Trabalho Final de Programção IV

Visão Geral do Projeto

Seu desafio é construir um **sistema de blog completo** usando o framework web Flask. Além de todas as funcionalidades (autenticação, perfil de usuário e controle de permissões), você deve adicionar a integração com uma API externa de forma **assíncrona** e a tradução automática do conteúdo obtido.

Requisitos Funcionais

A aplicação deve atender aos seguintes requisitos:

1. Página Inicial (Home)

- Exibir uma lista de todos os posts do blog, ordenados do mais recente para o mais antigo.
- Cada post deve mostrar o título, o conteúdo parcial e o nome do autor.
- **Integração com API Externa:**
  - A página deve exibir uma "Citação do Dia"obtida da API Quotable (ou semelhante).
  - A requisição para a API Quotable deve ser **assíncrona**, para não bloquear o carregamento da página principal.
  - A citação obtida (em inglês) deve ser **traduzida para o português** usando a API de tradução (**free_translate_api**) (ou semelhante).

2. Autenticação de Usuários e Controle de Permissões

- **Registro e Validação por E-mail:**
  - Novos usuários se registram com nome de usuário, senha e e-mail.
  - A conta só será ativada após a confirmação por e-mail, via um link com token de validação.
  - **Todos os novos usuários são criados com o papel de 'usuário padrão' por padrão.**
- **Login:** Usuários com contas ativadas podem fazer login com suas credenciais.

- **Sistema de Papéis:**
  - **Usuário Padrão:** Pode criar, editar e excluir **apenas seus próprios posts**.
  - **Moderador:** Tem todas as permissões de um usuário padrão, e pode **excluir qualquer post** na plataforma.
  - **Administrador:** Tem todas as permissões de um moderador, e pode **editar o papel de qualquer usuário**.

3. Gerenciamento de Posts (CRUD)

- **Criação (Create):** Qualquer usuário logado pode criar novos posts.
- **Leitura (Read):** A página inicial lista todos os posts, e uma página de detalhes exibe o conteúdo completo.
- **Edição (Update):** Um autor pode editar seus próprios posts.
- **Exclusão (Delete):** Um autor pode excluir seus próprios posts. **Um moderador ou administrador pode excluir posts de qualquer autor.**

4. Gestão do Perfil do Usuário

- Crie uma página de perfil (`/perfil`) para visualizar e editar as informações do usuário.
- **Página de Administração de Usuários:** Crie uma página exclusiva para administradores que liste todos os usuários do sistema e permita a um administrador mudar o papel de qualquer um deles.

Tecnologias e Ferramentas

Para desenvolver este projeto, você deve usar as seguintes ferramentas:

- **Flask**, **Flask-SQLAlchemy**, **Flask-Login**, **Flask-WTF**.
- **Flask-Mail** e **itsdangerous**.
- **requests**.
- **Jinja2, HTML & CSS**.

Estrutura do Projeto

Organize seu código de maneira modular para facilitar a manutenção e o crescimento. Uma estrutura de projeto recomendada seria:

Estrutura de Pastas

```
/nome-do-projeto
|-- app.py # Arquivo principal para inicializar o Flask
|-- models.py # Define as tabelas do banco de dados
|-- forms.py # Define os formularios
|-- /templates # Armazena todos os arquivos HTML
|-- /static # Armazena arquivos CSS e JS
|-- ... (outros arquivos de configuracao)
```

Critérios de Avaliação

Seu projeto será avaliado com base em:

Entrega

- **Código-fonte:** O código completo deve ser entregue compactado ou em um repositório Git, com um arquivo `README.md` que contenha instruções claras sobre como configurar e executar o projeto.
- **Vídeo de Demonstração:** Deve ser gravado um vídeo com a captura de tela que se inicia desde a inicialização do serviço, demonstrando as seguintes funcionalidades:
  1. **Criação de um Administrador:** Mostre a criação do usuário administrador por linha de comando (`flask shell`) ou via a interface gráfica, caso tenha sido implementada.
  2. **Demonstração de Papéis:**
     - Crie dois usuários (ambos com o papel padrão).
     - Faça login com o administrador, acesse a página de administração de usuários e altere o papel de um dos novos usuários para 'moderador'.
     - Demonstre as permissões de cada papel:
       * **Usuário Padrão:** Crie um post, edite-o e mostre que ele não pode excluir o post de outro usuário.
       * **Moderador:** Faça login com a conta que teve o papel alterado. Crie um novo post e demonstre que, além de poder editá-lo e excluí-lo, ele consegue excluir o post criado pelo usuário padrão.
  3. **Outras funcionalidades:** Demonstre as demais funcionalidades esperadas pelo trabalho (registro, login, CRUD de posts, gestão de perfil e a exibição da citação traduzida).
- *Observação:* O vídeo não precisa ter narração ou áudio.

Funcionalidade e Estrutura (Pontuação)

- **Funcionalidade (40%):** A aplicação deve executar todas as funcionalidades listadas.
- **Controle de Permissões (20%):** As permissões de cada papel de usuário devem ser respeitadas.
- **Validação por E-mail (10%):** A funcionalidade de registro, envio de e-mail e ativação de conta deve funcionar corretamente.
- **Integração com API Assíncrona (10%):** A requisição e a tradução da citação devem ser executadas em segundo plano.
- **Gestão de Perfil (5%):** A página de perfil e a funcionalidade de edição devem operar conforme o esperado.
- **Estrutura do Código (5%):** O código deve ser organizado, limpo e seguir a estrutura recomendada.