LH Pets no Visual Studio

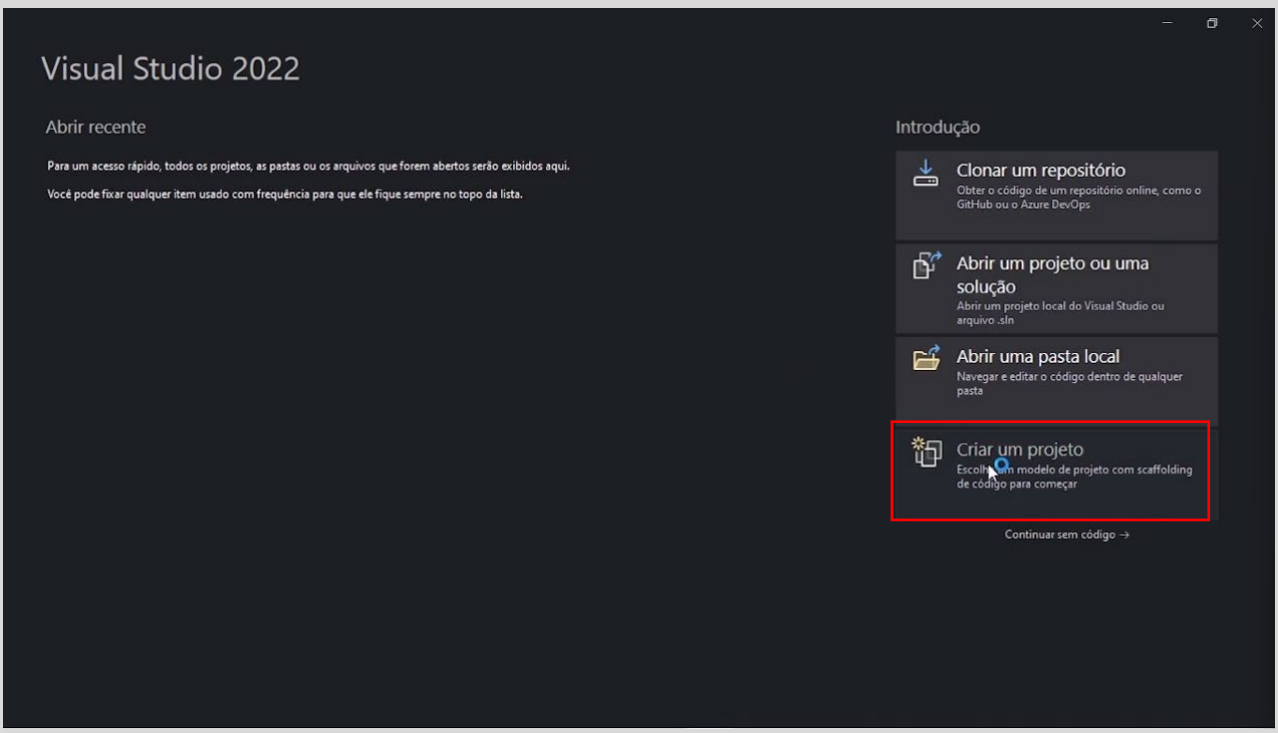
Neste vídeo, você vai acompanhar o desenvolvimento da aplicação LH Pets para clínica veterinária no Visual Studio 2022, utilizando um projeto do tipo aplicativo web do ASP.NET Core MVC.

O ASP.NET Core MVC é uma estrutura usada para a criação de aplicativos web e APIs, usando o padrão Model-View-Controller.

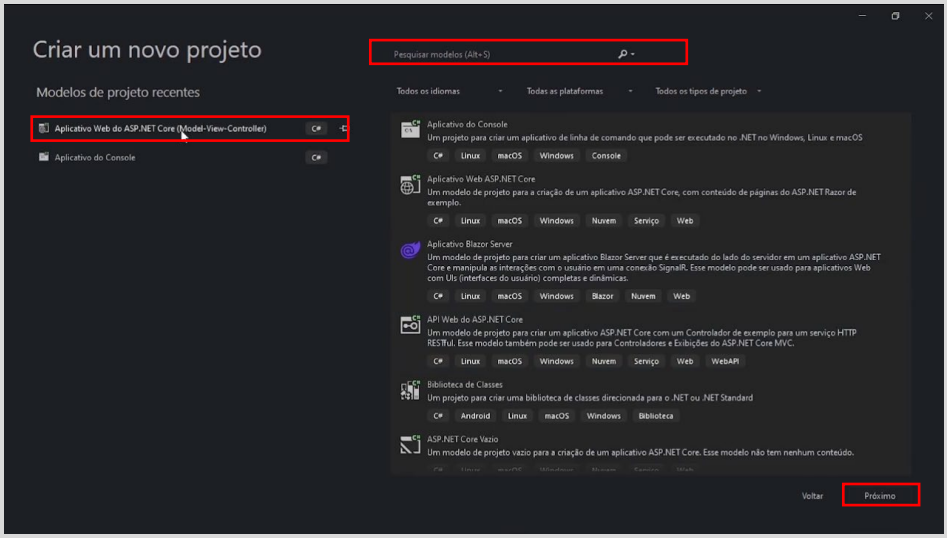
A aplicação será desenvolvida no Visual Studio. Fique atento para não confundir com o Visual Studio Code. Os ícones são parecidos, mas os programas são bem diferentes.

1. Com um duplo clique no **ícone**, inicie o Visual Studio.

2. Clique em **Criar um projeto**.

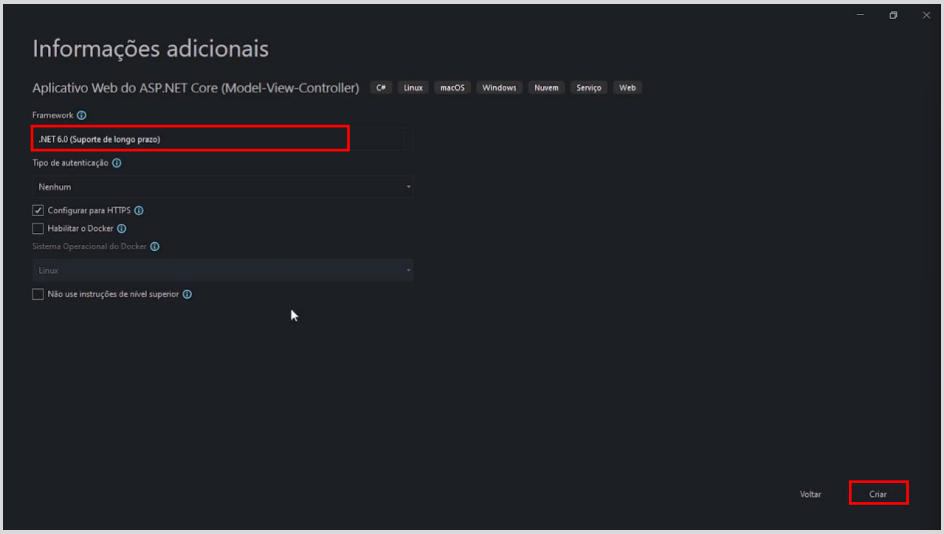


3. Verifique se o modelo de projeto está correto**: Aplicativo Web do ASP.NET Core (Model-View-Controller)** e clique em **Próximo**. Se não estiver, digite **mvc** na barra de pesquisa.

