

# Manual de Utilizador



**OFIPEÇAS**

**Aluno: Rodrigo Ferreira**

**Turma: TGPSI23S**

**N.º Aluno: 2223233**

# **Índice**

## **1. Visão Geral da Arquitetura**

- **1.1. Arquitetura em Camadas**
- **1.2. Organização do Projeto**

## **2. Base de Dados**

- **2.1. Diagrama Entidade-Relação (ER)**
- **2.2. Script de Criação (SQL)**

## **3. Camada de Serviços**

- **3.1. Visão Geral dos Serviços**
- **3.2. AuthService.cs**
- **3.3. StoreService.cs**
- **3.4. CartService.cs**
- **3.5. OrderService.cs**
- **3.6. UserService.cs**
- **3.7. AdminService.cs**

## **4. Dependências e Instalação**

- **4.1. Requisitos do Sistema**
- **4.2. Bibliotecas Externas (NuGet)**
- **4.3. Configuração do Ambiente**
- **4.4. Criar Utilizador Administrador**

## **1. Visão Geral da Arquitetura**

**Este capítulo descreve a estrutura arquitetónica da aplicação OfiPeças, concebida para promover a manutenção, escalabilidade e separação de responsabilidades.**

### **1.1. Arquitetura em Camadas**

**A aplicação foi desenvolvida seguindo um padrão de arquitetura em 3 camadas, que isola as diferentes responsabilidades do software:**

#### **1. Camada de Apresentação (UI - User Interface):**

- **Componentes:** Formulários Windows Forms (Login.cs, Loja.cs, PainelAdmin.cs, etc.) e UserControls (ProdutoCard.cs, ItemCarrinho.cs, etc.).
- **Responsabilidade:** Apresentar a informação ao utilizador e capturar as suas interações. Esta camada não contém lógica de negócio; ela apenas invoca os métodos da camada de serviços e exibe os resultados.

#### **2. Camada de Lógica de Negócio (Serviços):**

- **Componentes:** Classes estáticas de serviço (AuthService.cs, StoreService.cs, CartService.cs, OrderService.cs, UserService.cs, AdminService.cs).
- **Responsabilidade:** Orquestrar toda a lógica da aplicação. É aqui que as regras de negócio são aplicadas (ex: verificar stock, validar passwords, calcular totais). Esta camada serve como ponte entre a interface e o acesso aos dados.

### **3. Camada de Acesso a Dados:**

- **Componentes:** Classe `DatabaseConnection.cs` e o uso direto de `Microsoft.Data.SqlClient` dentro dos serviços.
- **Responsabilidade:** Lidar com toda a comunicação com a base de dados SQL Server. Centraliza a criação de conexões e a execução de comandos SQL.

### **1.2. Organização do Projeto**

O projeto no Visual Studio 2022 está organizado da seguinte forma para refletir a arquitetura:

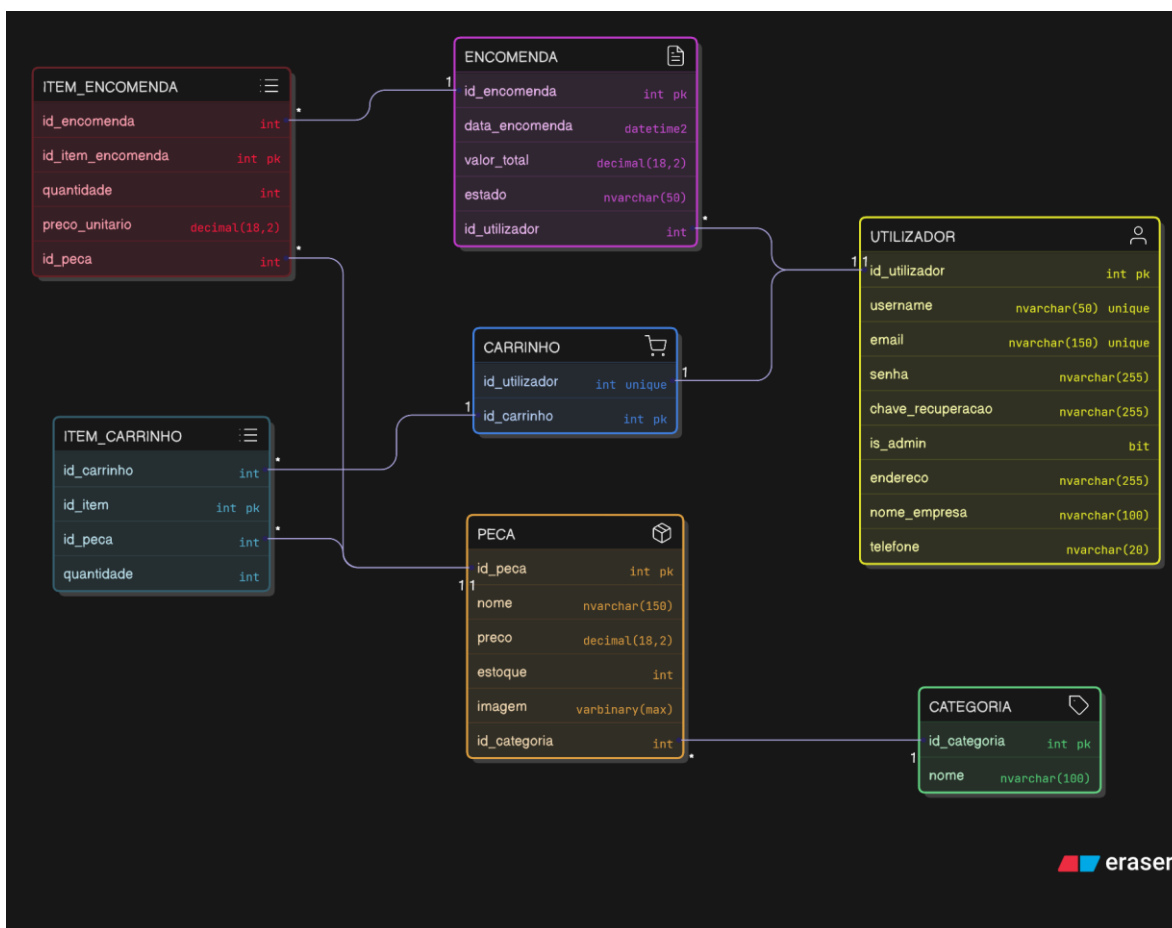
- **Formulários (.cs):** Cada ecrã principal da aplicação.
- **UserControls (.cs):** Componentes visuais reutilizáveis.
- **Serviços (.cs):** As classes estáticas que compõem a camada de lógica de negócio.
- **Modelos (Models.cs):** Classes simples (`Peca`, `UserInfo`, etc.) usadas para transportar dados entre as camadas.
- **Ficheiro .env:** Ficheiro de configuração que armazena as credenciais de acesso à base de dados.

## 2. Base de Dados

A base de dados, denominada OfiPecas, foi implementada em Microsoft SQL Server.

### 2.1. Diagrama Entidade-Relação (ER)

O diagrama abaixo ilustra a estrutura das tabelas e as suas relações.



[Diagrama Entidade-Relação]

