DESENVOLVIMENTO DE UM SISTEMA DE GERENCIAMENTO DE BANCO DE DADOS, UTILIZANDO O PYTHON INTEGRADO COM O MYSQL

Kiara Silva Gomes (UFCG-CDA) kiaragomeseng@hotmail.com
José Caio Santos de Souza (UFCG-CDSA) jose.caio@estudante.ufcg.edu.br
Monique Gabrielly de Araújo (UFCG-CDSA) monique.gabrielly@estudante.ufcg.edu.br
Cecir Barbosa de Almeida Farias (UFCG-CDA) cecir.barbosa@professor.ufcg.edu.br

Resumo

Com o mundo globalizado, tornou-se imprescindível às empresas investirem em tecnologia e informação para se manterem competitivas ou no mínimo sobreviverem, tendo em vista que precisam melhorar continuamente seus produtos e serviços. O presente trabalho tem o intuito de realizar uma análise de sistema por meio de diagramas de entidades e relacionamentos (DER) e, em seguida, desenvolver e testar o sistema de banco de dados de uma farmácia em expansão. Esse método foi realizado através de uma abordagem quantitativa de dados fictícios dessa empresa, e dividida em quatro etapas. Sendo primeiramente a identificação do problema, a descrição de objetivos e funcionalidades de um sistema de dados. Em seguida, houve a criação de um Diagrama de Entidades e Relacionamentos (DER) e transformação para o Modelo Relacional (MR). Após isto, os dados fictícios foram armazenados em um Sistema Gerenciador de Banco de Dados MySQL, e finalmente, foram realizadas consultas e testes no sistema aplicando alguns comandos da linguagem SQL - *Structures Query Language*, como *Create Table, Select, Insert, Update e Delete.*

Palavras-Chaves: *Banco de Dados; SGBD; Farmácia; MySql; Python*

1. Introdução

Bancos de dados desempenham um papel fundamental na organização, armazenamento e recuperação de informações em sistemas computacionais. Eles são o alicerce tecnológico que sustenta uma ampla gama de aplicações, desde sistemas de gerenciamento de estoque até redes sociais. No coração desse ecossistema de gerenciamento de dados, linguagens como SQL (Structured Query Language) desempenham um papel fundamental. O SQL é uma linguagem

de consulta que permite aos usuários interagir com bancos de dados de maneira eficiente, executando operações como inserção, consulta, atualização e exclusão de dados.

Conforme Cardoso (2017), "a linguagem SQL - *STRUCTURED QUERY LANGUAGE* (Linguagem de consulta estruturada) surgiu na década de 70 com base nos modelos de dados relacionais, quando a empresa IBM, desenvolveu a linguagem SEQUEL - *SEQUENCE QUERY LANGUAGE* (Linguagem de consulta sequencial), cujo objetivo era implementar o modelo relacional de dados, e assim posteriormente o termo evoluiu e se reduziu para SQL".

A linguagem SQL sempre vem sendo atualizada e busca armazenar e processar informações em um Banco de Dados Relacional. Ela permite armazenar informações em formato tabular, com linhas e colunas representando diferentes atributos de dados e as várias relações entre os valores dos dados, podendo utilizar instruções SQL para armazenar, atualizar, remover, pesquisar e recuperar informações do banco de dados, também pode-se usar SQL para otimizar a performance do banco de dados.

Em um nível mais avançado, empresas e organizações muitas vezes recorrem a sistemas gerenciadores de banco de dados (SGBD) robustos e escaláveis, como o MySQL, para atender às suas necessidades de armazenamento e processamento de dados em larga escala.

A evolução contínua dessas tecnologias de gerenciamento de dados desempenha um papel essencial na capacitação de empresas para tomar decisões informadas, impulsionar a inovação e atender às demandas cada vez maiores por informações precisas e acessíveis. De acordo com Codd (1970), o pioneiro dos bancos de dados relacionais, afirmou: "Um banco de dados é uma coleção de fatos armazenados que podem ser processados por métodos disponíveis para revelar informações".

De acordo com Santos (2017) "Um banco de dados toda entidade possui um ou mais atributos chave, cujos valores são distintos para cada entidade do conjunto de entidades. Portanto, os valores desses atributos podem identificar uma única entidade".

O Modelo de Entidade-Relacional (MER) é uma estrutura conceitual fundamental para o projeto e representação de bancos de dados relacionais. Proposto pelo cientista da computação Peter Chen (1976), o MER continua a ser uma ferramenta essencial na modelagem de dados. O diagrama entidade relacionamento (ER) se concentra na representação de entidades, atributos e relacionamentos entre essas entidades, fornecendo uma visualização clara e abstrata dos dados de um sistema.

O "Diagrama de Entidade e Relacionamento (DER) trata de um modelo popular para modelagem conceitual de dados, baseado em três conceitos básicos: entidades, atributos e relacionamentos, onde as entidades representam um elemento do mundo real, o atributo se refere as características ou propriedades que descrevem as entidades do banco de dados e os relacionamentos descrevem a forma como uma entidade se relaciona com a outra e os seus atributos", conforme descreve Ribeiro (1992).

A transformação de um diagrama Entidade-Relacional (ER) para um modelo Relacional é um processo crucial no projeto de banco de dados, que visa traduzir a representação abstrata de entidades, atributos e relacionamentos do ER em uma estrutura de tabelas que pode ser implementada em um SGBD. Esse processo de transformação permite uma implementação eficaz da estrutura de dados em um SGBD, tornando os dados acessíveis e manipuláveis por meio de consultas SQL, preservando a integridade e a consistência dos dados.

