PONTIFÍCIA UNIVERSIDADE CATÓLICA DE SÃO PAULO

FACULDADE CIÊNCIAS EXATAS E TECNOLOGIA

Curso de Graduação em Ciência da Computação

PROJETO CONTROLE DE ALIMENTAÇÃO DIÁRIA

ANNE YUMI KAGAWA RA00319228

CINTHIA ALVES BARRETO RA00325161

ISABELLA RUBIO VENANCIO RA00319673

LEONARDO FAJARDO GRUPIONI RA00319703

São Paulo

2024

SUMÁRIO

1. INTRODUÇÃO

Foi proposto um trabalho desenvolvido que consiste em criar uma aplicação de gerenciador de alimentos, capaz de contar a quantidade de calorias na sua refeição. Ele foi desenvolvido em Python e utiliza um banco de dados do MySQL Workbench. Neste projeto, para atender os requisitos, foi incluído funcionalidades como adicionar, editar, buscar e excluir (CRUD), além de outras funcionalidades necessárias para uma melhor utilização da aplicação.

2. OBJETIVO

A aplicação de Controle de Alimentação Diária tem como objetivo auxiliar os usuários a gerenciar e controlar sua alimentação diária. As principais funcionalidades incluem:

- Registro de Usuários: Permite que novos usuários se cadastrem fornecendo informações básicas.
- Autenticação: Usuários podem fazer login usando CPF e senha.
- Registro de Refeições: Usuários podem registrar suas refeições (almoço ou jantar), selecionando alimentos e proporções consumidas.
- Cálculo de Calorias: O aplicativo calcula automaticamente as calorias ingeridas com base nos alimentos selecionados.
- Histórico de Alimentação: Usuários podem visualizar o histórico de suas refeições e calorias consumidas.
- Gerenciamento de Alimentos: Possibilidade de adicionar novos alimentos ao banco de dados.
- Gerenciamento de Conta: Usuários podem atualizar suas informações pessoais ou excluir sua conta.

3. ARQUITETURA E TECNOLOGIAS UTILIZADAS

A aplicação foi desenvolvida a partir The Clean Arquitecture (Arquitetura Limpa), utilizando as seguintes tecnologias:

- ➢ Linguagem de Programação: Python 3;
- ➢ Front-end: Streamlit (biblioteca Python para criação de interfaces web interativas);
- ➢ Back-end: MySQL (gerenciado através do MySQL Workbench)
- ➢ Bibliotecas Adicionais: mysql-connector-python: Conexão entre Python e MySQL; bcrypt: Criptografia de senhas para armazenamento seguro.
- ➢ Gerenciamento de Sessão: Utilização do st.session_state do Streamlit para manter o estado de login do usuário.

4. SISTEMA CRUD (Create, Read, Update, Delete) EM SQL

A aplicação implementa um sistema completo de CRUD (Create, Read, Update, Delete) conectado ao banco de dados MySQL. A seguir, detalhamos como cada operação é realizada, utilizando o sistema de cadastro de usuário como base.

4.1. Cadastro de Usuário

Operação: Create

- Permite que novos usuários se cadastrem no aplicativo, inserindo informações como CPF, nome, email, celular, senha, idade, peso e sexo.

- Processo:
    - O usuário preenche um formulário de cadastro na interface do Streamlit.
    - A senha é criptografada usando bcrypt antes de ser armazenada.
    - Os dados são inseridos na tabela usuarios do banco de dados MySQL.

- Consulta SQL Utilizada:

```
INSERT INTO usuarios (cpf, nome, email, celular, senha, idade, peso, sexo)
VALUES (%s, %s, %s, %s, %s, %s, %s, %s);
```

4.2. Autenticação do Login

Operação: Read

- Usuários já cadastrados podem fazer login usando seu CPF e senha.

- Processo:
    - O usuário insere seu CPF e senha na interface.
    - O aplicativo busca a senha criptografada correspondente no banco de dados.
    - A senha inserida é comparada com a armazenada usando bcrypt.

- Consulta SQL utilizada:

```
SELECT senha FROM usuarios WHERE cpf = %s;
```

4.3. Atualização de Dados do Usuário

Operação: Update

- Usuários autenticados podem atualizar suas informações pessoais, como nome, email, celular, senha, idade, peso e sexo.

- Processo:
    - O usuário acessa a seção "Gerenciar Conta" na interface.
    - Os dados atuais são carregados e exibidos nos campos correspondentes.
    - O usuário faz as alterações desejadas e confirma.
    - Os dados são atualizados na tabela usuarios no banco de dados.

