

Atividade

Criar CRUD de usuários

Nesta atividade, você continuará a implementação dos códigos no projeto Web API da empresa **ExoApi**. Nesta etapa, sua tarefa será o desenvolvimento dos códigos dos métodos para o novo **CRUD** (**create**, **read**, **update** e **delete**) de usuários.

Os testes desta atividade continuarão a ser executados no aplicativo **Insomnia** (ou **Postman**).

Sua tarefa será implementar os códigos dos métodos nas duas classes, **UsuariosController.cs** e **UsuarioRepository.cs**. Será necessário, também, o desenvolvimento do model **Usuario.cs**, a referência do model **Usuario** na classe **ExoContext.cs** e, por último, a inserção do serviço na classe **Program.cs**.

Importante

Para realizar este tutorial, é necessário baixar e descompactar o arquivo **ATIVIDADE-05.zip**, que está disponível no AVA.



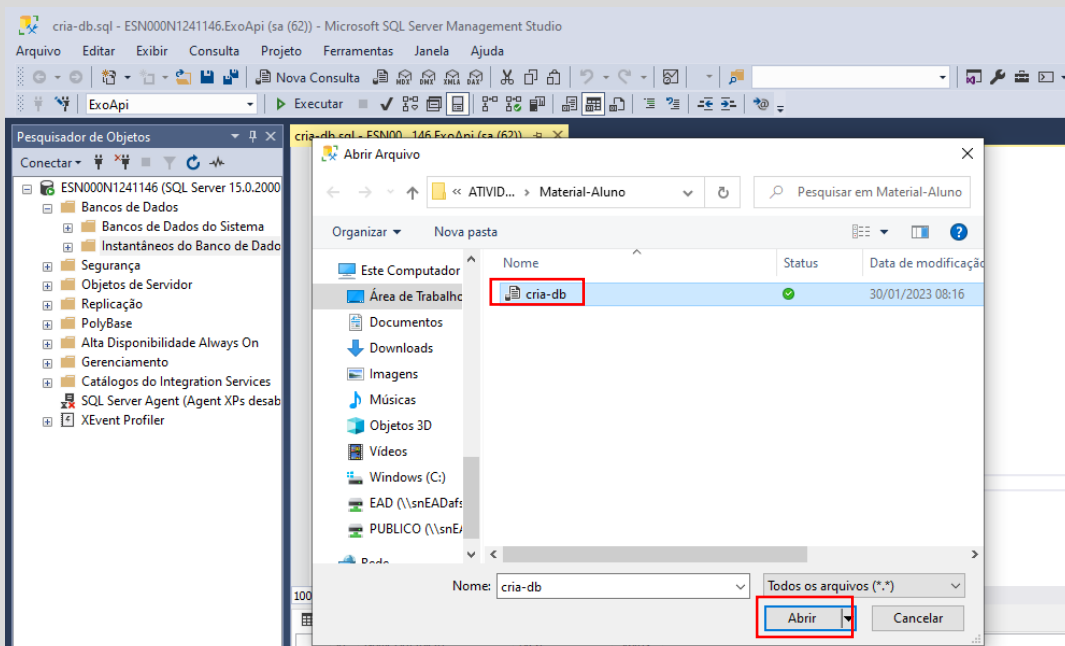
Criando o banco no SSMS

Importante

Os passos a seguir devem ser realizados caso você tenha apagado o banco das atividades anteriores ou queira criar um novo banco. Caso já esteja com o banco criado, pule essa etapa e vá para [Preparação dos arquivos no VSCode](#).



1. Abra o SQL Server Management Studio (SSMS). Clique em **Arquivo > Abrir Arquivo...** e localize a pasta baixada para realizar a atividade. Na pasta **Material-Aluno**, selecione o script **cria-db.sql** e clique em **Abrir**.

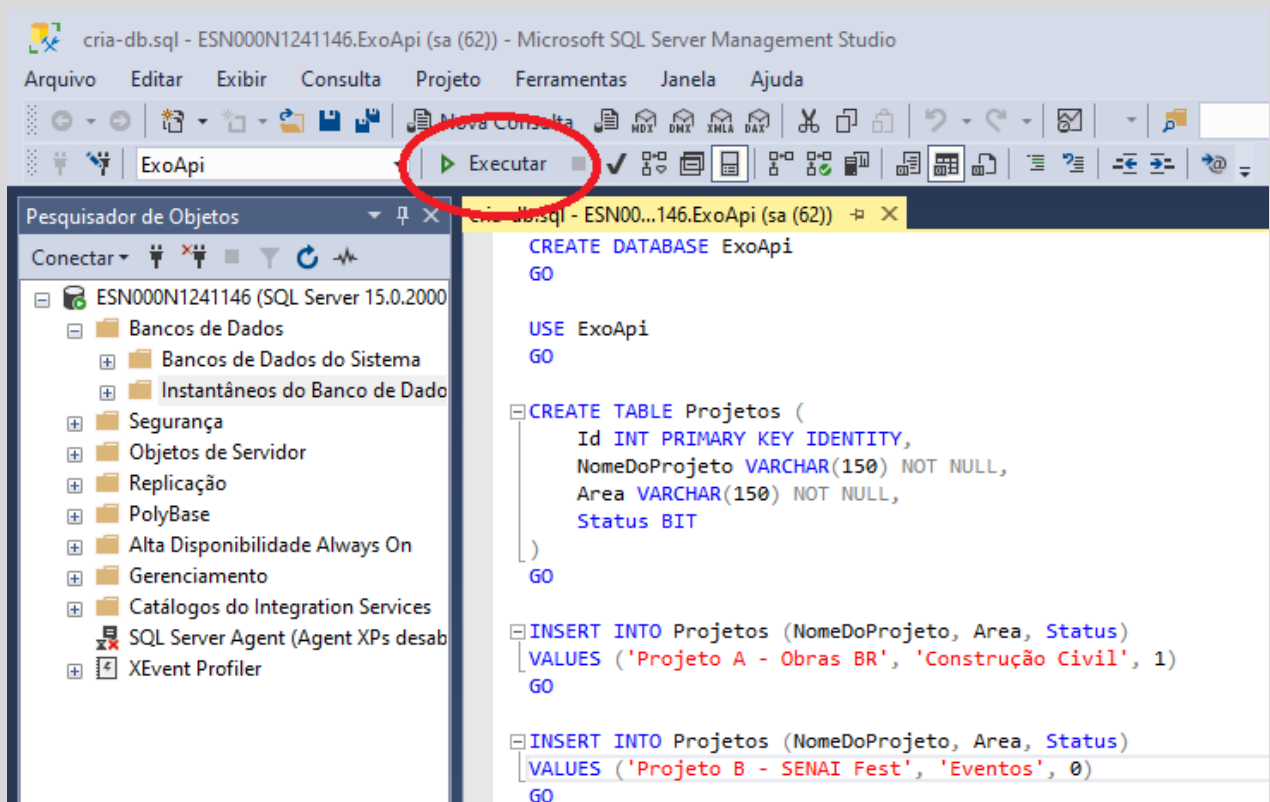


Dica!

Caso já exista um banco de dados e você quiser criar um novo, será necessário deletar o existente antes de executar esse script.



