**UNIVERSIDADE POSITIVO**

**TÓPICOS ESPECIAIS EM DESENVOLVIMENTO DE SISTEMAS**

**TUTORIAL ENTITYFRAMEWORK**

Sumário

[ETAPA1: CONFIGURANDO E CRIANDO A PRIMEIRA CLASSE MODELO DO PROJETO 1](#_Toc167714371)

[ETAPA 2: ALTERANDO A ESTRUTURA DA MODEL PRODUTOS 5](#_Toc167714372)

[ETAPA 3: INSERINDO, ALTERANDO E CONSULTANDO DADOS DOS PRODUTOS 6](#_Toc167714373)

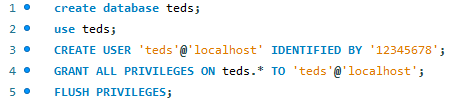
[ETAPA 4: PRATICANDO 11](#_Toc167714374)

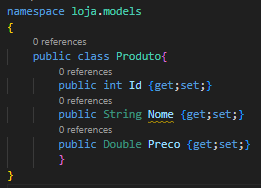
[DESAFIO 13](#_Toc167714375)

**Neste tutorial, você desenvolverá uma aplicação capaz de manter dados sobre produtos.**

**Importante: O MySql será usado com banco dedados**

# ETAPA1: CONFIGURANDO E CRIANDO A PRIMEIRA CLASSE MODELO DO PROJETO

1. Crie o banco de dados para este tutorial usanto o MySql Workbench
   1. Será necessário criar um usuário específico para o novo sistema
   2. Comandos para criar o banco de dados e o usuário:  
      
   3. Agora seu banco de dados possui um usuário que será utilizado pelo sistema para acessar as tabelas e demais operações de dados.
2. Inicie um projeto do tipo *webapi* chamado **Loja**
3. Entrar na pasta Loja (pasta criada para o seu projeto, conforme o item 1)
4. Instale os pacotes do EntityFramework Core:
   1. dotnet tool install --global dotnet-ef
   2. dotnet add package Microsoft.EntityframeworkCore.Design
   3. dotnet add package Microsoft.EntityframeworkCore
   4. dotnet add package Pomelo.EntityframeworkCore.MySql
5. Abra o VSCode
6. Crie uma pasta **models** dentro do seu projeto
7. Dentro da pasta **models**, implemente a classe **Produto.cs**



1. Crie uma pasta (dentro do projeto) com o nome **data**
2. Dentro da pasta **data**, implemente a classe LojaDbContext.cs

namespace loja.data{

    public class LojaDbContext : DbContext{

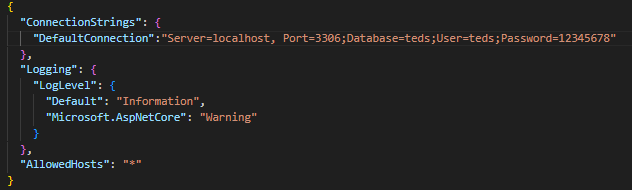
            public LojaDbContext(DoContextOptions<LojaDbContext> options) : base(options){}

            publicDbSet<ProducesResponseTypeMetadata> Produtos {get;set;}

    }

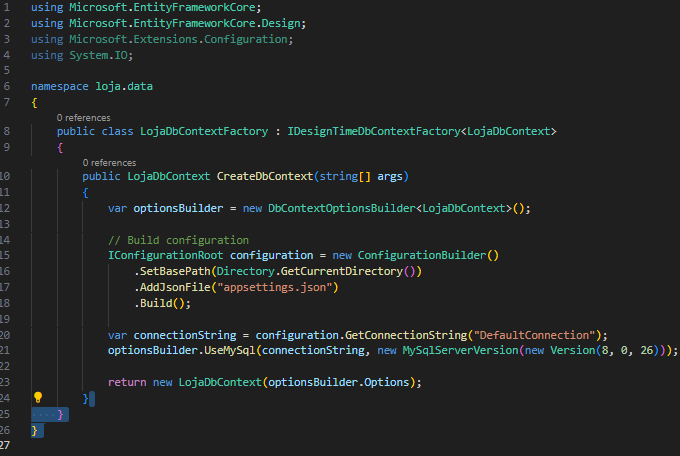
}

1. Atualize o arquivo appsettings.json



* 1. ATENÇÃO: Note que a conexão exemplo (acima) define o usuário do banco de dados como **teds** e senha como **12345678**. Você deve criar um usuário próprio e alterar os dados diretamente na linha *DefaultConnection*.