Um Sistema de Gerenciamento de Banco de Dados ou Sistema de Gestão de Bases de Dados é o software responsável pelo gerenciamento de um ou mais bancos de dados. Seu principal objetivo é retirar da aplicação cliente a responsabilidade de gerenciar o acesso, a persistência, a manipulação e a organização dos dados. "O SGBD que manipula banco de dados relacionais, ou SQL, exige esquemas predefinidos, não permitindo a inserção de dados sem definir os tipos de dados e relacionamentos" segundo Date (2014). Em suma, o SGBD relacional disponibiliza uma interface para que seus clientes possam incluir, alterar ou consultar dados previamente armazenados. Em bancos de dados relacionais a interface é constituída pelas API (Application Programming Interface) ou drivers do SGBD que executam comandos na linguagem SQL, de acordo com Nascimento (2014), onde os dados são ordenados em tabelas (relações) com colunas (atributos) e linhas (registros).

Este artigo demonstra como o SQL e um Sistema de Gerenciamento de Banco de Dados se alinham a uma visão que capacita as organizações a explorar todo o potencial dos dados em um contexto orientado para a informação. Dentro desse cenário, o artigo explora a importância do SQL na gestão de dados, destacando sua relevância no panorama atual da tecnologia da informação. Portanto, o presente trabalho tem por objetivo desenvolver um sistema de gerenciamento de banco de dados com o SGBD MySQL integrado com a linguagem de programação Python. A farmácia decidiu expandir sua variedade de produtos, além de medicamentos. A finalidade da farmácia é cadastrar clientes e impulsionar as vendas por meio de descontos. A implementação de um programa de descontos voltado para clientes idosos demanda a definição de critérios de elegibilidade e a implementação de um sistema eficaz para

verificar a idade. Isso garante que apenas clientes que atendam aos requisitos estabelecidos tenham acesso aos descontos.

2. Referencial teórico

2.1 Diagrama de entidades e relacionamentos (DER)

O Diagrama de Entidade-Relacionamento (DER) é uma ferramenta crucial na modelagem de dados, representando entidades, atributos e relacionamentos de um sistema. Amplamente utilizado em projetos de banco de dados, esse diagrama cria uma representação visual clara e concisa da estrutura subjacente dos dados. Uma técnica que permite modelar conceitualmente as informações necessárias em um sistema, identificando entidades relevantes e as relações entre elas. De acordo com Elmasri e Navathe (2016), "DER é uma ferramenta importante para a comunicação entre analistas, projetistas de sistemas de informações e usuários finais".

No DER, as entidades representam objetos do mundo real, enquanto os atributos definem suas características. Segundo Connolly e Begg (2014), "entidades podem ser pessoas, lugares, objetos do mundo real ou conceitos", e atributos são "características que descrevem essas entidades".

Os relacionamentos nos diagramas de Entidade-Relacionamento mostram como as entidades estão conectadas e interagem. Elmasri e Navathe (2016) explicam que "um relacionamento é uma associação entre entidades" e que "os relacionamentos podem ser classificados quanto ao grau, cardinalidade e opção". Cardinalidade descreve quantos objetos de uma entidade estão relacionados a objetos de outra entidade, enquanto a opção indica se a participação é obrigatória ou opcional. Connolly e Begg (2014) destacam que "a cardinalidade e a opção são importantes para entender a natureza dos relacionamentos".

Na prática, Diagramas de Entidade-Relacionamento são amplamente usados no projeto de bancos de dados relacionais. Eles ajudam a criar um modelo conceitual dos dados, base para esquemas de banco de dados. Segundo Elmasri e Navathe (2016), "o processo de modelagem ER é fundamental para garantir que o esquema de banco de dados reflita com precisão os requisitos da organização". O Diagrama de Entidade-Relacionamento é uma poderosa ferramenta visual usada na modelagem de dados. O DER oferece uma visualização intuitiva, facilitando a compreensão de estruturas de dados complexas em sistemas de informação.

2.3 Sistema de banco de dados

Um Banco de Dados é uma poderosa tecnologia de gerenciamento de dados que permite o armazenamento, manipulação e recuperação de informações em formato tabular. Segundo Date (2014), "Um banco de dados se trata de uma coleção persistente de dados usada pelos sistemas de aplicação de uma empresa, é um local onde informações necessárias para as operações de uma organização são armazenadas". Ele é amplamente utilizado em sistemas de gerenciamento de banco de dados relacionais, como Microsoft SQL Server, Oracle Database, MySQL e PostgreSQL.

O MySQL - SGBD escolhido para este trabalho, "é um sistema gerenciador de banco de dados relacional de código aberto usado na maioria das aplicações gratuitas para gerir suas bases de dados. O serviço utiliza a linguagem SQL (Structure Query Language – Linguagem de Consulta Estruturada), que é a linguagem mais popular para inserir, acessar e gerenciar o conteúdo armazenado num banco de dados" segundo Pisa (2012).

No SQL, os dados são organizados em tabelas compostas por linhas e colunas, e as consultas são realizadas usando a linguagem SQL para recuperar, atualizar ou inserir dados. Além disso, o SQL oferece recursos avançados para garantir a integridade dos dados, realizar transações seguras e otimizar consultas para melhor desempenho. O SQL permite a criação e consulta dessas tabelas, facilitando o acesso eficiente aos dados para várias aplicações, desde sistemas de comércio eletrônico até sistemas de gerenciamento de recursos humanos.

