

Criando uma regra de negócio para não permitir que clientes com o mesmo email sejam cadastrados na API

Repositório:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.SqlClient; //importando
using System.Configuration; //importando
using Projeto.Entities; //importando
using Dapper; //importando

namespace Projeto.DAL
{
    public class ClienteRepository
    {
        private string connectionString;

        public ClienteRepository()
        {
            connectionString = ConfigurationManager
                .ConnectionStrings["projeto"].ConnectionString;
        }

        public void Insert(Cliente cliente)
        {
            using (var conn = new SqlConnection(connectionString))
            {
                string query = "insert into Cliente(Nome, Email, DataCriacao) "
                    + "values(@Nome, @Email, GETDATE())";

                conn.Execute(query, cliente);
            }
        }

        public void Update(Cliente cliente)
        {
            using (var conn = new SqlConnection(connectionString))
            {
                string query = "update Cliente set Nome = @Nome, Email = @Email "
                    + "where IdCliente = @IdCliente";

                conn.Execute(query, cliente);
            }
        }

        public void Delete(int id)
        {
            using (var conn = new SqlConnection(connectionString))
            {
                string query = "delete from Cliente
                    where IdCliente = @IdCliente";

                conn.Execute(query, new { IdCliente = id });
            }
        }
    }
}
```

```

public List<Cliente> SelectAll()
{
    using (var conn = new SqlConnection(connectionString))
    {
        string query = "select * from Cliente";

        return conn.Query<Cliente>(query).ToList();
    }
}

public Cliente SelectById(int id)
{
    using (var conn = new SqlConnection(connectionString))
    {
        string query = "select * from Cliente
                        where IdCliente = @IdCliente";

        return conn.QuerySingleOrDefault<Cliente>(query,
            new { IdCliente = id });
    }
}

public bool HasEmail(string email)
{
    using (var conn = new SqlConnection(connectionString))
    {
        string query = "select count(Email) from Cliente "
            + "where Email = @Email";

        return conn.QuerySingleOrDefault<int>(query,
            new { Email = email }) > 0;
    }
}
}
}

```

Camada de Regras de Negócio:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Projeto.Entities;
using Projeto.DAL;

namespace Projeto.BLL
{
    public class ClienteBusiness
    {
        //atributo..
        private ClienteRepository repository;

        //construtor -> ctor + 2x[tab]
        public ClienteBusiness()
        {
            repository = new ClienteRepository();
        }
    }
}

```

```
//método para cadastrar cliente
public void CadastrarCliente(Cliente cliente)
{
    //verificar se o email informado já está cadastrado
    if(repository.HasEmail(cliente.Email))
    {
        throw new Exception($"O Email {cliente.Email}
                               já está cadastrado.");
    }
    else
    {
        repository.Insert(cliente);
    }
}

//método para atualizar cliente
public void AtualizarCliente(Cliente cliente)
{
    repository.Update(cliente);
}

//método para excluir cliente
public void ExcluirCliente(int id)
{
    repository.Delete(id);
}

//método para listar todos os clientes
public List<Cliente> ConsultarTodos()
{
    return repository.SelectAll();
}

//método para consultar cliente por id
public Cliente ConsultarPorId(int id)
{
    return repository.SelectById(id);
}
}
```

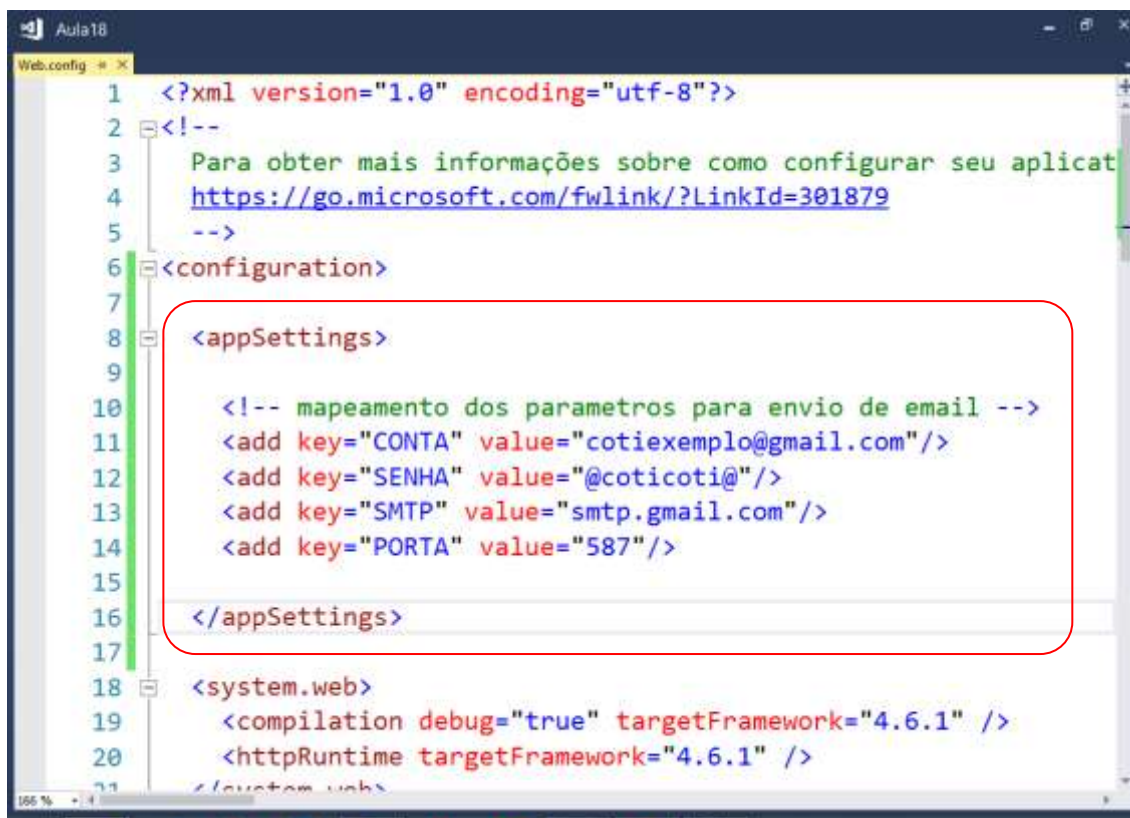
\Web.config.xml

Mapeando os parametros necessários
para realizar o envio do email.

<appSettings>

```
<!-- mapeamento dos parametros para envio de email -->
<add key="CONTA" value="cotiexemplo@gmail.com"/>
<add key="SENHA" value="@coticoti"/>
<add key="SMTP" value="smtp.gmail.com"/>
<add key="PORTA" value="587"/>
```

