Documentação do Projeto de E-commerce

Esta documentação apresenta uma visão geral e detalhada do projeto de E-commerce, uma aplicação web desenvolvida com Flask (Python) que visa oferecer uma plataforma robusta para a venda de produtos online. São abordadas as tecnologias empregadas, a estrutura de diretórios, a configuração do ambiente, os principais módulos e as funcionalidades implementadas.

---

1. Introdução

O projeto de E-commerce tem como objetivo criar uma plataforma intuitiva e funcional, englobando desde o cadastro e autenticação dos usuários até a gestão completa de produtos, carrinho de compras, processo de checkout e administração. A aplicação foi desenvolvida utilizando o micro-framework Flask, visando facilidade de manutenção e escalabilidade.

---

2. Tecnologias Utilizadas

2.1. Backend

- **Python:** Linguagem principal utilizada para o desenvolvimento.
- **Flask:** Micro-framework para a criação de aplicações web.
- **SQLite:** Banco de dados relacional leve utilizado para armazenamento.
- **Werkzeug:** Conjunto de ferramentas WSGI que complementa o Flask.

2.2. Frontend

- **HTML5:** Estruturação das páginas da aplicação.
- **CSS3:** Estilização com foco em design moderno e responsivo.
- **Bootstrap 5:** Framework CSS para componentes UI e sistema de grid responsivo.

- **Font Awesome:** Biblioteca de ícones para uma melhor experiência visual.
- **JavaScript:** Adiciona interatividade às páginas.

2.3. Gestão de Pacotes e Ambientes

- **pip:** Gerenciador de pacotes Python.
- **virtualenv:** Criação de ambientes virtuais isolados para o projeto.

2.4. Controle de Versão

- **Git:** Sistema de controle de versão distribuído, possibilitando o versionamento e a colaboração no código.

---

3. Estrutura do Projeto

A estrutura do projeto é organizada para facilitar a manutenção e a escalabilidade da aplicação. Abaixo, apresenta-se a estrutura padrão de diretórios e arquivos:

php

CopiarEditar

```
ecommerce/
├── venv/                    # Ambiente virtual Python
├── app.py                   # Arquivo principal da aplicação Flask
├── config.py                # Configurações da aplicação (ex.: chaves secretas, upload, etc.)
├── database.py              # Conexão com o DB e funções de operações básicas e logs
├── ecommerce.db             # Banco de dados SQLite (criado automaticamente, se não existir)
├── requirements.txt         # Dependências Python do projeto
├── routes/                  # Módulos de rotas organizados via Blueprints
│   ├── __init__.py
```

- │   ├── admin.py   # Rotas do painel de administração
- │   ├── auth.py   # Autenticação: login, cadastro e logout
- │   ├── cart.py   # Gestão do carrinho de compras
- │   ├── checkout.py   # Processos de checkout e finalização de pedidos
- │   ├── main.py   # Rotas principais (homepage, listagem e detalhes dos produtos)
- │   └── user.py   # Perfil do usuário e histórico de pedidos
- ├── static/   # Arquivos estáticos (CSS, JavaScript e imagens)
- │   ├── css/
- │   │   ├── admin.css   # Estilos para o painel administrativo
- │   │   ├── cadastro.css
- │   │   ├── carrinho.css
- │   │   ├── confirmacao_pedido.css
- │   │   ├── contato.css   # Estilos para a página de contato
- │   │   ├── forgot_password.css# Estilos para a recuperação de senha
- │   │   ├── global.css   # Estilos globais aplicados em todo o site
- │   │   ├── historico.css
- │   │   ├── index.css
- │   │   ├── login.css
- │   │   ├── produto.css
- │   │   └── sobre.css   # Estilos para a página "Sobre nós"
- │   ├── js/   # Arquivos JavaScript
- │   └── uploads/   # Pasta para imagens de produtos (configurável)
- └── templates/   # Arquivos de templates HTML (Jinja2)
-     ├── base.html   # Template base para o site (área pública)
-     ├── base_admin.html   # Template base para o painel administrativo

```
    ├── cadastro.html
    ├── carrinho.html
    ├── confirmacao_pedido.html
    ├── contato.html           # Template da página de
contato
    ├── detalhes_produto.html
    ├── forgot_password.html   # Template para
recuperação de senha
    ├── historico_pedidos.html
    ├── index.html             # Página inicial com
listagem dos produtos
    ├── login.html
    ├── sobre.html             # Página "Sobre nós"
    └── admin/                 # Templates exclusivos da
área administrativa
        ├── dashboard.html     # Painel de estatísticas
        ├── detalhes_pedido.html
        ├── editar_produto.html
        ├── logs.html
        ├── novo_produto.html
        ├── pedidos.html
        ├── produtos.html
        └── usuarios.html
```

Diagrama

Diagrama de Casos de Uso para o seu sistema de e-commerce com Flask, contemplando tanto o **usuário comum** quanto o **administrador**.

