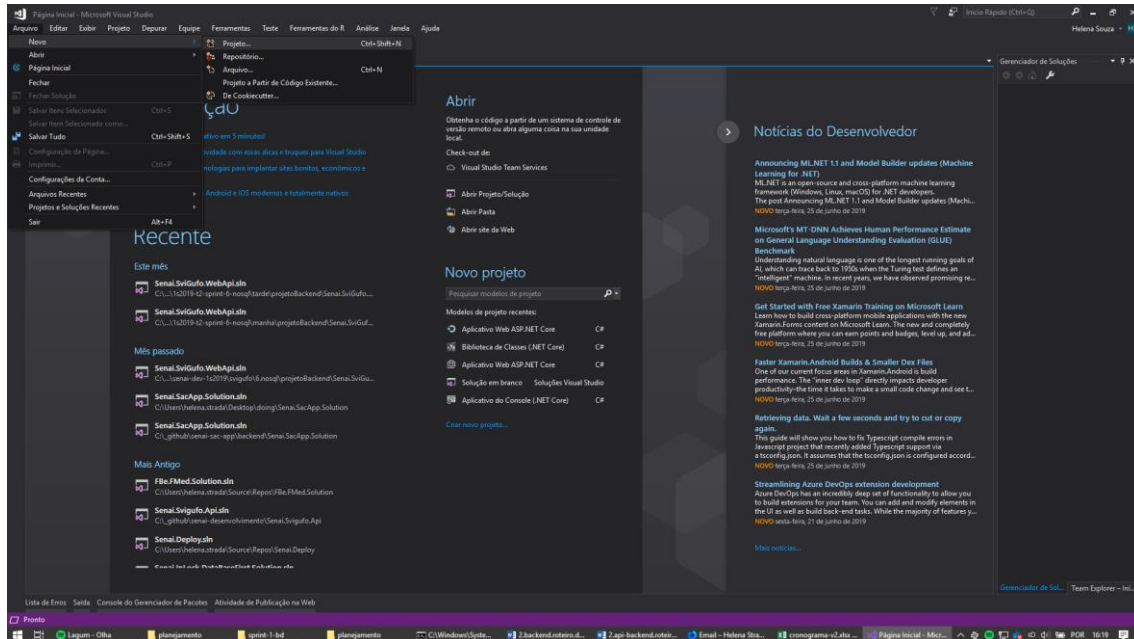
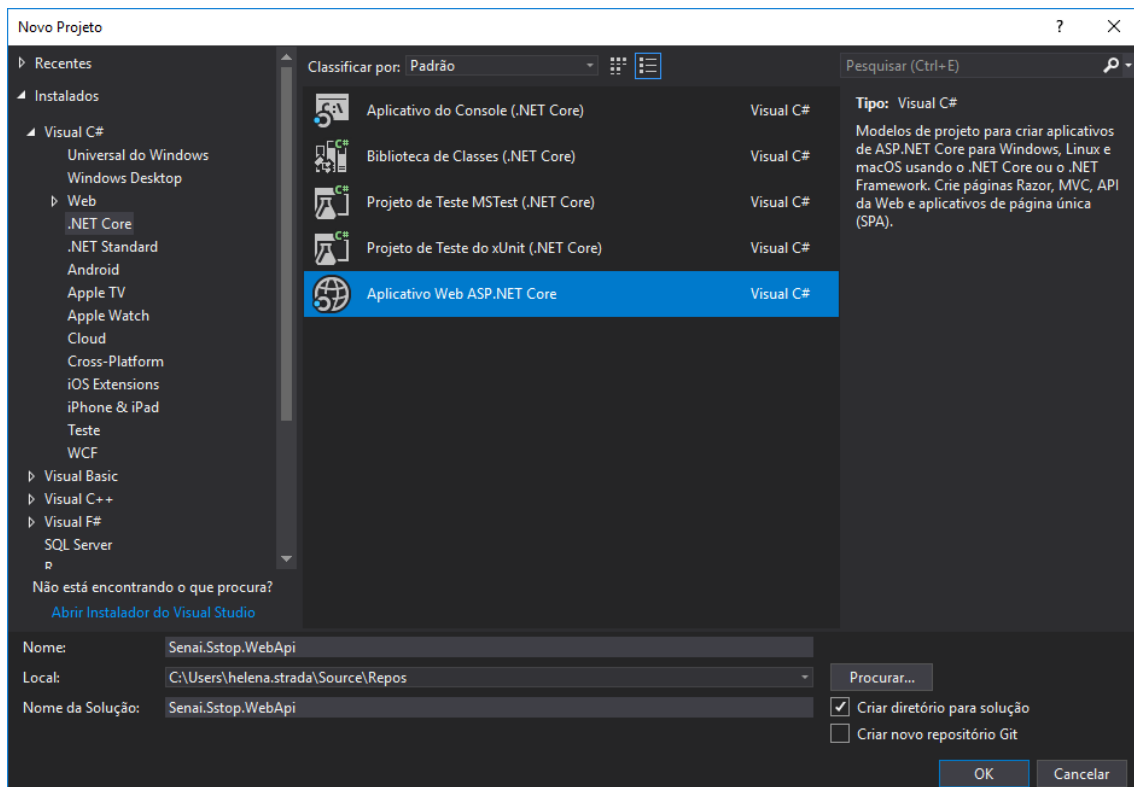


