DOCUMENTAÇÃO DE BANCO DE DADOS

Documentação elaborada para composição da nota da Disciplina de Projeto de Banco de Dados, orientado pelo profª Tacyana Batista.

Sumário

Minimundo

O EasyOrder é um sistema de cardápio digital integrado para restaurantes, com foco em otimizar o atendimento e proporcionar uma experiência moderna aos clientes.

No restaurante, cada mesa possui um QR Code exclusivo. Quando o cliente escaneia esse código com seu smartphone, é automaticamente direcionado ao sistema, que reconhece de qual mesa o pedido será feito.

Antes de realizar o pedido, o cliente precisa se cadastrar com informações básicas (nome, e-mail ou telefone, senha e data de nascimento). Esse cadastro permite que o sistema registre seu histórico de pedidos, atribua pontos de fidelidade e personalize a experiência.

O cardápio digital é organizado em categorias (entradas, pratos principais, sobremesas, bebidas, etc.), e cada item do cardápio possui descrição, preço, custo, tempo de preparo estimado, imagem e possibilidade de personalização (como adicionar/remover ingredientes).

Durante a realização do pedido, o cliente pode:

- Selecionar múltiplos itens;
- Adicionar observações ou personalizações;
- Revisar e confirmar o pedido;
- Realizar o pagamento pelo próprio aplicativo (cartão, Pix, carteiras digitais) ou deixar para pagar na mesa.

Assim que o pedido é confirmado, ele é enviado automaticamente para o sistema da cozinha, que organiza e prioriza os pedidos. O cliente recebe notificações em tempo real sobre o andamento (pedido recebido, em preparo, pronto e entregue).

O sistema também coleta feedbacks e avaliações após a refeição, registrando comentários e notas de 1 a 5, para que o restaurante possa melhorar seu atendimento e cardápio.

No lado administrativo, os gestores do restaurante têm acesso a:

- Gerenciamento completo do cardápio;
- Controle de estoque, com movimentações de entrada, saída e ajustes;
- Registro de promoções e programas de fidelidade;
- Relatórios de vendas, consumo por categoria e preferências de clientes;
- Registro de logs de atividades feitas tanto por clientes quanto por funcionários.

Benefícios para o restaurante:

- Redução de erros e tempo de atendimento;
- Melhor controle de estoque e pedidos;
- Relatórios estratégicos para tomada de decisão;
- Sistema de fidelidade que aumenta o engajamento dos clientes.

Benefícios para o cliente:

- Autonomia e agilidade no pedido;
- Facilidade no pagamento digital;
- Transparência com notificações de status;
- Participação em promoções e programas de fidelidade.

MER (Modelo de Entidade e Relacionamento)

O Modelo de Entidade e Relacionamento (MER) é o projeto conceitual do banco de dados. Ele é apresentado em duas partes: a descrição detalhada das entidades e seus atributos, e um resumo dos relacionamentos e suas cardinalidades.

Entidades e Atributos

- **usersAdmin**
    - idUser: primary key autoincrement
    - name: varchar(255) not null
    - email: varchar(255) unique not null
    - password: varchar(255) not null
    - type: enum('dono', 'gerente', 'garçom', 'cozinha') not null
    - activo: boolean default true
    - dateRegister: datetime default current_timestamp
    - lastLogin: datetime

- **clientes**
    - idCliente: primary key autoincrement
    - name: varchar(255) not null
    - email: varchar(255) unique
    - telefone: varchar(20) unique not null
    - password: varchar(255) not null
    - birthData: date
    - dateRegister: datetime default current_timestamp
    - loyaltyPoints: int default 0

- **desks**
    - idDesk: primary key autoincrement
    - deskNumber: varchar(10) unique not null
    - qrcodeUld: varchar(50) unique not null
    - capacity: int not null
    - condition: enum ('livre', 'ocupada', 'reservada', 'manutencao') default 'livre'

- **categories**
    - idCategory: primary key autoincrement
    - name: varchar(50) not null
    - descricao: text
    - order: int default 0
    - activo: boolean default true

- **menultens**
    - idItem: primary key autoincrement
    - idCategory: foreign key → categories (idCategory)
    - name: varchar(100) not null
    - descricao: text
    - price: decimal(10,2) not null
    - cost: decimal(10,2)
    - timePreparo: int (minutos)
    - emphasis: boolean default false
    - activo: boolean default true
    - imagePath: varchar(255)
    - options: json