2. Com o arquivo **cria-db.sql** aberto, clique em **Executar** para executar o banco e criar os usuários.



Preparação dos arquivos no VSCode

Importante

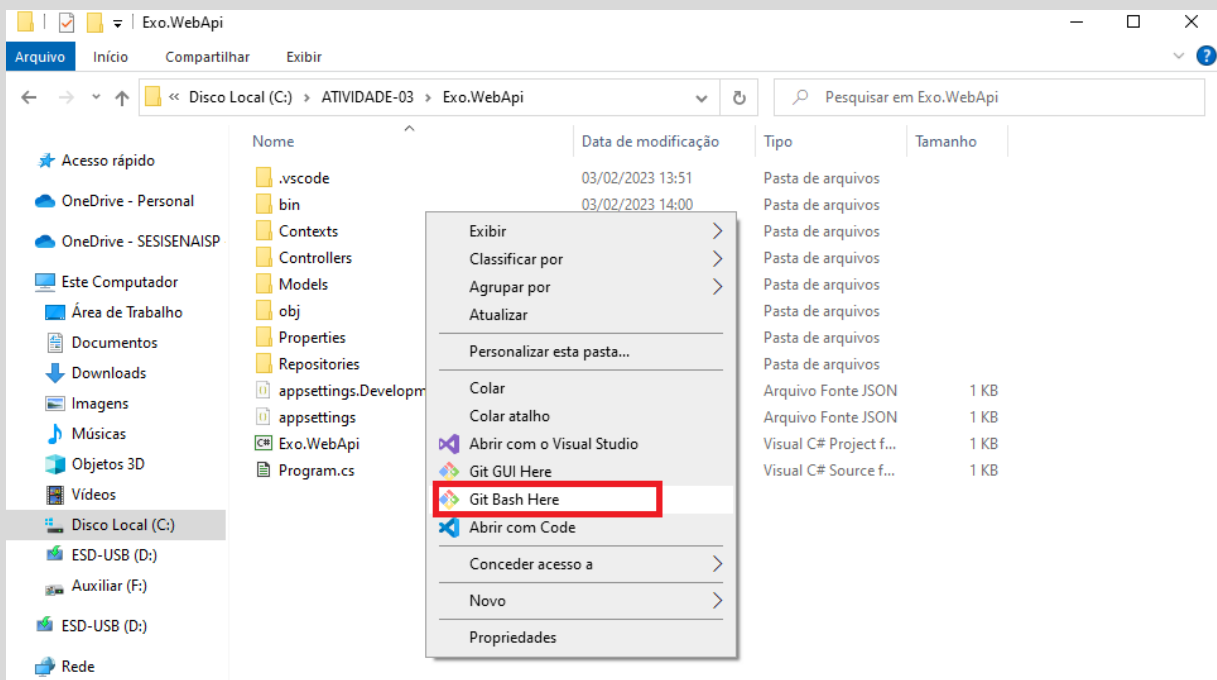
O projeto de API dessa atividade será o mesmo da antecedente. Na etapa anterior, foram realizadas algumas implementações e continuaremos com esse mesmo projeto.

Caso seja necessário, os arquivos estão anexados à atividade (na parte em que paramos).

Caso você queira usar o seu projeto, pule a próxima etapa e vá diretamente para [Implementação dos códigos das classes](#).



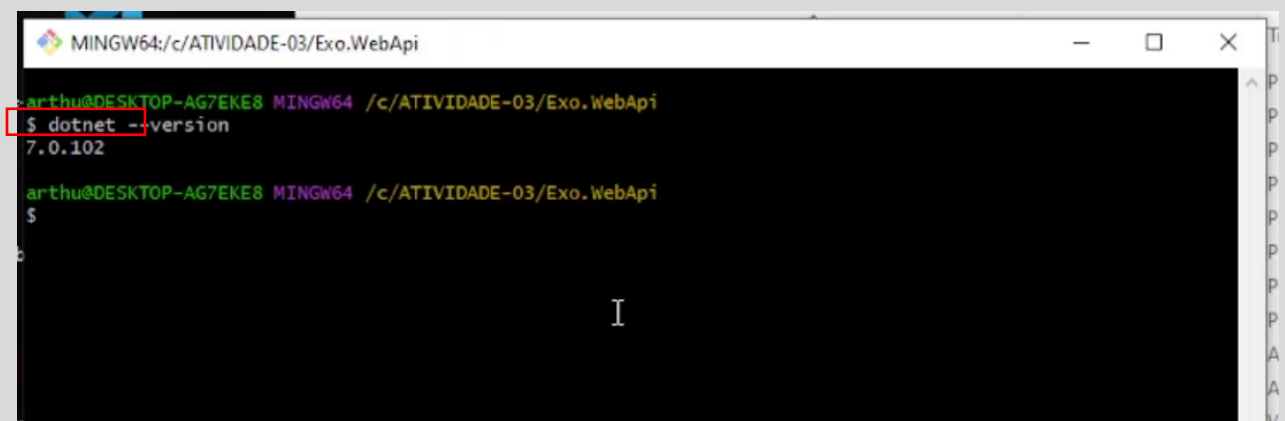
1. Abra a pasta que você usou para baixar a atividade e localize a **Exo.WebApi**. Clique com o botão direito dentro dela e escolha **Git Bash Here** para abrir o terminal.



2. No terminal que será aberto, digite o comando abaixo e dê **Enter** para verificar a versão do dotnet instalada em sua máquina.

```
dotnet --version
```

3. Certifique-se de que sua versão do dotnet seja a 6 ou superior. No nosso caso, a versão é 7.0.102, como mostra a imagem.

A screenshot of a terminal window titled 'MINGW64:/c/ATIVIDADE-03/Exo.WebApi'. The prompt is 'arthu@DESKTOP-AG7EKE8 MINGW64 /c/ATIVIDADE-03/Exo.WebApi'. The command '\$ dotnet --version' is entered, with 'dotnet' highlighted by a red box. The output is '7.0.102'. The prompt '\$' is shown again on the next line.

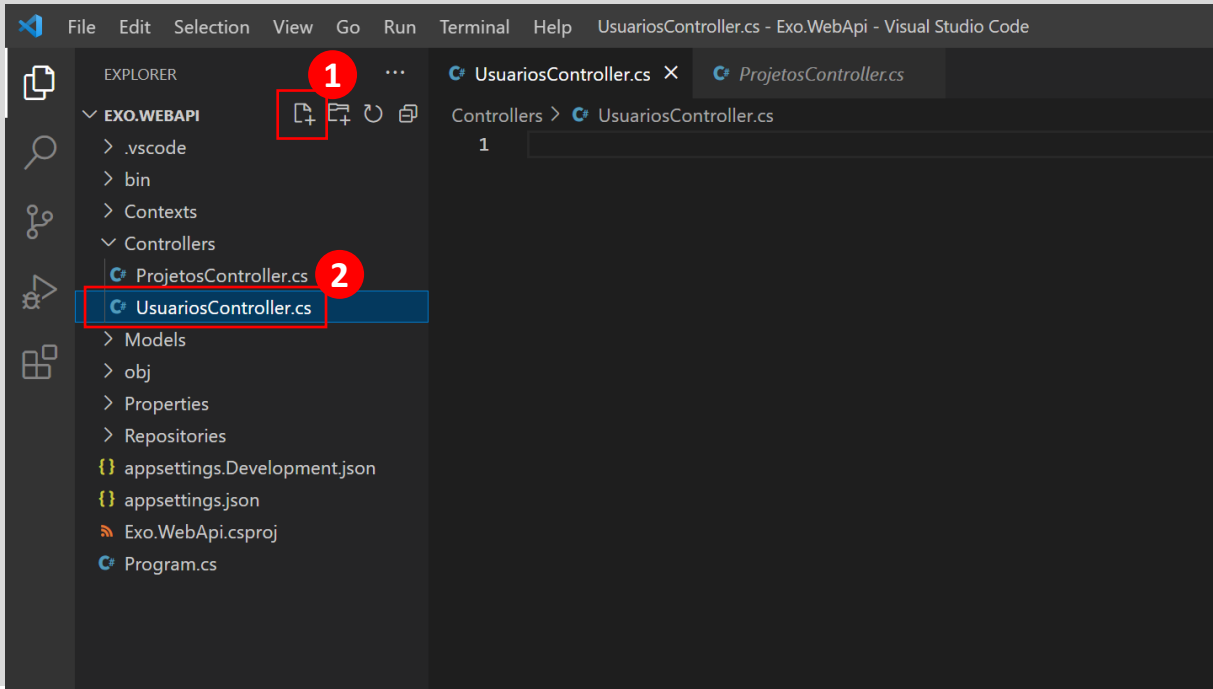
```
arthu@DESKTOP-AG7EKE8 MINGW64 /c/ATIVIDADE-03/Exo.WebApi
$ dotnet --version
7.0.102
arthu@DESKTOP-AG7EKE8 MINGW64 /c/ATIVIDADE-03/Exo.WebApi
$
```