Descrição dos Atores

- **Usuário**: Cliente que acessa o site para navegar, comprar e visualizar seus pedidos.
- **Administrador**: Usuário com privilégios especiais para gerenciar produtos, usuários e pedidos.

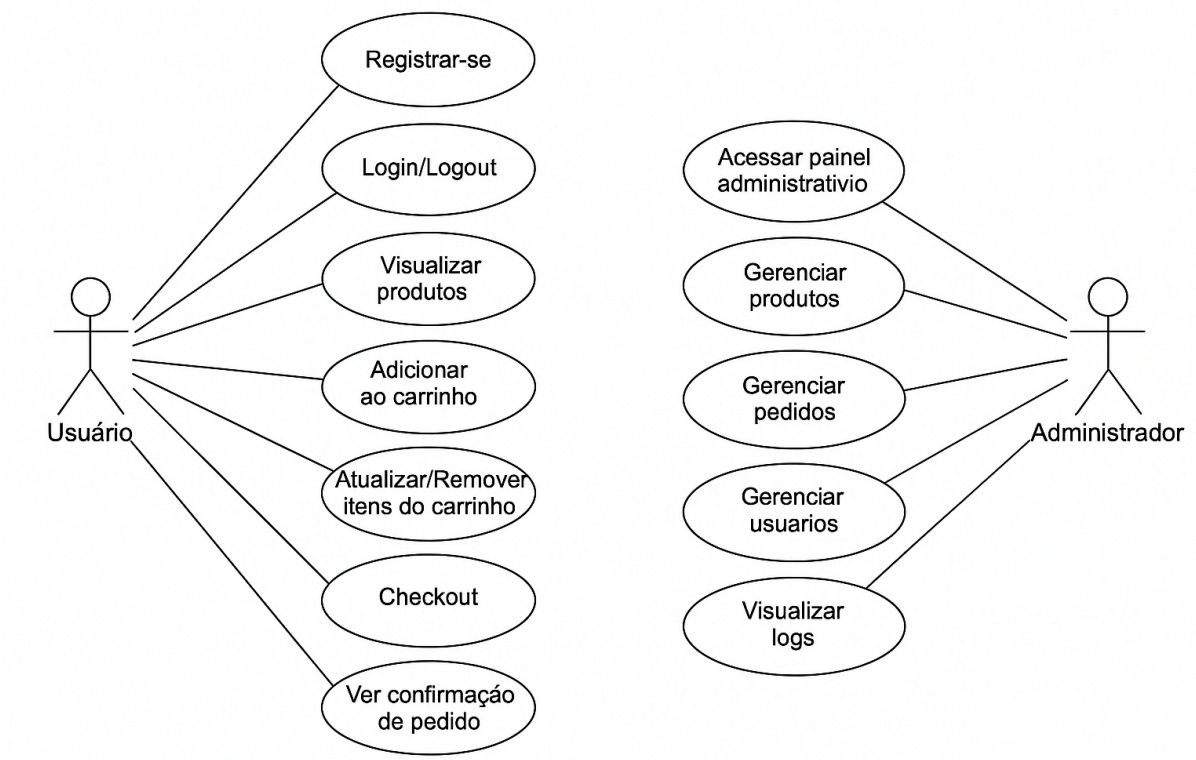
Principais Casos de Uso

Para o Usuário:

- Registrar-se
- Fazer login/logout
- Visualizar produtos
- Ver detalhes de um produto
- Adicionar produto ao carrinho
- Atualizar/remover itens do carrinho
- Finalizar compra (checkout)
- Ver confirmação de pedido
- Ver histórico de pedidos
- Recuperar senha
- Entrar em contato (página de contato)
- Ver página "Sobre"

Para o Administrador (além dos acima, se ele também for usuário):

- Acessar painel administrativo
- Gerenciar produtos (Criar, Editar, Excluir)
- Gerenciar pedidos (Ver, Atualizar status)
- Gerenciar usuários (listar, alterar permissões)
- Visualizar logs de atividades



---

4. Configuração do Ambiente

Siga as etapas abaixo para configurar e executar o projeto em sua máquina:

4.1. Clonar o Repositório

Caso o projeto esteja hospedado em um repositório Git, clone-o utilizando o comando:

bash

CopiarEditar

- `git clone <url_do_repositorio>`
- `cd ecommerce`

4.2. Criar e Ativar o Ambiente Virtual

Crie um ambiente virtual para isolar as dependências do projeto:

bash

CopiarEditar

```
python3 -m venv venv
source venv/bin/activate   # Linux/macOS
```

Ou, no Windows:

bash

CopiarEditar

```
venv\Scripts\activate
```

4.3. Instalar Dependências

Com o ambiente virtual ativado, instale as bibliotecas necessárias:

bash

CopiarEditar

```
pip install -r requirements.txt
```

> **Observação:** Caso o arquivo `requirements.txt` não exista, crie-o manualmente incluindo no mínimo as dependências:
>
> - Flask
> - Werkzeug

4.4. Inicializar o Banco de Dados

O arquivo do banco de dados `ecommerce.db` será gerado automaticamente na primeira execução do `app.py`, através da função `inicializar_db()` (definida em `database.py`). Esta função criará as tabelas necessárias e, se desejado, poderá inserir dados de exemplo.

---

5. Principais Módulos e Componentes

5.1. `app.py`

- **Função:** Ponto de entrada da aplicação.
- **Responsabilidades:**
    - Instanciar a aplicação Flask.
    - Carregar as configurações do `config.py`.
    - Registrar os Blueprints contidos na pasta `routes/`.
    - Inicializar o banco de dados.
    - Configurar a sessão.

5.2. `config.py`

- **Conteúdo:** Variáveis de configuração essenciais, tais como:
    - `SECRET_KEY`: Chave para segurança das sessões (recomenda-se armazená-la via variável de ambiente em produção).
    - `UPLOAD_FOLDER`: Diretório para armazenamento de imagens de produtos.
    - `ALLOWED_EXTENSIONS`: Extensões de arquivos permitidas para upload.
    - `DB_NAME`: Nome do arquivo do banco de dados SQLite.

5.3. `database.py`

- **Funções principais:**
    - `conectar_db()`: Estabelece conexão com o banco SQLite e configura `sqlite3.Row` para facilitar o acesso às colunas.
    - `inicializar_db()`: Cria tabelas (usuarios, produtos, pedidos, pedido_itens e logs_atividades) e insere dados de exemplo se necessário.

- `registrar_log()`: Registra atividades importantes para auditoria e monitoramento do sistema.

5.4. Decoradores e Utilitários

- **Arquivo `utils/decorators.py`:**
  - **`login_required`:** Assegura que somente usuários autenticados acessem determinadas rotas.
  - **`admin_required`:** Garante que apenas administradores autorizados acessem as rotas do painel de administração.
- **Arquivo `utils/helpers.py`:**
  - **`allowed_file(filename)`:** Valida a extensão do arquivo para upload conforme definido em `config.py`.

5.5. Rotas (Blueprints)

A pasta `routes/` organiza as diferentes funcionalidades da aplicação:

- **`main.py`:**
  - Rotas para a página inicial (`/`), listagem de produtos e detalhes de produtos.
  - Inclusão das novas páginas: contato e sobre.
- **`auth.py`:**
  - Gerencia cadastro (`/cadastro`), login (`/login`), logout (`/logout`) e recuperação de senha (`/forgot_password`).
- **`cart.py`:**
  - Funções para adicionar produtos ao carrinho, visualizar e atualizar quantidades, e remover itens.
- **`checkout.py`:**
  - Processo de finalização da compra com registro do pedido e exibição da confirmação (`/confirmacao_pedido`).

- **`user.py`**:
    - Gerencia o perfil do usuário e o histórico de pedidos.
- **`admin.py`**:
    - Rotas prefixadas com `/admin`, protegidas pelo decorador `admin_required`, para:
        - Visualização do dashboard com estatísticas do sistema.
        - Gestão completa de produtos (adição, edição, exclusão).
        - Monitoramento e alteração do status dos pedidos.
        - Administração de usuários (incluindo alternância do status de administrador).
        - Acesso aos logs de atividades.

---

6. Templates (Frontend)

A aplicação utiliza o mecanismo de templates Jinja2 para renderizar páginas HTML dinâmicas.

6.1. Template Base

- **`templates/base.html`**:
    - Estrutura padrão para as páginas do site não administrativas.
    - Inclui a importação de Bootstrap, Font Awesome e o arquivo `global.css`.
    - Define blocos personalizáveis: `title`, `head`, `content` e `scripts`.
- **`templates/base_admin.html`**:
    - Template específico para a área administrativa, com seu próprio estilo (ex.: `admin.css`).