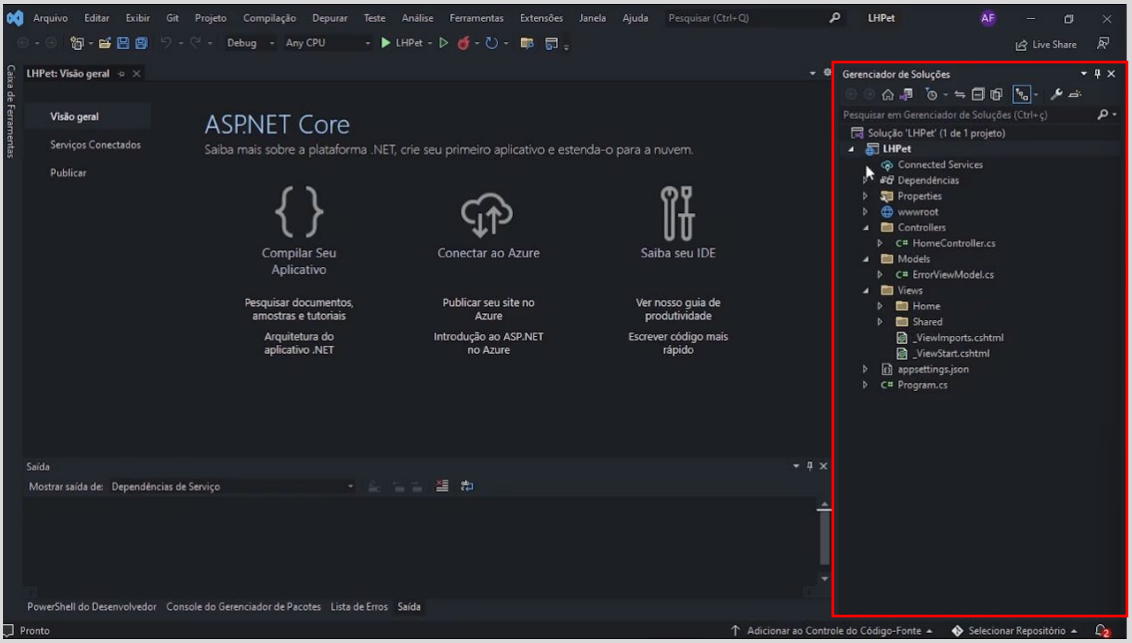


4. Coloque **LHPet** em Nome do projeto e escolha a **pasta do projeto** em Local. Clique em **Próximo**.

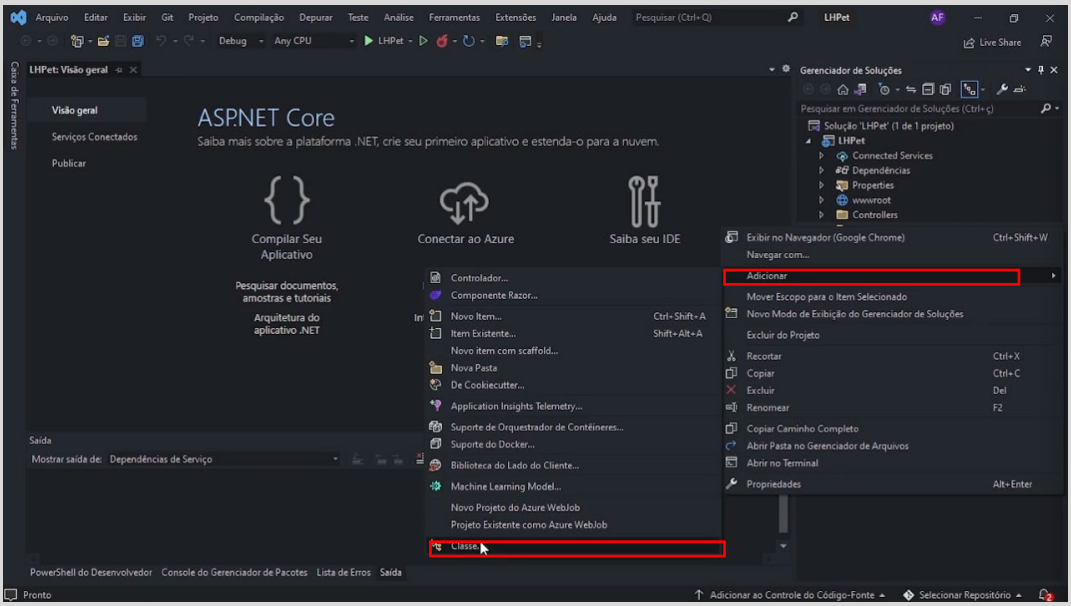
5. Verifique se a versão do Framework é 6.0 e deixe as configurações-padrão. Clique em **Criar**.



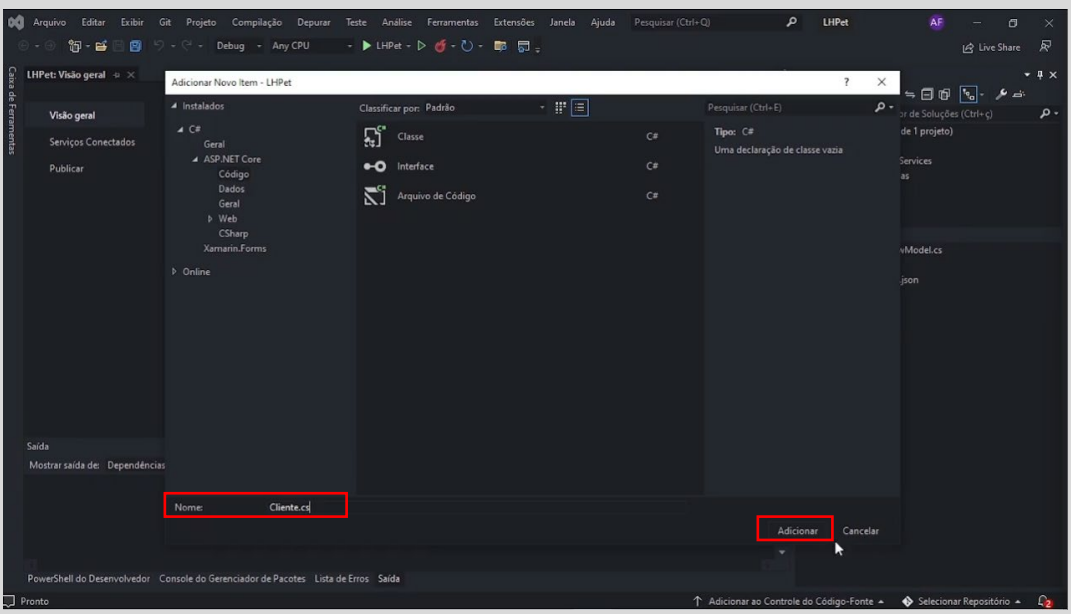
6. À direita está o “Gerenciador de Soluções”, com a estrutura do projeto: controllers, models e views.



7. No “Gerenciador de Soluções”, clique com botão direito em **Model**. Em seguida, selecione **Adicionar** e, depois, **Classe**.



8. Nomeie a classe Model criada como **Cliente.cs** e clique em **Adicionar**.



9. No arquivo **Clientes.cs**, digite o seguinte código:

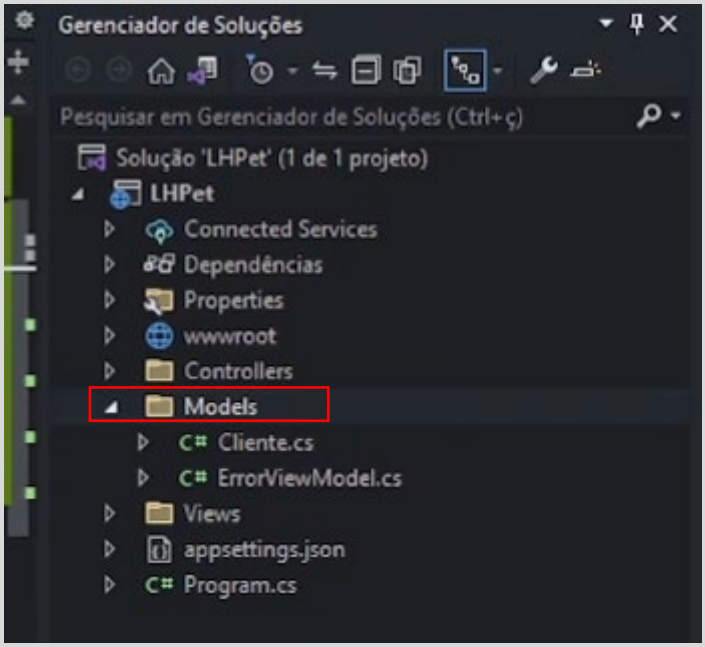
```
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace LHPet.Models
{
    [Table("Cliente")]
    public class Cliente
    {

            [Key]
            [Column("Id")]
            [Display(Name = "Id")]
            public int Id { get; set; }

            [Column("Nome")]
            [Display(Name = "Nome")]
            public string Nome { get; set; }

            [Column("Cpf")]
            [Display(Name = "Cpf")]
            public string Cpf { get; set; }

            [Column("Email")]
            [Display(Name = "Email")]
            public string Email { get; set; }

            [Column("Paciente")]
            [Display(Name = "Paciente")]
            public string Paciente { get; set; }

        }

    }
```

Nesse código, importamos as bibliotecas necessárias com **using**, usamos o **namespace** da aplicação, anotamos que essa classe vai formar uma tabela, determinamos a coluna **Id** como chave primária (**Key**) e definimos como o dado vai aparecer. Então, definimos o restante das colunas e como será mostrado o dado.