4. Agora, digite o comando abaixo e dê **Enter** para abrir o VSCode com o projeto já aberto.

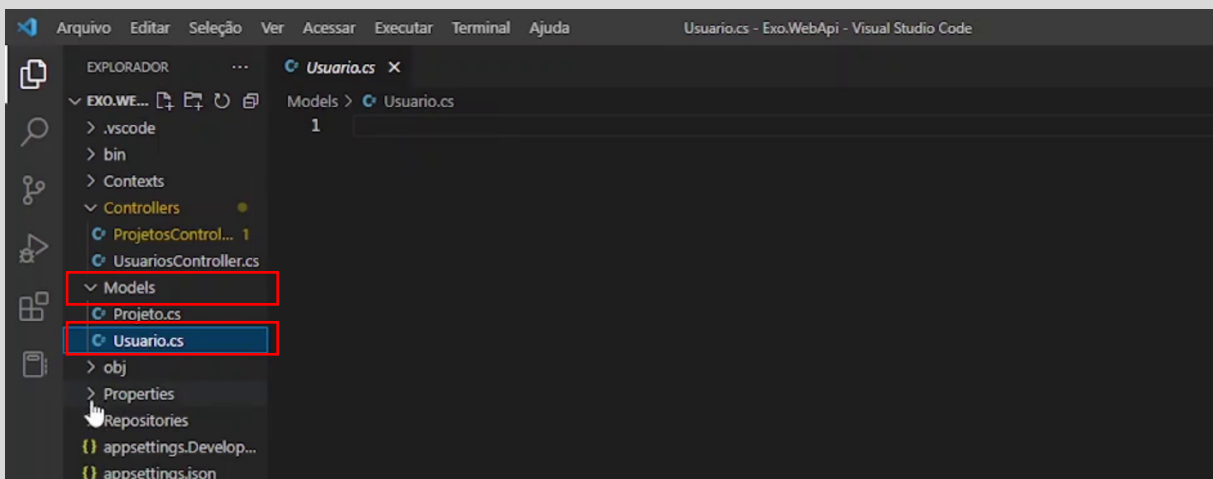
```
code .
```

Implementação dos códigos

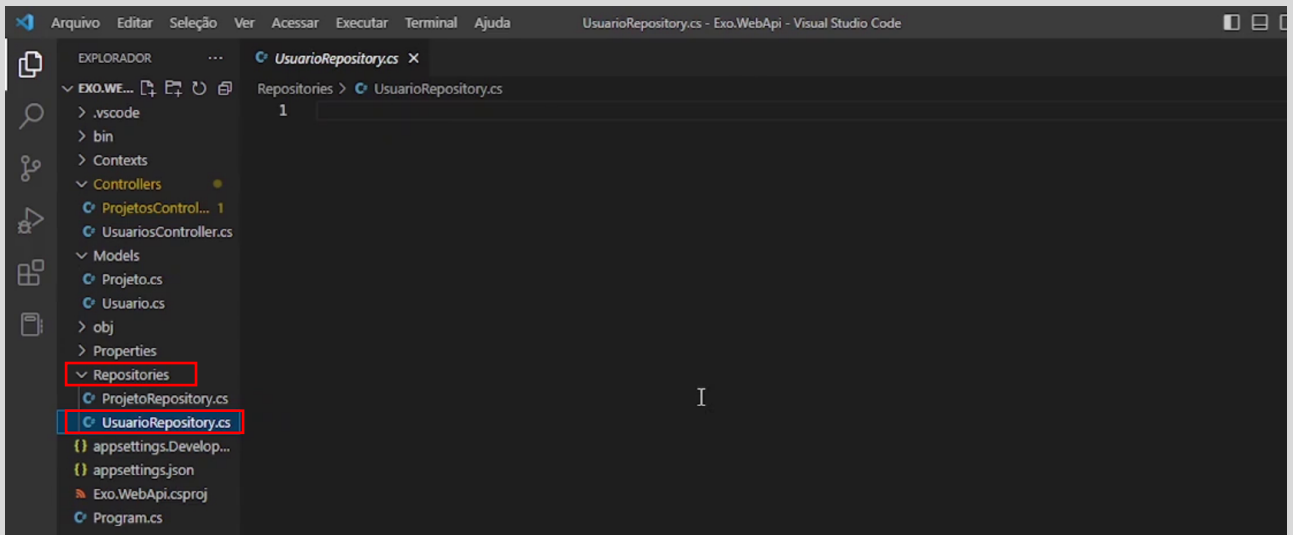
1. Com o VSCode aberto no projeto, clique na pasta **Controllers** e crie o controlador necessário ao projeto, clicando em **New File** (1) e nomeando-o **UsuariosController.cs** (2).



2. Realize o mesmo processo do passo 1 para criar a classe **Usuario.cs** dentro da pasta **Models**.



3. Novamente, realize o mesmo processo do passo 1 para criar a classe **UsuarioRepository.cs** dentro da pasta **Repositories**.



4. Abra o model **Usuario.cs** e substitua o código pelo que está abaixo:

```
namespace Exo.WebApi
{
    public class Usuario
    {
        public int Id { get; set; }
        public string Email { get; set; }
        public string Senha { get; set; }
    }
}
```

5. Abra o model **UsuariosController.cs** e insira o código a seguir:

```
using Exo.WebApi.Models;
using Exo.WebApi.Repositories;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using System;

namespace Exo.WebApi.Controllers
{
    [Produces("application/json")]
    [Route("api/[controller]")]
    [ApiController]
    public class UsuariosController : ControllerBase
    {
        private readonly UsuarioRepository _usuarioRepository;

        public UsuariosController(UsuarioRepository
usuarioRepository)
        {
            _usuarioRepository = usuarioRepository;
        }

        // get -> /api/usuarios
        [HttpGet]
        public IActionResult Listar()
        {
            return Ok(_usuarioRepository.Listar());
        }

        // post -> /api/usuarios
        [HttpPost]
        public IActionResult Cadastrar(Usuario usuario)
        {
            _usuarioRepository.Cadastrar(usuario);
            return StatusCode(201);
        }
    }
}
```

Continuação do código na próxima página >>>

>>> continuação do código da página anterior

```
// get -> /api/usuarios/{id}
[HttpGet("{id}")] // Faz a busca pelo ID.
public IActionResult BuscarPorId(int id)
{
    Usuario usuario = _usuarioRepository.BuscaPorId(id);
    if (usuario == null)
    {
        return NotFound();
    }
    return Ok(usuario);
}

// put -> /api/usuarios/{id}
// Atualiza.
[HttpPut("{id}")]
public IActionResult Atualizar(int id, Usuario usuario)
{
    _usuarioRepository.Atualizar(id, usuario);
    return StatusCode(204);
}

// delete -> /api/usuarios/{id}
[HttpDelete("{id}")]
public IActionResult Deletar(int id)
{
    try
    {
        _usuarioRepository.Deletar(id);
        return StatusCode(204);
    }
    catch (Exception e)
    {
        return BadRequest();
    }
}
}
```

6. Abra a classe **UsuarioRepository.cs** e insira o código abaixo:

```
using Exo.WebApi.Contexts;
using Exo.WebApi.Models;
using System.Collections.Generic;
using System.Linq;

namespace Exo.WebApi.Repositories
{
    public class UsuarioRepository
    {
        private readonly ExoContext _context;

        public UsuarioRepository(ExoContext context)
        {
            _context = context;
        }

        public Usuario Login(string email, string senha)
        {
            return _context.Usuarios.FirstOrDefault(u => u.Email ==
email && u.Senha == senha);
        }

        public List<Usuario> Listar()
        {
            return _context.Usuarios.ToList();
        }

        public void Cadastrar(Usuario usuario)
        {
            _context.Usuarios.Add(usuario);
            _context.SaveChanges();
        }
    }
}
```