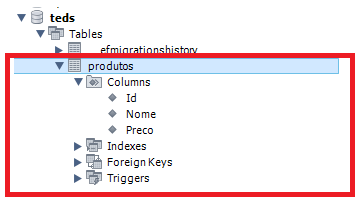
1. Dentro da pasta **data**, crie uma classe DbContextFactory para permitir que os comandos *dotnet ef* funcionem:



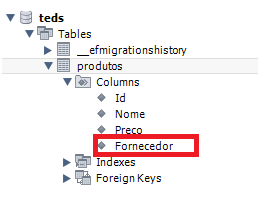
1. Atualize o Program.cs, para configurar o acesso ao banco de dados, conforme os trechos destacados em vermelho



1. Use o **Terminal**, e aplique as definições do banco de dados (o DotNet chama de *Migrations*)
   1. dotnet ef migrations add InitialCreate
   2. dotnet ef database update
   3. dotnet run
2. Vá até o MySql Workbench e verifique o banco de dados **teds** (seu banco de dados pode ter outro nome)
   1. Se tudo deu certo, o Entityframework criou uma pasta produtos no banco de dados



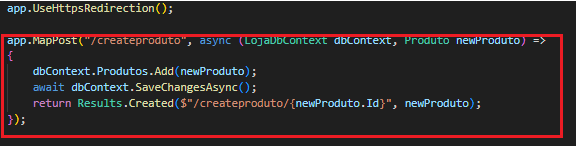
# ETAPA 2: ALTERANDO A ESTRUTURA DA MODEL PRODUTOS

1. Adicione a propriedade Fornecedor à classe Produto.cs
   1. Xx
2. Atualize o banco de dados, criando uma nova *Migration* e atualizando o banco de dados
   1. dotnet ef migrations add AddFornecedorToProduto
   2. dotnet ef database update
3. Verifique a tabela produtos no banco de dados, ela deve possuir uma nova coluna Fornecedor
   1. 

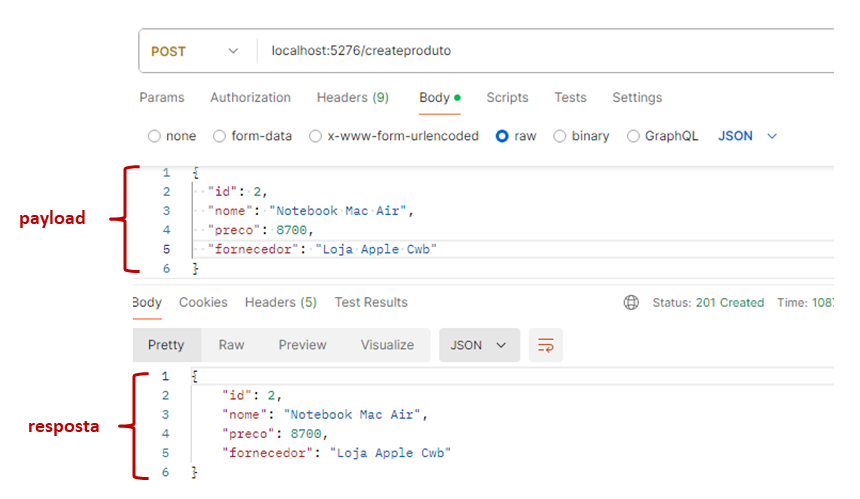
# ETAPA 3: INSERINDO, ALTERANDO E CONSULTANDO DADOS DOS PRODUTOS

**INSERIR DADOS**

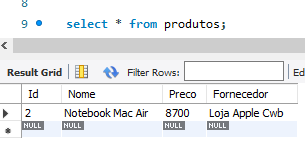
1. Implemente um endpoint para criar um novo produto na base de dados. Observe que o código do endpoint foi adicionado abaixo da **linha app.UseHttpsRedirection()**;



1. Rode o seu programa
   1. dotnet run
2. Envie dados para a API usando o Postman
   1. Note que os dados enviados no *payload* do *request* obedecem exatamente os atributos da classe produto.
   2. Atenção para a porta usada no request. No exemplo abaixo, foi utilizada a porta 5276. Verifique em qual porta seu programa está rodando.