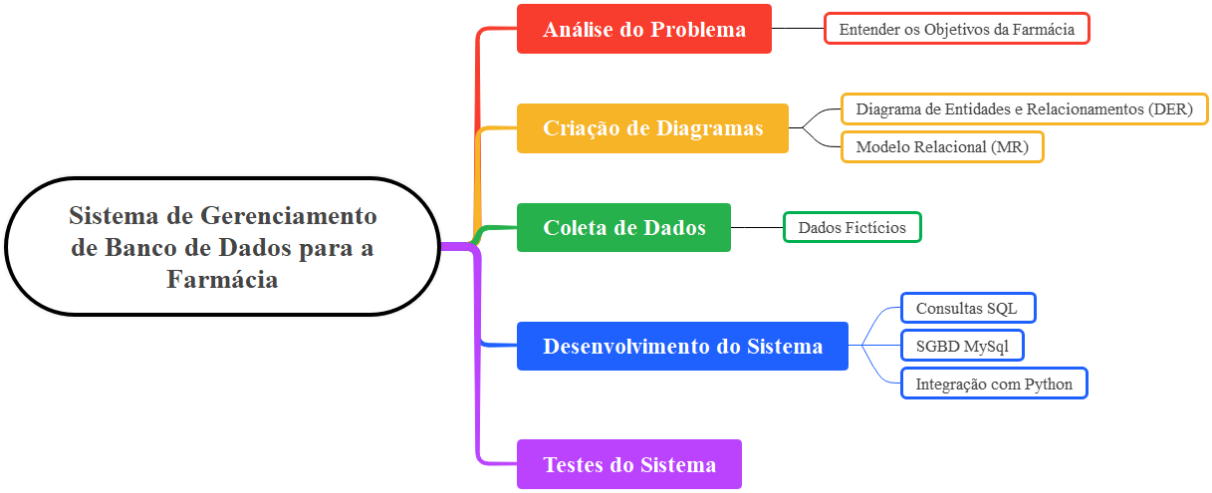
A linguagem Python foi escolhida para implementação do sistema em questão. "A linguagem inclui diversas estruturas de alto nível, como listas e dicionários, e coleções de módulos prontos para uso, além de frameworks de terceiros que podem ser adicionados. A linguagem é interpretada através de bytecode pela máquina virtual Python, tornando o código portável. Com isso, é possível compilar aplicações em uma plataforma e rodar em outros sistemas ou executar o direto do código da fonte" (Borges, 2010).

3. Metodologia

O presente estudo é uma pesquisa de natureza quantitativa com dados fictícios e caráter exploratório dividida em quatro etapas. Na primeira etapa, é feita uma análise do problema para entender qual serão os objetivos da farmácia. A segunda etapa envolve a criação do Diagrama

de Entidades e Relacionamentos (DER) e a sua transformação para o Modelo Relacional (MR). A Figura 1 apresenta o fluxo metodológico dessa pesquisa.

Figura 1 - DER do Sistema



Fonte: Autores (2023)

Na terceira etapa, os dados fictícios coletados foram armazenados em um banco de dados SQL, utilizando scripts Python. A combinação de SQL e Python permite um acesso eficaz e programático aos dados, sendo inseridos nas tabelas do banco de dados SQL com a ajuda de bibliotecas como o SQL Alchemy ou o psycopg2.

A quarta etapa consiste na realização de consultas e testes do sistema, aplicando comandos do banco de dados SQL, como Create Table, Select, Insert, Update e Delete. Diante dessas etapas, o presente trabalho propõe-se a desenvolver e analisar os dados coletados, aplicando consultas no sistema de banco de dados.

4. Resultados

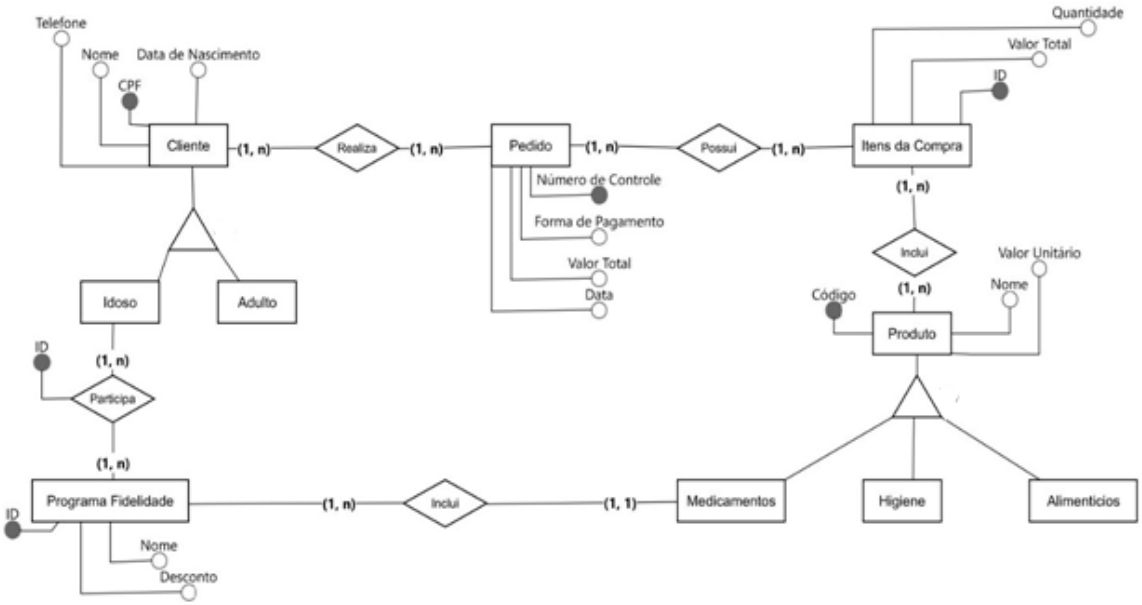
4.1 Escolha do SGBD

O sistema de gerenciamento de banco de dados (SGBD) escolhido para desenvolver as finalidades da farmácia foi o MySQL, uma opção de código aberto e gratuita, evitando custos adicionais. Destacando-se pelo desempenho, rapidez e eficiência, o MySQL é capaz de lidar com grandes volumes de dados e transações simultâneas de forma altamente escalável. Sua compatibilidade com a linguagem SQL padrão foi um fator determinante na escolha, além da facilidade de integração com a linguagem de programação Python.

4.2 Diagrama de Entidade – Relacionamento (DER)

O Diagrama de Entidade-Relacionamento (DER) na Figura 2 mostra as entidades, relacionamentos, atributos e cardinalidades do sistema de gerenciamento da farmácia, fornecendo uma visão precisa da estrutura dos dados e das principais conexões entre as partes do sistema, além disso, é importante para a modelagem do sistema e a comunicação das informações necessárias para o seu funcionamento.

Figura 2 - DER do Sistema



Fonte: Autores (2023)

Após a criação do Diagrama de Entidades e Relacionamento, foi realizada a transformação para o Modelo Relacional e todos os atributos foram colocados nas tabelas (Tabela 1) corretas seguindo as regras de cardinalidade.