- **stock**
  - idStock: primary key autoincrement
  - idItem: foreign key → menultens(idItem)
  - amount: decimal(10,2) not null
  - unity: varchar(20) not null (ex: kg, L, un)
  - updateData: datetime default current_timestamp

- **stockMovement**
  - idMovement: primary key autoincrement
  - idStock: foreign key stock(idStock)
  - type: enum('entrada', 'saida', 'ajuste') not null
  - amount: decimal(10,2) not null
  - descricao: text
  - timeDate: datetime default current_timestamp

- **orders**
  - idOrder: primary key autoincrement
  - idCliente: foreign key clientes(idCliente)
  - idDesk: foreign key → desks(idDesk)
  - idPromocao: foreign key → promocoes (idPromocao)
  - timeData: datetime default current_timestamp
  - status: enum ('recebido', 'preparando', 'pronto', 'entregue', 'cancelado') default 'recebido'
  - total: decimal(10,2) not null
  - observacao: text

- **itemOrder**
  - idItemOrder: primary key autoincrement
  - idOrder: foreign key → orders(idOrder)
  - idItem: foreign key → menultens(idltem)
  - amount: int default 1 not null
  - unityPrice: decimal(10,2) not null
  - custom: json
  - observacao: text

- **feedbacks**
  - idFeedback: primary key autoincrement
  - idOrder: foreign key orders(idOrder)
  - idCliente: foreign key clientes(idCliente)
  - avaliacao: int (1 a 5)
  - comments: text
  - timeData: datetime default current_timestamp
  - respondido: boolean default false
  - respostaAdmin: text

- **promocoes**
  - idPromocao: primary key autoincrement
  - titulo: varchar(100) not null
  - descricao: text
  - type: enum('desconto', 'brinde', 'fidelidade') not null
  - valorDesconto: decimal(10,2)
  - codigo: varchar(50) unique

- validoDe: datetime
- validoAte: datetime
- status: boolean default true

- **logs**
  - idLog: primary key autoincrement
  - idUser: foreign key → usersAdmin(idUser)
  - idCliente: foreign key clientes(idCliente)
  - acao: varchar(100) not null
  - descricao: text
  - ipAddress: varchar(50)
  - timeData: datetime default current_timestamp

Relacionamentos e Cardinalidades

- categories (1) --- (N) menultens: Uma categoria pode ter vários itens, mas um item pertence a uma única categoria.
- clientes (1) --- (N) orders: Um cliente pode fazer vários pedidos, mas um pedido pertence a um único cliente.
- desks (1) --- (N) orders: Uma mesa pode ter vários pedidos ao longo do tempo, mas um pedido é feito de uma única mesa.
- orders (N) --- (M) menultens: Um pedido pode conter vários itens, e um item pode estar em vários pedidos. Este relacionamento é implementado pela entidade associativa itemOrder.
- menultens (1) --- (1) stock: Cada item do cardápio possui um único registro de estoque correspondente.
- stock (1) --- (N) stockMovement: Um item em estoque pode ter várias movimentações (entradas, saídas, etc.).

- orders (1) --- (1) feedbacks: Um pedido pode ter no máximo um feedback.
- clientes (1) --- (N) feedbacks: Um cliente pode registrar vários feedbacks.

- promocoes (N) --- (M) orders: Uma promoção pode ser aplicada a vários pedidos, e um pedido pode ter várias promoções. Requer uma tabela associativa para implementação.

- usersAdmin (1) --- (N) logs: Um usuário administrador pode gerar vários registros de log.

- clientes (1) --- (N) logs: Um cliente pode gerar vários registros de log.

DER (Diagrama de Entidade e Relacionamento)

Esta seção descreve textualmente o diagrama, detalhando o significado de cada relacionamento e como ele é implementado através das chaves primárias (PK) e estrangeiras (FK).