Continuação do código na próxima página >>>

>>> continuação do código da página anterior

```
public Usuario BuscaPorId(int id)
{
    return _context.Usuarios.Find(id);
}

public void Atualizar(int id, Usuario usuario)
{
    Usuario usuarioBuscado = _context.Usuarios.Find(id);

    if (usuarioBuscado != null)
    {
        usuarioBuscado.Email = usuario.Email;
        usuarioBuscado.Senha = usuario.Senha;
    }

    _context.Usuarios.Update(usuarioBuscado);
    _context.SaveChanges();
}

public void Deletar(int id)
{
    Usuario usuarioBuscado = _context.Usuarios.Find(id);
    _context.Usuarios.Remove(usuarioBuscado);
    _context.SaveChanges();
}
}
```

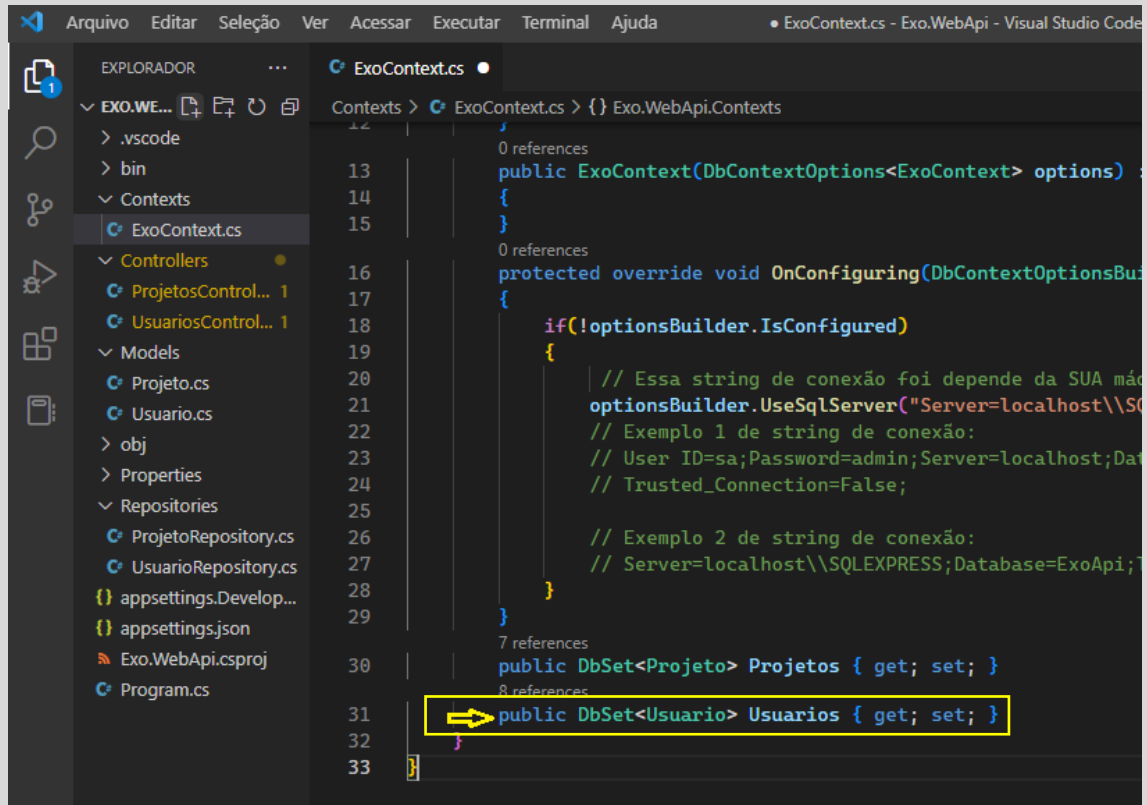
Importante

Além dos métodos do CRUD e o BuscaPorId(), nessa classe teremos o método login, que será utilizado na última atividade do projeto.



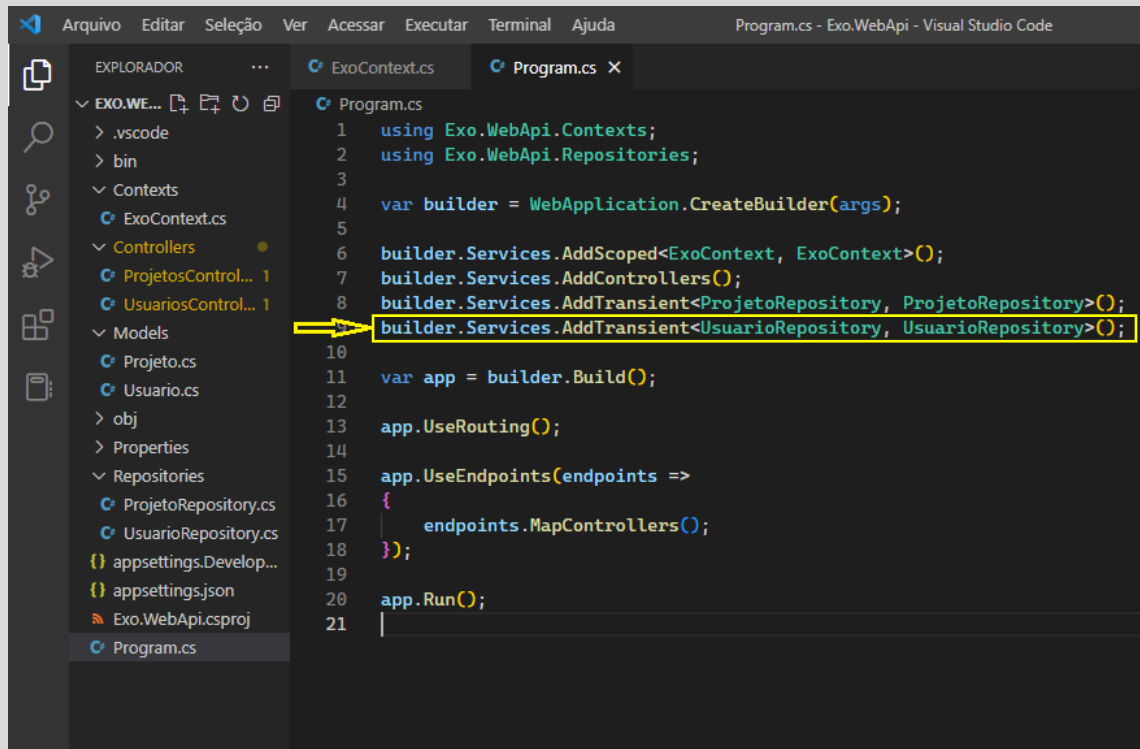
7. Na classe **ExoContext.cs**, inclua, no lugar indicado, a linha de código abaixo:

```
public DbSet<Usuario> Usuarios { get; set; }
```



8. Na classe **Program.cs**, inclua no lugar indicado a linha de código abaixo:

```
builder.Services.AddTransient<UsuarioRepository,  
UsuarioRepository>();
```