Tabela 1 – Tabelas do banco de dados

Tabelas (Entidades)	Colunas (Atributos)

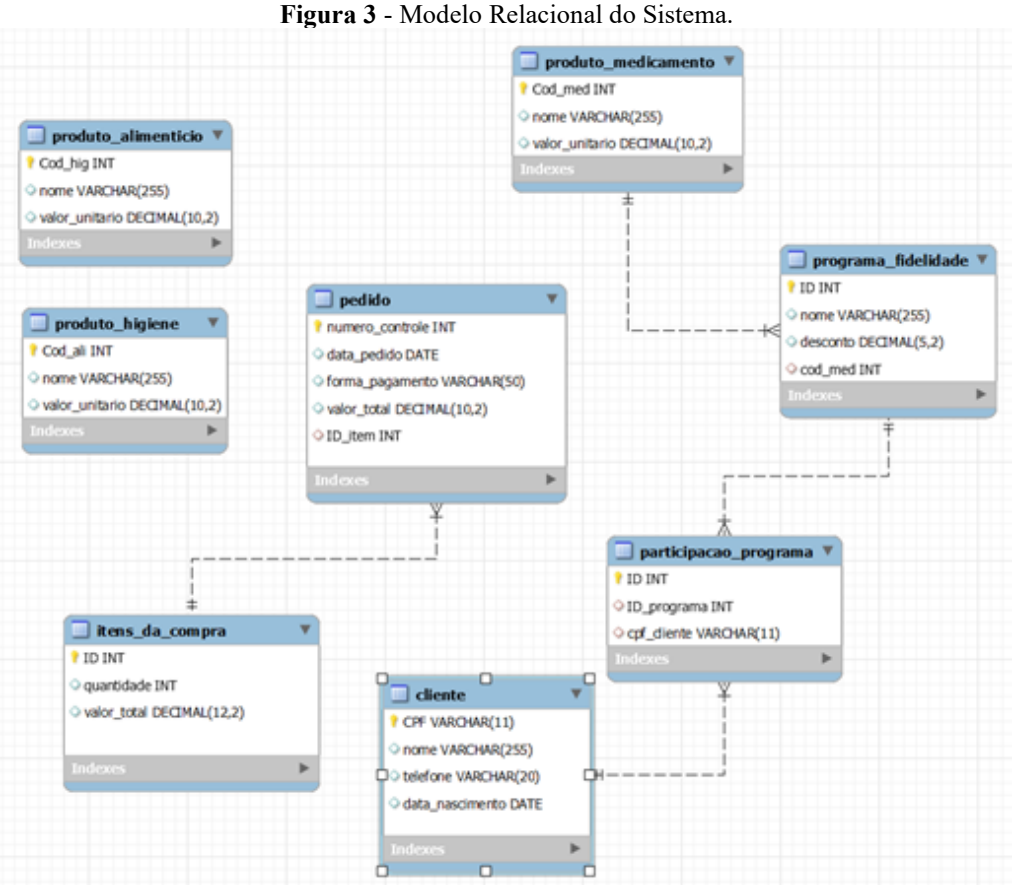
Cliente	(CPF, nome, telefone, data-nascimento)
Pedido	(numero-controle, data-pedido, forma-pagamento, valor-total)
Itens-da-compra	(ID, quantidade, valor-total)
Produto-medicamento	(Cod-med, nome, valor-unitario)
Produto-higiene	(Cod-ali, nome, valor-unitario)
Produto-alimenticio	(Cod-hig, nome, valor-unitario)

Fonte: Autores (2023)

4.3 Modelo Relacional do SGBD

O modelo relacional desempenha um papel fundamental no gerenciamento de bancos de dados, utilizando tabelas para representar entidades do mundo real no contexto do sistema de gerenciamento da farmácia, como clientes, produtos e pedidos. Cada tabela é caracterizada por colunas que definem seus atributos. As chaves primárias e estrangeiras são essenciais ao estabelecer conexões entre tabelas, facilitando consultas por meio da linguagem SQL para operações eficazes de CRUD (Create, Read, Update, Delete). Em resumo, o modelo relacional criado (Figura 3) oferece eficiência e desempenha um papel vital em diversas aplicações de banco de dados.

Figura 3 - Modelo Relacional do Sistema.



Fonte: Autores (2023).

4.3 Estrutura de tabelas do banco de dados.

A estrutura principal do banco de dados foi desenvolvida para atender às necessidades específicas da farmácia, visando armazenar e gerenciar informações relacionadas a clientes, pedidos, produtos e programas de fidelidade. Com um foco especial em clientes idosos que podem se beneficiar de descontos exclusivos, a Figura 4 apresenta a estrutura das tabelas do banco de dados. Essas tabelas foram criadas para permitir o registro detalhado de informações sobre clientes, seus pedidos e os produtos oferecidos pela farmácia. A integração do programa de fidelidade ao sistema, com descontos exclusivos para clientes idosos, fortalece a proposta. As tabelas são inter-relacionadas por meio de chaves estrangeiras, garantindo a integridade dos dados e possibilitando consultas e relatórios eficazes para o gerenciamento da loja.