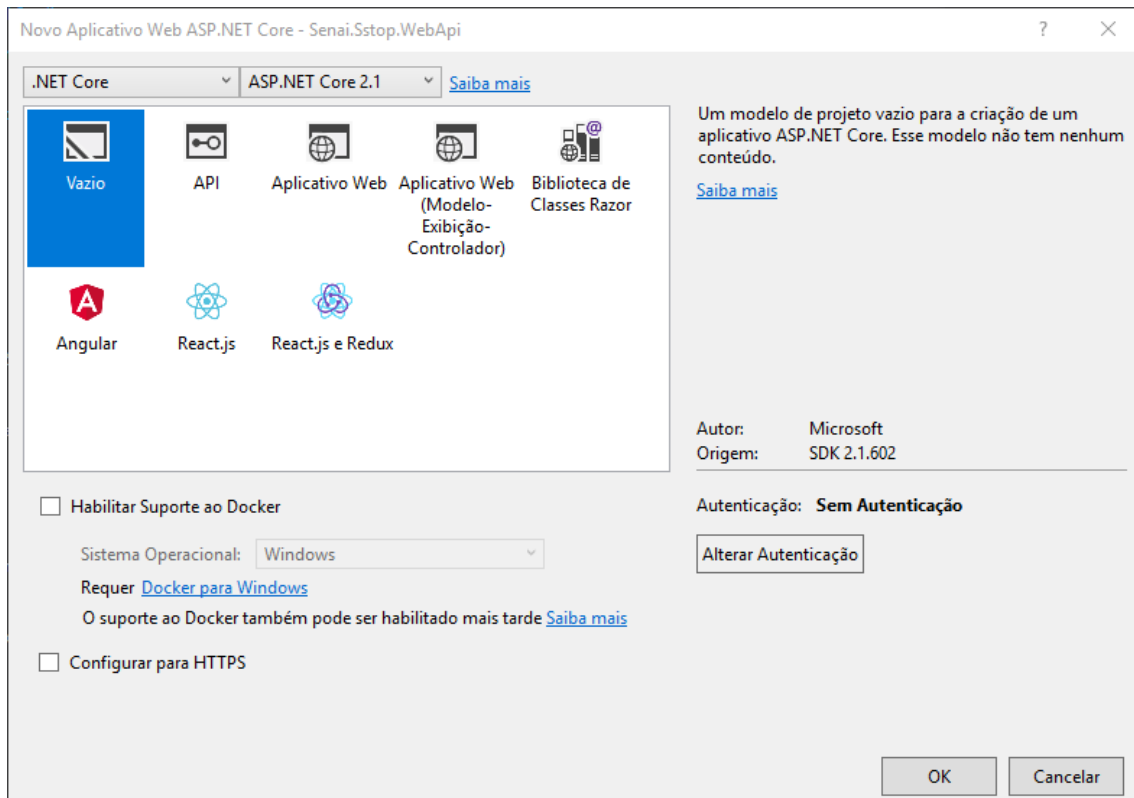
## Apresentação API

## Criar novo projeto



## Senai.Sstop.WebApi





## Habilitar MVC

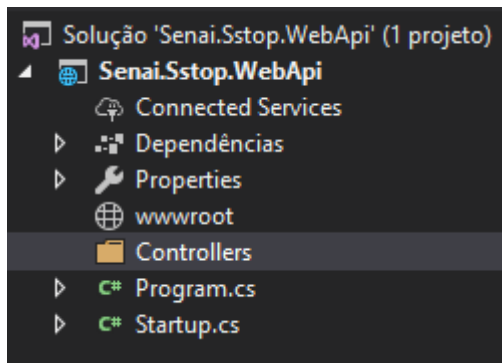
```
public class Startup
{
    // This method gets called by the runtime. Use this method to add services to the container.
    // For more information on how to configure your application, visit https://go.microsoft.com/fwlink/?LinkID=398940
    public void ConfigureServices(IServiceCollection services)
    {
        services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version_2_1);
    }

    // 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();
        }

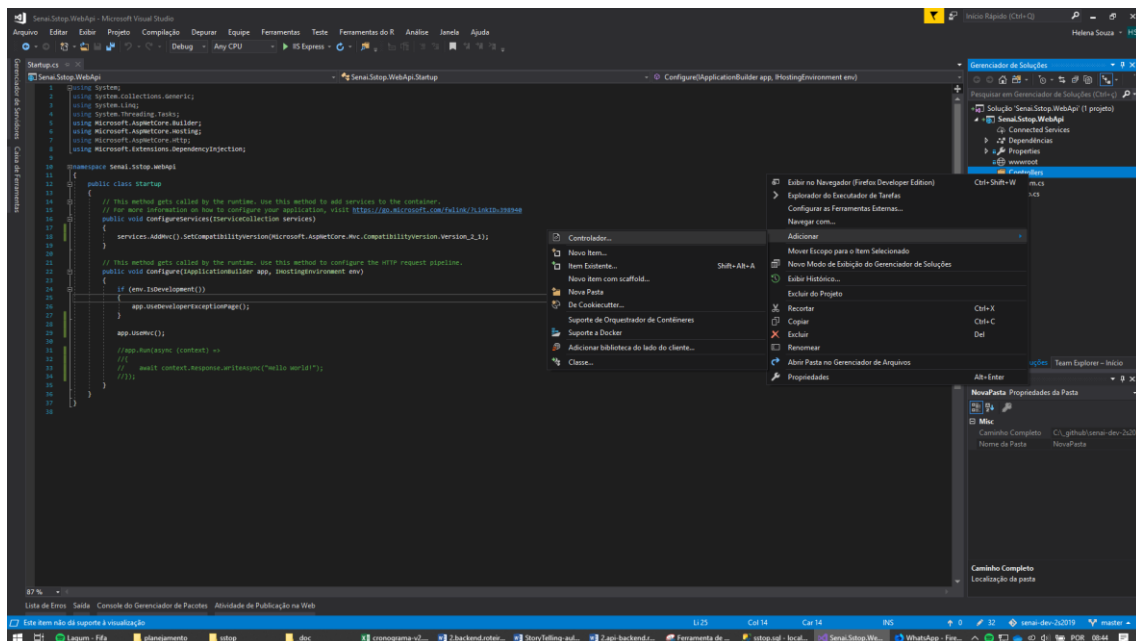
        app.UseMvc();

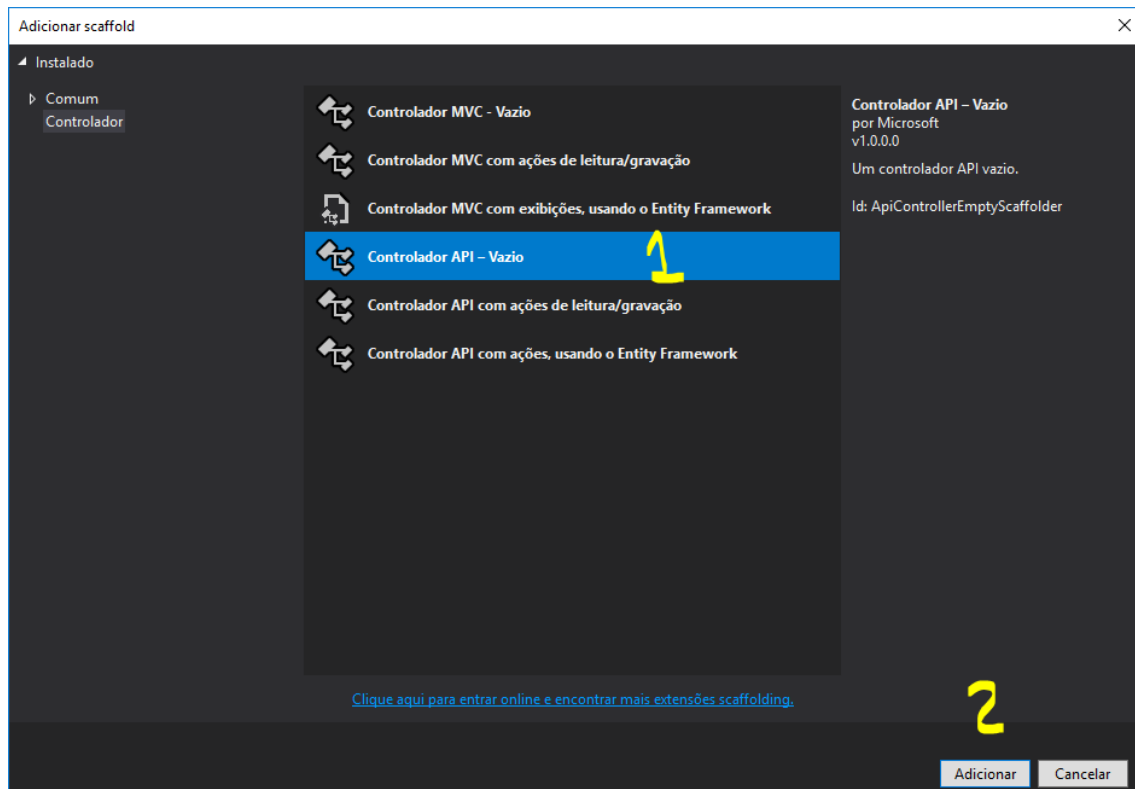
        //app.Run(async (context) =>
        //{
        //    await context.Response.WriteAsync("Hello World!");
        //});
    }
}
```