</appSettings>



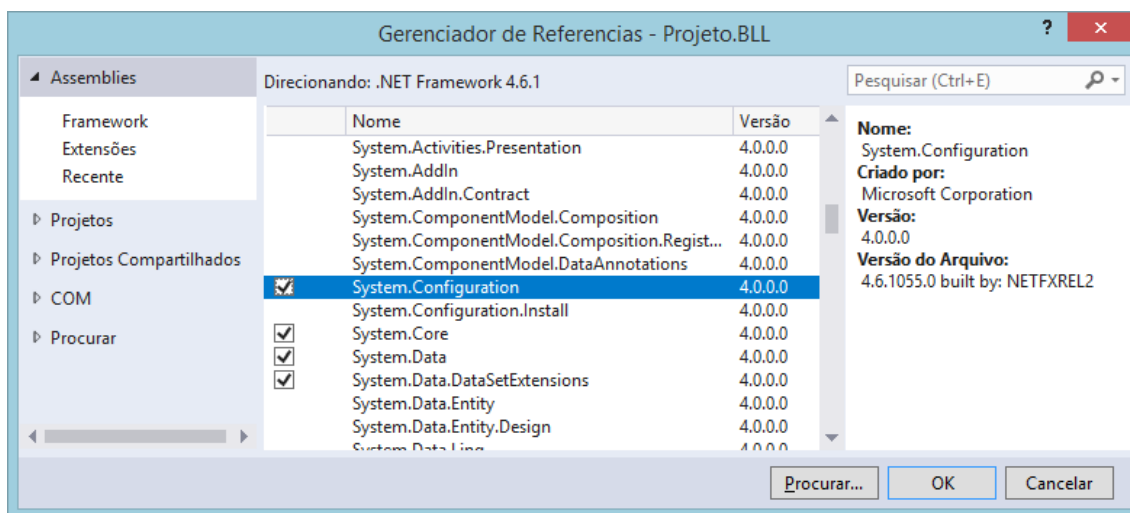
```

1 <?xml version="1.0" encoding="utf-8"?>
2 <!--
3 Para obter mais informações sobre como configurar seu aplicat
4 https://go.microsoft.com/fwlink/?LinkId=301879
5 -->
6 <configuration>
7
8 <appSettings>
9
10 <!-- mapeamento dos parametros para envio de email -->
11 <add key="CONTA" value="cotiexemplo@gmail.com"/>
12 <add key="SENHA" value="@coticoti@"/>
13 <add key="SMTP" value="smtp.gmail.com"/>
14 <add key="PORTA" value="587"/>
15
16 </appSettings>
17
18 <system.web>
19 <compilation debug="true" targetFramework="4.6.1" />
20 <httpRuntime targetFramework="4.6.1" />
21 </system.web>

```

Voltando na camada de regras de negócio:

Adicionando referencia para System.Configuration



```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Projeto.Entities;
using Projeto.DAL;
using System.Net;
using System.Net.Mail;
using System.Configuration;

```

```
namespace Projeto.BLL
{
    public class ClienteBusiness
    {
        //atributo..
        private ClienteRepository repository;

        //construtor -> ctor + 2x[tab]
        public ClienteBusiness()
        {
            repository = new ClienteRepository();
        }

        //método para cadastrar cliente
        public void CadastrarCliente(Cliente cliente)
        {
            //verificar se o email informado já está cadastrado
            if(repository.HasEmail(cliente.Email))
            {
                throw new Exception($"O Email {cliente.Email}
                                     já está cadastrado.");
            }
            else
            {
                repository.Insert(cliente);
                EnviarEmailDeBoasVindas(cliente);
            }
        }

        //método para atualizar cliente
        public void AtualizarCliente(Cliente cliente)
        {
            repository.Update(cliente);
        }

        //método para excluir cliente
        public void ExcluirCliente(int id)
        {
            repository.Delete(id);
        }

        //método para listar todos os clientes
        public List<Cliente> ConsultarTodos()
        {
            return repository.SelectAll();
        }

        //método para consultar cliente por id
        public Cliente ConsultarPorId(int id)
        {
            return repository.SelectById(id);
        }

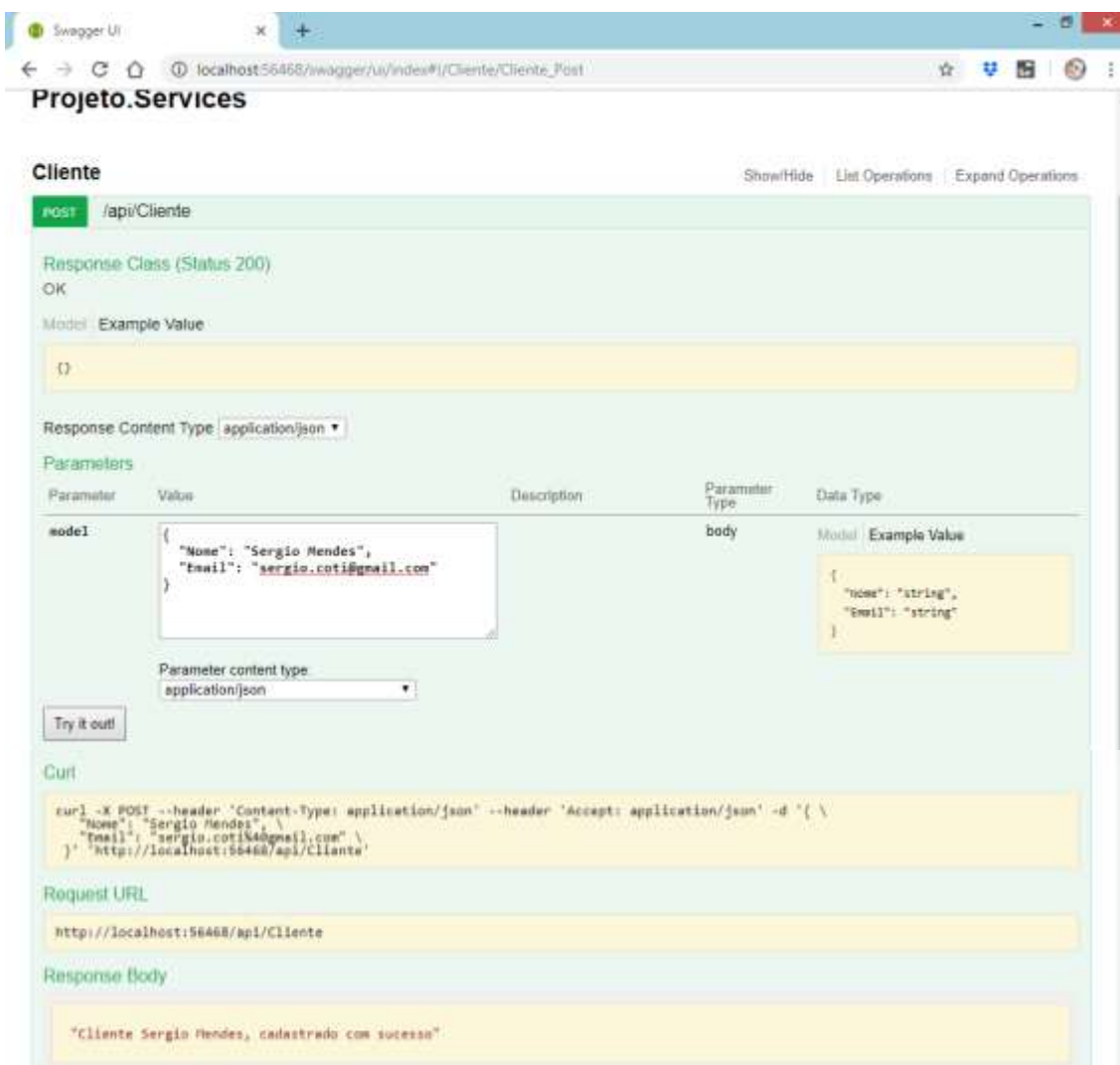
        //método para enviar um email de boas-vindas para o cliente
        private void EnviarEmailDeBoasVindas(Cliente cliente)
        {
            //capturando os dados mapeados no web.config.xml
            string conta = ConfigurationManager.AppSettings["CONTA"];
            string senha = ConfigurationManager.AppSettings["SENHA"];
            string smtp = ConfigurationManager.AppSettings["SMTP"];
            string porta = ConfigurationManager.AppSettings["PORTA"];
```

```
//passo 1) criando o email
MailMessage message = new MailMessage(conta, cliente.Email);

message.IsBodyHtml = true;
message.Subject = "Conta de cliente cadastrada com sucesso!";
message.Body = $"Seja bem vindo <strong>{cliente.Nome}</strong>"
    + "<br><br>"
    + $"Sua conta de cliente foi criada com sucesso"
    + "<br><br>"
    + $"Att<br>Sistema de Controle de Clientes";

//passo 2) enviando o email
SmtpClient client = new SmtpClient(smtp, int.Parse(porta));
client.EnableSsl = true; //habilitar criptografia do email
client.Credentials = new NetworkCredential(conta, senha);
client.Send(message); //enviando o email!
    }
}
}
```

Testando na API:



Swagger UI

localhost:56468/swagger/ui/index#/Cliente/Cliente_Post

Projeto.Services

Cliente

POST /api/Cliente

Response Class (Status 200)
OK

Model Example Value

```
{}
```

Response Content Type: application/json

Parameters

Parameter	Value	Description	Parameter Type	Data Type
model	{ "Nome": "Sergio Mendes", "Email": "sergio.coti@gmail.com" }		body	Model Example Value

Parameter content type: application/json

Try it out!

