Corsona - Rede Social para Artistas e Fãs

Corsona é uma plataforma de rede social desenvolvida para **artistas e seus fãs**, permitindo que artistas compartilhem suas publicações e interajam com seu público, e que fãs acompanhem seus artistas favoritos, curtam e comentem em suas postagens. A plataforma também permite que os artistas gerenciem seus perfis.

Funcionalidades Principais

- **Autenticação de Usuários:** Cadastro e Login seguro para artistas e fãs.
- **Gerenciamento de Perfil (Artistas):** Edição de nome, nome de usuário, bio e avatar com ajuste de posição e zoom.
- **Alteração de Senha:** Funcionalidade segura para alteração de senha.
- **Feed de Publicações:** Visualização de posts de artistas.
- **Criação de Posts (Artistas):** Artistas podem criar posts com texto e imagem.
- **Interações (Artistas e Fãs):**
    - Curtir e descurtir posts.
    - Comentar em posts.
- **API RESTful:** Endpoints para todas as funcionalidades principais.

Tecnologias Utilizadas

- **Backend:** Python, Flask
- **Frontend:** HTML, CSS, JavaScript
- **Banco de Dados:** MySQL
- **Gerenciamento de Sessão:** Flask-Session (baseado em sistema de arquivos)
- **Hashing de Senhas:** bcrypt
- **Variáveis de Ambiente:** python-dotenv

Estrutura do Projeto (Simplificada)

PROF

```
corsona/
├── app/                    # Pacote principal da aplicação Flask
│   ├── __init__.py         # Fábrica da aplicação, registro de
blueprints
│   ├── config.py           # Configurações da aplicação
│   ├── db.py               # Lógica de conexão e inicialização do BD
│   ├── utils.py            # Decorators e funções utilitárias
│   ├── main_routes.py      # Rotas principais/globais
│   │
│   ├── auth/               # Blueprint para autenticação (login,
registro)
│   ├── feed/               # Blueprint para o feed e posts
│   ├── profile/            # Blueprint para gerenciamento de perfil
│   │
│   ├── static/             # Arquivos estáticos (CSS, JS, Imagens)
```

```
│   │   ├── css/
│   │   ├── js/
│   │   └── images/
│   │
│   └── templates/          # Templates HTML (Jinja2)
│       ├── auth/
│       └── feed/
│
├── uploads/                # Pasta para uploads de usuários (avatares,
imagens de posts)
├── venv/                   # Ambiente virtual Python
├── flask_session/          # Arquivos de sessão do Flask
├── main.py                 # Ponto de entrada da aplicação
├── requirements.txt        # Dependências Python
├── .env                    # Arquivo para variáveis de ambiente
└── README.md               # Este arquivo
```

Configuração do Ambiente

Siga os passos abaixo para configurar e rodar a aplicação localmente.

1. Pré-requisitos

- Python 3.8 ou superior
- MySQL Server instalado e rodando
- `pip` (gerenciador de pacotes Python)

2. Clonar o Repositório (se aplicável)

```
git clone https://github.com/IgNicAl/corsona.git
cd corsona
```

PROF

3. Criar e Ativar o Ambiente Virtual

É altamente recomendável usar um ambiente virtual para isolar as dependências do projeto.

```
# No Linux/macOS
python3 -m venv venv
source venv/bin/activate

# No Windows
python -m venv venv
.\venv\Scripts\activate
```

4. Instalar Dependências

Com o ambiente virtual ativado, instale as dependências listadas no `requirements.txt`:

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

5. Configurar Variáveis de Ambiente

Crie um arquivo chamado `.env` na raiz do projeto (`corsona/.env`) e adicione as seguintes variáveis, substituindo pelos seus valores:

```
DB_HOST=localhost
DB_USER=seu_usuario_mysql
DB_PASSWORD=sua_senha_mysql
DB_NAME=corsona
SECRET_KEY=gere_uma_chave_secreta_forte_e_aleatoria_aqui
FLASK_APP=main.py
FLASK_DEBUG=True
```