Criar uma nova pasta chamada Controllers

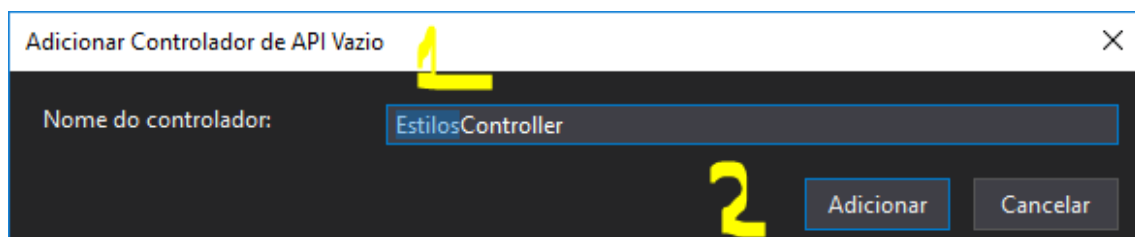


Botão direito na pasta Controllers -> Adicionar -> Controlador





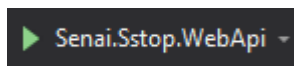
Escolher o nome de EstilosController



Alterar o EstilosController para realizar uma chamada no Get.

```
EstilosController.cs  Startup.cs
Senai.Sstop.WebApi
1  using Microsoft.AspNetCore.Mvc;
2
3  namespace Senai.Sstop.WebApi.Controllers
4  {
5      1 [Route("api/[controller]")]
6      [ApiController]
7      public class EstilosController : ControllerBase
8      {
9          [HttpGet]
10         public string Get()
11         {
12             2 return "Requisição recebida";
13         }
14     }
15 }
```

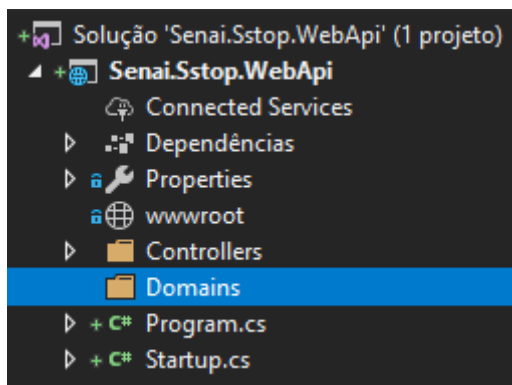
Play



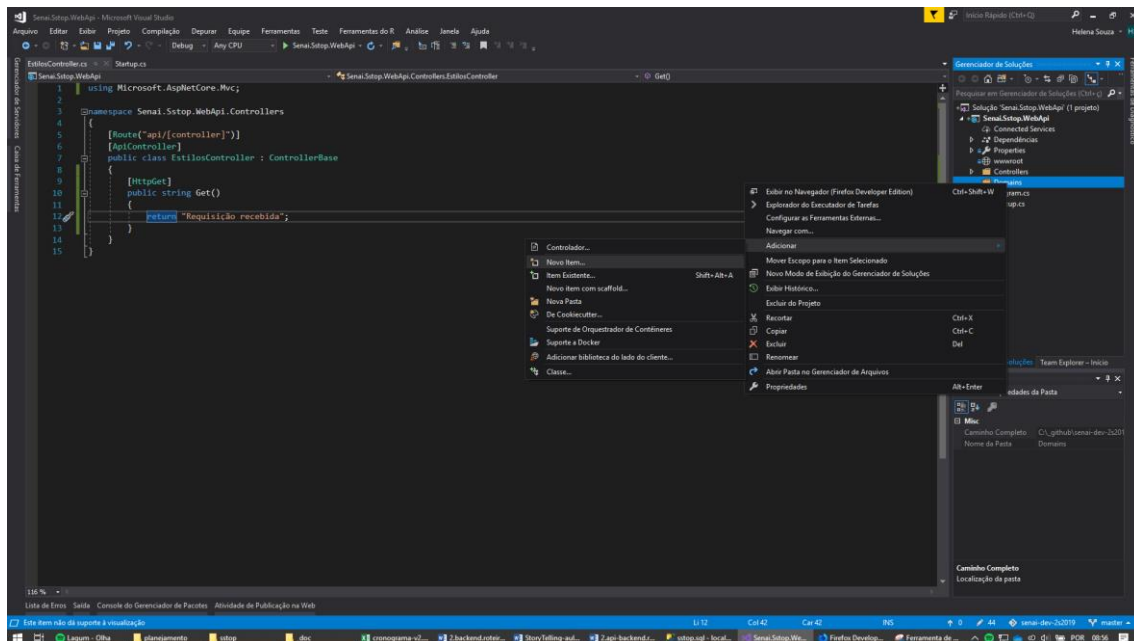
Abrir o navegador e realizar a seguinte chamada.



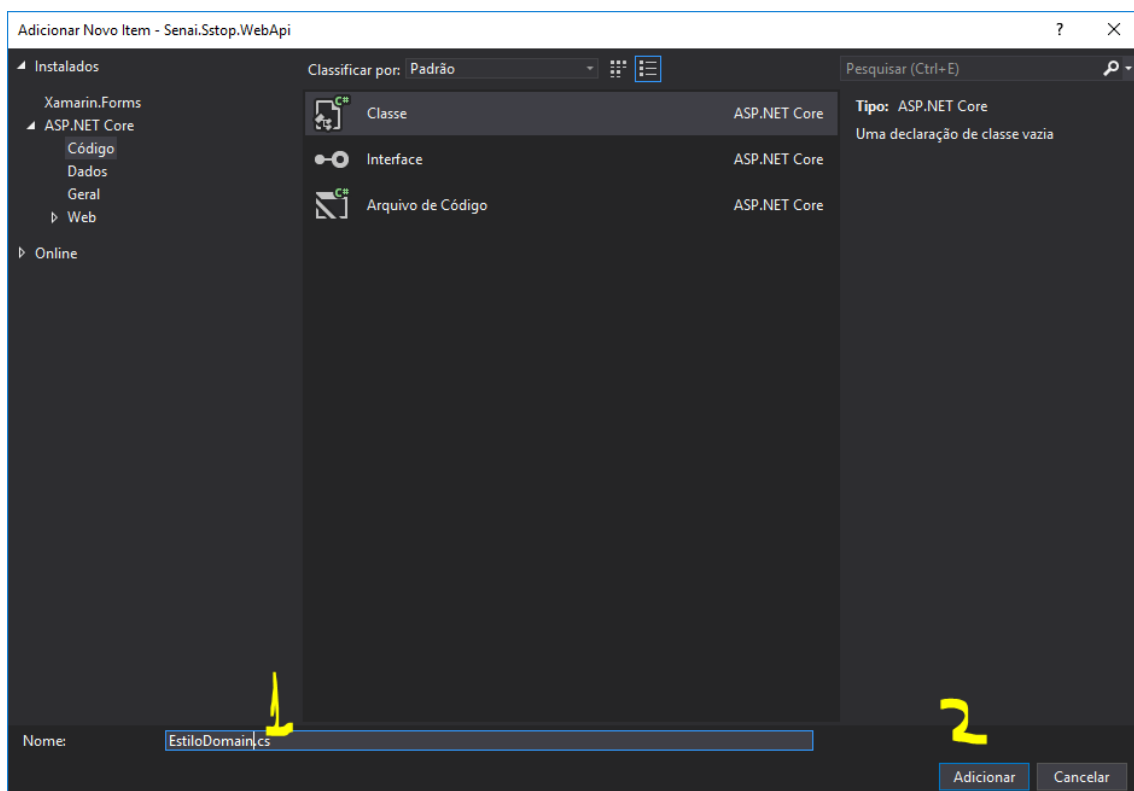
Criar uma pasta chamada Domains.



Adicionar novo item dentro da pasta Domains.



## Adicionar Classe



Alterar o EstiloDomain para coincidir com as informações do banco de dados.