Dica!

Use o atalho Ctrl + . para acessar dicas de como solucionar erros no código.

10. A próxima etapa é criar a classe Contexto em Model. Repita os passos para criar classes: clique com botão direito em **Model**. Em seguida, selecione **Adicionar** e depois **Classe**. Nomeie a classe criada como **Contexto.cs**.



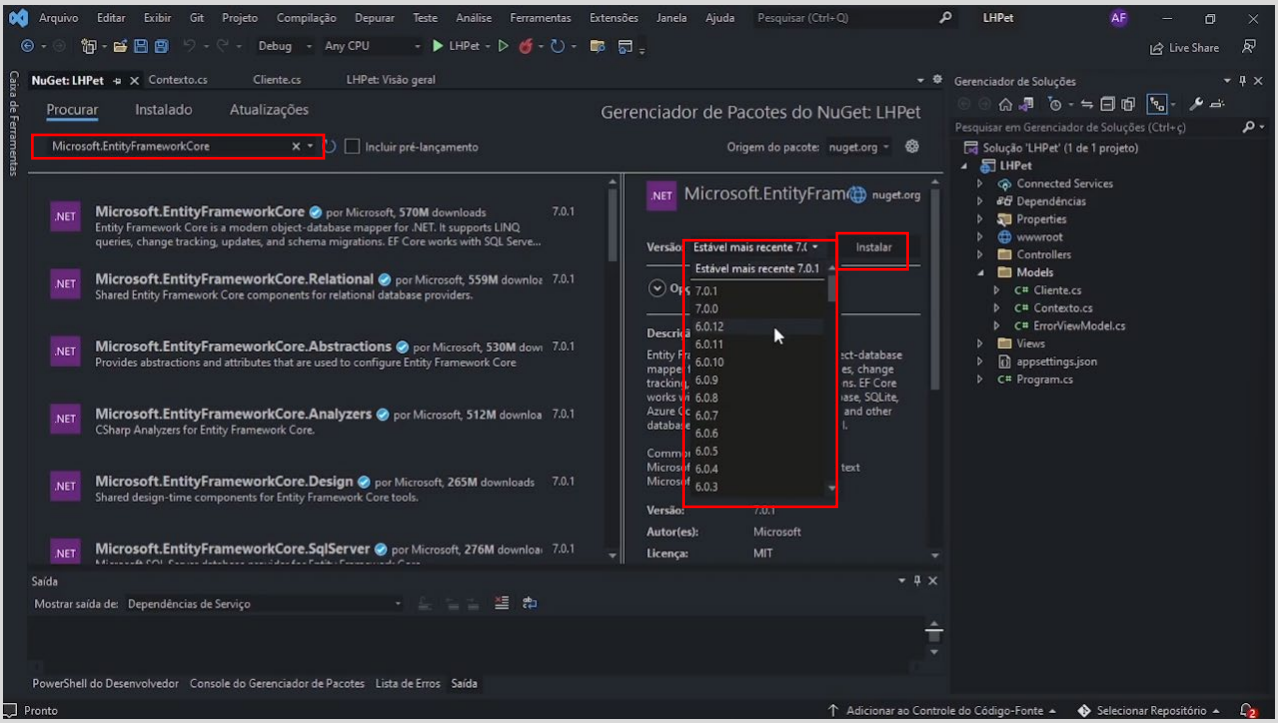
Essa classe vai ser responsável pela transação com o banco de dados e com os controllers, como um arquivo de guia ou apoio às outras classes.

11. No arquivo **Contexto.cs**, digite o seguinte código:

```
using Microsoft.EntityFrameworkCore;


namespace LHPet.Models
{
    public class Contexto : DbContext
    {
        public Contexto(DbContextOptions<Contexto> options) : base(options)
        {

        }
        public DbSet<Cliente> Cliente { get; set; }

    }
}
```

O Visual Studio vai dar alguns avisos de erros de referência devido à falta de pacotes e bibliotecas. Existe um gerenciador de bibliotecas no Visual Studio chamado **NuGet**, que sugere os pacotes necessários para o funcionamento do sistema. Basta usar o atalho **Ctrl + .** e selecionar **Instalar o pacote...** E, em seguida, **Instalar com o gerenciador de pacotes**.

12. Na janela do gerenciador de pacotes, vão aparecer algumas sugestões. Caso elas não apareçam, busque o pacote pelo **nome**, escolha a **versão** dele e clique em **Instalar**.

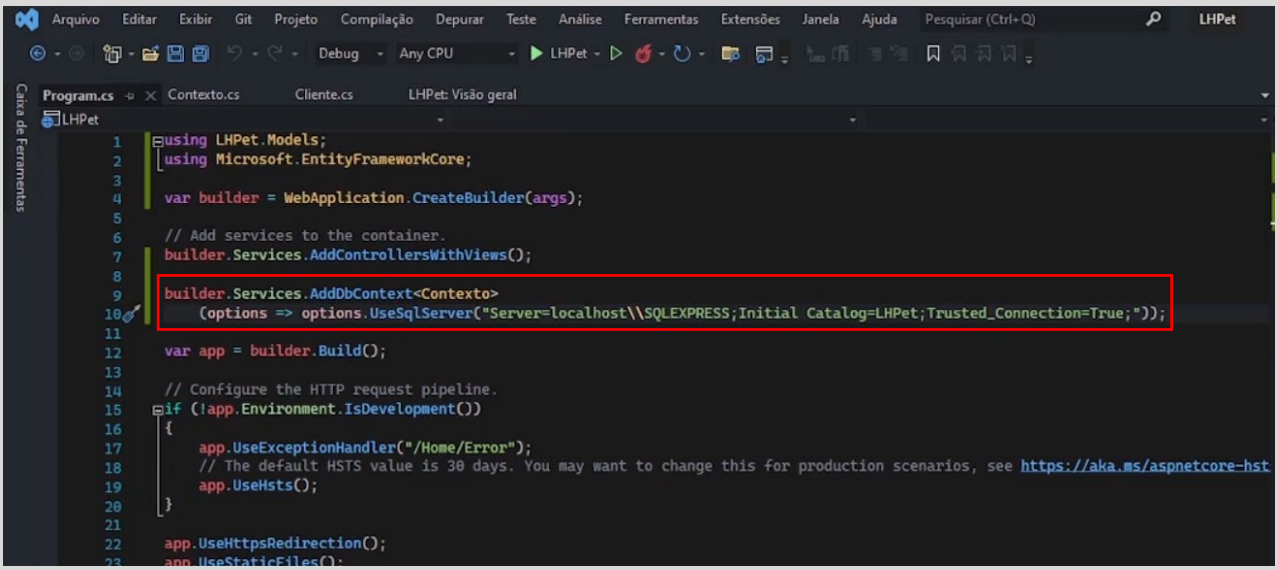


Instale os seguintes packages:

- Com.H.EF.SqlServer" Version="0.1.0"
- Microsoft.EntityFrameworkCore.Relational Version="6.0.12"
- Microsoft.EntityFrameworkCore.SqlServer Version="6.0.12"
- Microsoft.EntityFrameworkCore.Tools Version="6.0.12“
- Microsoft.VisualStudio.Web.CodeGeneration.Design Version="6.0.11"

13. Abra o arquivo da classe Program.cs e, após o comando **builder.Services.AddControllersWithViews();**, insira o seguinte código:

```
builder.Services.AddDbContext<Contexto>
    (options => options.UseSqlServer("Server=localhost\\SQLEXPRESS;Initial
Catalog=LHPet;Trusted_Connection=True;"));
```