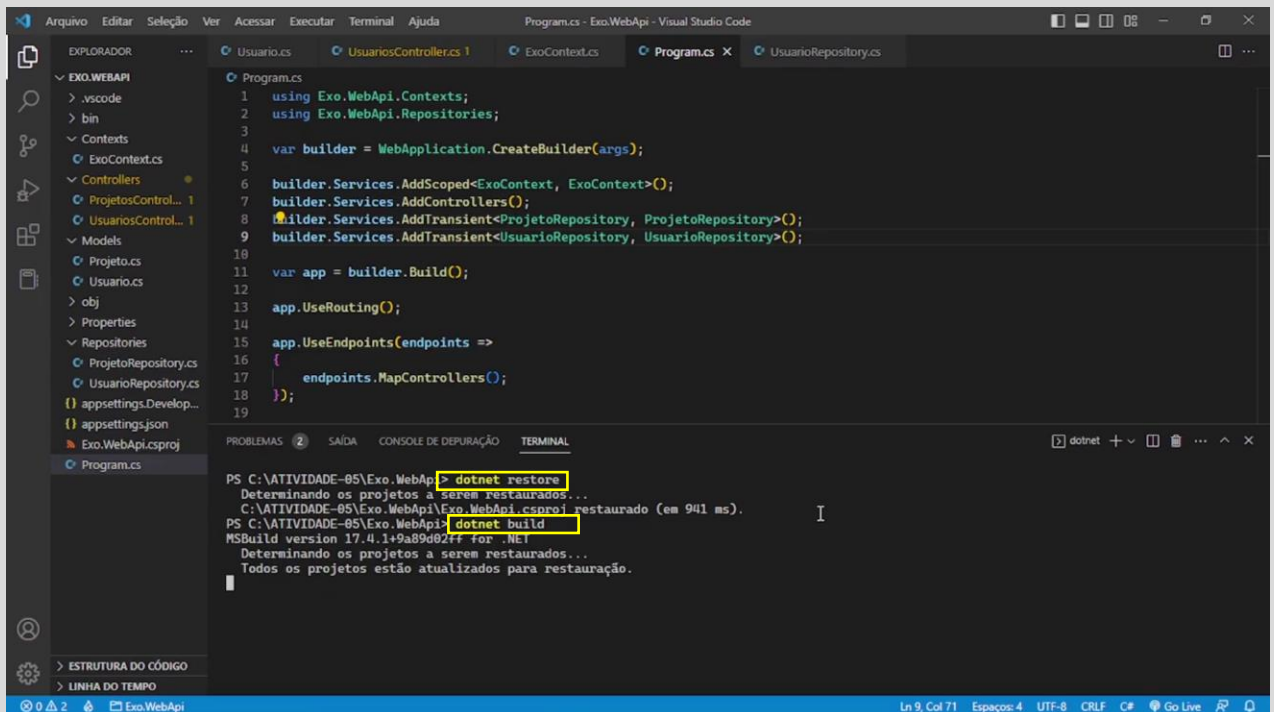


9. Nesse projeto já existem alguns pacotes instalados. Abra o terminal (menu **Terminal > Novo Terminal**) e, nele, digite o comando abaixo para restaurar o projeto e consolidar a instalação.

```
dotnet restore
```

10. Digite o comando abaixo para compilar a aplicação.

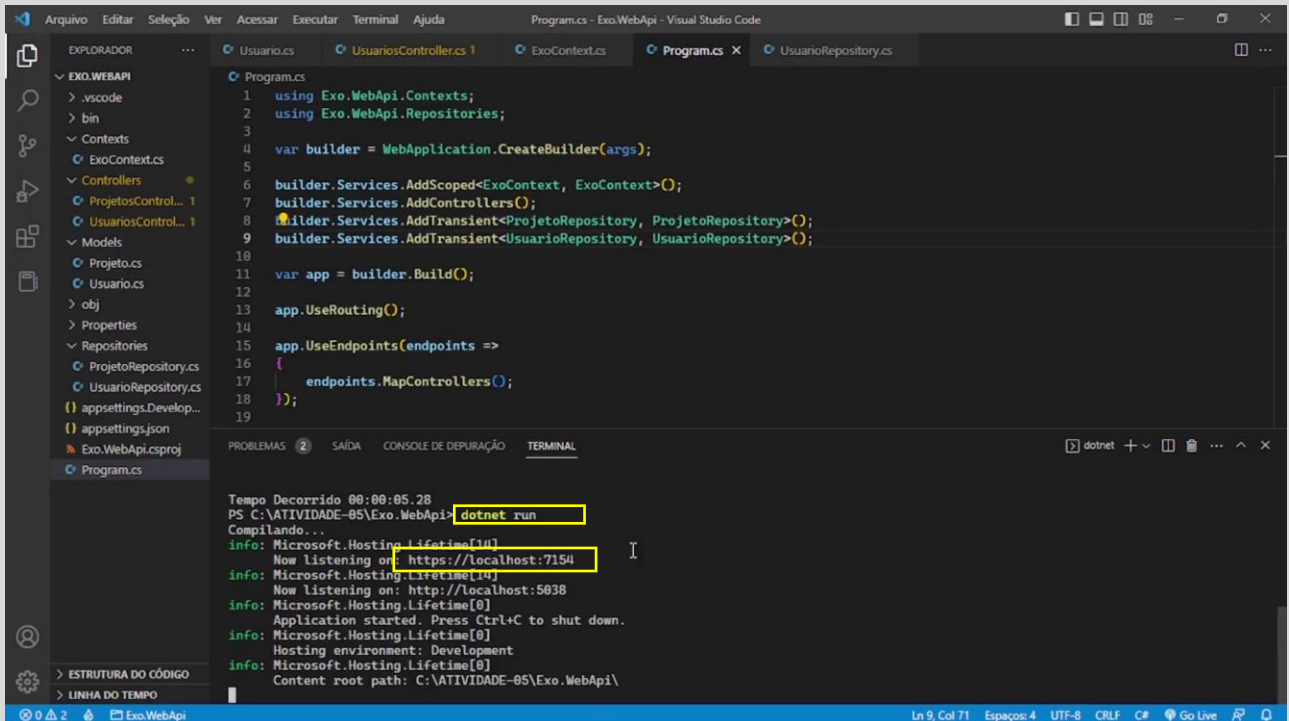
```
dotnet build
```



11. Digite o próximo comando para executar o projeto.

```
dotnet run
```

12. Anote o endereço que apareceu no terminal. Utilizaremos esse link seguido do sufixo **api/usuarios** para realizar a próxima etapa da atividade.



The screenshot shows the Visual Studio Code interface with the `Program.cs` file open. The code defines a Web API using the ASP.NET Core framework. The terminal at the bottom shows the output of the `dotnet run` command, indicating that the application is running on `https://localhost:7154`.

```
1 using Exo.WebApi.Contexts;
2 using Exo.WebApi.Repositories;
3
4 var builder = WebApplication.CreateBuilder(args);
5
6 builder.Services.AddScoped<ExoContext, ExoContext>();
7 builder.Services.AddControllers();
8 builder.Services.AddTransient<ProjetoRepository, ProjetoRepository>();
9 builder.Services.AddTransient<UsuarioRepository, UsuarioRepository>();
10
11 var app = builder.Build();
12
13 app.UseRouting();
14
15 app.UseEndpoints(endpoints =>
16 {
17     endpoints.MapControllers();
18 });
19
```

Terminal Output:

```
Tempo Decorrido 00:00:05.28
PS C:\ATIVIDADE-05\Exo.WebApi> dotnet run
Compilando...
info: Microsoft.Hosting.Lifetime[14]
Now listening on: https://localhost:7154
info: Microsoft.Hosting.Lifetime[14]
Now listening on: http://localhost:5038
info: Microsoft.Hosting.Lifetime[0]
Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
Content root path: C:\ATIVIDADE-05\Exo.WebApi\
```

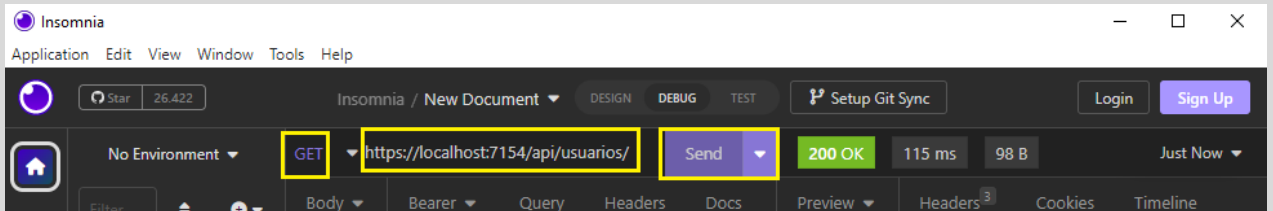
Você sabia?

Note que o resultado foi o mesmo da atividade anterior. Isso ocorre, porque esses dados fornecidos pela API podem ser visualizados – tanto quando chamamos via barra de endereços do **navegador**, quanto por alguma **página formatada** com front-end ou nos softwares que testam API como o **Insomnia** e o **Postman**.

