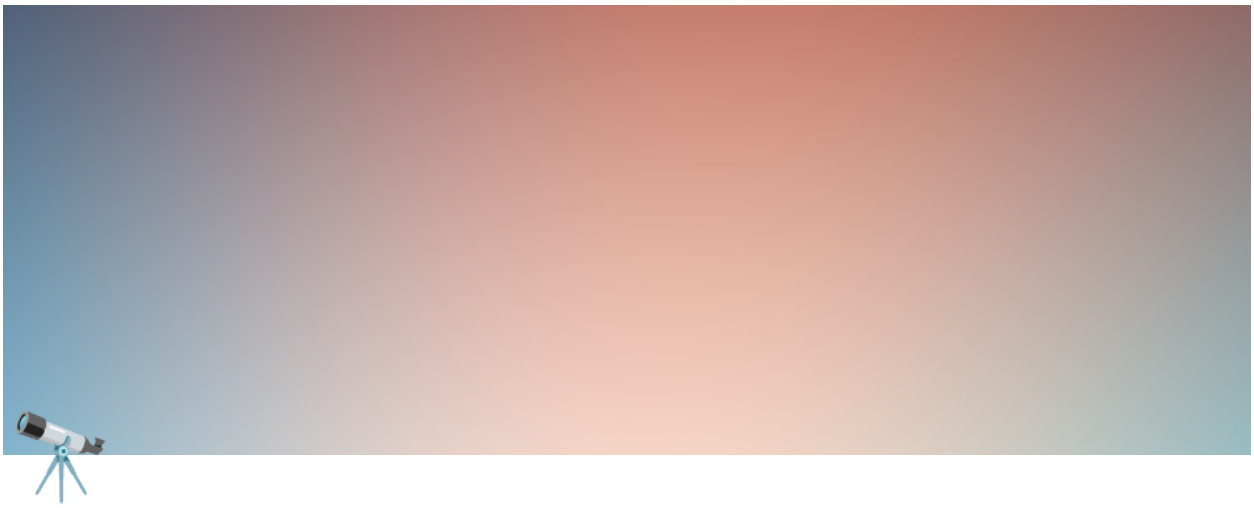

API-REST

Passo a passo do módulo 3 do curso de API's rest do professor Macoratti

1. **Criar um projeto do tipo web api c#**
2. **Excluir entidades desnecessárias e criar uma pasta Models com as classes Categoria.cs e Produtos.cs**
3. **No nuGet instalar os pacotes:**
    a. Pomelo.EntityFrameworkCore.MySql
    b. Microsoft.EntityFrameworkCore.Design
4. **Instalar a ferramenta Entity Framework Core .Net command-line Tools. Primeiramente, verifique se já não está instalada com o comando:**
    a. dotnet ef
5. **Após este comando deve aparecer uma mensagem da ferramenta, caso contrário, siga a instalação:**
    a. dotnet tool install --global dotnet-ef
    b. dotnet tool update --global dotnet-ef

O que é a classe de contexto?

É uma classe muito importante, herda do DbContext (do EntityFrameworkCore) essa classe que define o mapeamento entre as entidades e as tabelas

DbContext- Representa uma sessão com o banco de dados, é a ponte entre as entidades e o banco

DbSet<T>- Representa uma coleção de entidades no contexto que podem ser consultadas no bando de dados

6. **Criar uma classe de contexto chamada APICatalogoContext.cs na pasta Context**
7. **Definir a string de conexão no arquivo appsettings.json:**

```
{
  "ConnectionStrings": {
    "ConexaoPadrao": "stringconexão"
  },
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*"
}
```

8. **Pegar a string de conexão na classe Program**

```
//linha 1
string stringDeConexaoBD = builder.Configuration.GetConnectionString("ConexaoPadrao");

//linha 2
builder.Services.AddDbContext<APICatalogoContext>
        (options => options.UseMySql(stringDeConexaoBD, ServerVersion.AutoDetect(stringDeConexaoBD)));
```

9. **Antes de fazer o migrations nós precisamos definir o relacionamento entre as tabelas** **Produtos** e **Categorias**. Uma categoria vai ter uma coleção de produtos e um produto vai ter somente uma categoria.
10. **Para fazer isso, definimos na classe Categoria a propriedade:**

```
public ICollection<Produto>? Produtos { get; set; }
```

E na classe **Produto**, as propriedades:

```
public int CategoriaId { get; set; }
public Categoria? Categoria { get; set; }
```

11. **======== Fazendo o Migrations ========**

O que é o migrations no EF core ?

É uma maneira de atualizar as tabelas do banco de dados para mantê-lo sempre em sincronia com o modelo de dados do aplicativo.

IMPORTANTÍSSIMO: Sempre que você alterar as classes do modelo domínio, precisará executar o Migrations para manter o banco de dados atualizado.

Comandos do EF Core Tools:

- **dotnet ef:**

  Verifica se o EF Core Tools está instalado (se não estiver, volte ao passo de número 5)

- **dotnet ef migrations add 'nome':**

  Sendo 'nome' uma variável a escolha do usuário, que pode ser declarada com as aspas simples ou não, representa o nome do script de migração (com os métodos Up() e Down())

- **dotnet ef migrations remove 'nome':**

  Remove o script criado pelo comando migrations add, se nescessário ..

- **dotnet ef database update**

  Por fim, aplica as mudanças que você configurou.

12. **Passo 12, saber o que é DataAnnotations**

DataAnnotations, Para que servem ?

As convenções do **Entity Framework** core ditam que deve-se usar o maior valor possível para guardar uma coluna no banco de dados se o tamanho não estiver especificado, como por exemplo, uma entidade definida com o tipo string vai ser guardada no banco de dados como longtext que é um valor muito maior do que o que precisamos.

Para solucionar este problema, podemos sobrescrever essas convenções podemos usar DataAnnotations para:

- Definir regras de validação para o modelo
- Definir como os dados devem ser exibidos na interface em aplicações ASP.NET CORE MVC
- Especificar o relacionamento entre as entidades do modelo
- **Sobrescrever as convenções padrão do Entity Framework Core**

Estes atributos estão disponíveis nos namespaces:

- System.ComponentModel.DataAnnotations
- System.ComponentModel.DataAnnotations.Schema

Após definir os valores para o tamanho da string e dos decimal, faça uma nova migration e execute-a.

13. Popular as Tabelas do banco de dados

Criar uma migração vazia usando instruções de insert into para incluir dados nas tabelas

Começaremos pela tabela **Categorias**
->mb.Sql("Insert into Categorias(Nome, ImagemUrl) values('Bebidas', 'bebidas.jpg')");
E um código semelhante para a tabela Produtos ...