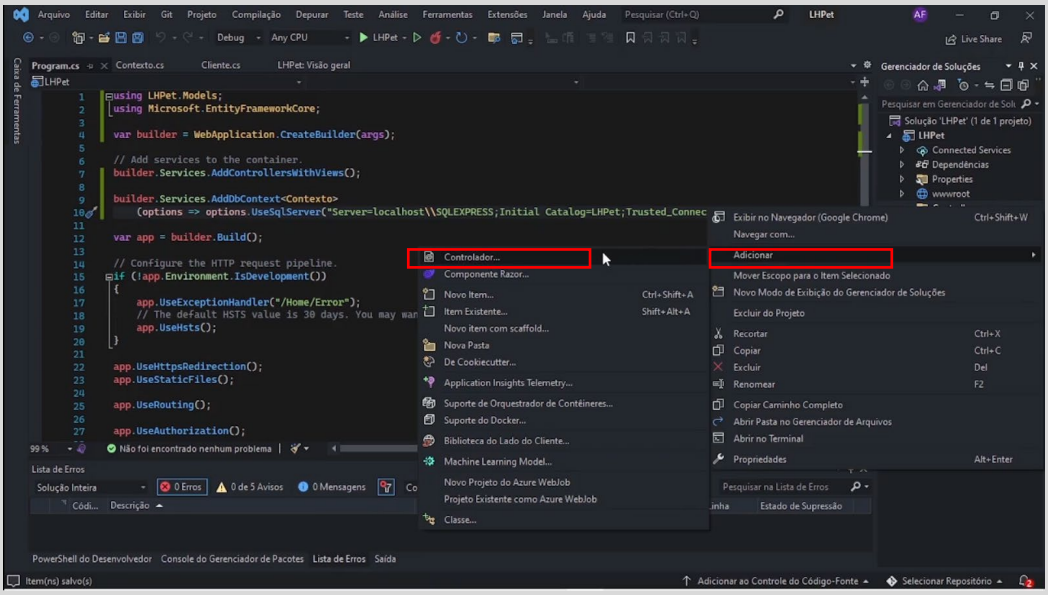


Esse bloco instala e abre a **string de conexão** com o banco de dados. Essa string aparece na instalação do banco de dados ou no visualizador do banco de dados.

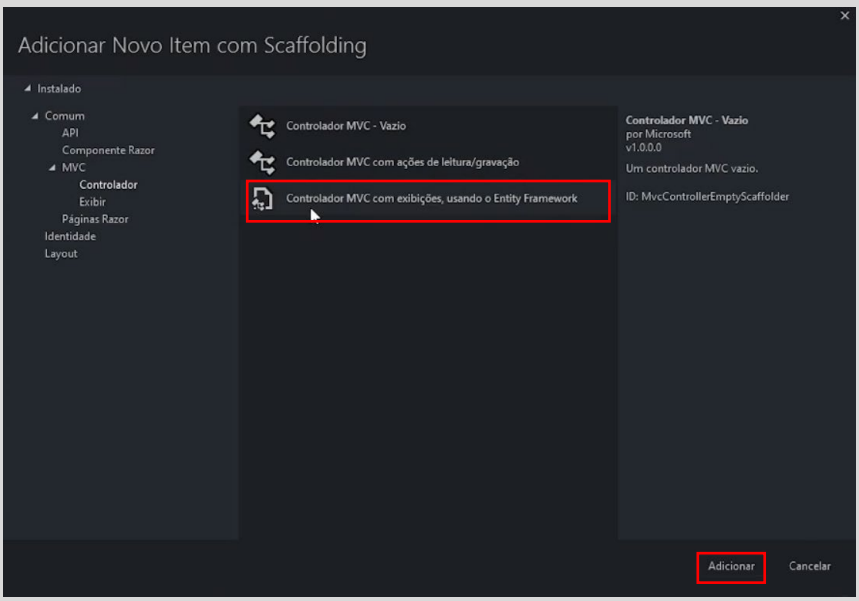
Verifique se as bibliotecas estão referenciadas no início do Program.cs.

```
using Microsoft.EntityFrameworkCore;
using LHPet.Models.Models;
```

14. A próxima etapa é criar uma classe em Controller. Então, clique com botão direito em **Controller**, em seguida, selecione **Adicionar** e depois **Controlador**.



15. Na janela do controlador, selecione **Controlador MVC com exibições, usando o Entity Framework**. Clique em **Adicionar**.

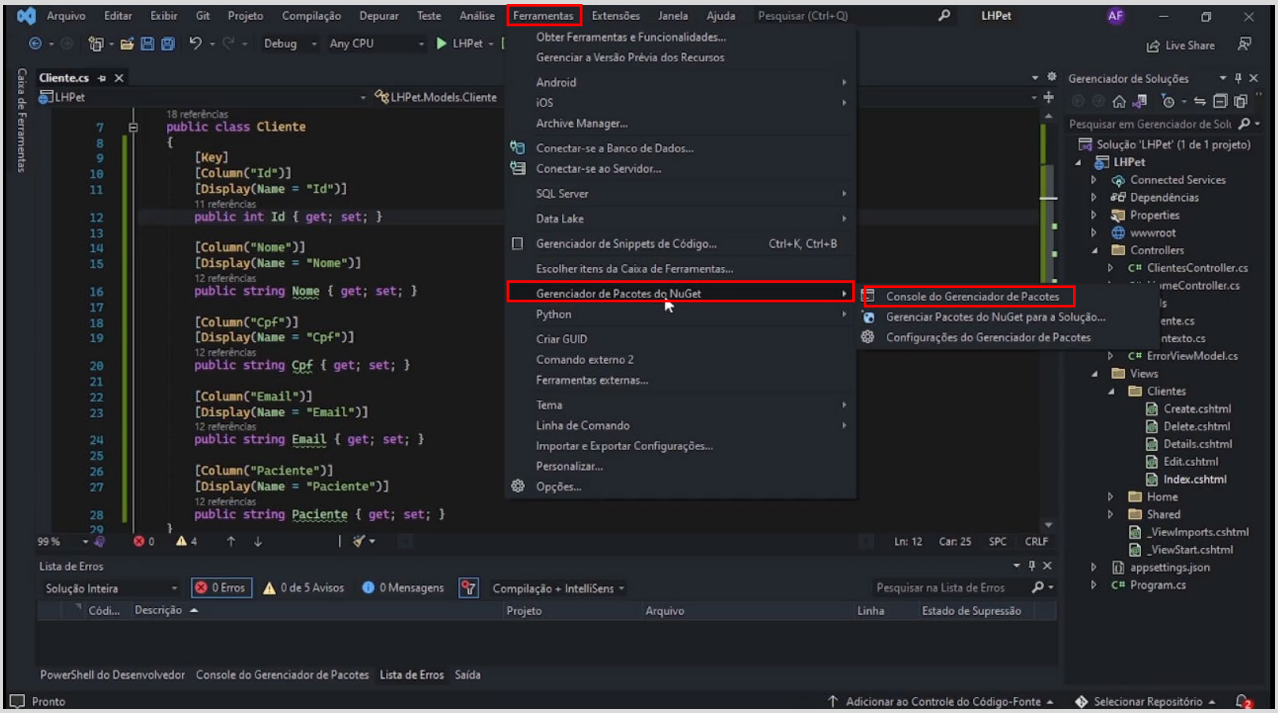


16. Na janela seguinte, deixe **Cliente (LHPet.Model)** para Classe do modelo e **Contexto (LHPet.Model)** para Classe de contexto de dados. Clique em **Adicionar**. Para nome do controller, digite **ClientesController**.

No arquivo ClientesController.cs estão todos os métodos de CRUD (create-read-update-delete), que manipulam dados em banco de dados.