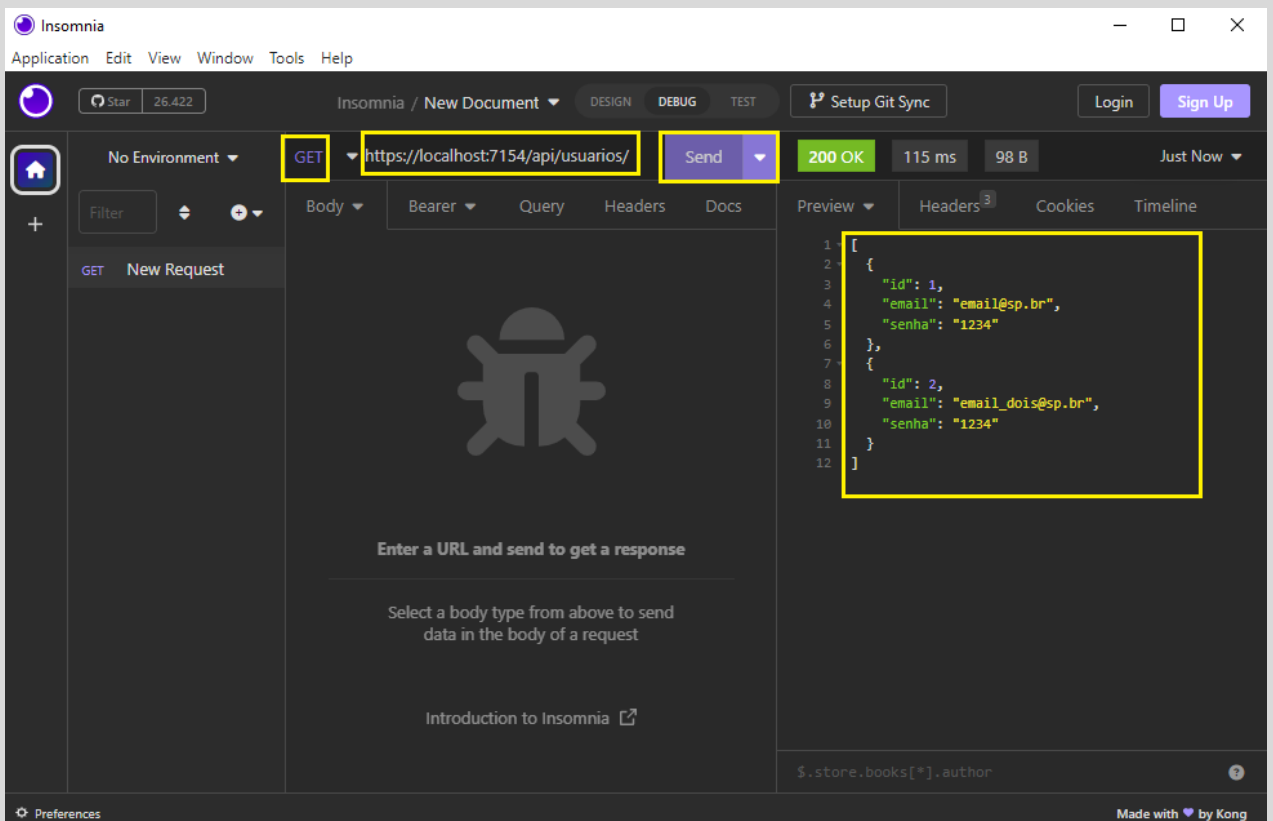


Utilizando o Insomnia

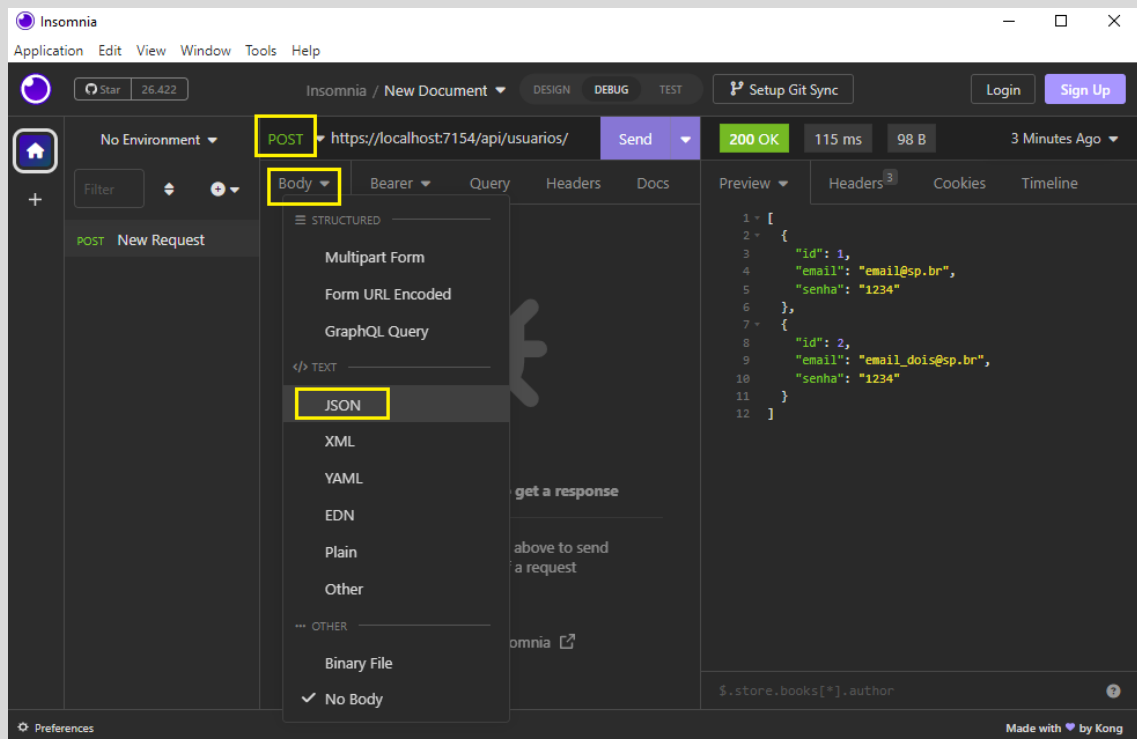
1. Baixe o programa **Insomnia** em <https://insomnia.rest/download>. Abra-o e configure os campos conforme o indicado abaixo.



2. Ao pressionar o botão **SEND**, o resultado aparecerá na janela à direita. O **200 OK** é o código de resposta para requisição bem-sucedida (caso ocorresse algum problema, a resposta seria outro código).