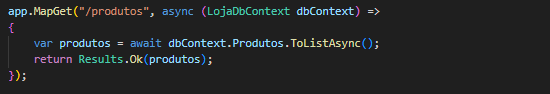


1. Verifique o banco de dados
   1. Se o *endpoint* está funcionando corretamente e o *request* enviado está correto, os dados deve, estar gravados na tabela produtos

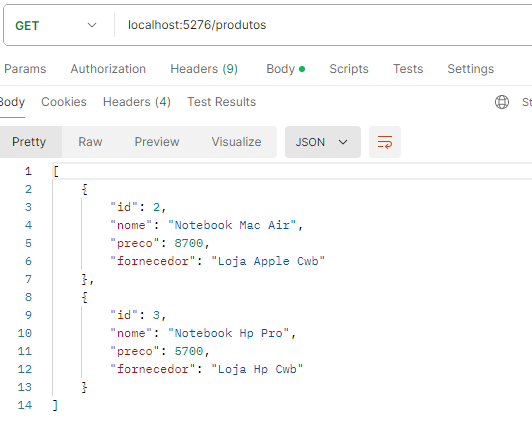


**CONSULTAR DADOS**

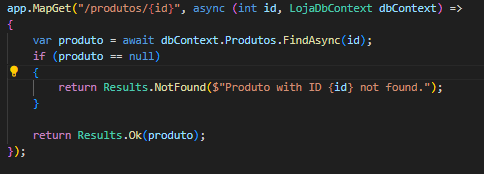
1. Para consultar os dados de todos os produtos
   1. Implemente um *endpoit* GET que retorna todos os dados da classe Produto



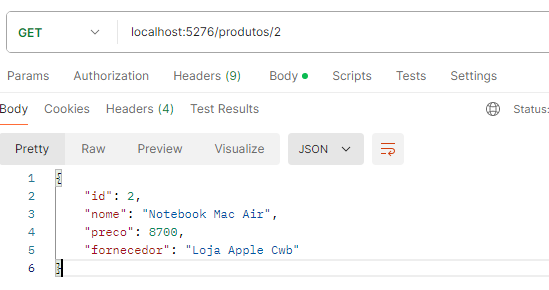
* 1. Teste o *endpoint* usando o Postman



1. Para consultar um produto a partir de seu ID
   1. Implemente um endpoint do tipo GET que receba o ID e retorne os dados conforme o banco de dados

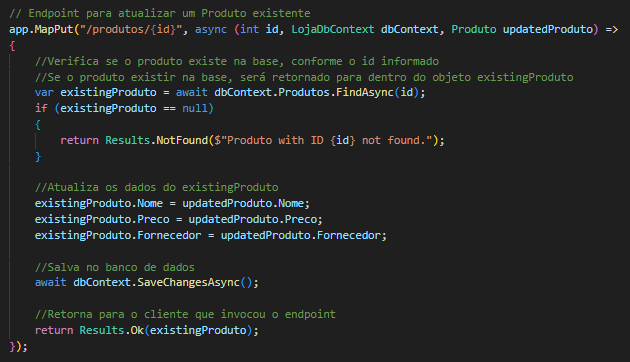


* 1. Teste com o Postman
     1. A url do request (GET) será: **localhost:5276/produtos/2**
     2. Atenção: O valor **2**, após a barra [**/2**] representa um dos ids de produtos gravados na sua base de dados
     3. Lembre-se de substituir a Porta 5276 pela correta (conforme sua aplicação)

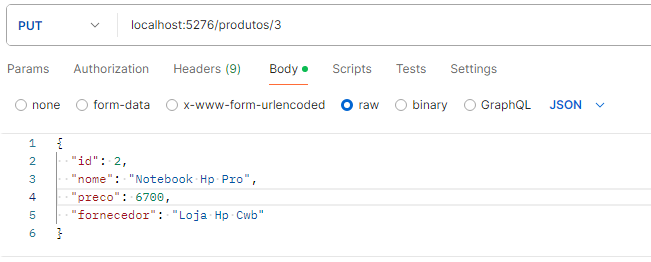


**ATUALIZANDO DADOS**

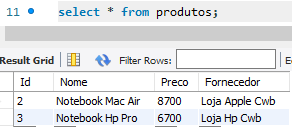
1. Para alterar os dados de um produto, faça:
   1. Implemente um enpoint do tipo PUT que receberá o ID e os novos dados do produto



