Trabalho II - Implementação do Banco de Dados

Allison Chistian Silva Parentes, Amanda Letícia Pereira Medeiros, Giovanna Alves da Silva

Curso Tecnólogo em Gestão de Dados – Centro de Educação Aberta e a Distância (CEAD)
Universidade Federal do Piauí (UFPI) – Teresina – PI – Brazil
{allison.chistian@ufpi.edu.br amandamedeiros, giovanna.da@ufpi.edu.br}

Abstract. *This paper proposes an information system for the Jardim Confeitado Sweet Shop, aimed at automating and optimizing the purchasing and selling process. Through the implementation of electronic orders and integration with sales management system, the goal is to streamline service, ensure order accuracy, and facilitate the payment process. The system aims to improve customer experience, increase operational efficiency, and provide greater sales control.*

Resumo. *Este artigo propõe um sistema de informação para a Loja de Doces Jardim Confeitado, com o objetivo de automatizar e otimizar o processo de compra e venda. Através da implementação de comandas eletrônicas e integração com um sistema de gestão de vendas, pretende-se agilizar o atendimento, garantir a precisão dos pedidos e facilitar o processo de pagamento. O sistema visa melhorar a experiência do cliente, aumentar a eficiência operacional e proporcionar um maior controle das vendas.*

1. Informações Gerais

O sistema de informação proposto visa automatizar o processo de compra e venda, proporcionando uma experiência mais eficiente tanto para os clientes quanto para os funcionários. Os clientes serão fornecidos de comandas eletrônicas ao entrarem na loja; os funcionários registrarão os pedidos dos clientes nas comandas, incluindo os itens escolhidos e a quantidade desejada. As comandas estarão integradas com um sistema de gestão de vendas, onde serão atualizadas conforme os pedidos são adicionados, sendo processados e gerenciados. As comandas eletrônicas permitirão que os funcionários registrem os pedidos de forma rápida e eficiente, reduzindo o tempo de espera dos clientes. Também, proporciona uma redução de erros no registro de pedidos, uma vez que as comandas serão preenchidas de forma digital. Ao finalizar seus pedidos, os clientes poderão se dirigir ao caixa para efetuar o pagamento, onde terá acesso aos pedidos registrados nas comandas eletrônicas, facilitando a cobrança e o fechamento da conta. Diferentes métodos de pagamento serão aceitos, proporcionando flexibilidade aos clientes. A implementação deste sistema de informação será essencial para modernizar e otimizar as operações da loja Jardim Confeitado, aumentando a

eficiência operacional, trazendo a melhoria na qualidade do atendimento e na satisfação do cliente.