Curl

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' -d '{  
  "Nome": "Sergio Mendes",  
  "Email": "sergio.coti@gmail.com"  
}' "http://localhost:56468/api/Cliente"
```

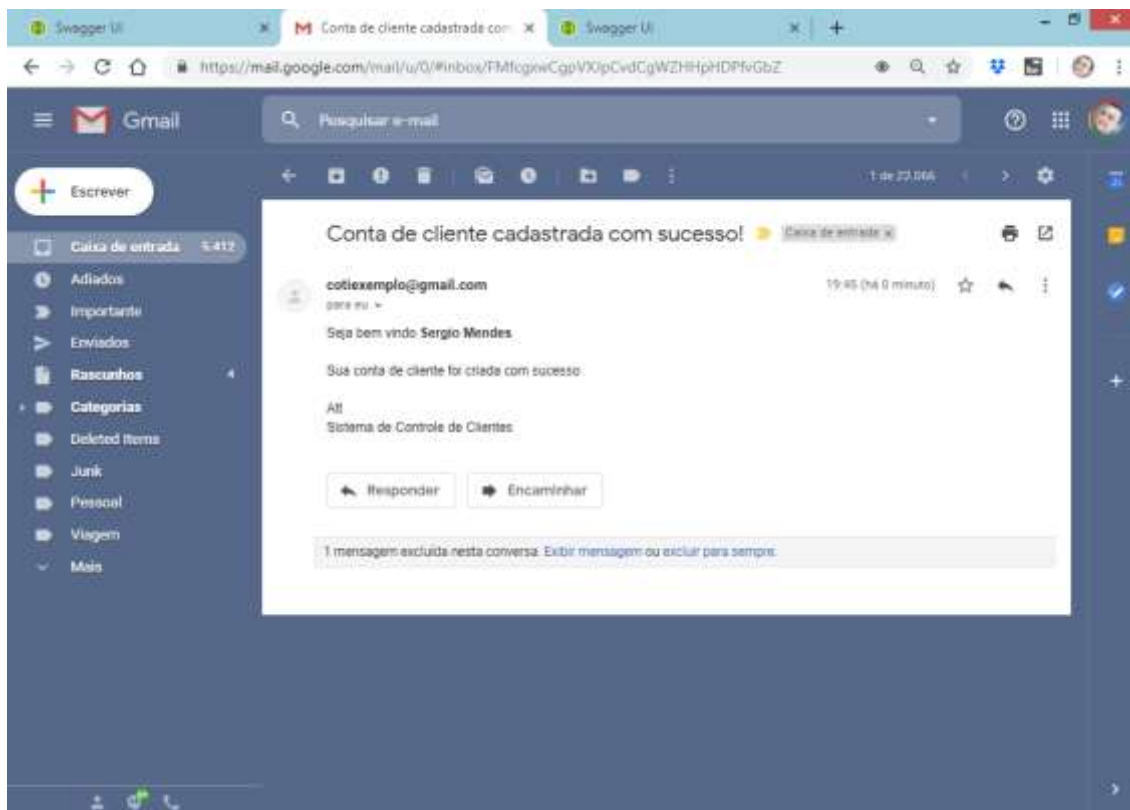
Request URL

http://localhost:56468/api/Cliente

Response Body

```
"Cliente Sergio Mendes, cadastrado com sucesso!"
```

Email recebido:



Criando os demais métodos da API:

Atualizar Cliente

/Controllers/ClienteController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Net.Http;
using System.Web.Http;
using AutoMapper; //importando
using Projeto.BLL; //importando
using Projeto.Entities; //importando
using Projeto.Services.Models; //importando

namespace Projeto.Services.Controllers
{
    [RoutePrefix("api/Cliente")]
    public class ClienteController : ApiController
    {
        private ClienteBusiness business;

        public ClienteController()
        {
            business = new ClienteBusiness();
        }
    }
}
```

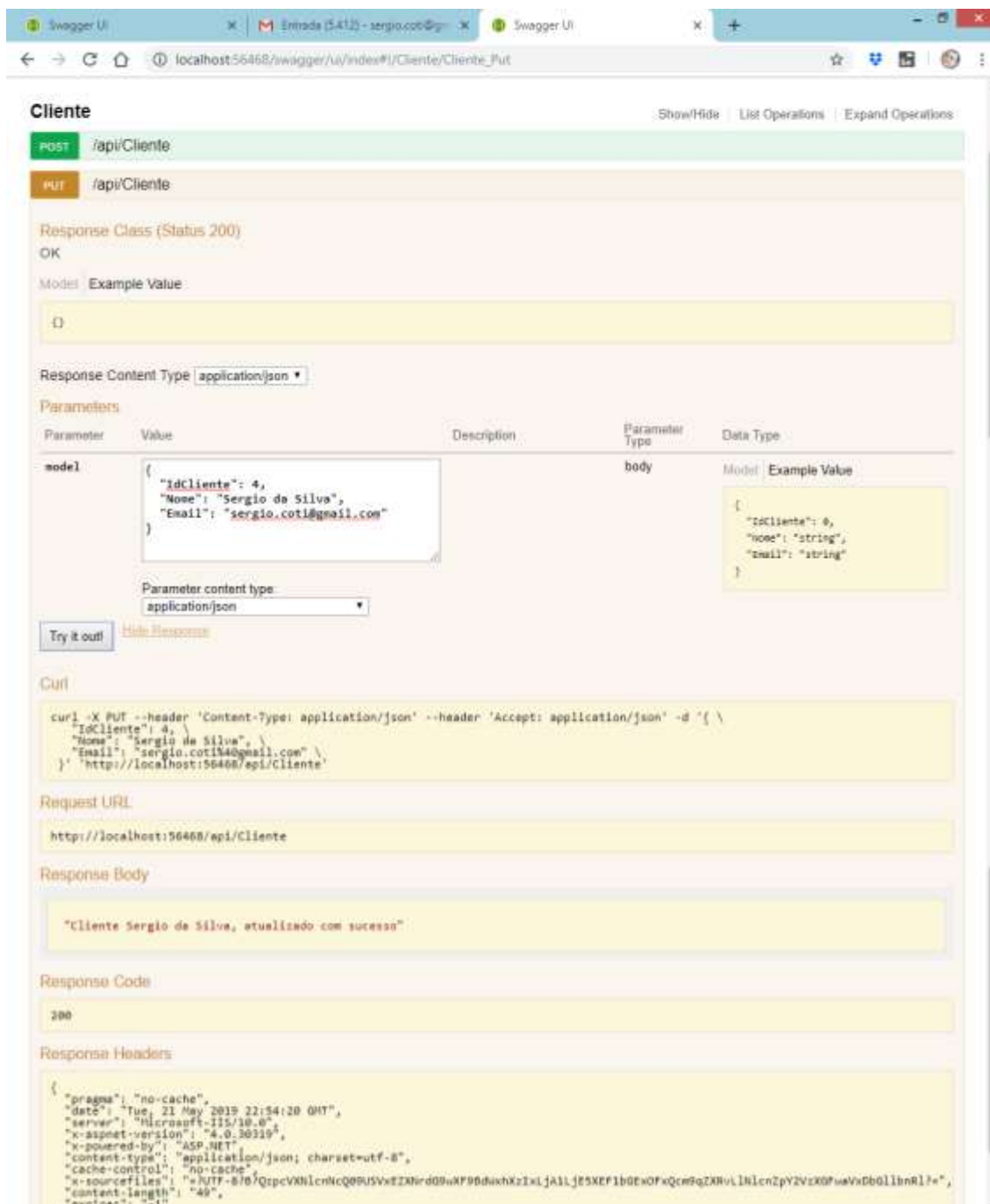
```
[HttpPost]
public HttpResponseMessage Post(ClienteCadastroViewModel model)
{
    if(ModelState.IsValid)
    {
        try
        {
            var cliente = Mapper.Map<Cliente>(model);
            business.CadastrarCliente(cliente);

            return Request.CreateResponse(HttpStatusCode.OK,
                $"Cliente {cliente.Nome}, cadastrado com sucesso");
        }
        catch(Exception e)
        {
            //retornar um status de erro HTTP 500 (InternalServerError)
            return Request.CreateResponse
                (HttpStatusCode.InternalServerError,
                "Erro interno de servidor: " + e.Message);
        }
    }
    else
    {
        //retornar um status de erro HTTP 400 (BadRequest)
        return Request.CreateResponse(HttpStatusCode.BadRequest,
            "Ocorreram erros de validação.");
    }
}

[HttpPut]
public HttpResponseMessage Put(ClienteEdicaoViewModel model)
{
    if (ModelState.IsValid)
    {
        try
        {
            var cliente = Mapper.Map<Cliente>(model);
            business.AtualizarCliente(cliente);

            //retornar um status de erro HTTP 200 (OK)
            return Request.CreateResponse(HttpStatusCode.OK,
                $"Cliente {cliente.Nome}, atualizado com sucesso");
        }
        catch (Exception e)
        {
            //retornar um status de erro HTTP 500 (InternalServerError)
            return Request.CreateResponse
                (HttpStatusCode.InternalServerError,
                "Erro interno de servidor: " + e.Message);
        }
    }
    else
    {
        //retornar um status de erro HTTP 400 (BadRequest)
        return Request.CreateResponse(HttpStatusCode.BadRequest,
            "Ocorreram erros de validação.");
    }
}
}
```


