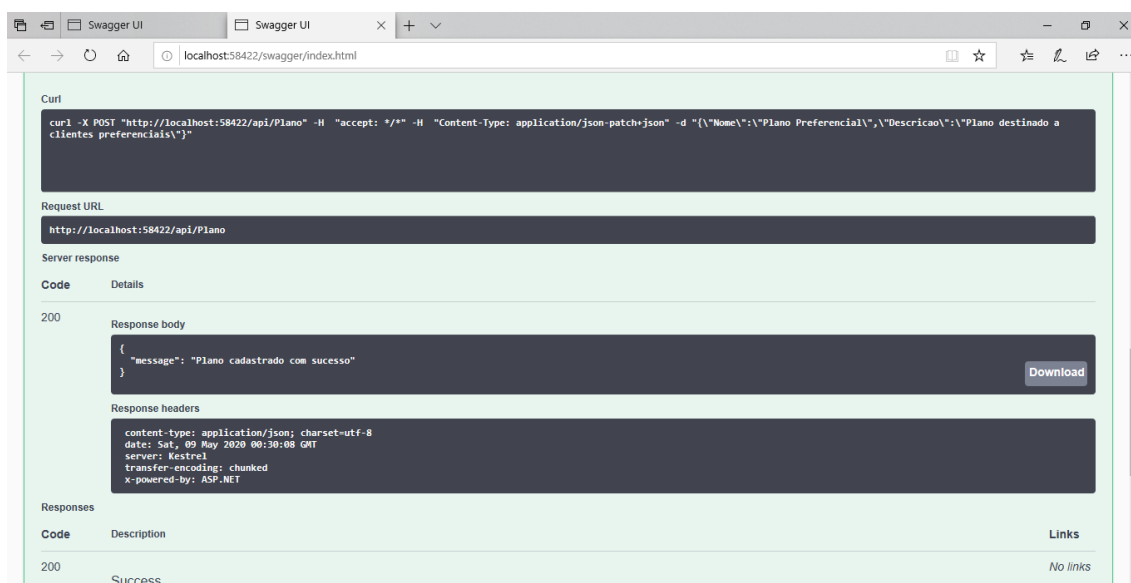
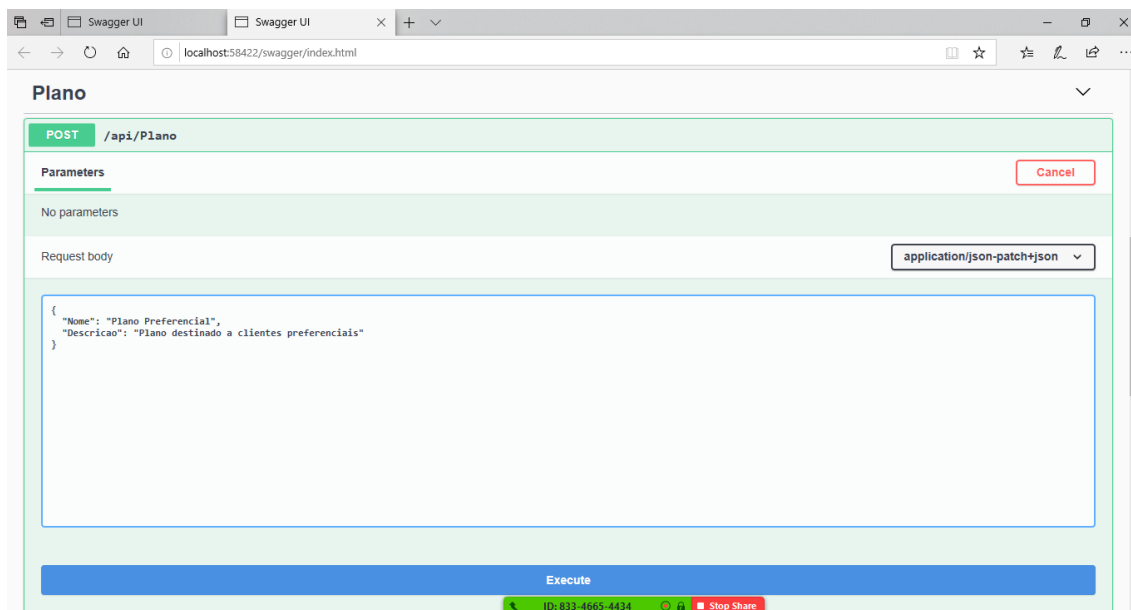
```
[HttpGet("{id}")]
public IActionResult GetById(int id)
{
    try
    {
        return Ok(service.ObterPorId(id));
    }
    catch (Exception e)
    {
        return StatusCode(500, e.Message);
    }
}
```

Executando:

<http://localhost:58422/swagger>



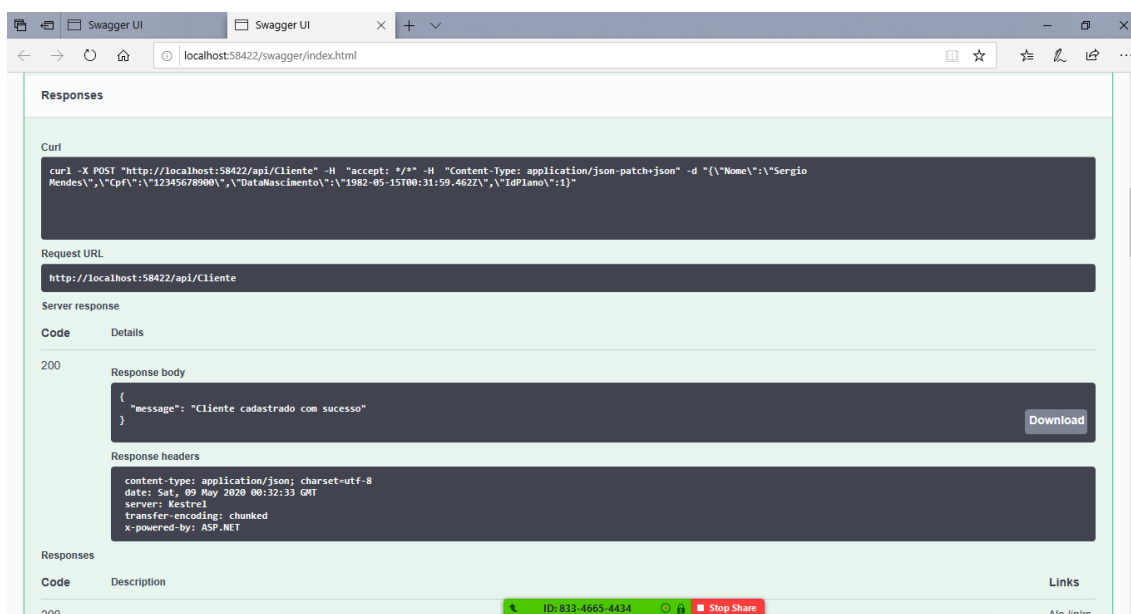
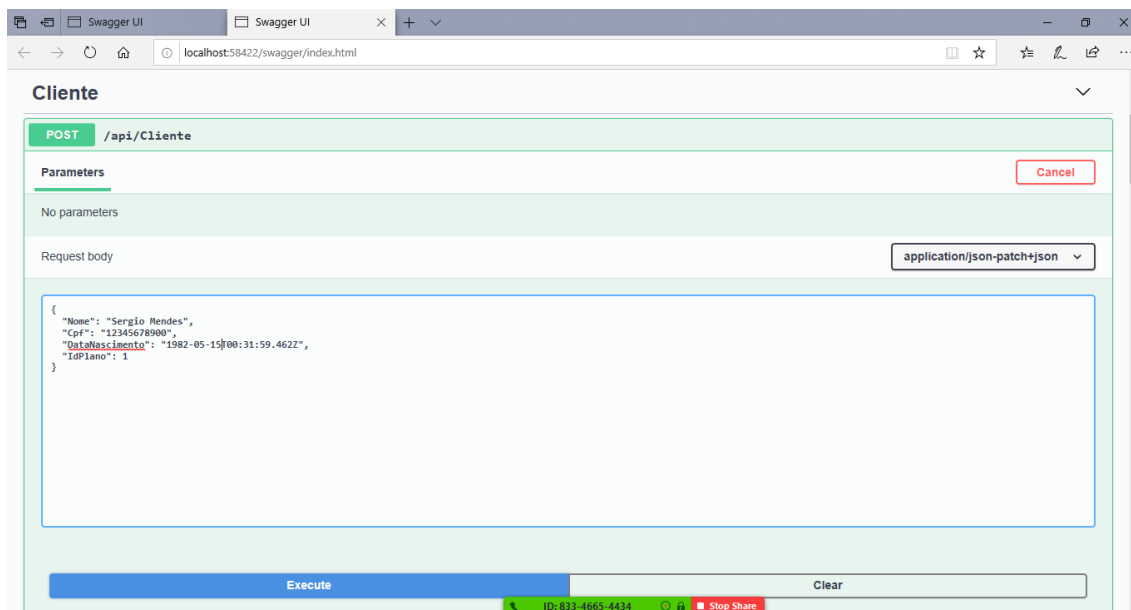
Testando os serviços:



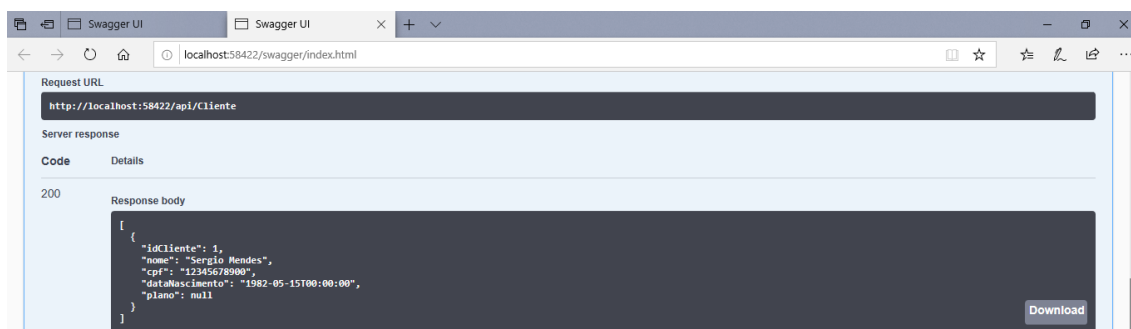
Consulta de Planos (GET)