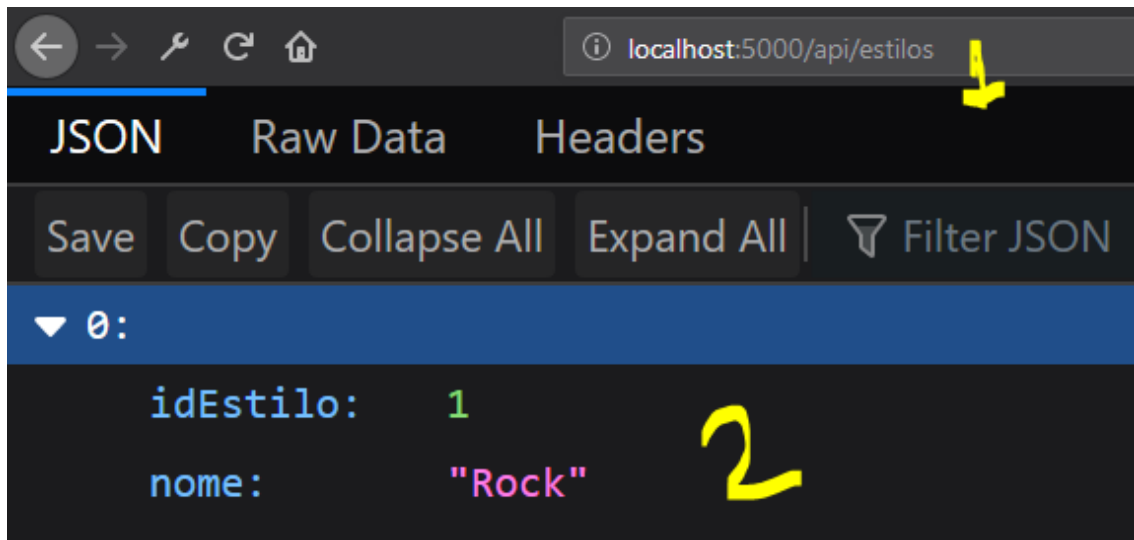
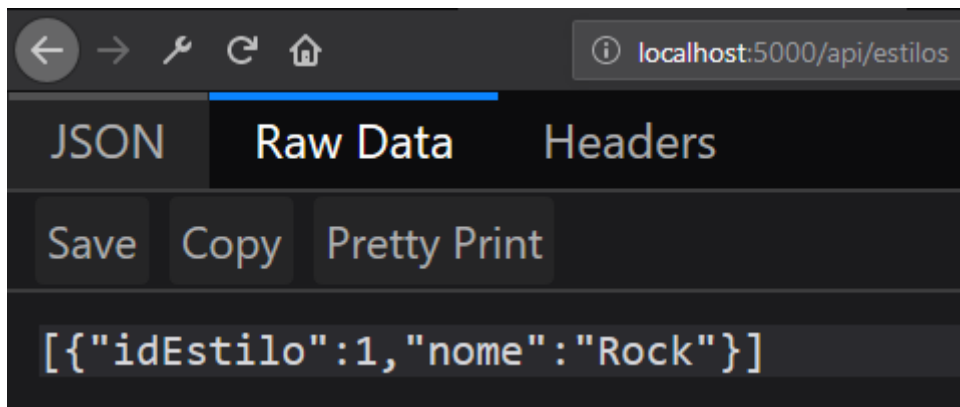
```
public class EstiloDomain
{
    public int IdEstilo { get; set; }
    public string Nome { get; set; }
}
```

No EstilosController, alterar para retornar uma lista fixa.

Adicionar o produces.

```
[Produces("application/json")]
[HttpGet]
public IEnumerable<EstiloDomain> Get()
{
    List<EstiloDomain> estilos = new List<EstiloDomain>()
    {
        new EstiloDomain { IdEstilo = 1, Nome = "Rock" }
    };
    return estilos;
}
```

Abrir no navegador.



Adicionar o cors e realizar uma requisição no frontend.

```
// This method gets called by the runtime. Use this method to add services to the container.
// For more information on how to configure your application, visit https://go.microsoft.com/fwlink/?LinkID=398940
public void ConfigureServices(IServiceCollection services)
{
    services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version_2_1);
    services.AddCors(options =>
    {
        options.AddPolicy("CorsPolicy", builder => builder.AllowAnyOrigin().AllowAnyMethod().AllowAnyHeader().AllowCredentials());
    });
}

// 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();
    }
    app.UseCors("CorsPolicy");
    app.UseMvc();

    //app.Run(async (context) =>
    //{
    //    await context.Response.WriteAsync("Hello World!");
    //});
}
```

Incluir dois registros no backend.

```
[Produces("application/json")]
[HttpGet]
public IEnumerable<EstiloDomain> Get()
{
    List<EstiloDomain> estilos = new List<EstiloDomain>()
    {
        new EstiloDomain { IdEstilo = 1, Nome = "Rock" },
        new EstiloDomain { IdEstilo = 2, Nome = "Pop" }
    };
    return estilos;
}
```

The screenshot shows a web browser window with the address bar displaying `localhost:5000/api/estilos`. The browser is showing the JSON response of the API. The response is a list of two objects, each representing a style domain. The first object has `idEstilo: 1` and `nome: "Rock"`. The second object has `idEstilo: 2` and `nome: "Pop"`.

Index	idEstilo	nome
0	1	"Rock"
1	2	"Pop"



Realizar a busca por id.

```
[Route("api/[controller]")]
[ApiController]
public class EstilosController : ControllerBase
{
    // [HttpGet]
    // public string Get()
    // {
    //     return "Requisição recebida";
    // }

    List<EstiloDomain> estilos = new List<EstiloDomain>()
    {
        new EstiloDomain { IdEstilo = 1, Nome = "Rock" },
        new EstiloDomain { IdEstilo = 2, Nome = "Pop" }
    };

    [Produces("application/json")]
    [HttpGet]
    public IEnumerable<EstiloDomain> Get()
    {
        return estilos;
    }

    [Produces("application/json")]
    [HttpGet("{id}")]
    public IActionResult BuscarPorId(int id)
    {
        EstiloDomain Estilo = estilos.Find(x => x.IdEstilo == id);
        if (Estilo == null)
        {
            return NotFound();
        }
        return Ok(Estilo);
    }
}
```

The screenshot shows a web browser at the URL `localhost:5000/api/estilos/1`. The JSON response is displayed as `{idEstilo: 1, nome: 'Rock'}`. The browser's developer tools are open, showing the network tab with the request details. The request is a GET method to the URL `http://localhost:5000/api/estilos/1` with a status code of 200 (OK).

Remover o produces dos métodos e colocar em cima do controller.

```

[Route("api/[controller]")]
[Produces("application/json")]
[ApiController]
public class EstilosController : ControllerBase
{
    // [HttpGet]
    // public string Get()
    // {
    //     return "Requisição recebida";
    // }

    List<EstiloDomain> estilos = new List<EstiloDomain>()
    {
        new EstiloDomain { IdEstilo = 1, Nome = "Rock" },
        new EstiloDomain { IdEstilo = 2, Nome = "Pop" }
    };

    [HttpGet]
    public IEnumerable<EstiloDomain> Get()
    {
        return estilos;
    }

    [HttpGet("{id}")]
    public IActionResult BuscarPorId(int id)
    {
        EstiloDomain Estilo = estilos.Find(x => x.IdEstilo == id);
        if (Estilo == null)
        {
            return NotFound();
        }
        return Ok(Estilo);
    }
}