Testando:



The screenshot shows the Swagger UI interface for a web API. The selected operation is a PUT request to the endpoint `/api/Cliente`. The response class is `OK` with a status of 200. The response content type is `application/json`. The parameters section shows a `model` parameter with a JSON body: `{ "IdCliente": 4, "Nome": "Sergio da Silva", "Email": "sergio.coti@gmail.com" }`. The request URL is `http://localhost:56468/api/Cliente`. The response body is `"Cliente Sergio da Silva, atualizado com sucesso"`. The response code is `200`. The response headers include `pragma: no-cache`, `date: Tue, 21 May 2019 22:54:20 GMT`, `server: Microsoft-IIS/10.0`, `x-aspnet-version: 4.0.30319`, `x-powered-by: ASP.NET`, `content-type: application/json; charset=utf-8`, `cache-control: no-cache`, `x-sourcefiles: %3F7F-8767QzpcVXNlcncQ09USVx8Z200rd09uXf96daxhKzIkljallj25KEF1b0ExOFxQcm9qZXRvLnRlcjZpY2VzX0FwaVxDb0llbnRl?*`, `content-length: 40`, and `expires: -1`.

Excluir Cliente

/Controllers/ClienteController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Net.Http;
using System.Web.Http;
using AutoMapper; //importando
```

```
using Projeto.BLL; //importando
using Projeto.Entities; //importando
using Projeto.Services.Models; //importando

namespace Projeto.Services.Controllers
{
    [RoutePrefix("api/Cliente")]
    public class ClienteController : ApiController
    {
        //atributo
        private ClienteBusiness business;

        //construtor -> ctor + 2x[tab]
        public ClienteController()
        {
            business = new ClienteBusiness();
        }

        [HttpPost]
        public HttpResponseMessage Post(ClienteCadastroViewModel model)
        {
            if(ModelState.IsValid)
            {
                try
                {
                    var cliente = Mapper.Map<Cliente>(model);
                    business.CadastrarCliente(cliente);

                    //retornar um status de erro HTTP 200 (OK)
                    return Request.CreateResponse(HttpStatusCode.OK,
                        $"Cliente {cliente.Nome}, cadastrado com sucesso");
                }
                catch(Exception e)
                {
                    //retornar um status de erro HTTP 500 (InternalServerError)
                    return Request.CreateResponse
                        (HttpStatusCode.InternalServerError,
                            "Erro interno de servidor: " + e.Message);
                }
            }
            else
            {
                //retornar um status de erro HTTP 400 (BadRequest)
                return Request.CreateResponse(HttpStatusCode.BadRequest,
                    "Ocorreram erros de validação.");
            }
        }

        [HttpPut]
        public HttpResponseMessage Put(ClienteEdicaoViewModel model)
        {
            if (ModelState.IsValid)
            {
                try
                {
                    var cliente = Mapper.Map<Cliente>(model);
                    business.AtualizarCliente(cliente);

                    //retornar um status de erro HTTP 200 (OK)
                    return Request.CreateResponse(HttpStatusCode.OK,
```

```

        $"Cliente {cliente.Nome}, atualizado com sucesso");
    }
    catch (Exception e)
    {
        //retornar um status de erro HTTP 500 (InternalServerError)
        return Request.CreateResponse
            (HttpStatusCode.InternalServerError,
             "Erro interno de servidor: " + e.Message);
    }
}
else
{
    //retornar um status de erro HTTP 400 (BadRequest)
    return Request.CreateResponse(HttpStatusCode.BadRequest,
        "Ocorreram erros de validação.");
}
}

[HttpDelete]
public HttpResponseMessage Delete(int id)
{
    try
    {
        //excluindo o cliente
        business.ExcluirCliente(id);

        //retornando sucesso..
        return Request.CreateResponse(HttpStatusCode.OK,
            "Cliente excluído com sucesso.");
    }
    catch (Exception e)
    {
        return Request.CreateResponse(HttpStatusCode.InternalServerError,
            "Erro interno de servidor: " + e.Message);
    }
}
}
}
}

```

Principais verbos HTTP utilizados pelo padrão REST

GET	Request for one or more pieces of data
POST	Insert a record into the database
PUT	Update a record in the database
DELETE	Delete a record from the database

Testando:

Projeto.Services

Cliente

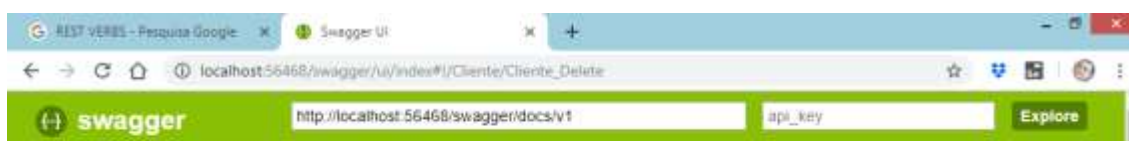
Show/Hide List Operations Expand Operations

POST /api/Cliente

PUT /api/Cliente

DELETE /api/Cliente/{id}

[BASE URL : API VERSION: v1]



Projeto.Services

Cliente

Show/Hide List Operations Expand Operations

POST /api/Cliente

PUT /api/Cliente

DELETE /api/Cliente/{id}

Response Class (Status 200)

OK

Model Example Value