## 2.2. Script de Criação (SQL)

O script seguinte pode ser executado para criar toda a estrutura da base de dados, incluindo tabelas, chaves primárias, chaves estrangeiras, restrições e índices.

```
-- 1) Criar o base de dados
IF DB_ID('OfiPecas') IS NULL
    CREATE DATABASE OfiPecas;
GO

USE OfiPecas;
GO

-- 2) Tabela UTILIZADOR
CREATE TABLE dbo.UTILIZADOR
(
    id_utilizador      INT IDENTITY(1,1) NOT NULL CONSTRAINT PK_UTILIZADOR
PRIMARY KEY,
    username           NVARCHAR(50)      NOT NULL CONSTRAINT
UQ_UTILIZADOR_USERNAME UNIQUE,
    email              NVARCHAR(150)     NOT NULL CONSTRAINT UQ_UTILIZADOR_EMAIL
UNIQUE,
    senha              NVARCHAR(255)     NOT NULL,
    chave_recuperacao  NVARCHAR(255)     NOT NULL,
    is_admin            BIT               NOT NULL CONSTRAINT DF_UTILIZADOR_ADMIN
DEFAULT 0,
    endereco           NVARCHAR(255)     NULL,
    nome_empresa       NVARCHAR(100)     NULL,
    telefone           NVARCHAR(20)      NULL
);
GO

-- 3) Tabela CATEGORIA
CREATE TABLE dbo.CATEGORIA
(
    id_categoria       INT IDENTITY(1,1) NOT NULL CONSTRAINT PK_CATEGORIA PRIMARY
KEY,
    nome               NVARCHAR(100)     NOT NULL
);
GO

-- 4) Tabela PECA (com campo BLOB para imagens)
CREATE TABLE dbo.PECA
(
    id_pecas           INT IDENTITY(1,1) NOT NULL CONSTRAINT PK_PECA PRIMARY KEY,
    nome               NVARCHAR(150)     NOT NULL,
    preco              DECIMAL(18,2)     NOT NULL,
    estoque            INT               NOT NULL CONSTRAINT DF_PECA_ESTOQUE DEFAULT
0,
    imagem             VARBINARY(MAX)    NOT NULL,
    id_categoria        INT              NOT NULL
CONSTRAINT FK_PECA_CATEGORIA FOREIGN KEY(id_categoria)
REFERENCES dbo.CATEGORIA(id_categoria)
```

```

        ON UPDATE CASCADE
        ON DELETE NO ACTION
    );
GO

-- 5) Tabela CARRINHO (1:1 com UTILIZADOR)
CREATE TABLE dbo.CARRINHO
(
    id_carrinho    INT IDENTITY(1,1) NOT NULL CONSTRAINT PK_CARRINHO PRIMARY
KEY,
    id_utilizador INT                NOT NULL
        CONSTRAINT FK_CARRINHO_UTILIZADOR FOREIGN KEY(id_utilizador)
        REFERENCES dbo.UTILIZADOR(id_utilizador)
        ON UPDATE CASCADE
        ON DELETE CASCADE,
    -- Garante que cada utilizador tenha apenas um carrinho
    CONSTRAINT UQ_CARRINHO_UTILIZADOR UNIQUE (id_utilizador)
);
GO

-- 6) Tabela ITEM_CARRINHO (1 carrinho : N itens)
CREATE TABLE dbo.ITEM_CARRINHO
(
    id_item        INT IDENTITY(1,1) NOT NULL CONSTRAINT PK_ITEM_CARRINHO
PRIMARY KEY,
    id_carrinho    INT                NOT NULL,
    id_peca        INT                NOT NULL,
    quantidade     INT                NOT NULL CONSTRAINT CK_ITEM_CARRINHO_QTD
CHECK(quantidade > 0),

    CONSTRAINT FK_ITEMCARRINHO_CARRINHO FOREIGN KEY(id_carrinho)
        REFERENCES dbo.CARRINHO(id_carrinho)
        ON UPDATE CASCADE
        ON DELETE CASCADE,

    CONSTRAINT FK_ITEMCARRINHO_Peca FOREIGN KEY(id_peca)
        REFERENCES dbo.Peca(id_peca)
        ON UPDATE CASCADE
        ON DELETE NO ACTION
);
GO

-- 7) Tabela ENCOMENDA (Guarda a informação geral da encomenda)
CREATE TABLE dbo.ENCOMENDA
(
    id_encomenda   INT IDENTITY(1,1) NOT NULL CONSTRAINT PK_ENCOMENDA PRIMARY
KEY,
    id_utilizador  INT                NOT NULL,
    data_encomenda DATETIME2          NOT NULL CONSTRAINT DF_ENCOMENDA_DATA
DEFAULT GETDATE(), -- Regista a data e hora automaticamente
    valor_total    DECIMAL(18,2)      NOT NULL,
    estado         NVARCHAR(50)       NOT NULL CONSTRAINT DF_ENCOMENDA_ESTADO
DEFAULT 'Pendente', -- Ex: Pendente, Em Processamento, Enviada, Entregue,
Cancelada

    CONSTRAINT FK_ENCOMENDA_UTILIZADOR FOREIGN KEY(id_utilizador)
        REFERENCES dbo.UTILIZADOR(id_utilizador)
        ON UPDATE CASCADE