Figura 4 – Tabelas do banco de dados

```sql
-- Tabela Cliente
CREATE TABLE Cliente (
    CPF VARCHAR(11) PRIMARY KEY,
    nome VARCHAR(255),
    telefone VARCHAR(20),
    data_nascimento DATE
);

-- Tabela Pedido
CREATE TABLE Pedido (
    numero_controle INT PRIMARY KEY,
    data_pedido DATE,
    forma_pagamento VARCHAR(50),
    valor_total DECIMAL(10, 2),
    ID_item INT,
    FOREIGN KEY (ID_item) REFERENCES Itens_da_compra(ID)
);

-- Tabela Itens_da_compra
CREATE TABLE Itens_da_compra (
    ID INT PRIMARY KEY,
    quantidade INT,
    valor_total DECIMAL(10, 2)
);

-- Tabela Produto-medicamento
CREATE TABLE Produto_medicamento (
    Cod_med INT PRIMARY KEY,
    nome VARCHAR(255),
    valor_unitario DECIMAL(10, 2)
);

-- Tabela Produto-higiene
CREATE TABLE Produto_higiene (
    Cod_all INT PRIMARY KEY,
    nome VARCHAR(255),
    valor_unitario DECIMAL(10, 2)
);

-- Tabela Produto-alimenticio
CREATE TABLE Produto_alimenticio (
    Cod_hig INT PRIMARY KEY,
    nome VARCHAR(255),
    valor_unitario DECIMAL(10, 2)
);

-- Tabela Programa-fidelidade
CREATE TABLE Programa_fidelidade (
    ID INT PRIMARY KEY,
    nome VARCHAR(255),
    desconto DECIMAL(5, 2),
    cod_med INT,
    FOREIGN KEY (cod_med) REFERENCES Produto_medicamento(Cod_med)
);

-- Tabela Participação-programa
CREATE TABLE Participacao_programa (
    ID INT PRIMARY KEY,
    ID_programa INT,
    cpf_cliente VARCHAR(11),
    FOREIGN KEY (ID_programa) REFERENCES Programa_fidelidade(ID),
    FOREIGN KEY (cpf_cliente) REFERENCES Cliente(CPF)
);
```

Fonte: Autores (2023)

4.4 Conexão com banco de dados

A escolha da linguagem de programação Python é motivada por sua simplicidade e pela existência de bibliotecas especializadas, tornando-a ideal para o desenvolvimento de aplicativos

de gerenciamento de banco de dados, tornando o processo mais eficiente e acessível. A interação entre o sistema e o banco de dados MySQL é facilitada pelo código Python, utilizando a biblioteca 'mysql.connector', o código estabelece uma conexão com o banco de dados, utilizando informações de autenticação como nome de usuário, senha e o nome do banco de dados. Essa operação é realizada com a seguinte seção do código que se encontra na Figura 5.

Figura 5 - Conexão com o MySQL.

```
import mysql.connector
from tabulate import tabulate
from datetime import datetime

conexao = mysql.connector.connect(
    host='localhost',
    user='root',
    password='1981',
    database='banco_dados_farmacia',
)

cursor = conexao.cursor()
```

Fonte: Autores (2023)

4.5 Códigos em Python para permitir as interações com o Banco de Dados.

4.5.1 Operações SQL integrado com Python.

A Figura 6 mostra trechos do código em Python que executam diversas operações no SGBD. Estas operações incluem a criação de tabelas, alterações nos tipos de dados, inserção de registros, modificação de dados existentes e realização de consultas. Cada uma dessas operações é implementada por meio de funções dedicadas no código.

Figura 6 – Algumas operações SQL do sitema em Python

```
def alternativa_b():
    comando = f"ALTER TABLE Itens_da_compra MODIFY valor_total DECIMAL(12, 2)"
    cursor.execute(comando)
    conexao.commit()
    print(">>> Adicionado com Sucesso. <<<")

def alternativa_c():
    comando1 = '''INSERT INTO Cliente (CPF, nome, telefone, data_nascimento)
                  VALUES ('12345678901', 'João Silva', '555-1234', '1990-05-15')'''
    cursor.execute(comando1)
    conexao.commit()
    print(">>> Adicionado com Sucesso. <<<")

    comando2 = '''INSERT INTO Pedido (numero_controle, data_pedido, forma_pagamento, valor_total, ID_item)
                     VALUES (2, '2023-11-06', 'Boleto Bancário', 75.50, 2)'''
    cursor.execute(comando2)
    conexao.commit()
    print(">>> Pedido adicionado com Sucesso. <<<")

def alternativa_d():
    comando = "UPDATE Cliente SET telefone = '555-9999' WHERE CPF = '12345678901'"
    cursor.execute(comando)
    conexao.commit()

def alternativa_e():
    comando = "SELECT nome, telefone FROM cliente WHERE cpf = 12345678901"
    cursor.execute(comando)
    resultado = cursor.fetchall()
    print(resultado)

def alternativa_f():
    comando = "SELECT * FROM pedido WHERE valor_total BETWEEN 50 AND 150"
    cursor.execute(comando)
    resultado = cursor.fetchall()
    print(resultado)


def alternativa_h():
    comando = "SELECT C.nome, P.desconto" \
            "FROM Cliente C " \
            "JOIN Participacao_programa PP ON C.CPF = PP.cpf_cliente" \
            "JOIN Programa_fidelidade P ON PP.ID_programa = P.ID"
    cursor.execute(comando)
    resultado = cursor.fetchall()
    print(resultado)
```

```
#Loop principal do programa
while True:
    print("=" *41)
    print("-"*10,"Consultas do roteiro.", "-"*8)
    print("=" *41)
    print("-"*18,"MENU", "-"*17)
    print(" ")
    print("1. Criar tabelas.")
    print("2. Alter o tipo de dado.")
    print("3. Inserções de dados")
    print("4. Alterações em dados já cadastrados.")
    print("5. Consulta a determinado dado de uma tabela.")
    print("6. Consulta que pertença a um intervalo entre valor mínimo e valor máximo.")
    print("7. Consulta que mostra o resultado em ordem crescente.")
    print("8. Consulta envolvendo dados de duas ou três tabelas.")
    print("9. Exclusão de um registro de uma tabela.")
    print("10. Finalizar Sistema")
    print(" ")
    print("=" *40)

    opcao = int(input("Digite a opção desejada: "))
    print("=" *40)

    if opcao == 1:
        alternativa_a()
    elif opcao == 2:
        alternativa_b()
    elif opcao == 3:
        alternativa_c()
    elif opcao == 4:
        alternativa_d()
    elif opcao == 5:
        alternativa_e()
    elif opcao == 6:
        alternativa_f()
    elif opcao == 7:
        alternativa_g()
    elif opcao == 8:
        alternativa_h()
    elif opcao == 9:
        alternativa_i()
    elif opcao == 10:
        print("Sistema Fechado")
        cursor.close()
        conexao.close()
        break
    else:
        print("Opção Invalidada")
```

Fonte: Autores (2023)

O programa continua em execução até que o usuário opte por encerrá-lo, proporcionando uma abordagem interativa e flexível para interagir com o SGBD. Esse design modular, com funções específicas para cada operação, contribui para a clareza e manutenção do código, facilitando sua compreensão e modificação quando necessário.