- **Relacionamento Categoria-Item**:
    - **Regra:** Uma categories agrupa N menuItens.
    - **Descrição:** Garante que todo item do cardápio esteja classificado, como "Entrada" ou "Prato Principal".
    - **Implementação:** A chave primária categories.idCategory (PK) é usada como chave estrangeira menuItens.idCategory (FK).
- **Relacionamento Cliente-Pedido**:
    - **Regra:** Um clientes realiza N orders.
    - **Descrição:** Associa um pedido a quem o fez, permitindo histórico de pedidos e programas de fidelidade.
    - **Implementação:** clientes.idCliente (PK) é usada como orders.idCliente (FK).
- **Relacionamento Mesa-Pedido**:
    - **Regra:** Uma desks sedia N orders.
    - **Descrição:** Identifica a localização física de um pedido dentro do restaurante.
    - **Implementação:** desks.idDesk (PK) é usada como orders.idDesk (FK).
- **Relacionamento Pedido-Item (N:M)**:
    - **Regra:** Um orders contém M menuItens.
    - **Descrição:** É o relacionamento central que forma o pedido. A entidade associativa itemOrder detalha quais itens, em qual quantidade (amount) e com qual preço (unityPrice) compõem cada pedido.
    - **Implementação:** itemOrder possui duas chaves estrangeiras: idOrder (de orders) e idItem (de menuItens).
- **Relacionamento Item-Estoque (1:1)**:
    - **Regra:** Um menuItens possui um stock.
    - **Descrição:** Controla a quantidade disponível de cada item do cardápio.
    - **Implementação:** menuItens.idItem (PK) é usada como stock.idItem (FK), que também deve ser definida como UNIQUE para garantir a cardinalidade 1:1.

- **Relacionamento Estoque-Movimentação**:
    - **Regra:** Um stock sofre N stockMovement.
    - **Descrição:** Rastreia todas as entradas, saídas e ajustes no estoque de um item.
    - **Implementação:** stock.idStock (PK) é usada como stockMovement.idStock (FK).
- **Relacionamento Pedido-Feedback (1:1)**:
    - **Regra:** Um orders recebe um feedbacks.
    - **Descrição:** Permite que a avaliação do cliente seja diretamente ligada ao pedido que a originou.
    - **Implementação:** orders.idOrder (PK) é usada como feedbacks.idOrder (FK), que deve ser UNIQUE.
- **Relacionamento Cliente-Feedback**:
    - **Regra:** Um clientes fornece N feedbacks.
    - **Descrição:** Cria um histórico de todas as avaliações feitas por um cliente.
    - **Implementação:** clientes.idCliente (PK) é usada como feedbacks.idCliente (FK).
- **Relacionamento Promoção-Pedido (N:M)**:
    - **Regra:** Uma promocoes é aplicada em M orders.
    - **Descrição:** Permite que cupons ou ofertas sejam aplicados a um pedido. Como um pedido pode ter múltiplos cupons e um cupom pode ser usado várias vezes, é necessário uma tabela de ligação.
    - **Implementação:** Requer uma tabela associativa (ex: Order_Promocoes) com as chaves estrangeiras idOrder e idPromocao.

Scripts DDL (Data Definition Language)