```

```

        ON DELETE NO ACTION -- Não queremos apagar encomendas se um utilizador
for apagado
);
GO

-- 8) Tabela ITEM_ENCOMENDA (Guarda os produtos específicos de cada encomenda)
CREATE TABLE dbo.ITEM_ENCOMENDA
(
    id_item_encomenda INT IDENTITY(1,1) NOT NULL CONSTRAINT PK_ITEM_ENCOMENDA
PRIMARY KEY,
    id_encomenda      INT          NOT NULL,
    id_peca           INT          NOT NULL,
    quantidade        INT          NOT NULL,
    preco_unitario    DECIMAL(18,2) NOT NULL, -- Guarda o preço da peça no
momento da compra

    CONSTRAINT FK_ITEMENCOMENDA_ENCOMENDA FOREIGN KEY(id_encomenda)
REFERENCES dbo.ENCOMENDA(id_encomenda)
ON UPDATE CASCADE
ON DELETE CASCADE, -- Se uma encomenda for apagada, os seus itens
também são

    CONSTRAINT FK_ITEMENCOMENDA_Peca FOREIGN KEY(id_peca)
REFERENCES dbo.PECA(id_peca)
ON UPDATE CASCADE
ON DELETE NO ACTION
);
GO

-- Índices adicionais para otimizar pesquisas
CREATE INDEX IX_ENCOMENDA_UTILIZADOR ON dbo.ENCOMENDA(id_utilizador);
CREATE INDEX IX_ITEM_ENCOMENDA_ENCOMENDA ON dbo.ITEM_ENCOMENDA(id_encomenda);
GO
CREATE INDEX IX_ITEM_CARRINHO_CARRINHO ON dbo.ITEM_CARRINHO(id_carrinho);
CREATE INDEX IX_Peca_CATEGORIA ON dbo.PECA(id_categoria);
GO

```

### 3. Camada de Serviços

A lógica de negócio está encapsulada em várias classes de serviço estáticas, cada uma com uma responsabilidade clara.

- **AuthService.cs:** Gere a autenticação (registo, login, recuperação) e a segurança de passwords.
- **StoreService.cs:** Responsável por obter os dados do catálogo (peças e categorias) para serem exibidos na loja.
- **CartService.cs:** Lida com todas as operações do carrinho de compras (adicionar, remover, atualizar itens).



- **OrderService.cs:** Orquestra o processo de finalização de uma encomenda e a consulta ao histórico.
- **UserService.cs:** Gere os dados da conta do utilizador (editar perfil, alterar password, apagar conta).
- **AdminService.cs:** Centraliza todas as operações exclusivas do painel de administração (CRUD de peças e categorias, gestão de utilizadores).

## **4. Dependências e Instalação**

### **4.1. Requisitos do Sistema**

- **Sistema Operativo:** Windows 10 ou superior.
- **Framework:** .NET Framework 4.7.2 ou superior.
- **Base de Dados:** Acesso a uma instância do Microsoft SQL Server.

### **4.2. Bibliotecas Externas (NuGet)**

O projeto utiliza as seguintes bibliotecas externas, que devem ser instaladas através do NuGet Package Manager:

- **Guna.UI2.WinForms:** Para a criação da interface de utilizador moderna.
- **DotNetEnv:** Para carregar as variáveis de ambiente do ficheiro “.env”
- **PDFsharp:** Para a geração de faturas em formato PDF.
- **Microsoft.Data.SqlClient:** O driver para a comunicação com a base de dados SQL Server.

### 4.3. Configuração do Ambiente

Antes de executar a aplicação, é necessário criar um ficheiro chamado “.env” na pasta principal do projeto. Este ficheiro deve conter as seguintes variáveis com os dados de acesso à base de dados local:

```
DB_SERVER=DESKTOP-64E3SOI
DB_USER=sa
DB_PASSWORD=123123Aa.
DB_NAME=OfiPecas
```

### 4.4. Criar Utilizador Administrador

Para facilitar a gestão inicial da aplicação, é recomendado criar um utilizador com permissões de administrador diretamente na base de dados. O script abaixo insere um utilizador padrão com as credenciais:

username: adm

password: 123

```
USE OfiPecas;
GO

-- Insere o novo utilizador administrador
INSERT INTO dbo.UTILIZADOR
    (username, email, senha, chave_recuperacao, is_admin, endereco,
    nome_empresa, telefone)
VALUES
    (
        'adm', -- username
        'admin@ofipeças.com', -- email de exemplo
        '1/1lB/ZGt7kfvlx4EVSmM+QrcBlgcYHx8k022ani6/AuRibRJKDXBJ/wMf/gLBC', --
0 hash correspondente à password '123'
        'CHAVEADMIN123', -- chave de recuperação de
exemplo
        1, -- is_admin = true
        'Sede da Empresa', -- endereço de exemplo
        'OfiPeças Admin', -- nome da empresa de
exemplo
        '912345678' -- telefone de exemplo
    );
```