4.5.2 Interface principal do sistema

O código Python é uma opção simples de sistema de gerenciamento para a farmácia que interage com o MySQL, a Figura 7 apresenta as várias funcionalidades do sistema, incluindo a opção de cadastrar clientes, inserir produtos, programas de fidelidade e a capacidade de realizar um pedido, acessar produtos comercializados, entre outras funcionalidades.

Figura 7 – Código da interface principal do sistema

```
#Loop principal do programa
while True:
    print("=" *41)
    print("-"*11,"SISTEMA FÁRMACIA.", "-"*11)
    print("=" *41)
    print("-"*18,"MENU", "-"*17)
    print(" ")
    print("1. Cadastrar Cliente.")
    print("2. Clientes")
    print("3. Programa Fidelidade.")
    print("4. Produtos Comercializados.")
    print("5. Cadastrar Produto.")
    print("6. Realizar Pedido.")
    print("0. Finalizar Sistema.")
    print(" ")
    print("=" *40)

    opcao = int(input("Digite a opção desejada: "))
    print("=" *40)

    if opcao == 1:
        cadastrar_cliente()
    elif opcao == 2:
        clientes()
    elif opcao == 3:
        programa_fidelidade()
    elif opcao == 4:
        produtos_comercializados()
    elif opcao == 5:
        cadastrar_produto()
    elif opcao == 6:
        realizar_pedido()
    elif opcao == 0:
        print("Sistema Fechado")
        cursor.close()
        conexao.close()
        break
    else:
        print("Opção Invalidada")

```

Fonte: Autores (2023)

A função `programa_fidelidade` (Figura 8) fornece uma interface para cadastrar programas de fidelidade e visualizar esses programas no banco de dados da farmácia. Esses programas são inseridos com base nas informações fornecidas pelo usuário, como nome, desconto e o código do medicamento associado. Além disso, a função oferece ao usuário a opção de visualizar quais clientes podem participar desses programas, proporcionando uma maneira intuitiva de gerenciar os programas de desconto da farmácia. O design modular e interativo da função facilita a inclusão e a manutenção desses programas no sistema, contribuindo para a eficiência e flexibilidade do gerenciamento de fidelidade na farmácia.

Figura 8 – Função programa fidelidade

```
def programa_fidelidade():
    while True:
        print("-"*40)
        print("MENU: PROGRAMA FIDELIDADE")
        print("-"*40)
        print("1. Cadastra Programa")
        print("2. Visualizar Programa Fidelidade")
        print("3. Participação dos clientes no Programa fidelidade")
        print("4. Encerrar")

        tipo = int(input("Escolha uma das opções acima:"))

        if tipo == 1:
            nome_prog = str(input("Qual o nome do Programa"))
            desconto = float(input("Informe o Desconto: "))

            comando = f'INSERT INTO programa_fidelidade (Nome, desconto)
                       VALUES ("{nome_prog}","{desconto}")'
            cursor.execute(comando)
            conexao.commit()

            print(" ")
            print("PROGRAMA CADASTRADO COM SUCESSO")
            print("-"*40)
        elif tipo == 2:
            comando = f'SELECT * FROM programa_fidelidade '

            cursor.execute(comando)
            resultado = cursor.fetchall()
            if resultado:
                headers = [desc[0] for desc in cursor.description]
                print(tabulate(resultado, headers, tablefmt="grid"))

        elif tipo == 3:
            print("-"*40)
            print("PARTCIPAÇÃO NO PROGRAMA")
            print("-"*40)
            print("1. De todos os Clientes")
            print("2. Apenas um Cliente")
            print("3. Encerrar")
            print(" ")

            opcao_fidelidade = int(input("Para ter acesso escolha uma das opções:"))

            if opcao_fidelidade == 1:
                print("Participação de todos os clientes no programa")
                comando = f'SELECT * FROM participacao_programa'
                cursor.execute(comando)
                resultado = cursor.fetchall()

            elif opcao_fidelidade == 2:
                participacao_cliente_progFidelidade()

            elif opcao_fidelidade == 3:
                break
        elif tipo == 4:
            break
```

Fonte: Autores (2023)

Os códigos apresentados na Figura 9 oferecem opções para adicionar produtos, incluindo medicamentos, produtos de higiene e alimentos ao banco de dados. O usuário fornece os dados do produto, como código, nome e valor unitário, que são então inseridos nas tabelas correspondentes. O menu 'cadastrar_produto' proporciona flexibilidade ao usuário, permitindo a escolha do tipo de produto que deseja adicionar, tornando o processo de gerenciamento de produtos na farmácia mais interativo e personalizado.