- Consulta SQL utilizada:

```
UPDATE usuarios
SET nome=%s, email=%s, celular=%s, senha=%s, idade=%s, peso=%s, sexo=%s
WHERE cpf=%s;
```

4.4. Exclusão de Usuário

Operação: Delete

- Usuários podem optar por excluir sua conta e todos os dados associados.

- Processo:
    - O usuário acessa a opção de exclusão na seção "Gerenciar Conta".
    - Uma confirmação é solicitada para evitar exclusões acidentais.
    - Os registros do usuário são removidos da tabela ‘usuarios’, bem como todas as refeições e detalhes associados (graças às chaves estrangeiras com ON DELETE CASCADE).

- Consultas SQL utilizadas:

```
-- Exclusão de detalhes das refeições
DELETE dr FROM detalhes_refeicao dr
JOIN refeicoes r ON dr.id_refeicao = r.id
WHERE r.cpf_usuario = %s;
```

```
-- Exclusão das refeições
DELETE FROM refeicoes WHERE cpf_usuario = %s;
-- Exclusão do usuário
DELETE FROM usuarios WHERE cpf = %s;
```

5. COMO USUFRUIR DA APLICAÇÃO

1. Para executar o aplicativo, é necessário instalar as seguintes bibliotecas Python:

    - **Streamlit** - Biblioteca para criação de interfaces web interativas em Python.

      Para instalar é necessário inserir no terminal:

      ```
      pip install streamlit
      ```

    - **mysql-connector-python** - Permite a conexão entre o Python e o banco de dados MySQL

      Para instalar é necessário inserir no terminal:

      ```
      pip install mysql-connector-python
      ```

    - **bcrypt** - Biblioteca para criptografia de senhas, garantindo armazenamento seguro.

      Para instalar é necessário inserir no terminal:

      ```
      pip install bcrypt
      ```

2. Agora, é necessário configurar o banco de dados. Instale o MySQL e o MySQL Workbench se ainda não o fez.

3. Execute o script SQL fornecido (script.sql) no MySQL Workbench para criar o banco de dados controle_alimentacao e as tabelas necessárias.
    - Abra o MySQL Workbench.
    - Crie uma nova conexão ou use uma existente.
    - Abra uma nova janela de script SQL.
    - Copie e cole o conteúdo do script.sql.
    - Execute o script (botão "Execute" ou tecla de atalho).

4. Configure as credenciais de acesso no arquivo database.py :

```
def criar_conexao():
    conexao = mysql.connector.connect(
        host='localhost',
        user='seu_usuario', # Substitua por seu usuário do MySQL
        password='sua_senha', # Substitua por sua senha do MySQL
        database='controle_alimentacao'
    )
    return conexao
```

5. Navegue até o diretório do projeto onde os arquivos .py estão localizados.

6. Execute o aplicativo usando o Streamlit:
   streamlit run app.py

7. Acesse o aplicativo no seu navegador, geralmente em http://localhost:8501.

6. ESTRUTURA DO PROJETO

A estrutura do projeto segue uma organização modular para facilitar a manutenção e a escalabilidade.

controle_alimentacao/

app.py # Arquivo principal que executa a aplicação Streamlit

auth.py # Módulo responsável pelas operações de autenticação e gerenciamento de usuários

alimentacao.py # Módulo com funções relacionadas ao registro e histórico de alimentação

database.py # Módulo para conexão com o banco de dados MySQL

7. DESCRIÇÃO DOS ARQUIVOS

- **app.py:** Contém a lógica da interface do usuário, gerenciando as rotas e a interação com o usuário através do Streamlit.
- **auth.py:** Implementa as funções de cadastro, autenticação, atualização e exclusão de usuários.
- **alimentacao.py:** Fornece funções para registro de refeições, cálculo de calorias e recuperação do histórico de alimentação.
- **database.py:** Gerencia a conexão com o banco de dados MySQL.
- **GERACAO_TABELAS.sql e DUMP_ALIMENTOS.sql:** Scripts SQL para criar o banco de dados, tabelas e inserir dados iniciais.

8. CONSIDERAÇÕES FINAIS

A aplicação criada oferece uma solução simples e eficaz para o controle da alimentação diária, permitindo que os usuários monitorem suas refeições e calorias consumidas. Além disso, a partir do cadastro com seus dados pessoais, a integração com o MySQL permite que os dados sejam armazenados e recuperados de forma persistente e segura. Foi uma aplicação interessante e muito produtiva ao mesmo tempo que houve maior aprendizado com o erros que apareceram e posteriormente corrigidos.