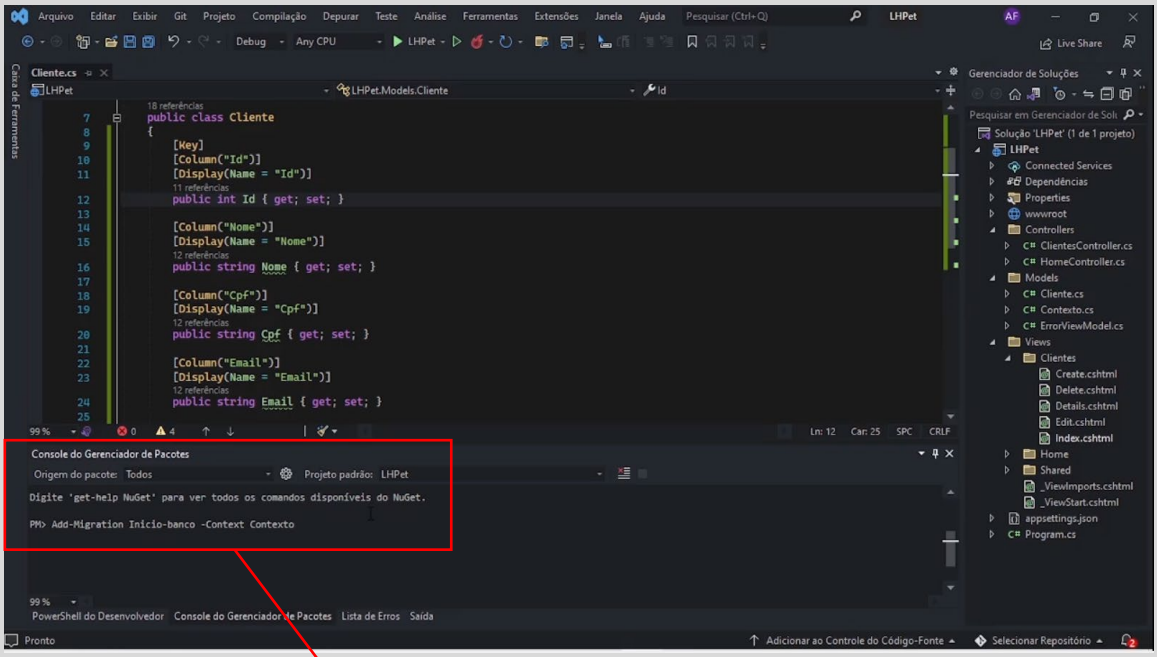
Além disso, o Scaffolding cria dentro da pasta Views um arquivo para cada ação CRUD.

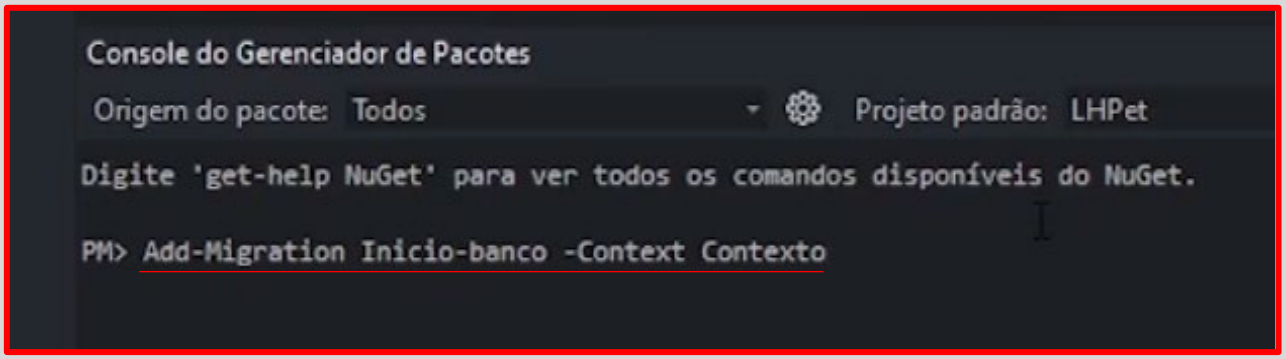
17. Agora você vai criar o banco de dados. Antes disso, instale os pacotes necessários. No menu superior, vá em **Ferramentas** e, depois, em **Gerenciador de Pacotes de NuGet** e, então, em **Console do Gerenciador de Pacotes**.



18. No console do gerenciador de pacotes, digite o código:

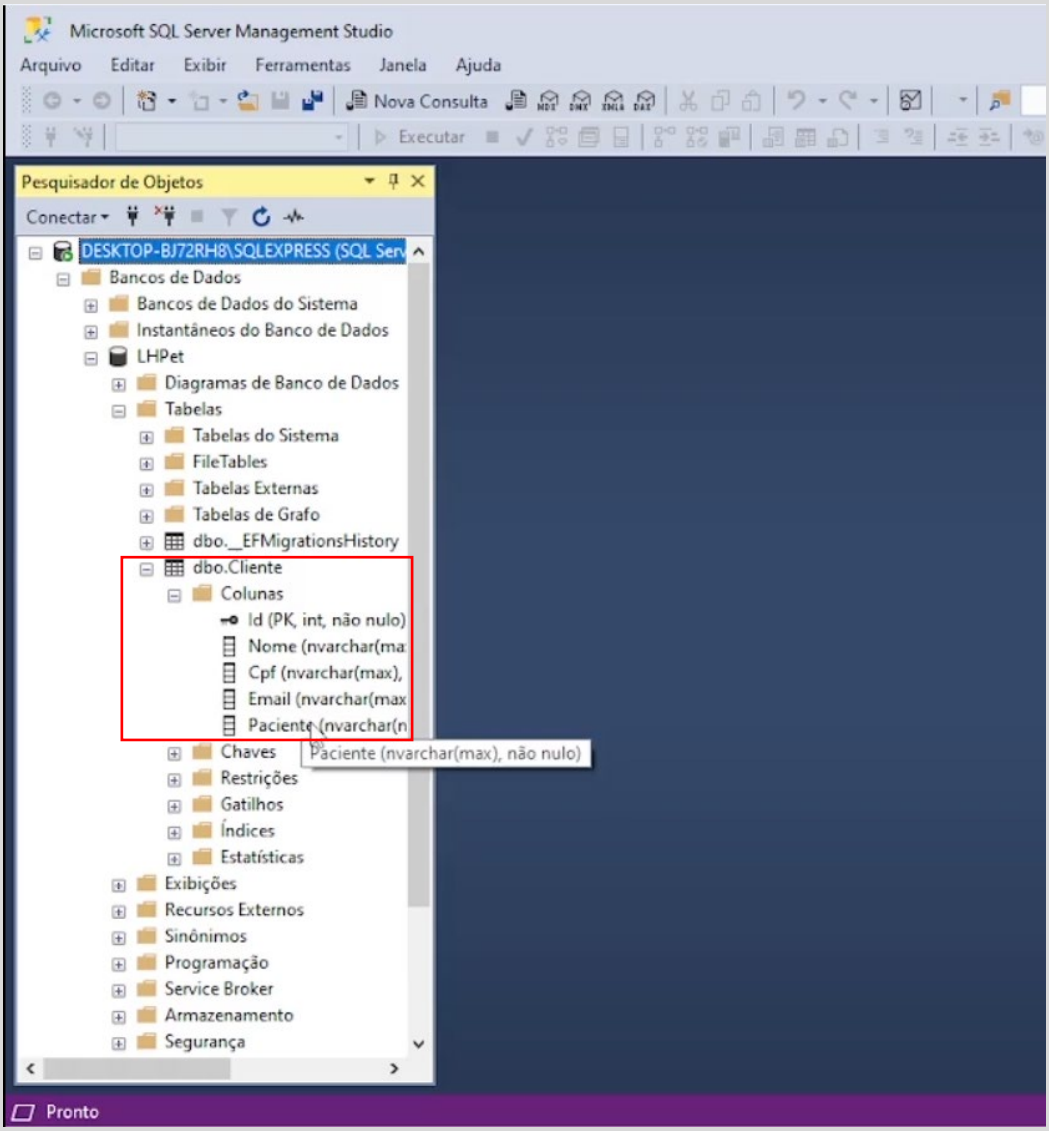
```
Add-Migration Inicio-banco -Context Contexto
```





Dê **Enter** e aguarde a criação da **migration**.