Figura 9 – Função cadastrar produto.

```
def cadastrar_produto():

    while True:
        print("-"*40)
        print(" "*7,"MENU: CADASTRAR PRODUTO")
        print("-"*40)
        print("1. Medicamento")
        print("2. Alimentício")
        print("3. Higiene")
        print("4. Encerrar")
        print("-"*40)

        tipo = int(input("Escolha uma das opções para cadastro: "))
        print("-"*40)

        if tipo == 1:
            cadastrar_medicamento()
        elif tipo == 2:
            cadastrar_alimento()
        elif tipo == 3:
            cadastrar_higiene()
        elif tipo == 4:
            break

def cadastrar_medicamento():
    nome_med = str(input("Informe o nome do medicamento: "))
    preco_unit = float(input("Qual o valor do medicamento? "))
    incl_prog_fidelidade = str(input("Incluir medicamento no programa fidelidade? [SIM] ou [NÃO]"))
    tipo_produto = "medicamento"

    if incl_prog_fidelidade.lower() == "sim":
        incl_prog_fidelidade = "sim"

    else:
        incl_prog_fidelidade = "nao"

    comando = f'''INSERT INTO produtos (Nome, preco_unitario, part_programa_fidelidade,
    tipo_produto) VALUES ("{nome_med}", "{preco_unit}", "{incl_prog_fidelidade}", "{tipo_produto}")'''

    cursor.execute(comando)
    conexao.commit()

    print(" ")
    print("-"*40)
    print(">>> Produto cadastrado com sucesso. <<<")
```

Fonte: Autores (2023)

O menu de cliente (Figura 10) fornece uma interface interativa para os funcionários da farmácia, tornando intuitivas e organizadas todas essas operações de gerenciamento de clientes. Isso facilita o processo para os funcionários, mesmo aqueles que não são especialistas em programação ou SQL.

Figura 10 – Menu cliente

```
if __name__ == "__main__":
    while True:
        print("Escolha uma opção:")
        print("1. Inserir novo cliente")
        print("2. Atualizar cliente")
        print("3. Listar clientes")
        print("4. Excluir cliente")
        print("5. Sair")
        opcao = input("Opção: ")

        if opcao == "1":
            cpf = input("CPF: ")
            nome = input("Nome: ")
            telefone = input("Telefone: ")
            data_nascimento = input("Data de Nascimento: ")
            inserir_cliente(cpf, nome, telefone, data_nascimento)
        elif opcao == "2":
            cpf = input("CPF do cliente a ser atualizado: ")
            novo_nome = input("Novo nome: ")
            novo_telefone = input("Novo telefone: ")
            nova_data_nascimento = input("Nova data de nascimento: ")
            atualizar_cliente(cpf, novo_nome, novo_telefone, nova_data_nascimento)
        elif opcao == "3":
            listar_clientes()
        elif opcao == "4":
            cpf = input("CPF do cliente a ser excluído: ")
            excluir_cliente(cpf)
        elif opcao == "5":
            print("Saindo do programa.")
            break
        else:
            print("Opção inválida.")
```

Fonte: Autores (2023)

A Figura 11 apresenta códigos que desempenham um papel fundamental na gestão de clientes, tornando a interação com o banco de dados mais eficiente e facilitando o atendimento ao cliente. Esses códigos garantem que os dados dos clientes estejam sempre atualizados e acessíveis, contribuindo para a melhoria do atendimento ao cliente e para a tomada de decisões com base em informações precisas. A função "inserir_cliente" permite que a farmácia adicione novos clientes ao banco de dados, crucial para manter um registro preciso de clientes e suas informações de contato, facilitando o atendimento e o relacionamento com os clientes. A função "atualizar_cliente" oferece a capacidade de manter os registros de clientes atualizados, permitindo ajustes nos dados conforme necessário. A função "listar_clientes" possibilita que a farmácia acesse uma lista de todos os clientes cadastrados, útil para conferência, pesquisa e identificação rápida de clientes durante vendas ou prestação de serviços. A função "excluir_cliente" é eficaz quando um cliente não deseja mais fazer parte da base de dados da farmácia, permitindo a remoção de registros de clientes de forma eficiente.

Figura 11 – Funções do meu Cliente

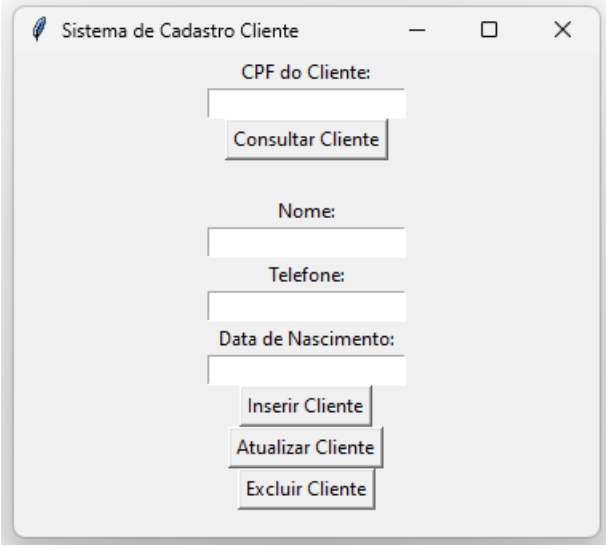
```python
# Função para inserir um novo cliente
def inserir_cliente(cpf, nome, telefone, data_nascimento):
    comando = "INSERT INTO Cliente (CPF, nome, telefone, data_nascimento) VALUES (%s, %s, %s, %s)"
    valores = (cpf, nome, telefone, data_nascimento)
    cursor.execute(comando, valores)
    conexao.commit()
    print("Cliente inserido com sucesso.")

# Função para atualizar os dados de um cliente
def atualizar_cliente(cpf, novo_nome, novo_telefone, nova_data_nascimento):
    comando = "UPDATE Cliente SET nome = %s, telefone = %s, data_nascimento = %s WHERE CPF = %s"
    valores = (novo_nome, novo_telefone, nova_data_nascimento, cpf)
    cursor.execute(comando, valores)
    conexao.commit()
    print("Cliente atualizado com sucesso.")

# Função para listar todos os clientes
def listar_clientes():
    cursor.execute("SELECT * FROM Cliente")
    clientes = cursor.fetchall()
    for cliente in clientes:
        print(f"CPF: {cliente[0]}, Nome: {cliente[1]}, Telefone: {cliente[2]}, Data de Nascimento: {cliente[3]}")

# Função para excluir um cliente pelo CPF
def excluir_cliente(cpf):
    comando = "DELETE FROM Cliente WHERE CPF = %s"
    valores = (cpf,)
    cursor.execute(comando, valores)
    conexao.commit()
    print("Cliente excluído com sucesso.")
```

Fonte: Autores (2023)

4.6 Interface gráfica com tkinter

O Tkinter é uma biblioteca padrão do Python para criar interfaces gráficas, possibilitando o desenvolvimento de interfaces interativas de maneira simples. No contexto do sistema de farmácia, o Tkinter (Figura 12) é utilizado para criar uma interface gráfica que facilita o gerenciamento de informações de clientes no SGBD. Mesmo sem estilização, essa interface torna as operações de banco de dados mais acessíveis e intuitivas para os funcionários da farmácia.