A seguir estão os scripts SQL para criar todas as tabelas do banco de dados EasyOrder.

```
CREATE TYPE userType AS ENUM ('dono', 'gerente', 'garcom', 'cozinha');
CREATE TYPE deskCondition AS ENUM ('livre', 'ocupada', 'reservada',
'manutencao');
CREATE TYPE stockMovementType AS ENUM ('entrada', 'saida', 'ajuste');
CREATE TYPE orderStatus AS ENUM ('recebido', 'preparando', 'pronto',
'entregue', 'cancelado');
CREATE TYPE orderOrigin AS ENUM ('app', 'balcao', 'telefone');
CREATE TYPE promotionType AS ENUM ('desconto', 'brinde', 'fidelidade');
CREATE TYPE paymentMethod AS ENUM ('credito', 'debito', 'pix', 'dinheiro',
'naConta');
CREATE TYPE paymentStatus AS ENUM ('pendente', 'pago', 'reembolsado',
'cancelado');
CREATE TYPE expenseType AS ENUM ('agua', 'luz', 'fornecedor', 'outros');

CREATE TABLE usersAdmin (
    idUser SERIAL PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    email VARCHAR(255) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL,
    type userType NOT NULL,
    active BOOLEAN DEFAULT TRUE,
    dateRegister TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    lastLogin TIMESTAMP
);

CREATE TABLE clients (
    idClient SERIAL PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    email VARCHAR(255) UNIQUE,
    phone VARCHAR(20) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL,
    birthDate DATE,
    dateRegister TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    loyaltyPoints INT DEFAULT 0
);

CREATE TABLE desks (
    idDesk SERIAL PRIMARY KEY,
    deskNumber INT UNIQUE NOT NULL,
    qrCodeUid VARCHAR(50) UNIQUE NOT NULL,
    capacity INT NOT NULL,
    condition deskCondition DEFAULT 'livre'
);

CREATE TABLE categories (
    idCategory SERIAL PRIMARY KEY,
    name VARCHAR(50) NOT NULL,
    description TEXT,
    menuOrder INT DEFAULT 0,
    active BOOLEAN DEFAULT TRUE
);
```

```
CREATE TABLE menuItems (
    idItem SERIAL PRIMARY KEY,
    idCategory INT NOT NULL,
    name VARCHAR(100) NOT NULL,
    description TEXT,
    price DECIMAL(10,2) NOT NULL,
    cost DECIMAL(10,2),
    timePreparation INT,
    emphasis BOOLEAN DEFAULT FALSE,
    active BOOLEAN DEFAULT TRUE,
    imagePath VARCHAR(255),
    options JSON,
    FOREIGN KEY (idCategory) REFERENCES categories(idCategory)
);

CREATE TABLE stock (
    idStock SERIAL PRIMARY KEY,
    idItem INT NOT NULL,
    amount DECIMAL(10,2) NOT NULL,
    unit VARCHAR(20) NOT NULL,
    updateDate TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (idItem) REFERENCES menuItems(idItem)
);

CREATE TABLE stockMovements (
    idMovement SERIAL PRIMARY KEY,
    idStock INT NOT NULL,
    type stockMovementType NOT NULL,
    amount DECIMAL(10,2) NOT NULL,
    description TEXT,
    timeDate TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (idStock) REFERENCES stock(idStock)
);

CREATE TABLE orders (
    idOrder SERIAL PRIMARY KEY,
    idClient INT NOT NULL,
    idDesk INT NOT NULL,
    timeDate TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    status orderStatus DEFAULT 'recebido',
    total DECIMAL(10,2) NOT NULL,
    observation TEXT,
    origin orderOrigin DEFAULT 'app',
    FOREIGN KEY (idClient) REFERENCES clients(idClient),
    FOREIGN KEY (idDesk) REFERENCES desks(idDesk)
);

CREATE TABLE orderItems (
    idOrderItem SERIAL PRIMARY KEY,
    idOrder INT NOT NULL,
    idItem INT NOT NULL,
    amount INT NOT NULL DEFAULT 1,
    unitPrice DECIMAL(10,2) NOT NULL,
    custom JSON,
    observation TEXT,
    FOREIGN KEY (idOrder) REFERENCES orders(idOrder) ON DELETE CASCADE,
    FOREIGN KEY (idItem) REFERENCES menuItems(idItem)
);
```

```
CREATE TABLE feedbacks (
    idFeedback SERIAL PRIMARY KEY,
    idOrder INT NOT NULL,
    idClient INT NOT NULL,
    rating INT CHECK (rating BETWEEN 1 AND 5),
    comments TEXT,
    timeDate TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    answered BOOLEAN DEFAULT FALSE,
    adminResponse TEXT,
    FOREIGN KEY (idOrder) REFERENCES orders(idOrder),
    FOREIGN KEY (idClient) REFERENCES clients(idClient)
);

CREATE TABLE promotions (
    idPromotion SERIAL PRIMARY KEY,
    title VARCHAR(100) NOT NULL,
    description TEXT,
    type promotionType NOT NULL,
    discountValue DECIMAL(10,2),
    code VARCHAR(50) UNIQUE,
    validFrom TIMESTAMP,
    validUntil TIMESTAMP,
    status BOOLEAN DEFAULT TRUE
);

CREATE TABLE logs (
    idLog SERIAL PRIMARY KEY,
    idUser INT NULL,
    idClient INT NULL,
    action VARCHAR(100) NOT NULL,
    description TEXT,
    ipAddress VARCHAR(50),
    timeDate TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (idUser) REFERENCES usersAdmin(idUser),
    FOREIGN KEY (idClient) REFERENCES clients(idClient)
);
```