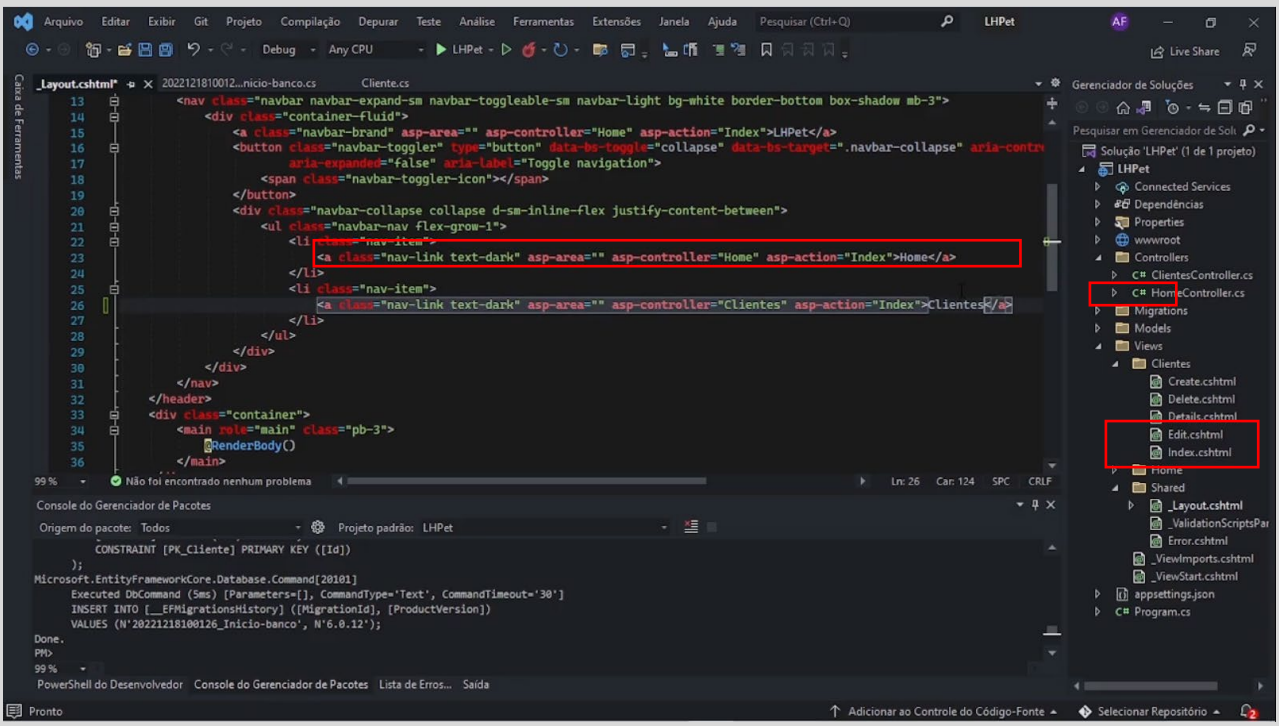
19. Entre no SSMS e verifique se o banco foi criado conforme definido na classe Clientes.cs.



20. No “Gerenciador de Soluções”, clique em **Views**, depois na pasta **Shared** e então no arquivo **_Layout.cshtml**.

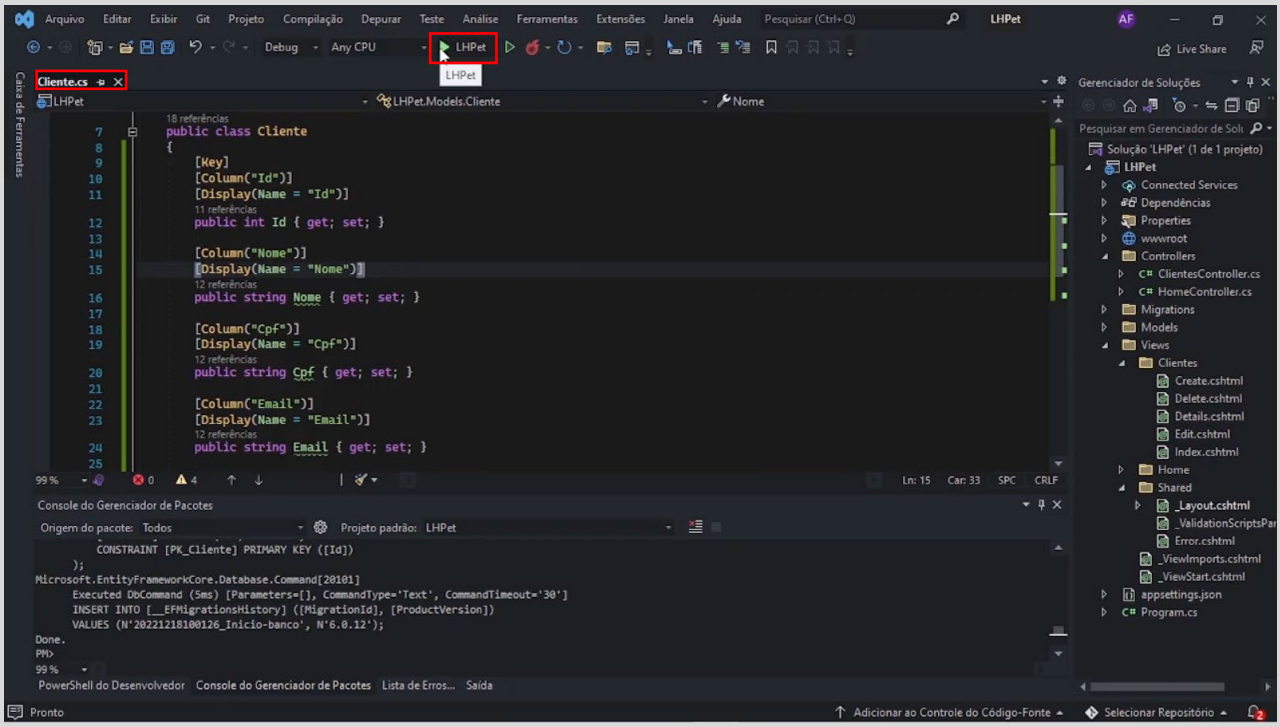
Na linha 26, altere alguns trechos do código conforme o seguinte:

```
 <a class="nav-link text-dark" asp-area="" asp-
controller="Clientes" asp-action="Index">Clientes</a>
```

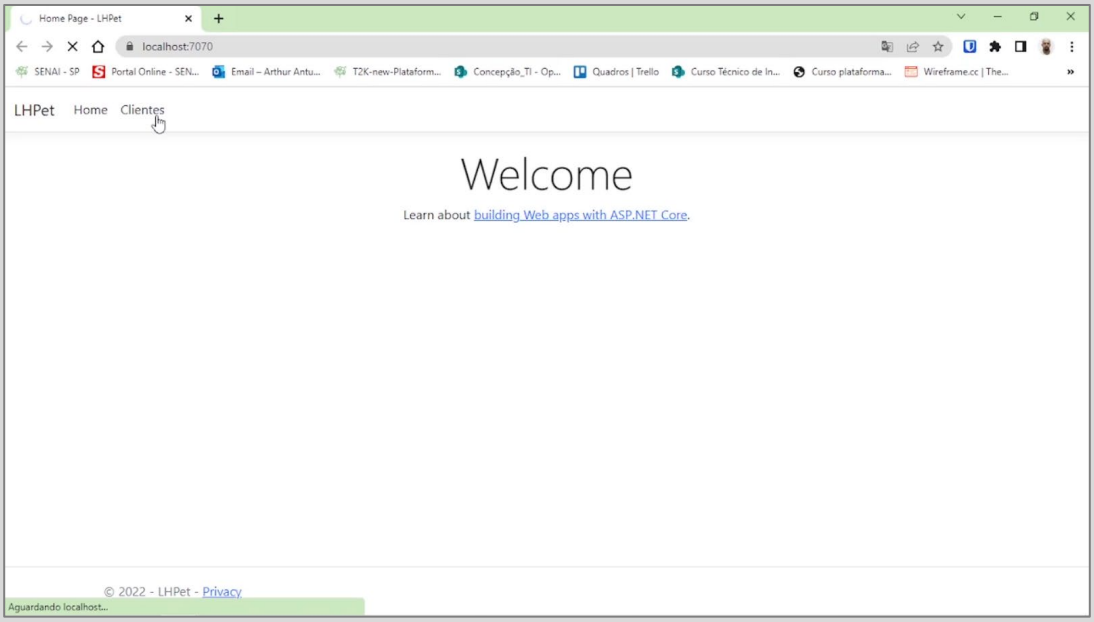


Salve o arquivo.