2. Diagrama lógico do banco de dados

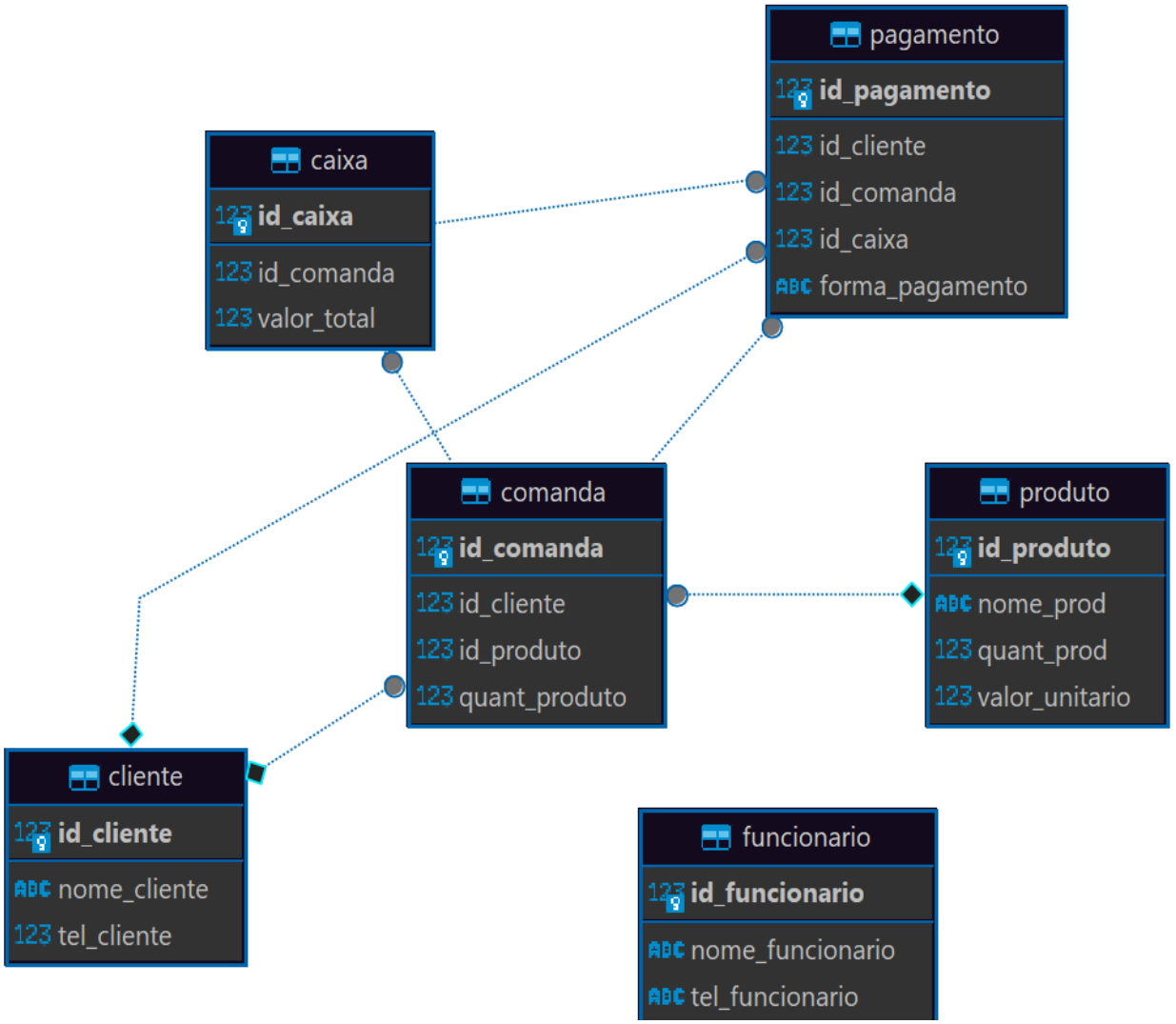

Figura 1. Diagrama lógico

3. *Script* SQL de criação e povoamento do banco de dados

Script SQL

```sql
CREATE TABLE cliente(
    id_cliente INTEGER PRIMARY KEY AUTOINCREMENT,
    nome_cliente VARCHAR NOT NULL,
    tel_cliente BIGINT NOT NULL
)
CREATE TABLE produto (
    id_produto INTEGER PRIMARY KEY AUTOINCREMENT,
    nome_prod VARCHAR NOT NULL,
    quant_prod INTEGER,
    valor_unitario REAL NOT NULL
)
CREATE TABLE comanda (
    id_comanda INTEGER PRIMARY KEY AUTOINCREMENT,
    id_cliente INTEGER,
    id_produto INTEGER,
    quant_produto BIGINT NOT NULL,
   FOREIGN KEY (id_cliente) REFERENCES cliente (id_cliente),
   FOREIGN KEY (id_produto) REFERENCES produto (id_produto)
);
CREATE TABLE funcionario (
    id_funcionario INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
    nome_funcionario VARCHAR NOT NULL,
    tel_funcionario INTEGER NOT NULL
)
CREATE TABLE caixa (
    id_caixa INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
    id_comanda INTEGER NOT NULL,
    valor_total REAL NOT NULL,
    FOREIGN KEY (id_comanda) REFERENCES comanda(id_comanda)
);
CREATE TABLE pagamento (
    id_pagamento INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
    id_cliente INTEGER,
    id_comanda INTEGER NOT NULL,
    id_caixa INTEGER NOT NULL,
    forma_pagamento VARCHAR NOT NULL,
    FOREIGN KEY (id_cliente) REFERENCES cliente (id_cliente),
    FOREIGN KEY (id_comanda) REFERENCES comanda (id_comanda),
    FOREIGN KEY (id_caixa) REFERENCES caixa (id_caixa)
);
```

Figura 1. Criando tabelas

Script SQL

```sql
INSERT INTO cliente (nome_cliente, tel_cliente) VALUES ('Antonio Jose', '9999-0000')
INSERT INTO cliente (nome_cliente, tel_cliente) VALUES ('Antonia Maria', '9999-1111')
INSERT INTO cliente (nome_cliente, tel_cliente) VALUES ('Jose Pedro', '9999-2222')
INSERT INTO cliente (nome_cliente, tel_cliente) VALUES ('Josefa Teresa', '9999-3333')
INSERT INTO cliente (nome_cliente, tel_cliente) VALUES ('Maria Julieta', '9999-4444')
INSERT INTO cliente (nome_cliente, tel_cliente) VALUES ('Antonella Chagas', '9999-4422');
INSERT INTO cliente (nome_cliente, tel_cliente) VALUES ('Patricia Medyna', '9999-4122');
INSERT INTO cliente (nome_cliente, tel_cliente) VALUES ('Legolas Elf', '0000-5432');
INSERT INTO cliente (nome_cliente, tel_cliente) VALUES ('Zé Carioca', '1299-5422');
INSERT INTO cliente (nome_cliente, tel_cliente) VALUES ('Patolino Shopping', '3499-4522');
```