Figura 12 – Interface gráfica Tkinter



Fonte: Autores (2023)

A Figura 13 mostra um trecho do código Tkinter, onde inicia importando os módulos necessários, utilizando o mysql.connector para estabelecer uma conexão com o banco de dados MySQL e o tkinter para criar a interface gráfica. Quatro funções de gerenciamento de clientes são definidas: consultar_cliente, inserir_cliente, atualizar_cliente e excluir_cliente. Cada uma dessas funções executa operações específicas no banco de dados com base nos dados fornecidos pelo usuário na interface.

A janela principal que contém os botões para operações de gerenciamento de clientes, essa janela é exibida ao executar o programa. O código inicia o loop principal da interface gráfica com window.mainloop(), permitindo a interação do usuário. Após a conclusão do loop, as conexões com o banco de dados são fechadas para garantir a liberação adequada de recursos. Esse design modular e interativo facilita a interação com o sistema de gerenciamento de clientes.

Figura 13 – Código do Tkinter

```python
import mysql.connector
import tkinter as tk
from tkinter import messagebox

def conectar_ao_banco():
    try:
        return mysql.connector.connect(
            host='localhost',
            user='root',
            password='1981',
            database='banco_dados_farmacia'
        )
    except mysql.connector.Error as err:
        messagebox.showerror("Erro de Conexão", f"Erro ao conectar ao banco de dados: {err}")
        exit(1)

def consultar_cliente():
    cpf = cpf_entry.get()
    cursor.execute("SELECT nome, data_nascimento FROM Cliente WHERE CPF = %s", (cpf,))
    result = cursor.fetchone()
    if result:
        nome, data_nascimento = result
        resultado_label.config(text=f"Nome: {nome}, Data de Nascimento: {data_nascimento}")
    else:
        resultado_label.config(text="Cliente não encontrado")

def inserir_cliente():
    cpf = cpf_entry.get()
    nome = nome_entry.get()
    telefone = telefone_entry.get()
    data_nascimento = data_nascimento_entry.get()
    try:
        cursor.execute("INSERT INTO Cliente (CPF, nome, telefone, data_nascimento) VALUES (%s, %s, %s, %s)",
                       (cpf, nome, telefone, data_nascimento))
        db.commit()
        resultado_label.config(text="Cliente inserido com sucesso")
    except mysql.connector.Error as err:
        db.rollback()
        resultado_label.config(text=f"Erro: {err}")
```

Fonte: Autores (2023)

5. Conclusão

O presente artigo foi desenvolvido com o objetivo de criar um sistema de gerenciamento que atendesse às necessidades de uma farmácia, alcançando com sucesso os principais objetivos estabelecidos. O sistema mostrou-se eficaz ao proporcionar uma interface intuitiva para o cadastro de clientes, produtos e a realização de pedidos. A implementação do programa de fidelidade destacou-se ao incentivar a fidelização dos clientes, oferecendo descontos, especialmente para idosos. Além disso, o sistema manteve um desempenho sólido, com tempos de resposta rápidos na execução de consultas e operações diversas. Olhando para o futuro, é possível identificar áreas que necessitam de melhorias, como o desenvolvimento de uma interface gráfica mais amigável e aprimoramentos no tratamento de erros no código. Essas melhorias são cruciais para manter a funcionalidade do sistema.

pythonpython

REFERÊNCIAS

BORGES, L. E. **Python para Desenvolvedores**, 2010. Rio de Janeiro. ed.2ª. Edição do autor. Disponível em: <https://ark4n.files.wordpress.com/2010/01/python_para_desenvolvedores_2ed.pdf_>. Acesso em: 12 set. 2023.

CARDOSO, GISELLE CRISTINA; CARDOSO, VIRGÍNIA M. **LINGUAGEM SQL**. Saraiva Educação SA, 2017.

CODD, E F. **A relational model of data for large shared data banks,** Communications of The ACM, v. 13, n. 6, p. 377–387, 1970.

CONNOLLY, Thomas; BEG, Carolyn. **Database Systems: A Practical Approach to Design, Implementation, and Management**. Editora Addison-Wesley, 2014.

CHEN, P. Peter. **The Entity-Relationship Model – Toward a Unified View of Data**. ACM Transactions on Database Systems (TODS), 9-36, 1976.

DATE, Christopher J. **Introdução a Sistemas de Banco de Dados**. Editora Campus. 1ª Edição, 2014.

ELMASRI, Ramez; NAVATHE, Shamkant. **Sistemas de Banco de Dados**. Editora Pearson, 6ª edição, 2016.

PISA, Pedro. O que é e como usar o MySQL? **Tech Tudo.** Abril, 2012. Disponível em: <https://www.techtudo.com.br/noticias/2012/04/o-que-e-e-como-usar-o-mysql.ghtml>. Acesso em: 04 out. 2023.

RIBEIRO, C. D. **Um gerador de transações voltado para o modelo entidade-relacionamento.** 119f. Tese (Doutorado em Ciências em Engenharia de Sistemas e Computação) – Universidade Federal do Rio de Janeiro, Rio de Janeiro, RJ, 1992.

SANTOS, P. H. **Diagrama ER é uma ferramenta voltada ao ensino de modelagem de dados.** 44p. Dissertação (Mestrado em Ciências da Computação) – Universidade Federal de Minas Gerais, Belo Horizonte, MG. 2017.

NASCIMENTO, Andersson. **O que é uma API (interface de programação de aplicações)?** Julho, 2014. Teletransporta. Disponível em: < https://canaltech.com.br/software/o-que-e-api/> Acesso em: 12 nov. 2023.