Cadastrando clientes:



Consultando os clientes: (GET)



Sobrescrita dos métodos de consulta de cliente na camada de InfraEstrutura de repositório:

/Repositories/ClienteRepository.cs

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

namespace Projeto.Infra.Data.Repositories
{
    public class ClienteRepository
        : BaseRepository<ClienteEntity>, IClienteRepository
    {
        private readonly DataContext context;

        public ClienteRepository(DataContext context)
            : base(context) //construtor da superclasse
        {
            this.context = context;
        }

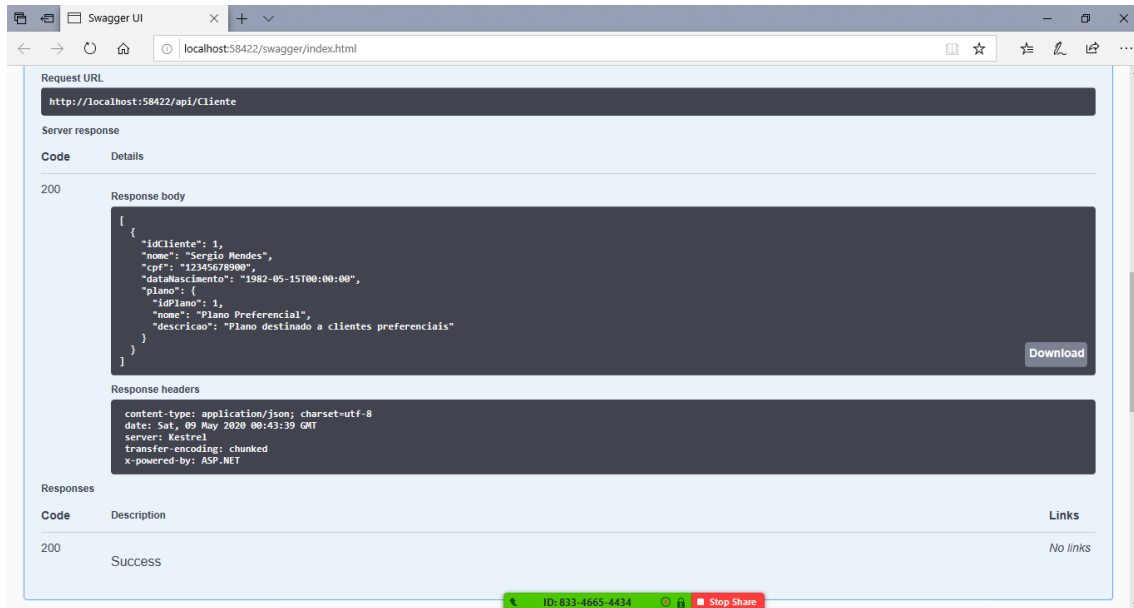
        public override List<ClienteEntity> GetAll()
        {
            return context.Cliente
                .Include(c => c.Plano) //INNER JOIN
                .ToList();
        }

        public override List<ClienteEntity>
            GetAll(Func<ClienteEntity, bool> where)
        {
            return context.Cliente
                .Include(c => c.Plano) //INNER JOIN
                .Where(where)
                .ToList();
        }

        public override ClienteEntity Get(Func<ClienteEntity, bool> where)
        {
            return context.Cliente
                .Include(c => c.Plano) //INNER JOIN
                .FirstOrDefault(where);
        }

        public override ClienteEntity GetById(int id)
        {
            return context.Cliente
                .Include(c => c.Plano) //INNER JOIN
                .FirstOrDefault(c => c.IdCliente == id);
        }
    }
}
```

Executando a consulta:



Request URL
`http://localhost:58422/api/Cliente`

Server response

Code	Details
200	<p>Response body</p> <pre>{ "idCliente": 1, "nome": "Sergio Mendes", "cpf": "12345678900", "dataNascimento": "1982-05-15T00:00:00", "plano": { "idPlano": 1, "nome": "Plano Preferencial", "descricao": "Plano destinado a clientes preferenciais" } }</pre> <p>Download</p> <p>Response headers</p> <pre>content-type: application/json; charset=utf-8 date: Sat, 09 May 2020 00:43:39 GMT server: Kestrel transfer-encoding: chunked x-powered-by: ASP.NET</pre>

Responses

Code	Description	Links
200	Success	No links

ID: 833-4665-4434 Stop Share