```
{}
```

Response Content Type **application/json**

Parameters

Parameter	Value	Description	Parameter Type	Data Type
id	(required)		path	integer

Try it out!

Curl

```
curl -X DELETE --header 'Accept: application/json' 'http://localhost:56468/api/Cliente/4'
```

Request URL

```
http://localhost:56468/api/Cliente/4
```

Response Body

```
"Cliente excluido com sucesso."
```

Response Code

```
200
```

Response Headers

```
{
  "pragma": "no-cache",
  "date": "Tue, 21 May 2019 23:05:52 GMT",
  "server": "Microsoft-IIS/10.0",
  "x-aspnet-version": "4.0.30319",
  "x-powered-by": "ASP.NET",
  "content-type": "application/json; charset=utf-8",
  "cache-control": "no-cache",
  "x-sourcefiles": "I:/UTF-8/7/QzpcV00lcnRlcQ00USVxZD0rd09wXF90d09wZlxiZjAlljE5XEF1bGEtOFxQcn9qZXRvLlNlcnZpV2VsX0FwaVxDb0llbnRlX0Q=",
  "content-length": "45"
}
```

Consultando Clientes

/Controllers/ClienteController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Net.Http;
using System.Web.Http;
using AutoMapper; //importando
using Projeto.BLL; //importando
using Projeto.Entities; //importando
using Projeto.Services.Models; //importando

namespace Projeto.Services.Controllers
{
    [RoutePrefix("api/Cliente")]
    public class ClienteController : ApiController
    {
        //atributo
        private ClienteBusiness business;

        //construtor -> ctor + 2x[tab]
        public ClienteController()
        {
            business = new ClienteBusiness();
        }

        [HttpPost]
        public HttpResponseMessage Post(ClienteCadastroViewModel model)
        {
            if(ModelState.IsValid)
            {
                try
                {
                    var cliente = Mapper.Map<Cliente>(model);
                    business.CadastrarCliente(cliente);

                    //retornar um status de erro HTTP 200 (OK)
                    return Request.CreateResponse(HttpStatusCode.OK,
                        $"Cliente {cliente.Nome}, cadastrado com sucesso");
                }
                catch(Exception e)
                {
                    //retornar um status de erro HTTP 500 (InternalServerError)
                    return Request.CreateResponse
                        (HttpStatusCode.InternalServerError,
                        "Erro interno de servidor: " + e.Message);
                }
            }
            else
            {
                //retornar um status de erro HTTP 400 (BadRequest)
                return Request.CreateResponse(HttpStatusCode.BadRequest,
                    "Ocorreram erros de validação.");
            }
        }
    }
}
```

```
[HttpPut]
public HttpResponseMessage Put(ClienteEdicaoViewModel model)
{
    if (ModelState.IsValid)
    {
        try
        {
            var cliente = Mapper.Map<Cliente>(model);
            business.AtualizarCliente(cliente);

            //retornar um status de erro HTTP 200 (OK)
            return Request.CreateResponse(HttpStatusCode.OK,
                $"Cliente {cliente.Nome}, atualizado com sucesso");
        }
        catch (Exception e)
        {
            //retornar um status de erro HTTP 500 (InternalServerError)
            return Request.CreateResponse
                (HttpStatusCode.InternalServerError,
                "Erro interno de servidor: " + e.Message);
        }
    }
    else
    {
        //retornar um status de erro HTTP 400 (BadRequest)
        return Request.CreateResponse(HttpStatusCode.BadRequest,
            "Ocorreram erros de validação.");
    }
}

[HttpDelete]
public HttpResponseMessage Delete(int id)
{
    try
    {
        //excluindo o cliente
        business.ExcluirCliente(id);

        //retornando sucesso..
        return Request.CreateResponse(HttpStatusCode.OK,
            "Cliente excluído com sucesso.");
    }
    catch (Exception e)
    {
        return Request.CreateResponse(HttpStatusCode.InternalServerError,
            "Erro interno de servidor: " + e.Message);
    }
}

[HttpGet]
public HttpResponseMessage GetAll()
{
    try
    {
        var clientes = business.ConsultarTodos();
        var lista = Mapper.Map<List<ClienteConsultaViewModel>>(clientes);

        return Request.CreateResponse(HttpStatusCode.OK, lista);
    }
}
```

```

        catch(Exception e)
        {
            return Request.CreateResponse(HttpStatusCode.InternalServerError,
                "Erro interno de servidor: " + e.Message);
        }
    }

    [HttpGet]
    public HttpResponseMessage GetById(int id)
    {
        try
        {
            var cliente = business.ConsultarPorId(id);
            var model = Mapper.Map<ClienteConsultaViewModel>(cliente);

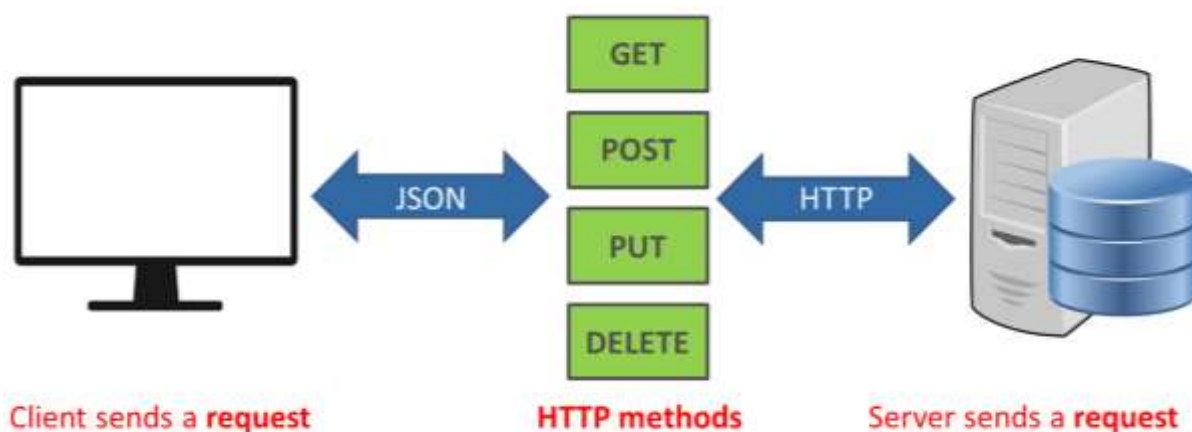
            return Request.CreateResponse(HttpStatusCode.OK, model);
        }
        catch(Exception e)
        {
            return Request.CreateResponse(HttpStatusCode.InternalServerError,
                "Erro interno de servidor: " + e.Message);
        }
    }
}