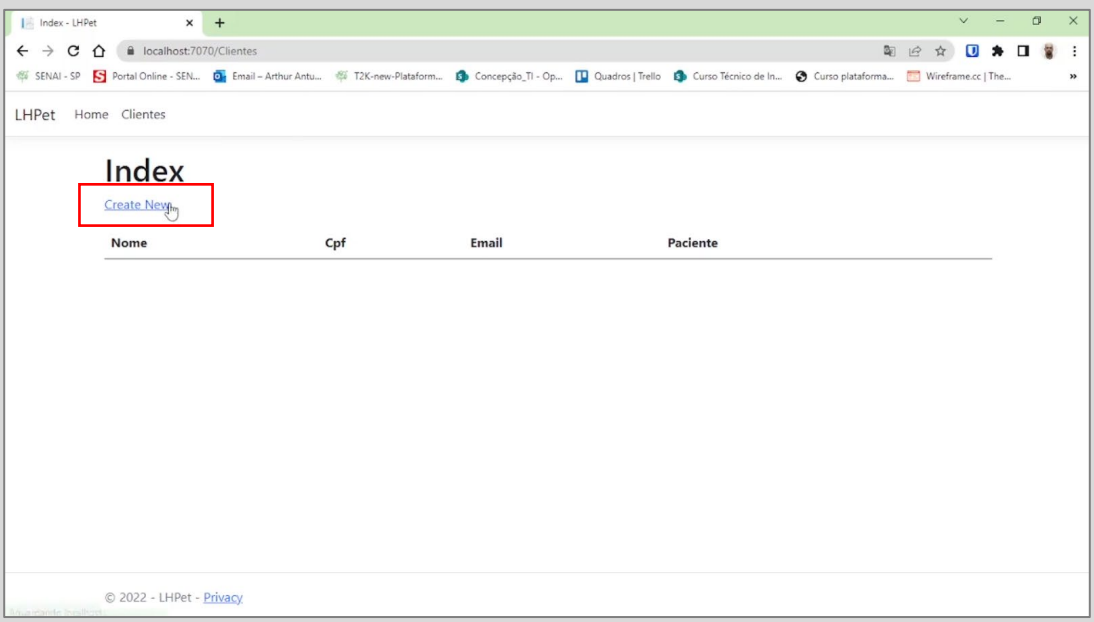
21. Abra o arquivo Clientes.cs e execute-o, clicando no triângulo verde abaixo do menu superior.



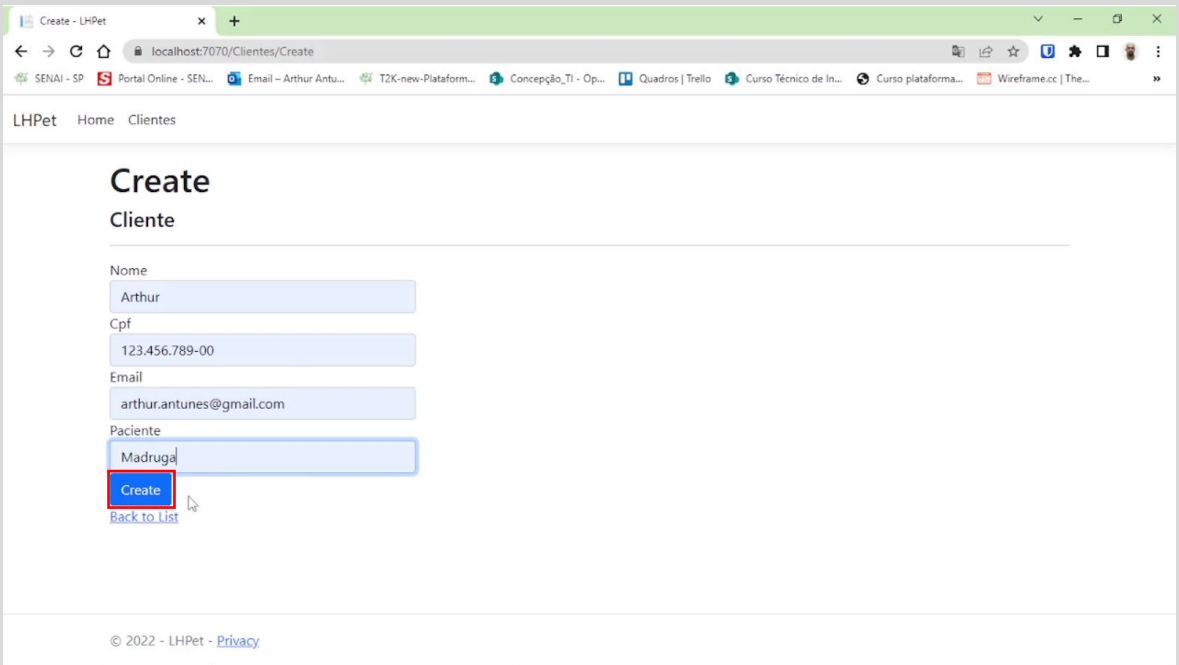
22. Se tudo correr bem, a aplicação vai abrir no navegador padrão da sua máquina.



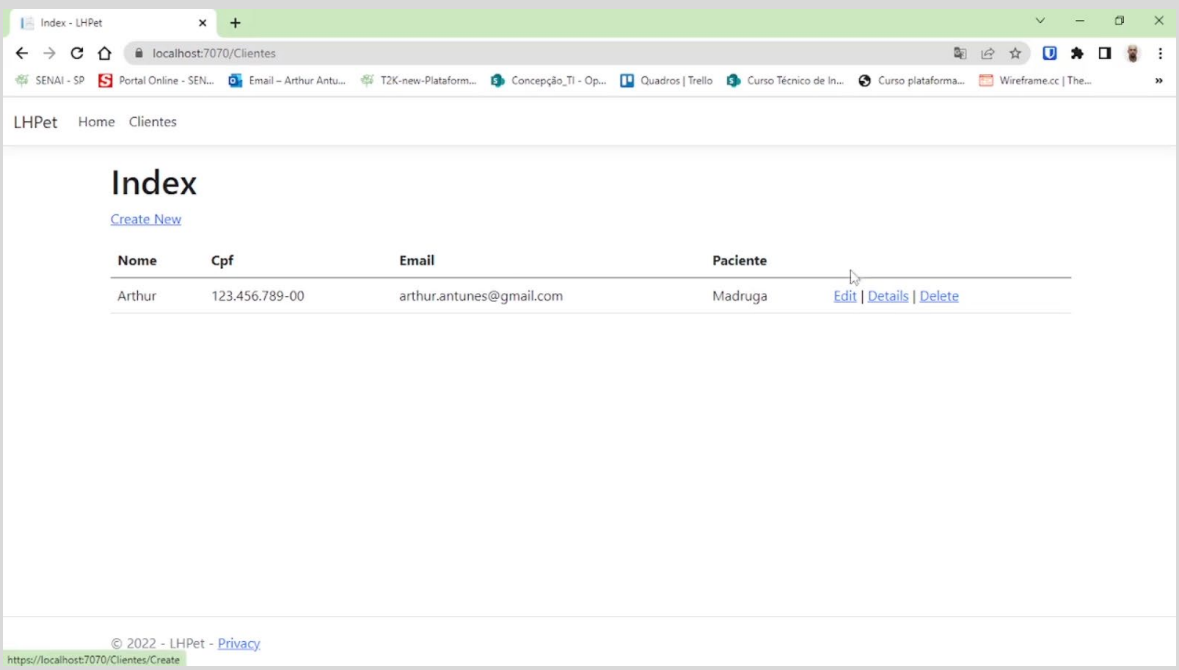
Clique em **Create New** para criar um novo cliente.