```

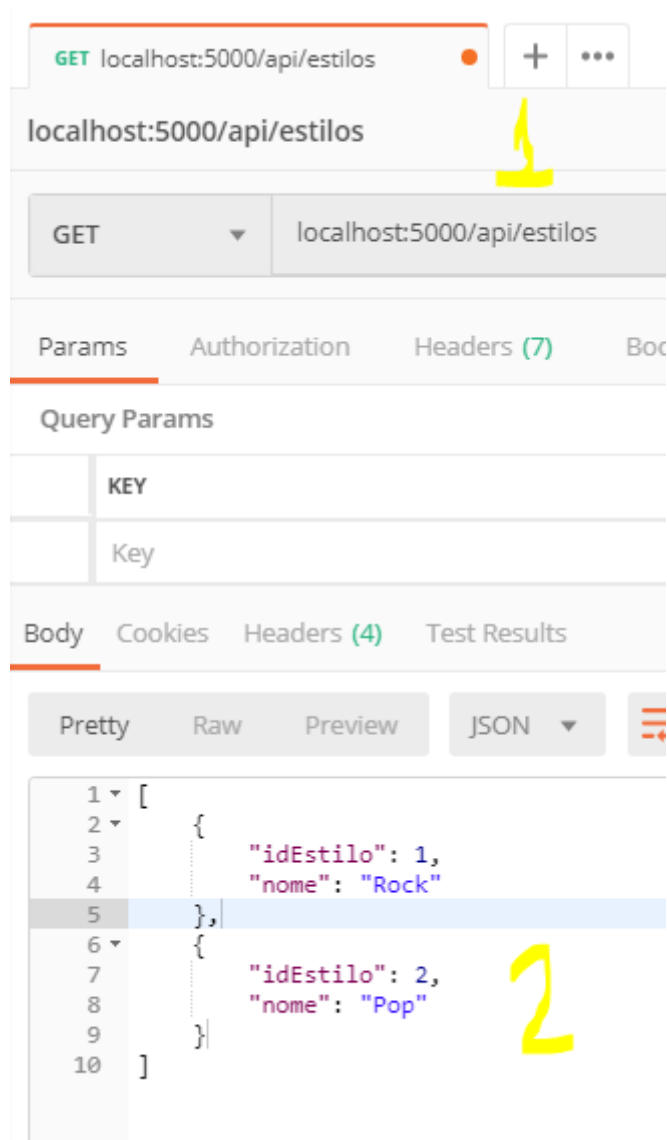
Realizar o cadastro de um novo item.

```

[HttpPost]
public IActionResult Cadadstrar()
{
    estilos.Add(new EstiloDomain { IdEstilo = estilos.Count + 1, Nome = "Eletrônica" });
    return Ok(estilos);
}

```

Apresentar o Postman.



Buscar por Id e mostrar uma requisição com erro.

The screenshot displays a REST client interface with two tabs at the top. The first tab shows a GET request to `localhost:5000/api/estilos`. The second tab, which is active, shows a GET request to `localhost:5000/api/estilos/1`. A yellow number '1' is written over the URL in the active tab. Below the URL bar, there are tabs for `Params`, `Authorization`, `Headers (7)`, `Body`, and `Pre-request Script`. The `Params` tab is selected, showing a table with two columns: `KEY` and `Value`. The `Value` column contains the text `Key`. Below the `Params` tab, there are tabs for `Body`, `Cookies`, `Headers (4)`, and `Test Results`. The `Body` tab is selected, showing a JSON response in `Pretty` format. The JSON response is `{ "idEstilo": 1, "nome": "Rock" }`. A yellow number '2' is written over the JSON response. The response is also shown in `Raw` format as `{ "idEstilo": 1, "nome": "Rock" }`.

GET localhost:5000/api/estilos

GET localhost:5000/api/estilos/1

localhost:5000/api/estilos/1

GET localhost:5000/api/estilos/1

Params Authorization Headers (7) Body Pre-request Script

Query Params

KEY	Value
	Key

Body Cookies Headers (4) Test Results

Pretty Raw Preview JSON

```
1 {  
2   "idEstilo": 1,  
3   "nome": "Rock"  
4 }
```

Realizar uma requisição com post.



Alterar para receber a requisição do cliente.

```
[HttpPost]
public IActionResult Cadadstrar(EstiloDomain estiloDomain)
{
    //estilos.Add(new EstiloDomain { IdEstilo = estilos.Count + 1, Nome = "Eletrônica" });
    estilos.Add(new EstiloDomain { IdEstilo = estilos.Count + 1, Nome = estiloDomain.Nome });
    return Ok(estilos);
}
```

Mostrar a requisição no postman enviando os dados no body.

localhost:5000/api/estilos

POST **1** localhost:5000/api/estilos **2**

Params Authorization Headers (9) **Body** Pre-request Script Tests

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary JSON (application/json) **3**

```
1 {  
2   "nome" : "Mpb"  
3 }
```

**4**

Body Cookies Headers (4) Test Results

Pretty Raw Preview JSON

```
1 [  
2   {  
3     "idEstilo": 1,  
4     "nome": "Rock" 5  
5   },  
6   {  
7     "idEstilo": 2,  
8     "nome": "Pop"  
9   },  
10  {  
11    "idEstilo": 3,  
12    "nome": "Mpb"  
13  }  
14 ]
```

## Atualização

```
[HttpPut]  
public IActionResult Atualizar(EstiloDomain estiloDomain) 1  
{  
    EstiloDomain estiloProcurado = estilos.Find(x => x.IdEstilo == estiloDomain.IdEstilo);  
    estiloProcurado.Nome = estiloDomain.Nome;  
    return Ok(estilos); 2  
}
```

Atualizar um valor no postman.

localhost:5000/api/estilos

PUT localhost:5000/api/estilos

Params Authorization Headers (9) Body Pre-request Script Tests

none form-data x-www-form-urlencoded raw binary JSON (application/json)

```
1 {
2   "idEstilo": 1,
3   "nome": "Mpb"
4 }
```

3

Body Cookies Headers (4) Test Results

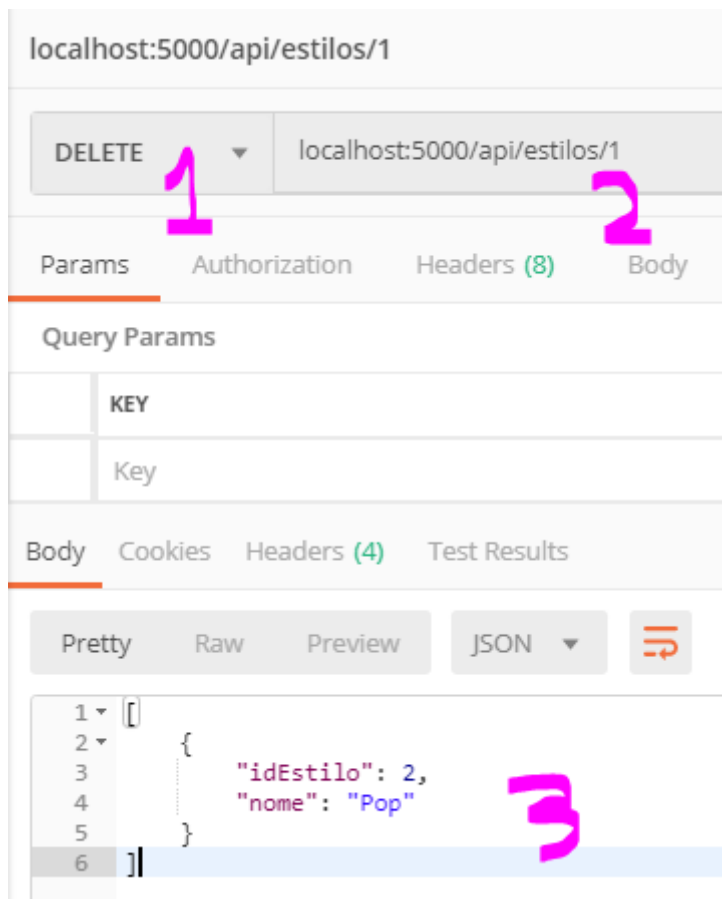
Pretty Raw Preview JSON

```
1 [
2   {
3     "idEstilo": 1,
4     "nome": "Mpb"
5   },
6   {
7     "idEstilo": 2,
8     "nome": "Pop"
9   }
10 ]
```