```

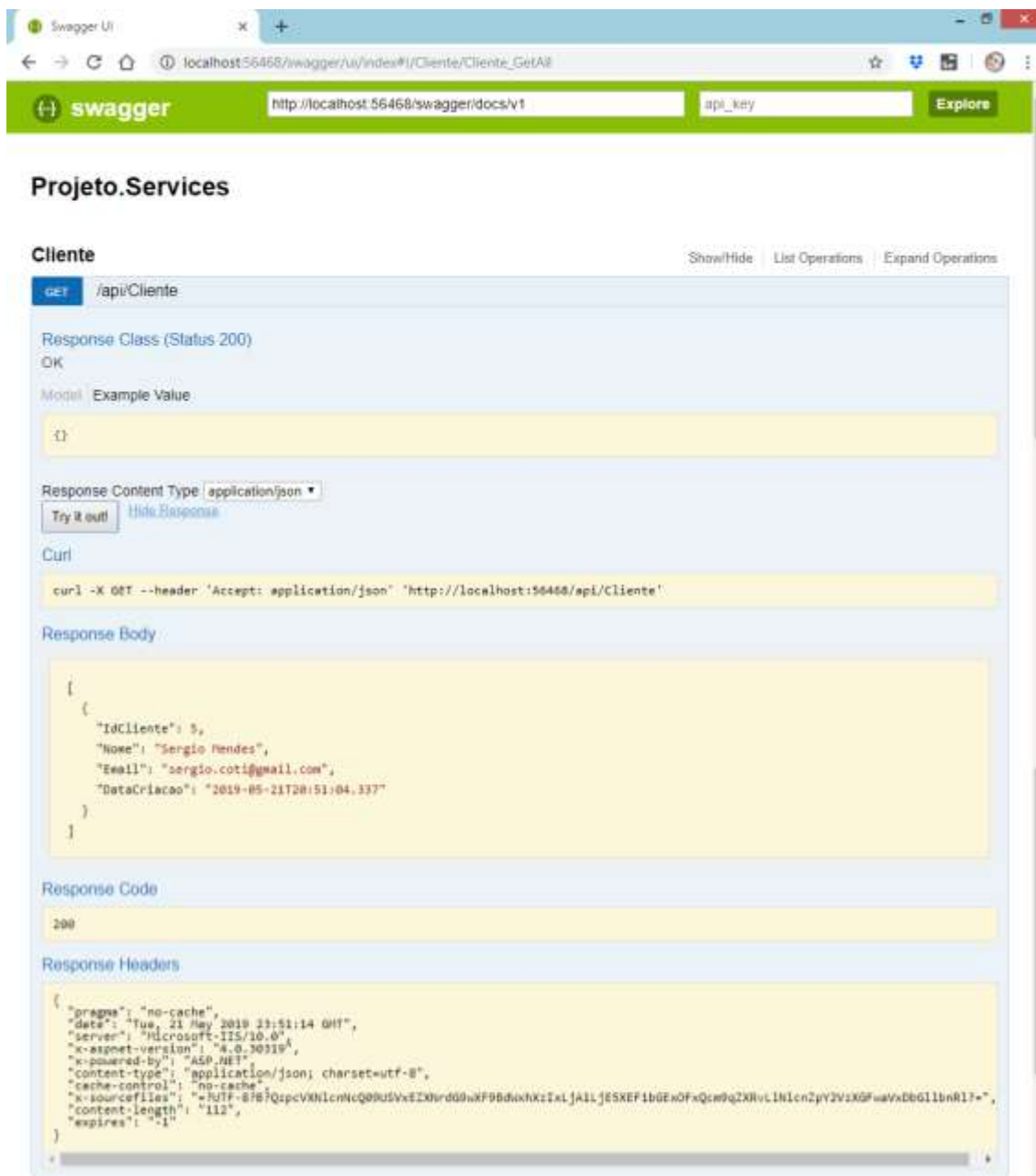
REST WEB API

O acrônimo API que provém do inglês Application Programming Interface (Em português, significa Interface de Programação de Aplicações), trata-se de um conjunto de rotinas e padrões estabelecidos e documentados por uma aplicação A, para que outras aplicações consigam utilizar as funcionalidades desta aplicação A, sem precisar conhecer detalhes da implementação do software.

Desta forma, entendemos que as APIs permitem uma interoperabilidade entre aplicações. Em outras palavras, a comunicação entre aplicações e entre os usuários.



Testando as consultas:



Swagger UI interface showing the GET /api/Cliente endpoint. The response body is a JSON object with the following structure:

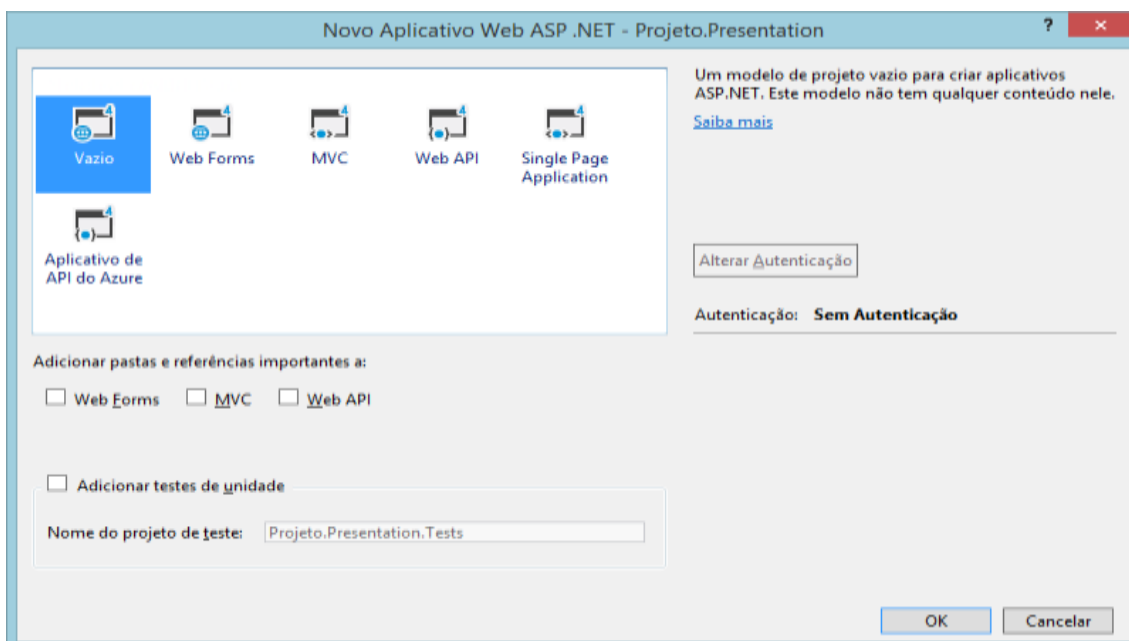
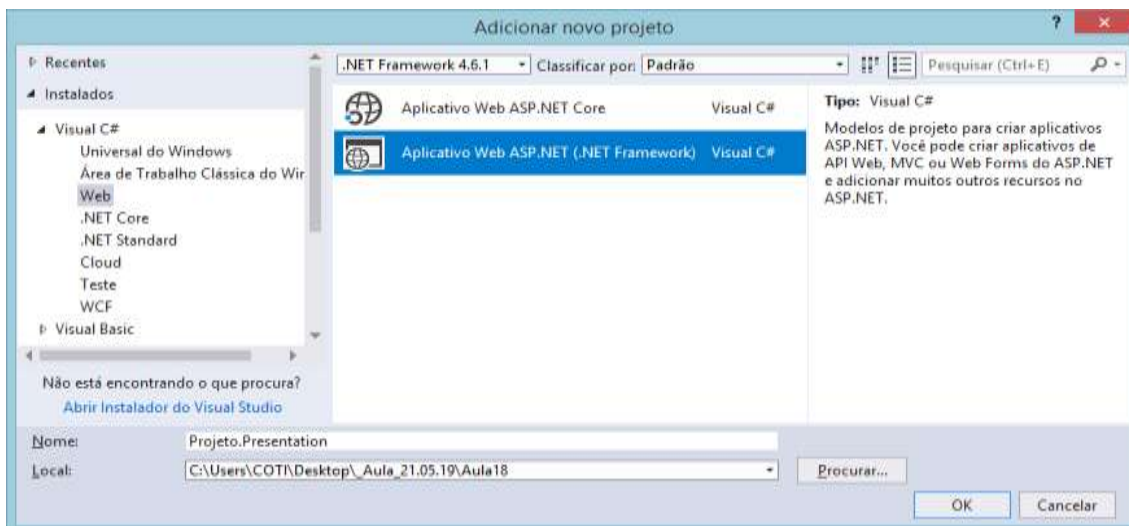
```
{
  "IdCliente": 5,
  "Nome": "Sergio Mendes",
  "Email": "sergio.coti@gmail.com",
  "DataCriacao": "2019-05-21T20:51:04.337"
}
```