* 1. Teste o endpoint com o Postman
     1. Note que o tipo de requet é PUT
     2. A url é **localhost:5276/produtos/3**
     3. Atenção: O valor **3**, após a barra [**/3**] representa um dos id do produto que será atualizado na sua base de dados
     4. O body recebe os novos dados

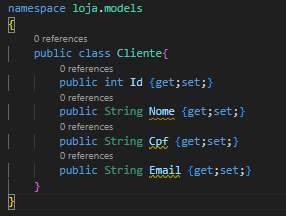


* + 1. Verifique o banco de dados

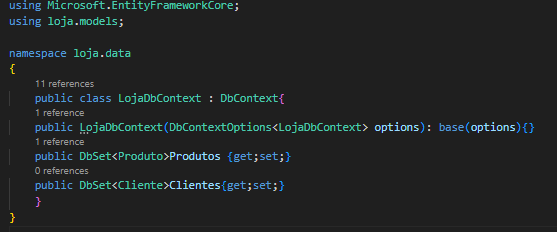


# ETAPA 4: PRATICANDO

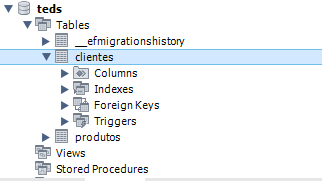
1. Implemente a classe Cliente no seu projeto
   1. Crie a classe Cliente.cs na pasta models



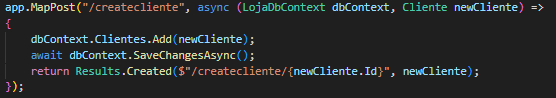
* 1. Adicionar a classe cliente na DbContext.cs



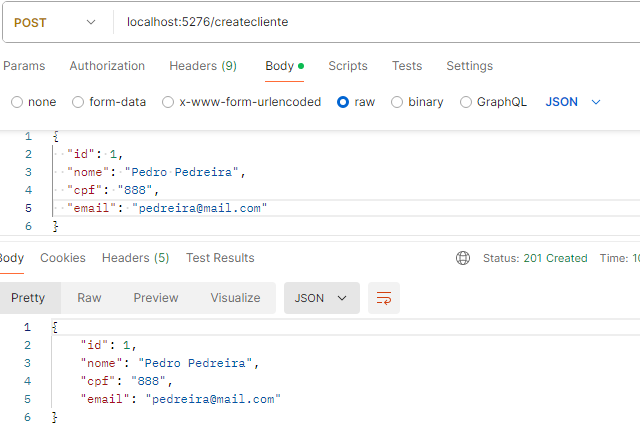
* 1. Adicionar a *Migration* (usando o terminal)
     1. dotnet ef migrations add AddCliente
     2. dotnet ef database update
  2. Verifique a tabela criada no banco de dados



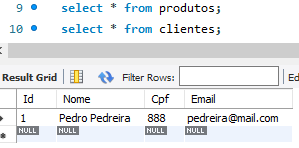
1. Implemente o *endpoint* para criar um novo cliente



* 1. Rode sua aplicação
     1. dotnet run
  2. Execute o endpoint usando o Postman



* 1. Consulte a tabela clientes no banco de dados



1. Implemente o endpoint para consultar todos os clientes do banco dedados
   1. Siga o exemplo criado para a classe Produto
2. Implemente o endpoint para consultar um cliente a partir de seu ID
   1. Siga o exemplo criado para a classe Produto
3. Implemente o endpint para altualizar os dados de um cliente
   1. Siga o exemplo criado para a classe Produto

# DESAFIO

1. Implemente seu projeto para possuir a classe Fornecedor com os atributos {id, cnpj, nome, endereco, email, telefone), com todas as operações de banco de dados explicadas neste tutorial.
2. Pesquise como se faz e altere o seu projeto de forma que os Ids do Produto e do Cliente sejam chaves primárias