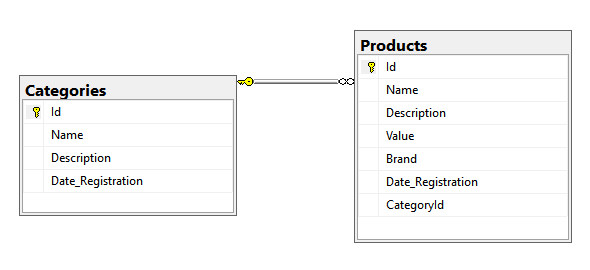
**DOCUMENTAÇÃO WEB API – ECOMMERCE**

**BANCO DE DADOS**

Para o projeto foi utilizado o Banco de Dados SQL Server 2019 e o Docker Desktop para instalar o SQL Server no container e utilizar no Visual Studio.

****

**USE DATABASE ECOMMERCE**

**CREATE TABLE [dbo].[Categories](**

**[Id] [int] IDENTITY(1,1) NOT NULL,**

**[Name] [varchar](100) NOT NULL,**

**[Description] [varchar](500) NOT NULL,**

**[Date\_Registration] [datetime] NULL**

**PRIMARY KEY CLUSTERED**

**(**

**[Id] ASC**

**)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]**

**) ON [PRIMARY]**

**GO**

**CREATE TABLE [dbo].[Products](**

**[Id] [int] IDENTITY(1,1) NOT NULL,**

**[Name] [varchar](100) NOT NULL,**

**[Description] [varchar](500) NOT NULL,**

**[Value] [decimal](18,2) NOT NULL,**

**[Brand] [varchar](50) NOT NULL,**

**[Date\_Registration] [datetime] NULL,**

**[CategoryId] [int] NOT NULL**

**PRIMARY KEY CLUSTERED**

**(**

**[Id] ASC**

**)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]**

**) ON [PRIMARY]**

**GO**

**ALTER TABLE [dbo].[Product] WITH CHECK ADD FOREIGN KEY([CategoryId])**

**REFERENCES [dbo].[Category] ([Id])**

**INFORMAÇÕES DA APLICAÇÃO**

A Web API foi desenvolvida com conceitos do DDD, a camada Service contém os controllers da API e acessa a camada de Application onde utiliza o mapeamento entre os data tranfers objects (DTO) e as entidades do domínio. Na Infrastructure fica o Sql Context para realizar a manipulação dos dados para o banco e o controle injeção de dependência.

Foi utilizado como um facilitador uma classe genérica Base para as entidades, pois neste projeto as duas entidades irão realizar um CRUD. Caso contrário o ideal seria criar nas próprias entidades.

* **Camada Service**

**Pacotes nuggets:**

Autofac (6.0.0)

Autofac.Extensions.DependencyInjection (7.1.0)

Swashbuckle.AspNetCore (5.6.3)

*- EcommerceWebAPI.API > Controllers > CategoryController/Product Controller*

Contém as Action dos métodos CRUD enviado os dados em formato DTO para camada de Application.

*- EcommerceWebAPI.API > Statup*

Foi inserido a conexão com o banco pegando os dados da conexão do SqlConnection do appsettings.json, a configuração do Swagger e a Injeção de Dependência da Infrastructure.

* **Camada Application**

**Pacotes nuggets:**

AutoMapper (10.1.1)

*- EccomerceWebAPI.Application > Dtos*

Contém as classes DtoBase, DtoCategory e DtoProduct para trabalhar com os dados que vem das Actions dos Controllers do Service.

*- EccomerceWebAPI.Application > Interface*

Interface de aplicação da entidade Category e Product para receber os dadaos das Actions dos Controllers do Service.

*- EccomerceWebAPI.Application > Mappers*

Foi desenvolvido classes para fazer o mapeamento dos dados DtoToModel e os dados do ModelToDto, ou seja, recebe os dados das Actions(DTO) e transfere para Domain(Model) e consulta os dados das entidades que estão na camada de Domain e transfere para os DTO.

As classes ApplicationServiceCategory e ApplicationServiceProduct faz a implementação do CRUD das interfaces criadas, trabalhando com o mapeamento dos DTOs e Model. É a ponte principal da manipulação dos dados entre os Controller e o Domain.

* **Camada Domain**

*EcommerceWebAPI.Domain > Entitys*

Classes Base, Category e Product com os atributos das entidades, a classe base é herda as outras duas classes.

*- EcommerceWebAPI.Domain.Core > Repositorys/Services*

No Domain Core fica as interfaces, tanto no Repository quanto em Services contém uma entidade anônima para herdar as outras duas classes. É utilizada para declarar o CRUD.

*- EcommerceWebAPI.Domain.Core.Services*

Implementação da interface Base (CRUD) que herda as Classes da categoria e do produto. Os métodos retornam as características da Interface do repositório.

* **Camada Infrastructure**

**Pacotes nuget:**

Autofac (6.0.0)

Microsoft.EntityFrameworkCore.SqlServer (5.0.0)

*- EcommerceWebAPI.Infrastructure > Data > Repository*

As Classes do repositório realizam a implementação da interface dos IRepositorys criada na camada de Domínio com os métodos CRUD no banco de dados.

*- EcommerceWebAPI.Infrastructure > Data*

SqlContext instancia os objetos das entidades e faz a manipulação com o CRUD para salvar no Banco de Dados. Ao salvar, o public override registra a data que o registro foi alterado ou inserido.

*- EcommerceWebAPI.Infrastructure > CrossCrutting > IOC*

ConfigurationIOC: registra as classes indicado a sua respectiva interface para fazer o load no container do Autofac.

ModuleIOC: sobrescreve o método antigo e protege a classe de configuração da injeção de dependência.