JSON Response:

```
[
  {
    "IdCliente": 5,
    "Nome": "Sergio Mendes",
    "Email": "sergio.coti@gmail.com",
    "DataCriacao": "2019-05-21T20:51:04.337"
  }
]
```


1.1 - Apresentação

Projeto Asp.Net .NetFramework vazio

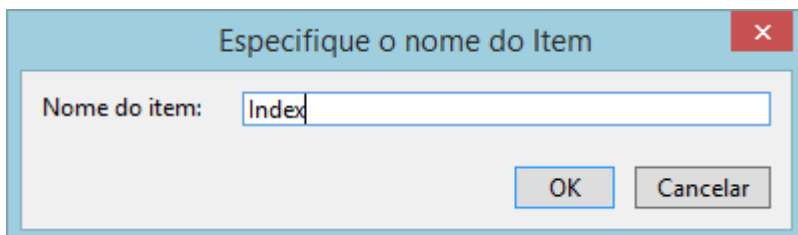


Instalando bootstrap:



Página inicial no projeto:

/Index.html



```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title>Projeto</title>

  <!-- folhas de estilo CSS -->
  <link href="Content/bootstrap.min.css" rel="stylesheet" />
</head>
<body>

  <div class="card card-body bg-dark">
    <h3 class="text-white">Sistema de Controle de Clientes</h3>
    <p class="text-white">
      Sistema Asp.Net WebAPI + Dapper + AngularJS
    </p>
  </div>

  <div class="container">

    <br/>

    <h4>Seja bem vindo ao Projeto</h4>
    Selecione a ação desejada:
    <hr/>

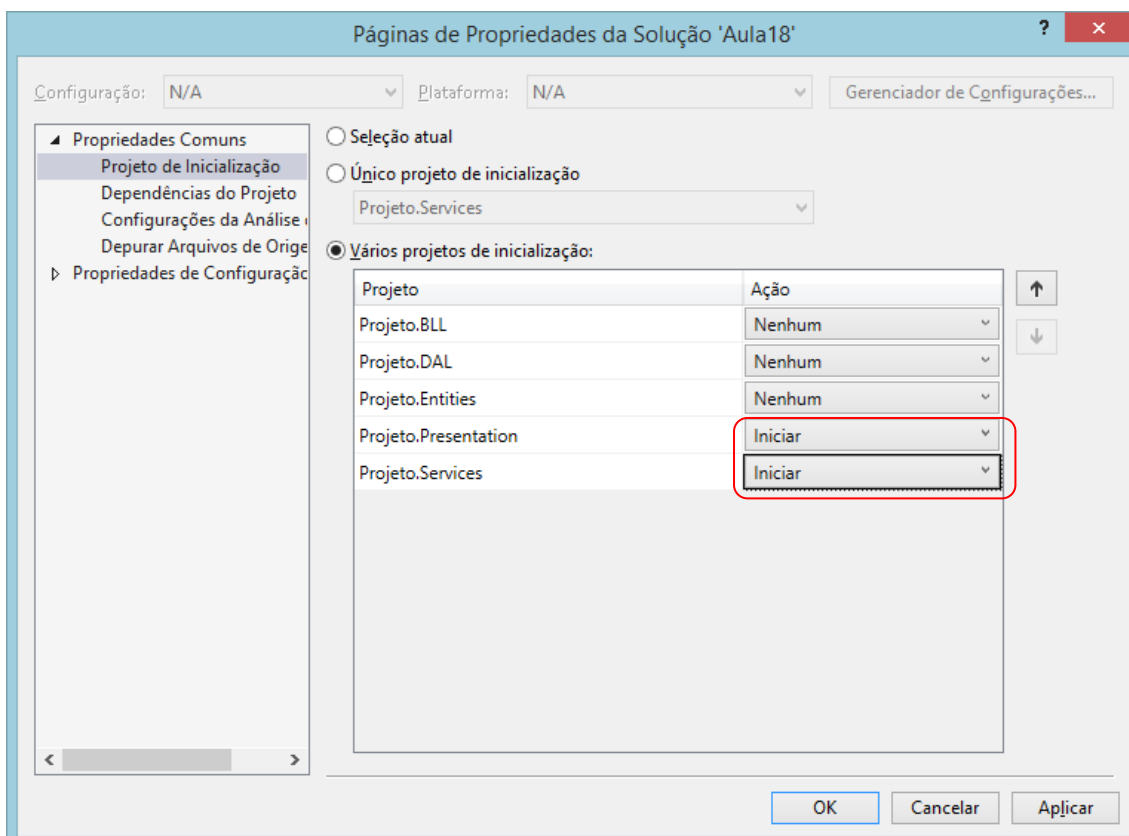
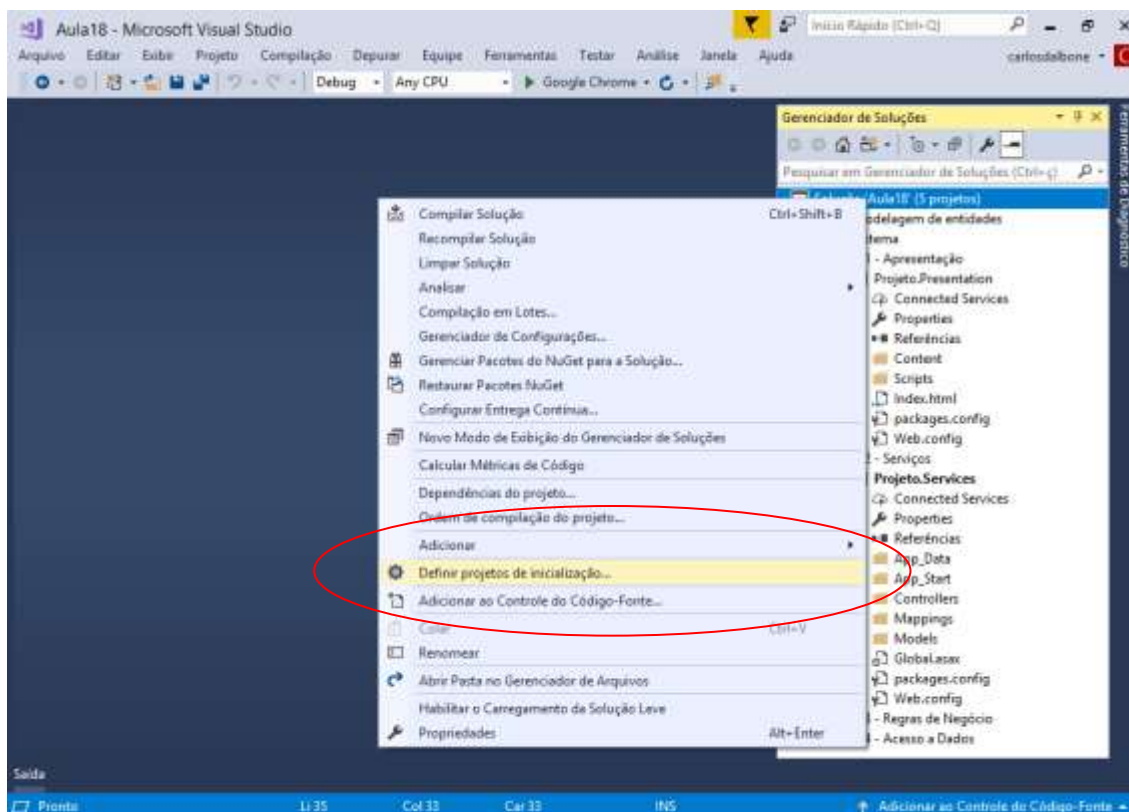
    <ul>
      <li>
        <a href="/Views/Cliente/Cadastro.html">
          Cadastrar Clientes
        </a>
      </li>
      <li>
        <a href="/Views/Cliente/Consulta.html">
          Consultar Clientes
        </a>
      </li>
    </ul>

  </div>

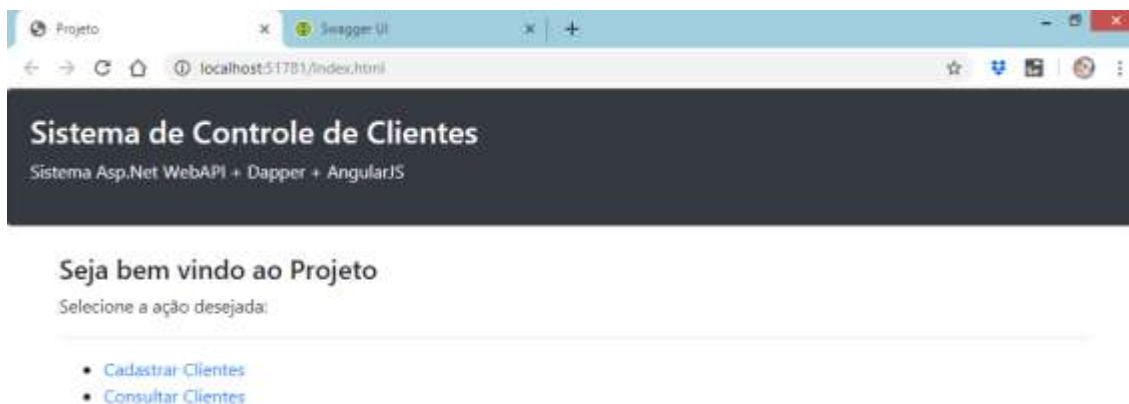
  <!-- arquivos javascript -->
  <script src="Scripts/jquery-3.0.0.min.js"></script>
  <script src="Scripts/bootstrap.min.js"></script>

</body>
</html>
```