Figura 2. Inserindo clientes

Script SQL

```sql
INSERT INTO funcionario (nome_funcionario, tel_funcionario) VALUES ('Allison Chistian', '8888-0000');
INSERT INTO funcionario (nome_funcionario, tel_funcionario) VALUES ('Amandinha Letícia', '8888-3333');
INSERT INTO funcionario (nome_funcionario, tel_funcionario) VALUES ('Gica Alves', '8888-8888');
INSERT INTO funcionario (nome_funcionario, tel_funcionario) VALUES ('Tio Patinhas', '8888-2222');
INSERT INTO funcionario (nome_funcionario, tel_funcionario) VALUES ('Tio Patinhas', '8888-2221');
```

Figura 4. Inserindo funcionários

Script SQL

```sql
INSERT INTO produto (nome_prod, quant_prod, valor_unitario) VALUES (“Cocada”, 10, 10.00)
INSERT INTO produto (nome_prod, quant_prod, valor_unitario) VALUES (“Bolo de Cenoura”, 5, 8.00)
INSERT INTO produto (nome_prod, quant_prod, valor_unitario) VALUES (“Bolo Red Velvet”, 8, 10.00)
INSERT INTO produto (nome_prod, quant_prod, valor_unitario) VALUES (“Cupcake de Pistache”, 20, 15.00)
INSERT INTO produto (nome_prod, quant_prod, valor_unitario) VALUES (“Brownie de Doce de Leite”, 15, 20.00)
INSERT INTO produto (nome_prod, quant_prod, valor_unitario) VALUES (“Coxinha de Brigadeiro”, 20, 9.00)
INSERT INTO produto (nome_prod, quant_prod, valor_unitario) VALUES (“Macaroon Vanilla”, 20, 14.00)
INSERT INTO produto (nome_prod, quant_prod, valor_unitario) VALUES (“Torta de Cookie Inteira”, 1, 70.00)
INSERT INTO produto (nome_prod, quant_prod, valor_unitario) VALUES (“Torta de Cookie Fatia”, 10, 9.00)
INSERT INTO produto (nome_prod, quant_prod, valor_unitario) VALUES (“Torta de Limão Inteira”, 2, 60.00)
INSERT INTO produto (nome_prod, quant_prod, valor_unitario) VALUES (“Torta de Limão Fatia”, 10, 8.00)
```

Figura 5. Inserindo produtos

Script SQL

```sql
INSERT INTO comanda (id_cliente, id_produto, quant_produto) VALUES (1, 4, 2);
INSERT INTO comanda (id_cliente, id_produto, quant_produto) VALUES (2, 2, 3);
INSERT INTO comanda (id_cliente, id_produto, quant_produto) VALUES (9, 8, 4);
INSERT INTO comanda (id_cliente, id_produto, quant_produto) VALUES (10, 5, 1);
INSERT INTO comanda (id_cliente, id_produto, quant_produto) VALUES (7, 11, 1);
INSERT INTO comanda (id_cliente, id_produto, quant_produto) VALUES (6, 6, 1);
INSERT INTO comanda (id_cliente, id_produto, quant_produto) VALUES (11, 10, 5);
```