- **`DB_USER`** e **`DB_PASSWORD`**: Suas credenciais do MySQL.
- **`DB_NAME`**: O nome do banco de dados que será usado (ex: `corsona`). Ele será criado se não existir.
- **`SECRET_KEY`**: Uma string longa, aleatória e secreta usada pelo Flask para segurança de sessões e outras funcionalidades. Você pode gerar uma usando Python:

  ```
  python -c "import secrets; print(secrets.token_hex(32))"
  ```

- **`FLASK_DEBUG=True`**: Habilita o modo de depuração do Flask, útil para desenvolvimento. Mude para `False` em produção.

6. Inicializar o Banco de Dados

PROF

O Flask precisa criar as tabelas no banco de dados. Com o ambiente virtual ativado e as variáveis de ambiente configuradas, execute o seguinte comando na raiz do projeto:

```
flask init-db
```

Isso irá criar o banco de dados (se não existir) e as tabelas `users`, `posts`, `likes` e `comments`.

Rodando a Aplicação

Após completar a configuração, você pode iniciar o servidor de desenvolvimento do Flask:

```
python main.py
```

A aplicação estará rodando em `http://127.0.0.1:5000/` (ou no endereço IP da sua máquina na rede local, como `http://192.168.x.x:5000/`).

Endpoints da API (Principais)

- **Autenticação:**
  - `POST /auth/register`: Cadastro de novo usuário.
  - `POST /auth/login`: Login de usuário.
  - `POST /auth/logout`: Logout de usuário.
- **Usuário/Perfil:**
  - `GET /feed/api/user`: Obter dados do usuário logado.
  - `POST /profile/api/user/update`: Atualizar perfil do usuário (nome, username, bio, avatar).
  - `POST /profile/api/user/password`: Alterar senha do usuário.
- **Posts:**
  - `POST /feed/api/posts`: Criar um novo post.
  - `GET /feed/api/posts`: Listar todos os posts.
- **Interações:**
  - `POST /feed/api/posts/<post_id>/like`: Curtir/descurtir um post.
  - `POST /feed/api/posts/<post_id>/comments`: Adicionar um comentário a um post.
  - `GET /feed/api/posts/<post_id>/comments`: Listar comentários de um post.
- **Outros:**
  - `GET /uploads/<filename>`: Servir arquivos de upload (avatares, imagens de posts).
  - `GET /debug/session`: (Apenas em modo DEBUG) Visualizar dados da sessão atual.

Solução de Problemas Comuns

- **`ModuleNotFoundError`**: Certifique-se de que o ambiente virtual está ativado e que todas as dependências foram instaladas (`pip install -r requirements.txt`). Verifique se você está executando os comandos `python main.py` ou `flask ...` da pasta raiz do projeto (`corsona/`).
- **Erro de Conexão com o Banco de Dados**:
  - Verifique se o seu servidor MySQL está rodando.
  - Confirme se as credenciais (`DB_USER`, `DB_PASSWORD`, `DB_HOST`) no arquivo `.env` estão corretas.
  - Certifique-se de que o usuário MySQL tem permissão para criar bancos de dados ou que o banco de dados especificado em `DB_NAME` já existe e o usuário tem permissões sobre ele.
- **`ValueError: DB_USER and DB_PASSWORD must be set in .env file`**: Garanta que o arquivo `.env` está na raiz do projeto e que as variáveis `DB_USER` e `DB_PASSWORD` estão definidas corretamente nele.
- **Problemas com Sessão (ex: `TypeError: cannot use a string pattern on a bytes-like object`)**: Verifique se a versão da `Flask-Session` no `requirements.txt` é `>=0.6.0`. Se não, atualize e reinstale as dependências.

Este `README.md` deve cobrir os pontos essenciais. Você pode adicionar mais seções conforme necessário, como "Contribuição", "Licença", ou detalhes mais aprofundados sobre a arquitetura.

PROF