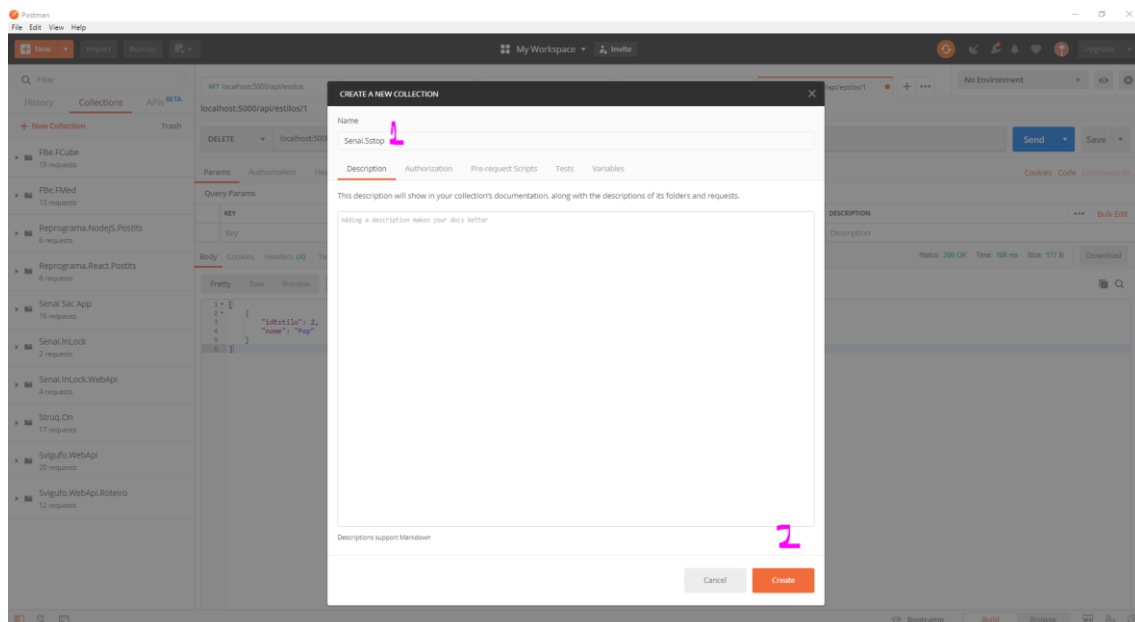
4

Deletar

```
1 [HttpDelete("{id}")]
public IActionResult Deletar(int id)
{
    estilos.Remove(estilos.Find(x => x.IdEstilo == id));
    return Ok(estilos);
}
2
```

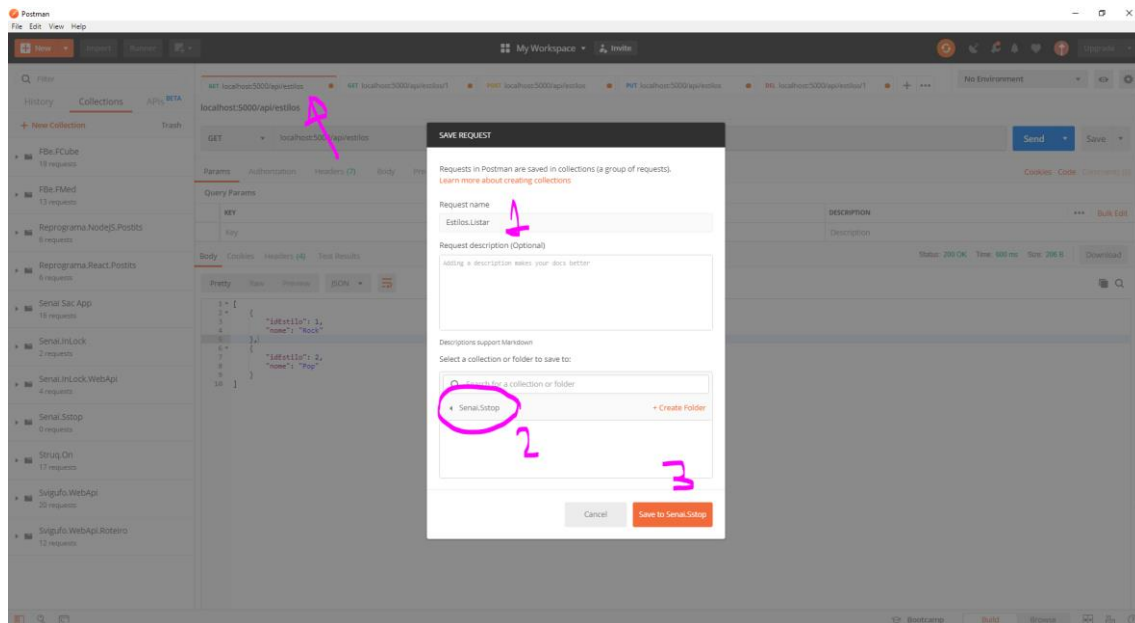


Organizar no postman uma nova collection.

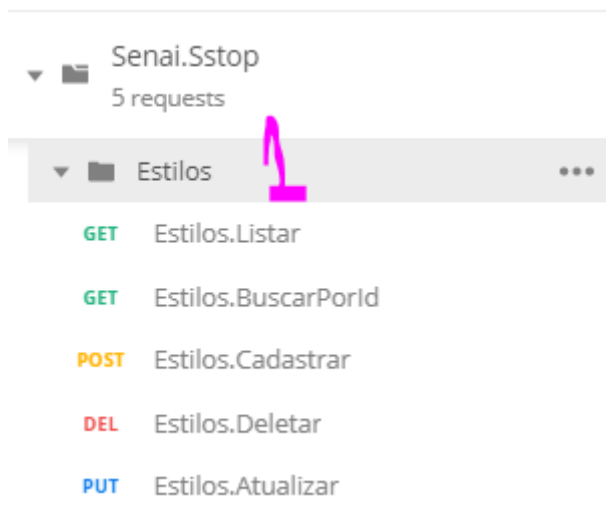


Salvar uma requisição na lista.



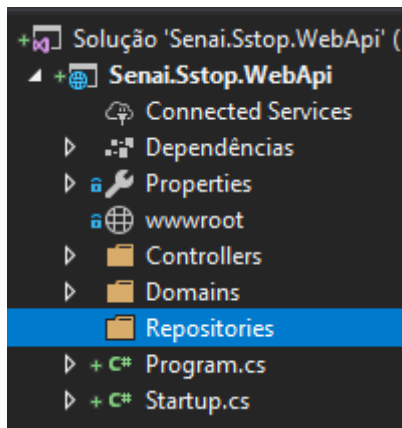


Para melhorar, criar uma pasta e adicionar as requisições dentro da pasta.