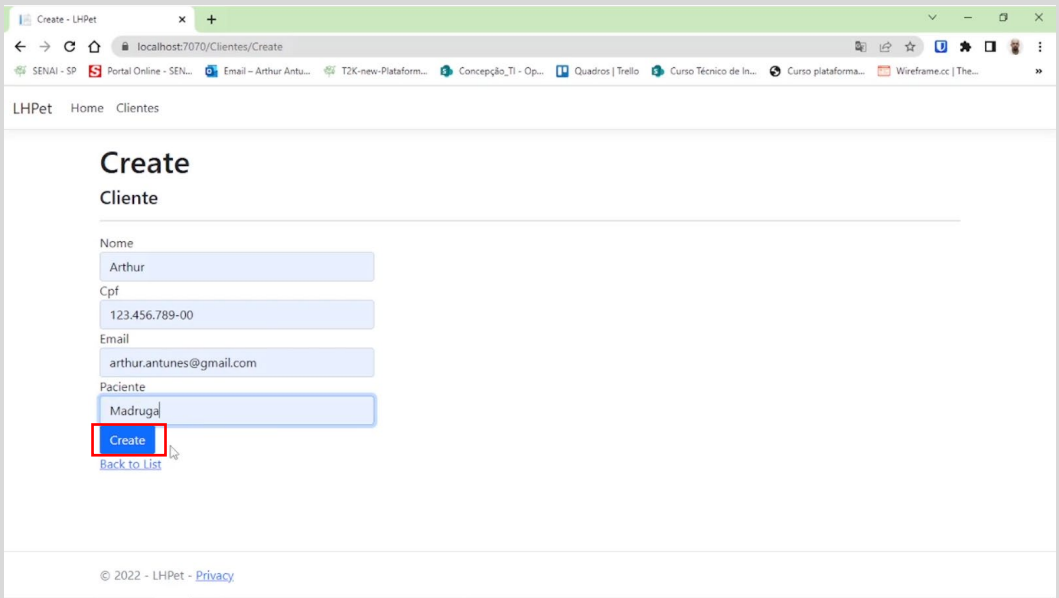
23. Preencha os dados e clique em **Save**.



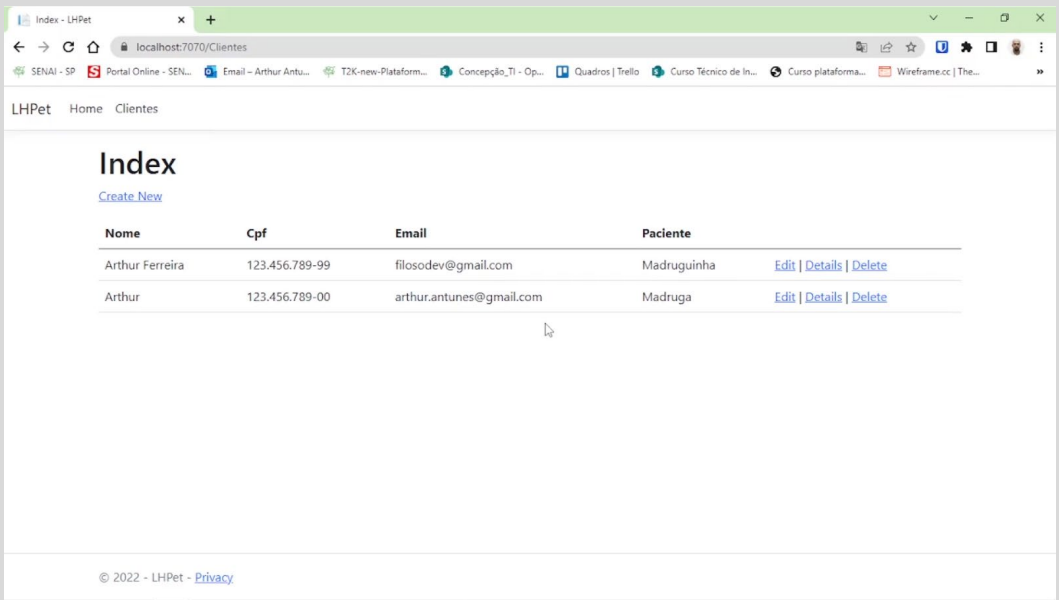
O cliente cadastrado deve aparecer na página Clientes.



24. Repita os passos e crie um novo cliente. Clique em **Save**.

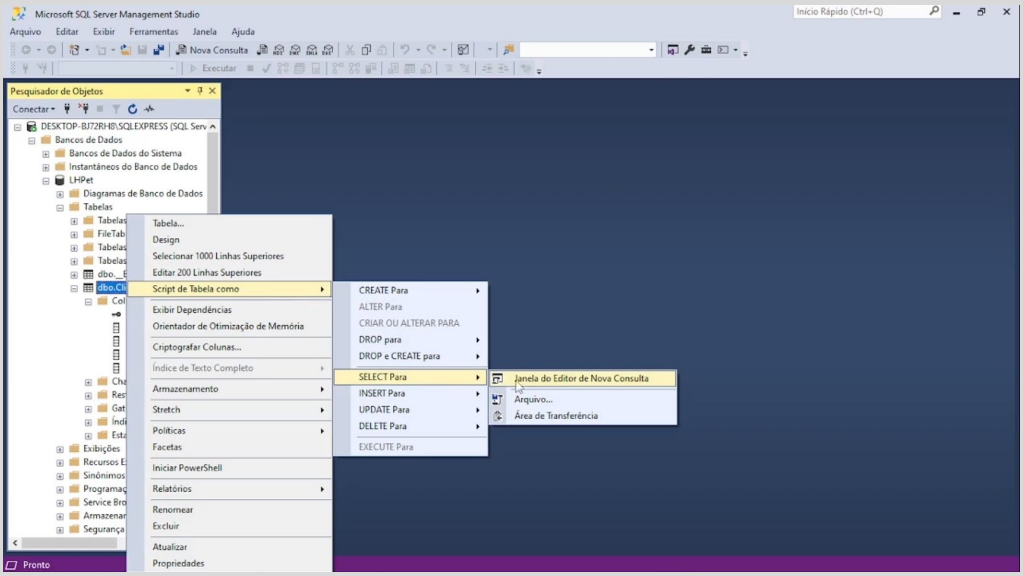


O cliente cadastrado deve aparecer na página Clientes.

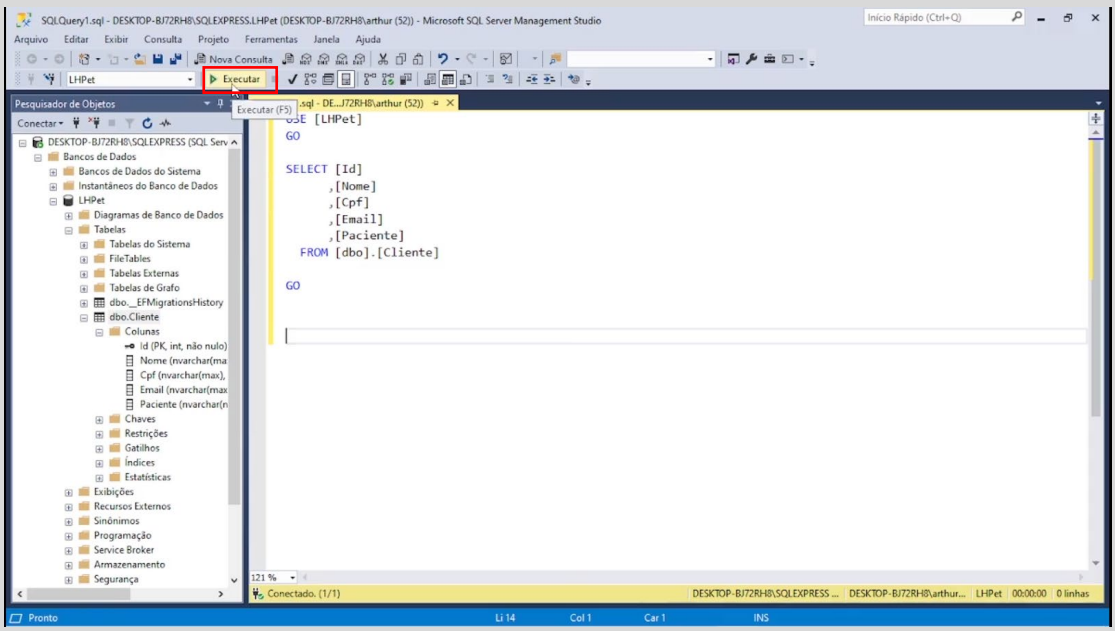


Você pode testar a aplicação alterando e deletando os dados também.

25. Verifique no SSMS se os clientes foram cadastrados. Clique em **dbo.Clientes** > **Script de Tabela como** > **SELECT Para** > **Janela do Editor de Nova Consulta**.



26. Na janela seguinte, clique em **Executar**.



27. Os clientes cadastrados devem aparecer corretamente no banco de dados.