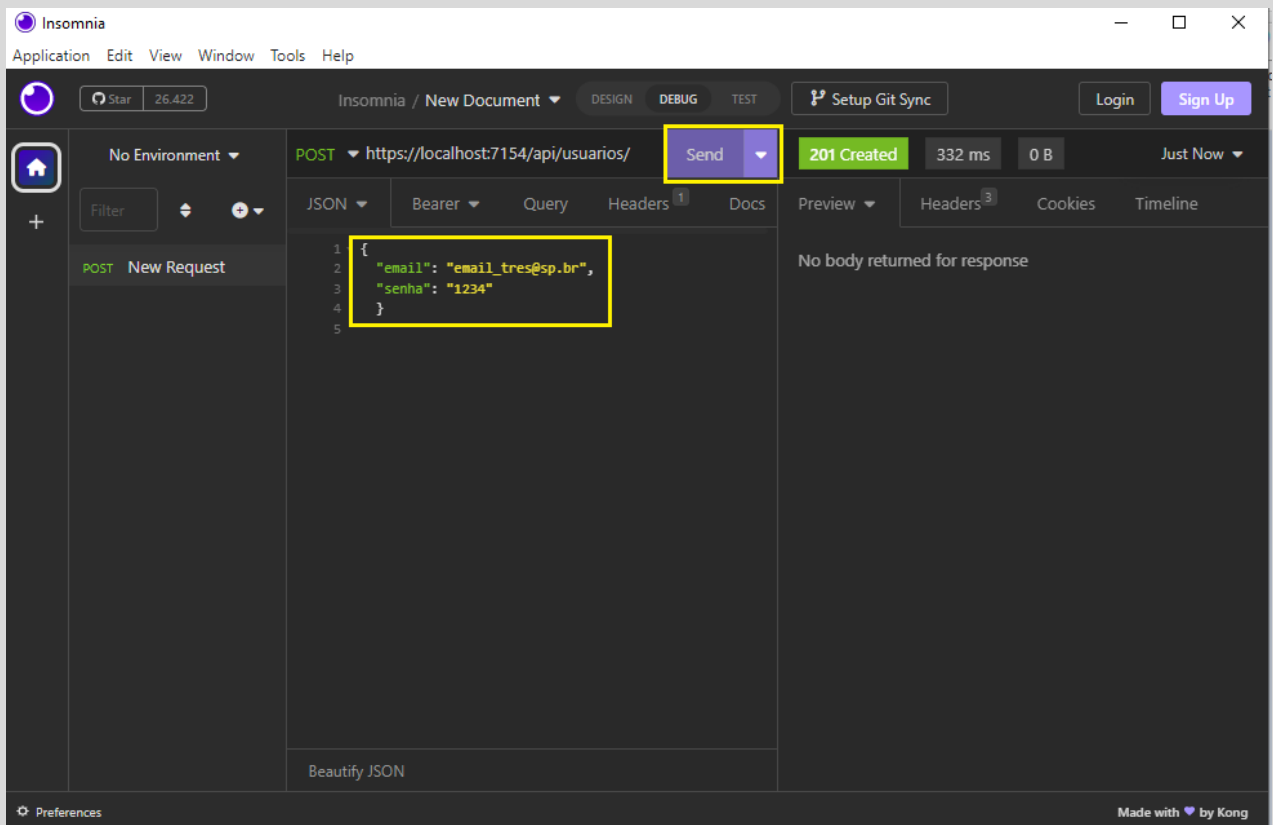


3. Agora, teste o método Cadastrar (POST). Escolha o método POST, insira a url **`https://localhost:7154/api/usuarios/`** e, em **Body**, troque para **JSON**, conforme a imagem:



4. Na janela JSON, insira o código abaixo para inserção de um novo cadastro no banco de dados da aplicação e clique em **SEND**. Aparecerá o código **201 Created** em verde, conforme programamos na função Cadastrar da classe **UsuariosController**.

```
{
  "email": "email_tres@sp.br",
  "senha": "1234"
}
```

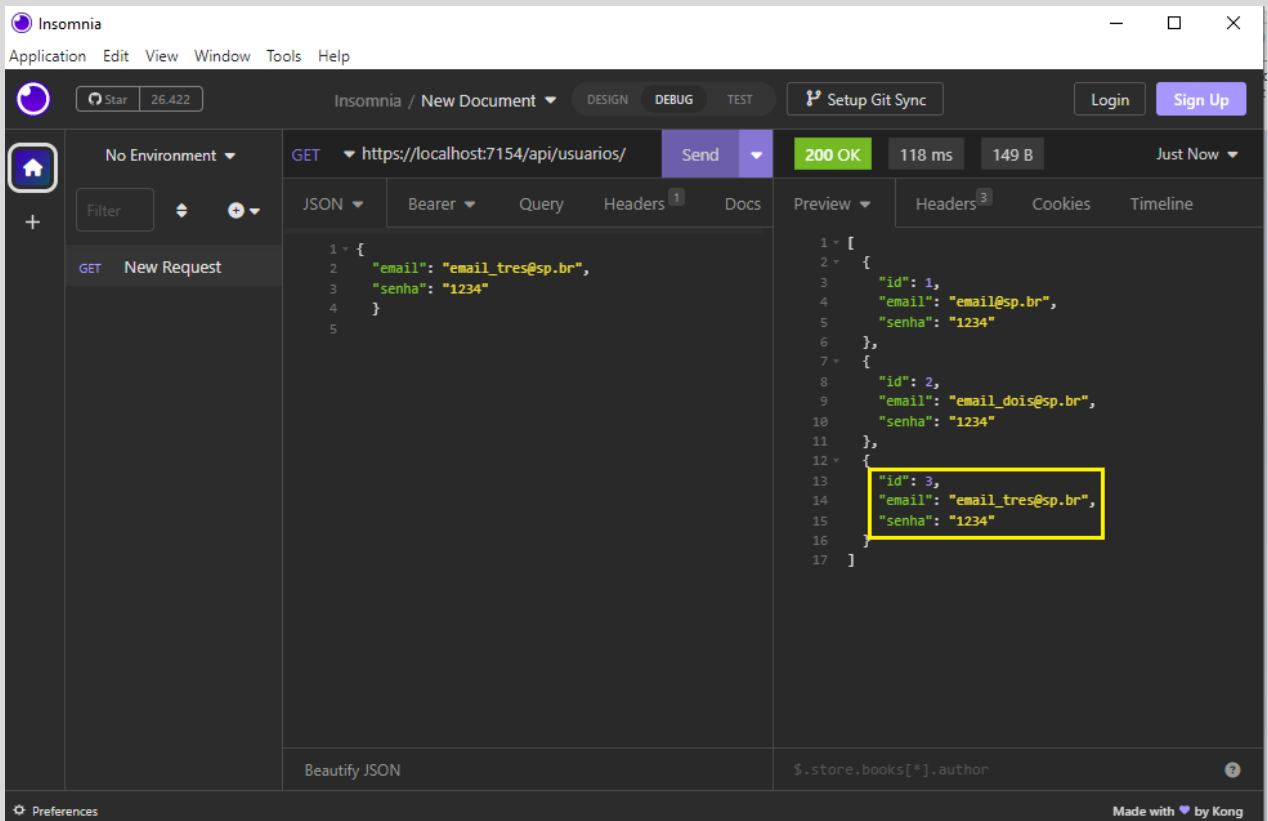


Importante

O campo **ID** não é inserido porque ele está configurado para autoincremento e, por isso, será colocado automaticamente.



5. Você pode consultar a listagem dos itens do banco de dados com o novo projeto alterando o método para GET e clicando em **SEND**. Eles aparecerão na janela lateral, conforme a imagem.



Você sabia?

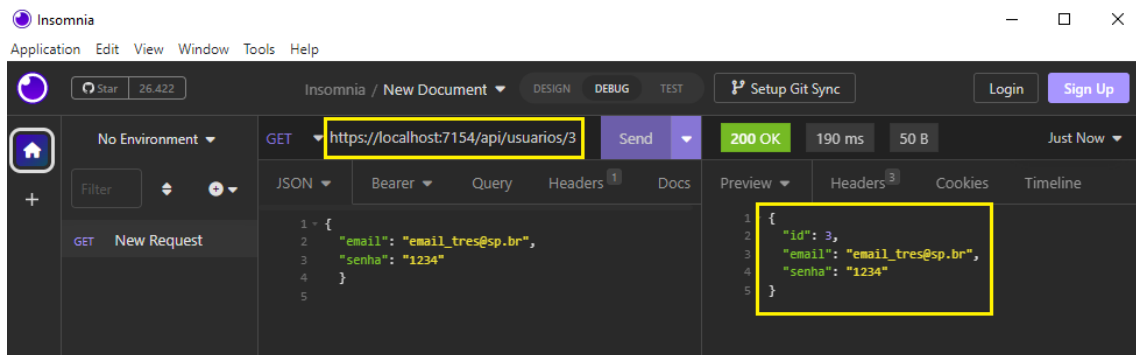
Você também pode consultar a listagem dos itens do banco de dados diretamente no **SSMS**.