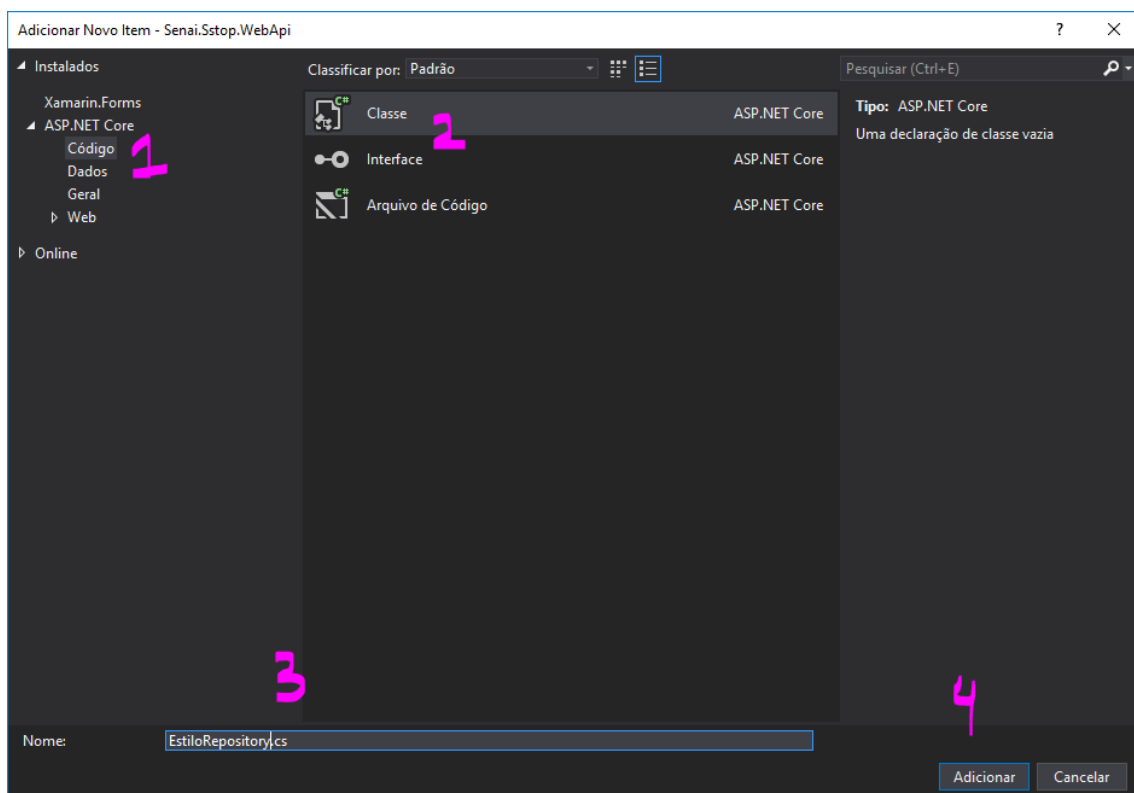


Vamos iniciar a alteração para que ao invés dos dados serem armazenados em uma lista, serem gravados/lidos do banco de dados.

Criar uma pasta chamada Repositories.



Adicionar uma nova classe chamada EstiloRepository.cs.



Remover a lista do controller e incluir no repositório.

```
public class EstiloRepository
{
    List<EstiloDomain> estilos = new List<EstiloDomain>()
    {
        new EstiloDomain { IdEstilo = 1, Nome = "Rock" },
        new EstiloDomain { IdEstilo = 2, Nome = "Pop" }
    };

    public List<EstiloDomain> Listar()
    {
        return estilos;
    }
}
```

Alterar no controller e incluir a instância do repositório.

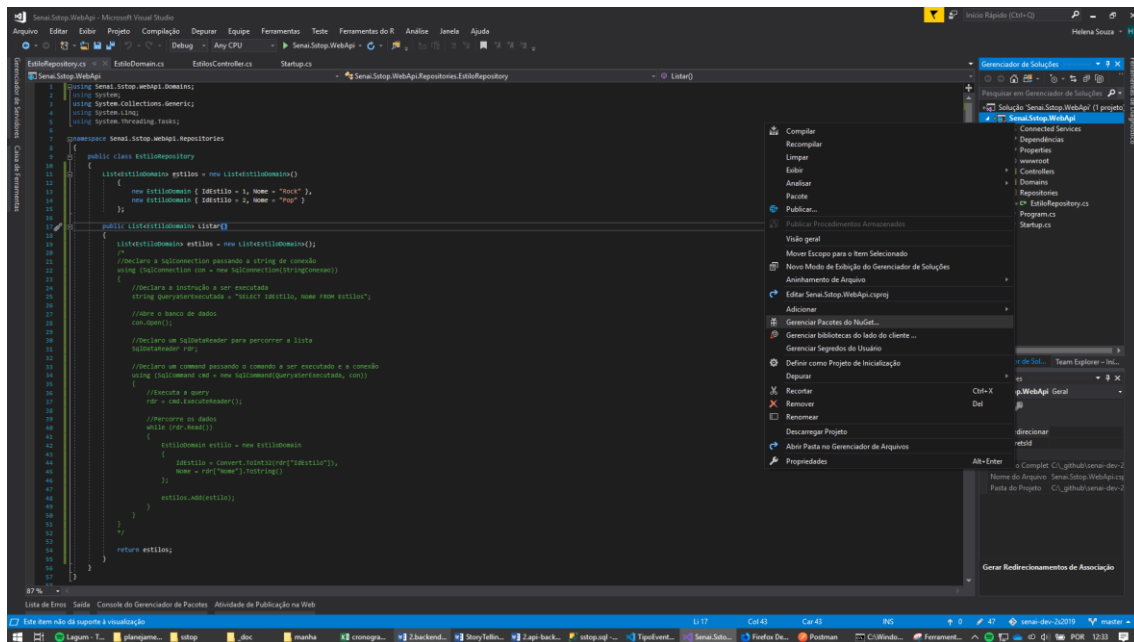
```
//[HttpGet]
//public string Get()
//{
//    return "Requisição recebida";
//}

List<EstiloDomain> estilos = new List<EstiloDomain>()
{
    new EstiloDomain { IdEstilo = 1, Nome = "Rock" },
    new EstiloDomain { IdEstilo = 2, Nome = "Pop" }
};

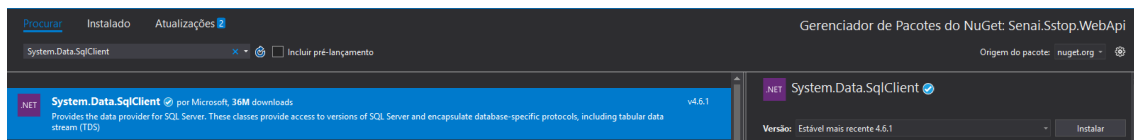
EstiloRepository EstiloRepository = new EstiloRepository();

[HttpGet]
public IEnumerable<EstiloDomain> Get()
{
    //return estilos;
    return EstiloRepository.Listar();
}
```

Gerenciar pacotes no nuget.



Adicionar o pacote System.Data.SqlClient – versão 4.6.1



Criar a conexão e realizar a leitura.

```

private string StringConexao = "Data Source=localhost; initial catalog=RoteirosSstop;Integrated Security=true";

public List<EstiloDomain> Listar()
{
    List<EstiloDomain> estilos = new List<EstiloDomain>();

    //Declaro a SqlConnection passando a string de conexão
    using (SqlConnection con = new SqlConnection(StringConexao))
    {
        //Declara a instrução a ser executada
        string QueryaSerExecutada = "SELECT IdEstilo, Nome FROM Estilos";

        //Abre o banco de dados
        con.Open();

        //Declaro um SqlDataReader para percorrer a lista
        SqlDataReader rdr;

        //Declaro um command passando o comando a ser executado e a conexão
        using (SqlCommand cmd = new SqlCommand(QueryaSerExecutada, con))
        {
            //Executa a query
            rdr = cmd.ExecuteReader();

            //Percorre os dados
            while (rdr.Read())
            {
                EstiloDomain estilo = new EstiloDomain
                {
                    IdEstilo = Convert.ToInt32(rdr["IdEstilo"]),
                    Nome = rdr["Nome"].ToString()
                };

                estilos.Add(estilo);
            }
        }
    }

    return estilos;
}

```

Incluir um novo registro no Sstop.

```
INSERT INTO Estilos VALUES ('Folk');
```

Realizar uma nova chamada à API.