- Estrutura com sidebar, navbar e blocos customizáveis: `title`, `head`, `admin_content` e `scripts`.

6.2. Templates Específicos

- **Público:**
  - `index.html`, `detalhes_produto.html`, `carrinho.html`, `checkout.html`, `confirmacao_pedido.html`, `login.html`, `cadastro.html`, `historico_pedidos.html`, `contato.html`, `forgot_password.html`, `sobre.html`.
- **Administração (pasta `templates/admin/`):**
  - `dashboard.html`, `produtos.html`, `novo_produto.html`, `editar_produto.html`, `pedidos.html`, `detalhes_pedido.html`, `usuarios.html`, `logs.html`.

---

7. Estilos (CSS)

Os arquivos CSS estão localizados na pasta `static/css/` e são organizados da seguinte forma:

- **`global.css`:** Define estilos globais que afetam tipografia, cores e layout geral.
- **`admin.css`:** Estiliza o painel administrativo com refinamentos de cores, espaçamentos e interações.
- **Outros arquivos específicos:** Cada página (como `index.css`, `produto.css`, `carrinho.css`, etc.) possui seu próprio arquivo de estilos para personalizações adicionais.

---

8. Banco de Dados

A aplicação utiliza um banco de dados SQLite denominado `ecommerce.db`.
O esquema é gerado e inicializado por meio do script `database.py` e inclui as seguintes tabelas:

- **usuarios:** Armazena informações dos usuários (nome, email, senha, e flag `is_admin`).
- **produtos:** Detalhes dos produtos (nome, preço, estoque, descrição, imagem).
- **pedidos:** Registro de pedidos realizados pelos usuários (ID do usuário, data, status e valor total).
- **pedido_itens:** Itens que compõem cada pedido (quantidade, preço unitário, ID do produto).
- **logs_atividades:** Armazena logs das atividades importantes para fins de auditoria e monitoramento.

---

9. Como Executar a Aplicação

9.1. Ativar o Ambiente Virtual

bash

CopiarEditar

```
source venv/bin/activate   # Linux/macOS
# ou
venv\Scripts\activate       # Windows
```

9.2. Definir a Variável de Ambiente do Flask

No terminal, defina a variável que indica o ponto de entrada da aplicação:

bash

CopiarEditar

```
export FLASK_APP=app.py   # Linux/macOS
# ou, no Windows:
# set FLASK_APP=app.py
```

9.3. Iniciar o Servidor

Para executar a aplicação, utilize:

bash

CopiarEditar

- `flask run`

A aplicação ficará acessível por padrão em http://127.0.0.1:5000. Para acessar o painel administrativo, navegue até http://127.0.0.1:5000/admin/dashboard e faça login com uma conta de administrador (caso existam dados de exemplo previamente cadastrados).

---

10. Funcionalidades Principais

10.1. Autenticação e Gestão de Usuários

- **Cadastro, Login e Logout:** Permite o registro de novos usuários, autenticação e encerramento de sessão.
- **Recuperação de Senha:** Funcionalidade para recuperação de senha com a página `forgot_password.html`.

10.2. Navegação e Visualização de Produtos

- **Página Inicial e Listagem de Produtos:** Exibe os produtos disponíveis para compra.
- **Detalhes do Produto:** Página dedicada para exibir informações detalhadas de cada produto.

10.3. Carrinho de Compras e Checkout

- **Gestão do Carrinho:** Adicionar, remover e atualizar a quantidade dos produtos.
- **Processo de Checkout:** Finaliza a compra, registra o pedido no banco de dados e redireciona para a página de confirmação.

10.4. Painel Administrativo

- **Dashboard:** Visualização de estatísticas e logs de atividades.

- **CRUD de Produtos:** Permite a criação, leitura, atualização e exclusão de produtos.
- **Gestão de Pedidos:** Exibição, detalhamento e alteração do status dos pedidos.
- **Administração de Usuários:** Gerenciamento dos usuários registrados com opção para alternar o status de administrador.
- **Logs de Atividades:** Registro de ações importantes para auditoria e monitoramento.

Esta versão aprimorada da documentação busca oferecer uma visão completa e organizada do projeto, facilitando a compreensão tanto para desenvolvedores envolvidos quanto para colaboradores que desejem entender a estrutura e as funcionalidades do sistema.

Em caso de dúvidas ou para contribuições, recomenda-se consultar o repositório oficial ou entrar em contato com a equipe responsável pelo desenvolvimento.