Figura 6. Inserindo comanda

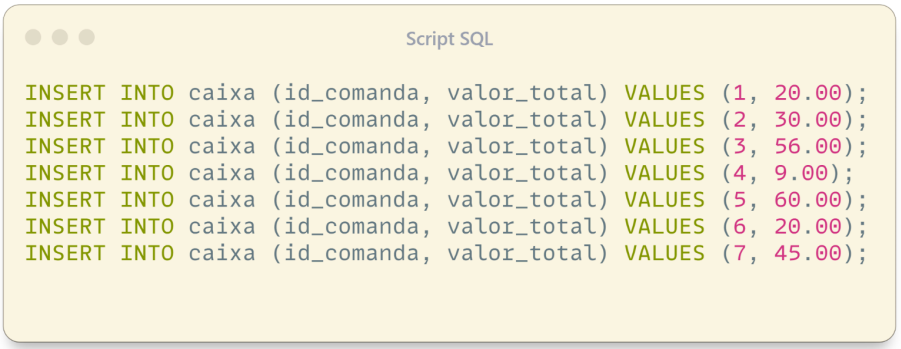

Figura 7. Inserindo caixa

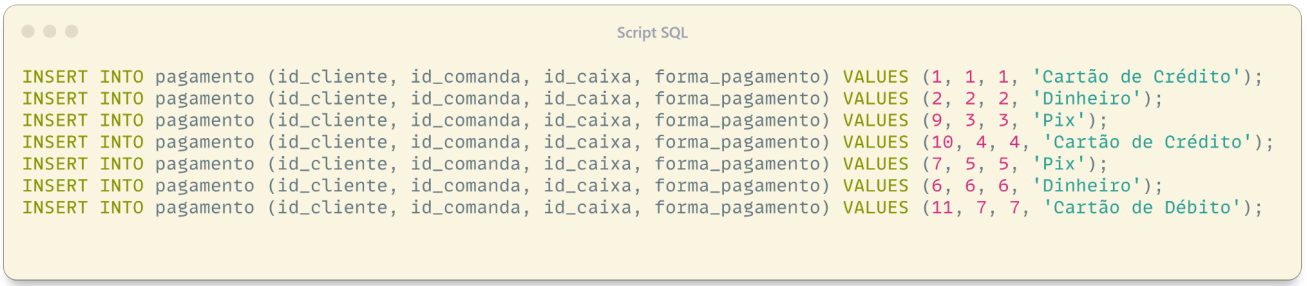

Figura 8. Inserindo pagamento

4. Consultas SQL

Script SQL

```sql
SELECT * FROM cliente;
SELECT * FROM produto;
SELECT * FROM comanda;
SELECT * FROM caixa;
SELECT * FROM pagamento;
SELECT * FROM funcionario;
-- Comparação:
SELECT
    *
FROM produto
WHERE valor_unitario  < 10;
SELECT
    *
FROM produto
WHERE quant_prod ⩾ 20;
SELECT
    *
FROM produto
WHERE quant_prod = 4;
-- Lógicos:
-- verificar valores altos
SELECT
    *
FROM produto
WHERE valor_unitario BETWEEN 15 AND 50;
-- hmm um bolinho ou cocada..
SELECT
    *
FROM produto
WHERE nome_prod = 'Bolo Red Velvet' OR nome_prod = 'Cocada Branca';
-- Bora usar o LIKE:
-- temos um funcionário bilionário?
SELECT nome_funcionario FROM funcionario WHERE nome_funcionario LIKE 'Tio%';
-- será que temos um personagem de filme famoso como cliente fiel?
SELECT nome_cliente FROM cliente WHERE nome_cliente LIKE 'Leg%';
-- JOINS
-- verificar numero da comanda por cliente
SELECT
    comanda.id_comanda,
    cliente.nome_cliente
FROM comanda
JOIN cliente ON comanda.id_cliente = cliente.id_cliente;
-- verificar tipo de pagamento por id do caixa
SELECT
    caixa.id_caixa,
    pagamento.forma_pagamento
FROM pagamento
JOIN caixa ON pagamento.id_caixa = caixa.id_caixa;
--verificar qual nome do produto e sua quantidade escolhida em cada comanda
SELECT
    comanda.id_comanda,
    produto.nome_prod,
    comanda.quant_produto
FROM comanda
INNER JOIN produto ON comanda.id_produto = produto.id_produto;
-- qual valor e qual nome do cliente a partir de uma comanda
SELECT
    comanda.id_comanda,
    cliente.nome_cliente,
    caixa.valor_total
FROM comanda
INNER JOIN cliente ON comanda.id_cliente = cliente.id_cliente
INNER JOIN caixa ON comanda.id_comanda = caixa.id_comanda;
-- verificar valor total, numero da comanda, numero do caixa e forma de pagamento
SELECT
    pagamento.id_pagamento,
    caixa.id_caixa,
    comanda.id_comanda,
    caixa.valor_total,
    pagamento.forma_pagamento
FROM pagamento
JOIN caixa ON pagamento.id_caixa = caixa.id_caixa
JOIN comanda ON pagamento.id_comanda = comanda.id_comanda;
```

Script SQL

```sql
-- Agregação:
SELECT
    COUNT(id_comanda) AS quant_comandas
FROM comanda;
SELECT
    SUM (valor_total) AS soma_de_pagamentos
FROM caixa;
SELECT
    AVG (valor_total) AS media_dos_valores
FROM caixa;
SELECT
    MIN (valor_total) AS menor_valor
FROM caixa;
SELECT
    MAX (valor_total) AS maior_valor
FROM caixa;
-- Order By:
-- verificar nomes dos clientes em ordem não alfabetica
SELECT
    nome_cliente
FROM cliente
ORDER BY nome_cliente DESC;
-- ordenar produtos pela sua quantidade, de forma decrescente
SELECT
    nome_prod,
    quant_prod
FROM produto
ORDER BY quant_prod DESC;
-- ordenar forma de pagamento pela mais usada
SELECT
    forma_pagamento,
    COUNT(forma_pagamento) AS quant_forma_pagamento
FROM pagamento
GROUP BY forma_pagamento
ORDER BY quant_forma_pagamento DESC;
```

5. Link da apresentação

Slides

Apresentação