Scripts DML (Data Manipulation Language)

```
-- Usuários Administradores
INSERT INTO usersAdmin (name, email, password, type)
VALUES
('Carlos Gerente', 'carlos@restaurante.com', '12345', 'gerente'),
('Ana Cozinha', 'ana@restaurante.com', 'abc123', 'cozinha');

-- Clientes
INSERT INTO clients (name, email, phone, password, birthDate)
VALUES
('João Silva', 'joao@email.com', '11999999999', 'senha123', '1995-03-10'),
('Maria Souza', 'maria@email.com', '11988888888', 'senha456', '1988-07-21');

-- Categorias
INSERT INTO categories (name, description)
VALUES
('Bebidas', 'Bebidas frias e quentes'),
('Lanches', 'Lanches rápidos e porções');

-- Itens do Cardápio
INSERT INTO menuItems (idCategory, name, description, price, cost)
VALUES
(1, 'Suco de Laranja', 'Natural, 300ml', 8.50, 2.50),
(1, 'Café Expresso', '200ml, puro', 5.00, 1.00),
(2, 'X-Burger', 'Pão, carne, queijo e salada', 18.00, 7.00),
(2, 'Batata Frita', 'Porção média crocante', 10.00, 3.00);

-- Mesas
INSERT INTO desks (deskNumber, qrCodeUid, capacity, condition)
VALUES
(1, 'QR123', 4, 'livre'),
(2, 'QR124', 2, 'ocupada');

-- Pedidos
INSERT INTO orders (idClient, idDesk, total, status)
VALUES
(1, 1, 28.00, 'preparando'),
(2, 2, 23.00, 'recebido');

-- Itens dos Pedidos
INSERT INTO orderItems (idOrder, idItem, amount, unitPrice)
VALUES
(1, 3, 1, 18.00),
(1, 1, 1, 8.50),
(2, 4, 1, 10.00),
(2, 2, 1, 5.00);

-- Feedbacks
INSERT INTO feedbacks (idOrder, idClient, rating, comments)
VALUES
(1, 1, 5, 'Comida excelente!'),
(2, 2, 4, 'Entrega rápida e lanche gostoso.');

-- Promoções
INSERT INTO promotions (title, description, type, discountValue, code,
validFrom, validUntil)
VALUES
('Promoção de Verão', 'Desconto de R$3,00 em bebidas', 'desconto', 3.00,
'VERAO2025', '2025-01-01', '2025-03-01');

-- Logs
INSERT INTO logs (idUser, action, description, ipAddress)
VALUES
(1, 'Login', 'Acesso ao sistema', '192.168.1.10'),
(2, 'Alteração', 'Atualizou cardápio', '192.168.1.11');
```

Scripts DQL (Data Query Language)

```
— 1. Listar todos os clientes
SELECT * FROM clients;

-- 2. Clientes com mais de 0 pontos
SELECT name, loyaltyPoints FROM clients WHERE loyaltyPoints > 0;

-- 3. Pedidos e nomes dos clientes
SELECT o.idOrder, c.name AS clientName, o.total, o.status, o.timeDate
FROM orders o
JOIN clients c ON o.idClient = c.idClient;

-- 4. Itens dos pedidos com o nome do produto
SELECT oi.idOrder, m.name AS itemName, oi.amount, oi.unitPrice
FROM orderItems oi
JOIN menuItems m ON m.idItem = oi.idItem;

-- 5. Total arrecadado por status de pedido
SELECT status, SUM(total) AS totalAmount
FROM orders
GROUP BY status;

-- 6. Mesas ocupadas
SELECT deskNumber, capacity, condition
FROM desks
WHERE condition = 'ocupada';

-- 7. Itens mais caros do cardápio
SELECT name, price FROM menuItems ORDER BY price DESC LIMIT 3;

-- 8. Média de avaliação dos clientes
SELECT AVG(rating) AS mediaAvaliacoes FROM feedbacks;

-- 9. Promoções válidas hoje
SELECT title, description, discountValue
FROM promotions
WHERE CURRENT_DATE BETWEEN validFrom AND validUntil;

-- 10. Logs do sistema com nome do usuário
SELECT l.action, l.description, u.name AS userName, l.timeDate
FROM logs l
JOIN usersAdmin u ON u.idUser = l.idUser;
```

Conclusão

O presente projeto teve como objetivo a concepção e a estruturação de um banco de dados relacional para o sistema **“EasyOrder”**, uma solução de cardápio digital voltada à modernização e otimização do atendimento em restaurantes. O propósito central foi desenvolver uma base de dados **robusta, íntegra e escalável**, capaz de suportar todas as funcionalidades essenciais do sistema — desde o cadastro de clientes e a navegação pelo cardápio até a realização de pedidos e o registro de feedbacks.

