Prova Prática - Desenvolvimento Web com Flask (3º Bimestre)

Objetivo:

Desenvolver uma aplicação web usando Flask, aplicando os conceitos de **middleware**, **gerenciamento de sessão**, **cookies**, **flash messages** e **tratamento de erros**. Você deve seguir a arquitetura **MVC** e usar templates **Jinja2** para renderizar as páginas.

Descrição do Sistema:

Você deverá criar uma aplicação de "**Agenda de Tarefas**" com as seguintes funcionalidades:

1. **Sessão (Tarefas):**
    a. A página principal deverá listar todas as tarefas cadastradas. Cada tarefa deve ter o título, a descrição e a data de vencimento.
        i. **Dica:** usar **JSON** para criar os objetos das tarefas.
    b. Você deverá criar um formulário para adicionar novas tarefas à agenda. As tarefas devem ser salvas na **sessão**.
    c. O usuário deve ser capaz de deslogar e assim perder todas as tarefas armazenadas na sessão.
2. **Flash Message (Adição de Tarefa):**
    a. O aluno deverá criar uma funcionalidade onde o usuário pode adicionar uma nova tarefa. O título e a descrição da tarefa serão inseridos através de um formulário.
    b. Caso a tarefa não tenha título ou descrição, o sistema deverá exibir uma ***flash message*** de **erro**, informando ao usuário o que faltou.
    c. Caso a tarefa seja adicionada corretamente, o sistema deverá exibir uma ***flash message*** de **sucesso**.
3. **Middleware (Gerenciamento de Sessão):**
    a. A aplicação deve armazenar as tarefas na sessão do usuário. As tarefas devem ser preservadas enquanto a sessão estiver ativa.
    b. Se o usuário acessar a página de agenda sem estar com a sessão ativa, ele será redirecionado para uma página com um aviso e um link para voltar à página inicial. Faça esse gerenciamento usando um **middleware**.

4. **Cookies:**

    a. Ao adicionar uma tarefa, o sistema deve salvar um cookie contendo o número total de tarefas adicionadas pelo usuário até aquele momento.

    b. Na página principal, o número total de tarefas será exibido, sendo lido do cookie.

5. **Gerenciamento de Erros:**

    a. A aplicação deve ter uma página de erro personalizada para **erros 404 (página não encontrada)**. Caso o usuário acesse uma URL inválida, ele será redirecionado para uma página de erro com uma mensagem amigável.

    b. Caso haja algum erro no servidor (por exemplo, ao tentar adicionar uma tarefa sem título ou descrição), o sistema deverá exibir uma **página de erro genérica**, com **uma mensagem informando o erro ocorrido**.

6. **Funcionalidades Extras (Opcional):**

    a. Funcionalidades extras serão consideradas para melhorar a nota final, funcionalidades como por exemplo, exclusão de tarefas, edição entre outras, questões visuais também podem ajudar como personalização com CSS, uso de Bootstrap, construção correta do HTML etc.

Requisitos Técnicos:

1. **Estrutura MVC:** Utilize a estrutura MVC (Model-View-Controller) para organizar o código. O modelo nessa prova é opcional, mas se usada deve ser responsável pela manipulação dos dados (Classes e funções JSON), a visão deve renderizar as páginas com Jinja2 e o controlador deve gerenciar as rotas usando Blueprints.

2. **Uso do Template Jinja:** Utilize o Jinja para renderizar as páginas e incluir a lógica necessária nas templates (ex: exibir o número de tarefas, mensagens de erro ou sucesso).

3. **Boas Práticas:** O código deve estar bem estruturado e comentado. Nomes de variáveis e funções devem ser claros, e o código deve seguir boas práticas de segurança e organização.

Critérios de Avaliação:

- Estruturação do código no geral do sistema no padrão MVC: 1,5 pontos
- Uso correto da sessão (criação, uso, atualização e exclusão) com as informações corretas: 2,5 pontos
- Configuração correta do middleware com todas a validações de rotas necessárias: 1,5 pontos
- Uso correto dos cookies (criação, uso, atualização e exclusão) com as informações corretas: 2 pontos
- Criação e exibição correta das flash messages: 1,5 pontos
- Configuração e funcionamento correto das páginas de erros: 1 pontos

Nota máxima: 10 pontos