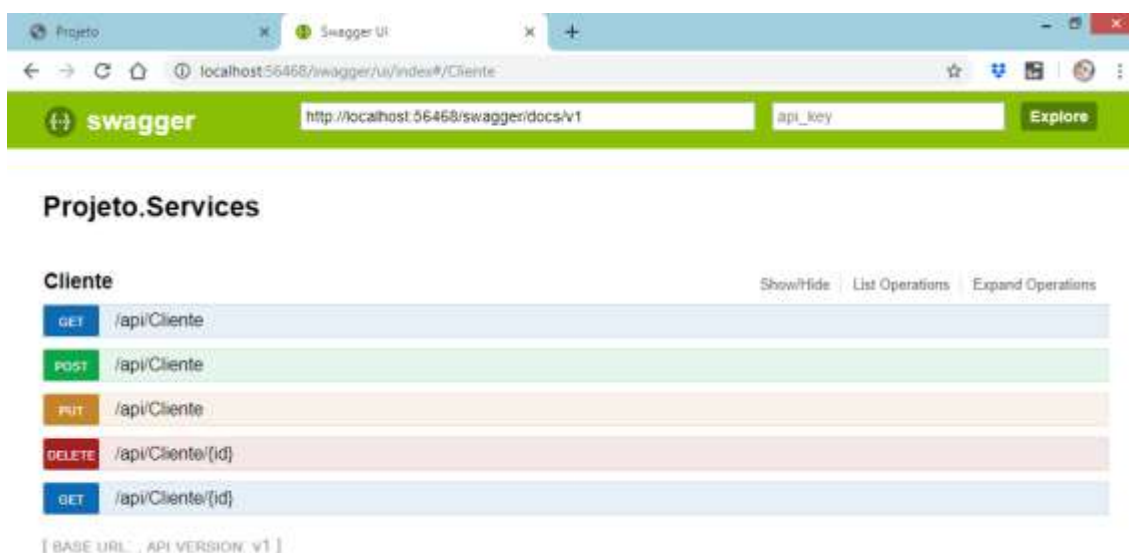
Definindo múltiplos projetos de inicialização na solution



<http://localhost:51781/Index.html>



<http://localhost:56468/swagger>



Criando as páginas de cadastro e consulta de clientes:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title>Projeto</title>

  <!-- folhas de estilo CSS -->
  <link href="/Content/bootstrap.min.css" rel="stylesheet" />
</head>
<body>

  <div class="card card-body bg-dark">
    <h3 class="text-white">Sistema de Controle de Clientes</h3>
    <p class="text-white">
```

Sistema Asp.Net WebAPI + Dapper + AngularJS

```
</p>
</div>

<div class="container">

    <br />

    <h4>Cadastro de Clientes</h4>
    <a href="/Index.html">Página inicial</a>
    <hr/>

    <div class="row">
        <div class="col-md-4">

            <label>Nome do Cliente:</label>
            <input type="text" class="form-control"
                placeholder="Digite aqui"/>
            <br/>

            <label>Email do Cliente:</label>
            <input type="text" class="form-control"
                placeholder="Digite aqui" />
            <br />

            <button class="btn btn-success">
                Cadastrar Cliente
            </button>

        </div>
    </div>

</div>

<!-- arquivos javascript -->
<script src="/Scripts/jquery-3.0.0.min.js"></script>
<script src="/Scripts/bootstrap.min.js"></script>

</body>
</html>
```

Sistema de Controle de Clientes

Sistema Asp.Net WebAPI + Dapper + AngularJS

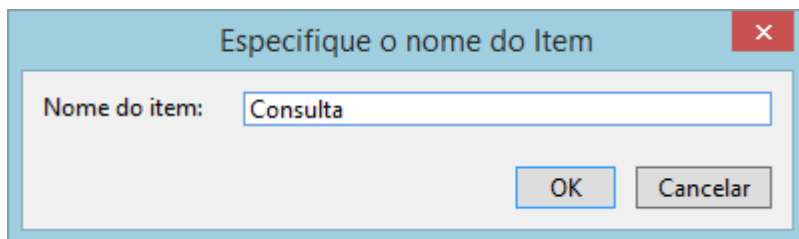
Cadastro de Clientes

[Página inicial](#)

Nome do Cliente:

Email do Cliente:

Consulta de Clientes:



```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title>Projeto</title>

  <!-- folhas de estilo CSS -->
  <link href="/Content/bootstrap.min.css" rel="stylesheet" />
</head>
<body>

  <div class="card card-body bg-dark">
    <h3 class="text-white">Sistema de Controle de Clientes</h3>
    <p class="text-white">
      Sistema Asp.Net WebAPI + Dapper + AngularJS
    </p>
  </div>

  <div class="container">

    <br />

    <h4>Consulta de Clientes</h4>
    <a href="/Index.html">Página inicial</a>
    <hr />

  </div>

  <!-- arquivos javascript -->
  <script src="/Scripts/jquery-3.0.0.min.js"></script>
  <script src="/Scripts/bootstrap.min.js"></script>

</body>
</html>
```

Continua...