O desenvolvimento do modelo seguiu as etapas fundamentais do projeto de banco de dados. Inicialmente, foi definido o **Minimundo**, responsável por delimitar o escopo do sistema e suas regras de negócio. A partir dessa análise, elaborou-se o **Modelo Entidade-Relacionamento (MER)**, que estruturou conceitualmente as entidades, seus atributos e os relacionamentos existentes. Em seguida, produziu-se o **Diagrama Entidade-Relacionamento (DER)**, representação gráfica que permitiu visualizar de forma clara e organizada a estrutura lógica do banco, a qual posteriormente foi traduzida para a **linguagem de definição de dados (DDL)** por meio da criação dos scripts SQL.

O modelo resultante foi construído com foco na **integridade e consistência dos dados**, utilizando **chaves primárias** para identificar de forma única cada registro e **chaves estrangeiras** para assegurar a coerência entre as tabelas relacionadas — como a ligação entre **clients** e **orders**. A estrutura foi projetada seguindo os princípios de **normalização**, com o intuito de eliminar redundâncias e evitar anomalias de atualização. Além disso, o relacionamento **muitos-para-muitos (N:M)**, identificado entre **orders** e **menuItems**, foi resolvido de maneira adequada por meio da entidade associativa **orderItems**.

Durante o processo de modelagem, um dos desafios foi definir o relacionamento entre **promotions** e **orders**. Embora, em um primeiro momento, pudesse parecer que cada pedido estaria vinculado a apenas uma promoção, optou-se por um modelo mais flexível de **muitos-para-muitos**, implementado através da tabela intermediária **orderPromotions**. Essa decisão de design, mais complexa, confere ao sistema maior escalabilidade e adaptabilidade, permitindo que no futuro múltiplas promoções possam ser aplicadas a um mesmo pedido, de acordo com novas estratégias de negócio e campanhas promocionais.

A execução deste projeto possibilitou a aplicação prática dos principais conceitos teóricos abordados na disciplina de **Projeto de Banco de Dados**, reforçando a importância de uma **modelagem conceitual bem estruturada** antes da implementação física. Como perspectivas futuras, o modelo atual serve como **base sólida** para a expansão do sistema, podendo incorporar novos módulos, como **controle de caixa**, **reserva de mesas** e **integração com gateways de pagamento digital**.

Em síntese, o banco de dados do **“EasyOrder”** representa uma **fundação sólida, eficiente e escalável**, plenamente capaz de sustentar as operações do sistema e agregar valor ao negócio, promovendo **maior eficiência operacional** para o restaurante e **melhor experiência** para o cliente final.

ANEXOS


usersAdmin
idUser INT
name VARCHAR(255)
email VARCHAR(255)
password VARCHAR(100)
type ENUM(...)
activo TINYINT
dateRegister DATETIME
lastLogin DATETIME
clientes
idCliente INT
name VARCHAR(255)
email VARCHAR(255)
telefone VARCHAR(20)
password VARCHAR(255)
birthDate DATETIME
dateRegister DATETIME
loyaltyPoints INT
logs
idLog INT
idUser INT
idClientes INT
acao VARCHAR(100)
descricao TEXT(255)
ipAddress VARCHAR(50)
timeData DATETIME
categories
idCategory INT
name VARCHAR(50)
descricao TEXT(255)
order INT
activo TINYINT
menuItens
idItem INT
idCategory INT
name VARCHAR(100)
descricao TEXT(255)
price DECIMAL(10,2)
cost DECIMAL(10,2)
timePreparo INT
emphasis TINYINT
activo TINYINT
imagePath VARCHAR(255)
options JSON
itemOrder
idItemOrder INT
idOrder INT
idItem INT
amount INT
unityPrice DECIMAL(10,2)
custom JSON
observacao TEXT(255)
orders
idOrder INT
idPromocoes INT
idDesk INT
idCliente INT
timeData DATETIME
status ENUM(...)
total DECIMAL(10,2)
observacao TEXT(255)
desks
idDesk INT
deskNumber INT
qrcodeUid VARCHAR(50)
capacity INT
condition ENUM(...)
feedbacks
idFeedback INT
idOrder INT
idCliente INT
avaliacao INT
comments TEXT(255)
timeData DATETIME
stock
idStock INT
idItem INT
amount DECIMAL(10,2)
unity VARCHAR(20)
updateData DATETIME
stockMovement
idMovement INT
idStock INT
type ENUM(...)
amount DECIMAL(10,2)
descricao TEXT(255)
timeDate DATETIME
promocoes
idPromocoes INT
title VARCHAR(100)
descricao VARCHAR(45)
type ENUM(...)
valorDesconto DECIMAL(10,2)
codigo VARCHAR(50)
validoDe DATETIME
validoAte DATETIME
status TINYINT