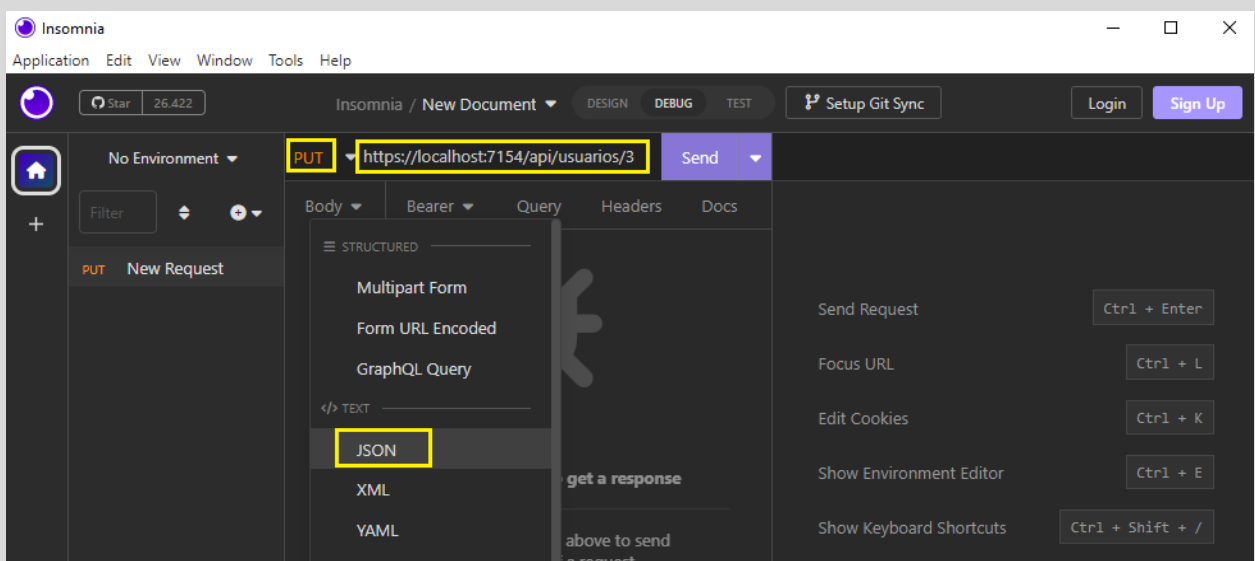
Dica!

Também é possível usar o **GET** para visualizar as ids individualmente. Para isso, insira a url

`https://localhost:7154/api/usuarios/3` – na qual **3** pode ser outro registro que se deseja consultar – e clique em **SEND**. Nesse exemplo, o resultado será o terceiro item do banco.

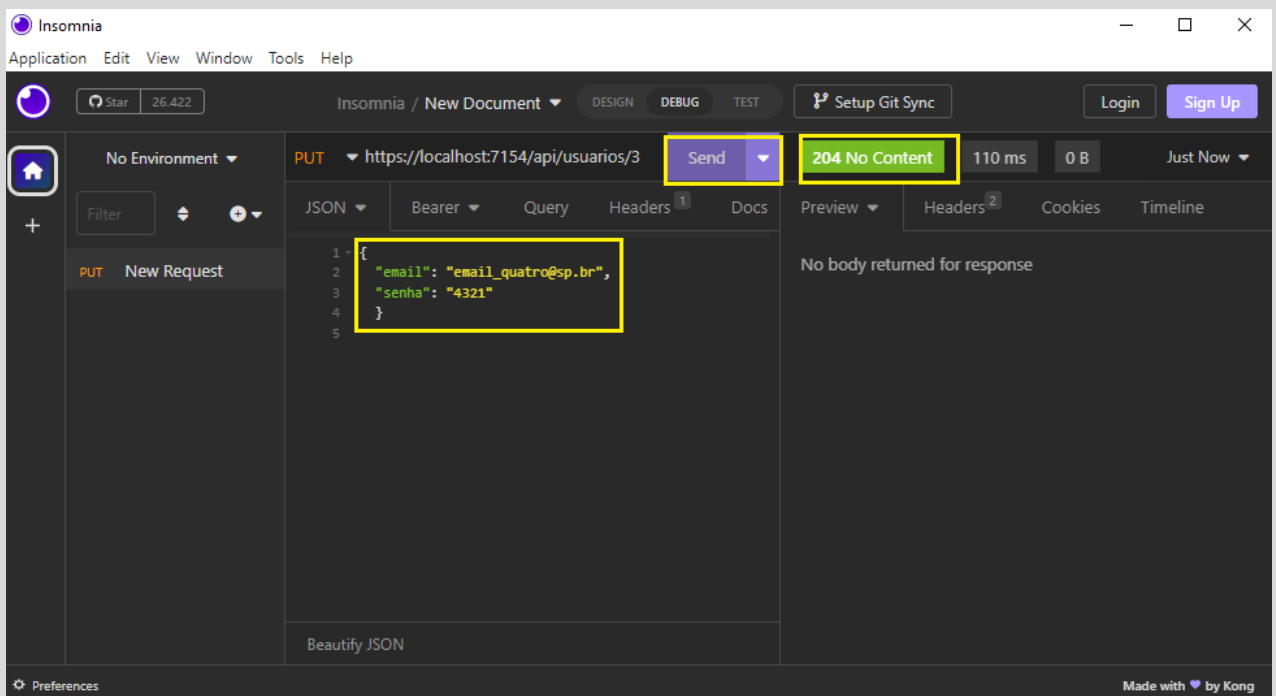


6. Agora, teste o método Atualizar (PUT) para alterar o último projeto que inserimos. Escolha o método PUT, insira a url **`https://localhost:7154/api/usuarios/3`** e mude o **Body** para **JSON**, conforme a imagem.



7. Substitua o código para o que está abaixo para atualização do item 3 e clique em **SEND**. O resultado será o **código 204** (a solicitação foi bem-sucedida), conforme programamos na função **Atualizar** da classe **UsuariosController.cs**.

```
{  
    "email": "email_quatro@sp.br",  
    "senha": "4321"  
}
```

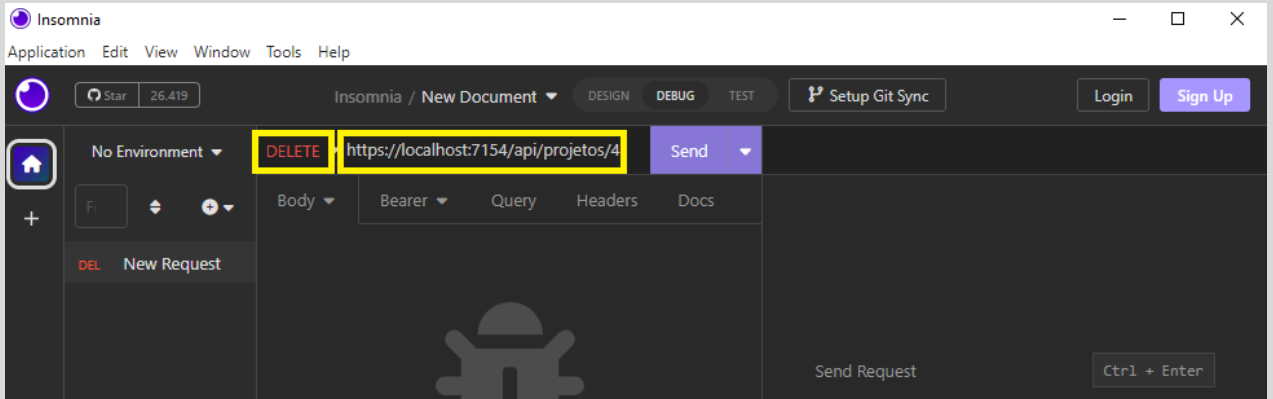


Importante

No exemplo, a url possui o identificador 3 no final, mas você pode atualizar outro projeto do seu banco.



8. Por último, testaremos a remoção no **Insomnia**. Escolha o método **DELETE**, insira a url **`https://localhost:7154/api/usuarios/3`** e clique em **SEND**. Será deletado o projeto com o **id 3**.



9. Você pode consultar a listagem dos itens do banco de dados sem o projeto deletado inserindo a url **`https://localhost:7154/api/usuarios/`** e clicando em **SEND**. Eles aparecerão na janela lateral, exceto o projeto deletado, conforme a imagem.