► Estilos.Listar

GET localhost:5000/api/estilos

Params Authorization Headers (7) Body

Query Params

KEY
Key

Body Cookies Headers (4) Test Results

Pretty Raw Preview JSON

```
[
  {
    "idEstilo": 1,
    "nome": "Folk"
  }
]
```

Cadastrar

EstiloRepository.cs

```

/// <summary>
/// Cadastra um novo estilo de música
/// </summary>
/// <param name="estiloDomain"></param>
public void Cadastrar(EstiloDomain estiloDomain)
{
    //Declara a conexão passando sua string de conexão
    using (SqlConnection con = new SqlConnection(StringConexao))
    {
        // string QuerySerExecutada = "INSERT INTO Estilos(Nome) VALUES (' + estiloDomain.Nome + "')";
        // Declara a query passando o valor como parametro
        string QueryAserExecutada = "INSERT INTO Estilos (Nome) VALUES (@Nome)";
        //Declara o command passando a query e a conexão
        SqlCommand cmd = new SqlCommand(QueryAserExecutada, con);
        //Passa o valor do parametro
        cmd.Parameters.AddWithValue("@Nome", estiloDomain.Nome);
        //abre a conexão
        con.Open();
        //Executa o comando
        cmd.ExecuteNonQuery();
    }
}

```

EstilosController

```

[HttpPost]
public IActionResult Cadastrar(EstiloDomain estiloDomain)
{
    //estilos.Add(new EstiloDomain { IdEstilo = estilos.Count + 1, Nome = "Eletrônica" });
    // estilos.Add(new EstiloDomain { IdEstilo = estilos.Count + 1, Nome = estiloDomain.Nome });
    EstiloRepository.Cadastrar(estiloDomain);
    // return Ok(estilos);
    return Ok();
}

```

Postman – Estilos.Cadastrar

Buscar Por Id

EstiloRepository.cs

```

public EstiloDomain BuscarPorId(int id)
{
    string QuerySelect = "SELECT IdEstilo, Nome FROM Estilos WHERE IdEstilo = @IdEstilo";

    using (SqlConnection con = new SqlConnection(StringConexao))
    {
        con.Open();
        SqlDataReader sdr;

        using (SqlCommand cmd = new SqlCommand(QuerySelect, con))
        {
            cmd.Parameters.AddWithValue("@IdEstilo", id);
            sdr = cmd.ExecuteReader();

            if (sdr.HasRows)
            {
                while (sdr.Read())
                {
                    EstiloDomain estilo = new EstiloDomain
                    {
                        IdEstilo = Convert.ToInt32(sdr["IdEstilo"]),
                        Nome = sdr["Nome"].ToString()
                    };

                    return estilo;
                }
            }

            return null;
        }
    }
}

```

EstilosController.cs

```
[HttpGet("{id}")]
public IActionResult BuscarPorId(int id)
{
    //EstiloDomain Estilo = estilos.Find(x => x.IdEstilo == id);
    EstiloDomain Estilo = EstiloRepository.BuscarPorId(id);
    if (Estilo == null)
    {
        return NotFound();
    }
    return Ok(Estilo);
}
```

Postman – Estilos.BuscarPorId

Alterar

EstiloRepository.cs

```
public void Alterar(EstiloDomain estiloDomain)
{
    using (SqlConnection con = new SqlConnection(StringConexao))
    {
        string Query = "UPDATE Estilos SET Nome = @Nome WHERE IdEstilo = @IdEstilo";

        SqlCommand cmd = new SqlCommand(Query, con);
        cmd.Parameters.AddWithValue("@Nome", estiloDomain.Nome);
        cmd.Parameters.AddWithValue("@IdEstilo", estiloDomain.IdEstilo);
        con.Open();
        cmd.ExecuteNonQuery();
    }
}
```

EstilosController.cs

```
[HttpPut]
public IActionResult Atualizar(EstiloDomain estiloDomain)
{
    //EstiloDomain estiloProcurado = estilos.Find(x => x.IdEstilo == estiloDomain.IdEstilo);
    //estiloProcurado.Nome = estiloDomain.Nome;
    EstiloRepository.Alterar(estiloDomain);
    return Ok();
}
```

Postman – Estilos.Atualizar

EstiloRepository.cs

```
public void Deletar(int id)
{
    string QueryDelete = "DELETE FROM Estilos WHERE IdEstilo = @IdEstilo;";
    using (SqlConnection con = new SqlConnection(StringConexao))
    {
        using (SqlCommand cmd = new SqlCommand(QueryDelete, con))
        {
            cmd.Parameters.AddWithValue("@IdEstilo", id);
            con.Open();
            cmd.ExecuteNonQuery();
        }
    }
}
```

EstilosController.cs

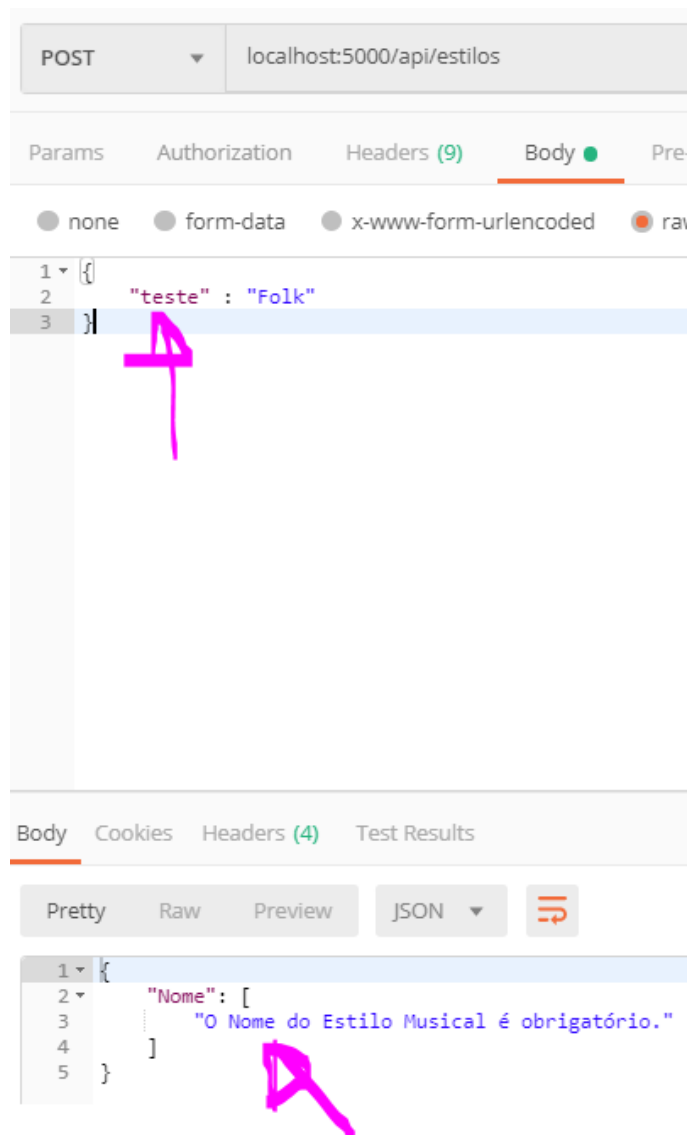
```
[HttpDelete("{id}")]
public IActionResult Deletar(int id)
{
    //estilos.Remove(estilos.Find(x => x.IdEstilo == id));
    EstiloRepository.Deletar(id);
    return Ok();
}
```

Postman – Estilos.Deletar

Melhorar o cadastro e incluir uma verificação para um novo cadastro de estilo musical.

```
public class EstiloDomain
{
    public int IdEstilo { get; set; }
    [Required(ErrorMessage = "O Nome do Estilo Musical é obrigatório.")]
    public string Nome { get; set; }
}
```

Realizar uma chamada sem o nome do estilo musical.



The screenshot shows a Postman interface for a POST request to `localhost:5000/api/estilos`. The request body is a JSON object: `{ "teste": "Folk" }`. The response is a JSON object indicating a validation error: `{ "Nome": [ "O Nome do Estilo Musical é obrigatório." ] }`. Hand-drawn pink annotations include an arrow pointing to the `"teste"` property in the request body and another arrow pointing to the error message in the response body.