Lab\_06

Pergunta 1:

Projeto\_1

As funções estão no app.py.

Um sistema de autenticação e autorização de utilizadores é implementado utilizando uma base de dados SQLite e a estrutura web do Flask. São tomadas as seguintes medidas para implementar o sistema:

- Inscrição: Quando um utilizador tenta registar-se, o ponto final de "registo" é chamado. O utilizador é chamado introduzido nome de utilizador, e-mail, e palavra-passe são extraídos dos dados do formulário e armazenados nas variáveis "username", "email", e "password", respetivamente. O código então valida as entradas, verificando se algum dos campos está vazio ou se o comprimento de algum dos campos têm menos de 3 caracteres. Se alguma destas condições for preenchida, o utilizador é redirecionado de volta para a página de registo com uma mensagem de erro.

- Se a validação de entrada for bem sucedida, a palavra-passe é acelerada utilizando o função "generate\_password\_hash" a partir do módulo "werkzeug.security".A “hashed\_password” é então armazenada na variável "hashed\_password". O código então verifica se o nome de utilizador ou e-mail introduzido já existe na base de dados, executando uma consulta SELECT. Se for encontrado um utilizador, o código devolve um erro mensagem para a página de registo.

- Se o nome de utilizador e o e-mail forem únicos, o código insere os dados do novo utilizador na tabela de "users" na base de dados SQLite.

- Login: O ponto final de "login" é chamado quando um utilizador tenta fazer o login. O código extrai o nome introduzido pelo utilizador e senha a partir dos dados do formulário. Uma consulta SELECT é executada para ir buscar os dados do utilizador à tabela "users", com base no nome de utilizador introduzido.

- Se o utilizador for encontrado na base de dados, o código compara a palavra-passe introduzida com a “hashed\_password” armazenada na base de dados, utilizando a função "check\_password\_hash" a partir do módulo "werkzeug.security". Se as palavras-passe corresponderem, o utilizador está registado, e o seu nome de utilizador é armazenado numa variável de sessão. O utilizador é então redirecionado para a página "Index".

- Logout: A função "logout" é chamada quando um utilizador tenta terminar a sessão. O código remove as chaves "logged\_in" e "username" da sessão, registando efetivamente o utilizador fora. O utilizador é então redirecionado para a página "Index".

Globalmente, o código implementa um sistema básico de autenticação e autorização de utilizadores, em que as credenciais do utilizador são armazenadas numa base de dados SQLite e a sessão do utilizador é gerida utilizando a gestão da sessão integrada do Flask.

Projeto\_2

Está com a arquitetura proposta MVC.

Pergunta\_2:

Esta é uma aplicação Flask com duas vias: "/post" e "/comment". A rota "/post" trata de pedidos de GET e POST, e recupera mensagens e comentários a partir de uma base de dados. Se o método de pedido é o POST, recebe a imagem carregada e o conteúdo do post e guarda-os numa base de dados. Depois recupera os posts atualizados e comentários da base de dados e remete-nos para a página "feed.html" com a atualização dados. A rota "/comment" trata de pedidos POST, recupera o ID do correio, ID do utilizador, e comenta o conteúdo do formulário, e insere o comentário na base de dados.

Pergunta\_3:

Esta é uma aplicação de Flask com uma rota "/search" que trata de pedidos de GET e POST. Se o método de pedido for POST, recupera a consulta de pesquisa a partir do formulário, consulta a base de dados para mensagens que correspondem à consulta de pesquisa, e recupera comentários da base de dados. Em seguida, apresenta a página "search.html" com os resultados da pesquisa, pesquisa consulta, e comentários. Se o método de pedido for GET, ele simplesmente retorna a página "search.html" com um formulário vazio e sem resultados de pesquisa.

Pergunta\_4:

(imagem)

Se o método de pedido for "POST", recupera o ficheiro de imagem do objeto do pedido formulário. Gera então um nome de ficheiro seguro para o ficheiro de imagem utilizando a função secure\_filename() da Werkzeug. A função guarda então o ficheiro de imagem numa pasta específica usando o método save(). O nome do ficheiro de imagem e a informação do utilizador são então passados para o módulo “register\_user”, composto por modelos para adicionar a informação do utilizador a uma base de dados.

Pergunta\_5:

A arquitetura MVC (Model-View-Controller) é um padrão de desenho de software que separa uma aplicação em três partes interligadas: o Modelo/Model, a Vista/View, e o Controlador/Controller. Esta arquitetura permite que vários “developers” trabalhem ao mesmo tempo na construção da aplicação, facilitando o seu desenvolvimento, a sua manutenção e os testes que têm de ser efetuados antes de ser aprovada. Este padrão de desenho permite ainda que o códigoseja reutilizado, uma vez que o mesmo Model e View pode ser utilizado por múltiplos Controllers e vice-versa.

Modelo/ Model: Esta é a camada de dados da aplicação, que representa os dados subjacentes e o seu comportamento. Encapsula os dados e fornece métodos para a sua manipulação. Por outras palavras, representa a lógica empresarial da aplicação.

Vista/ View: Esta é a camada de apresentação da aplicação, que é responsável por exibir os dados ao utilizador. Representa a interface do utilizador e fornece uma forma de o utilizador interagir com os dados.

Controlador/ Controller: Esta é a camada de controlo da aplicação, trata do input dos dados e como estes são exibidos. Atua como um intermediário entre o Model e a View, e fornece uma forma de manipular os dados e atualizá-los na base de dados.

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application

Description automatically generatedA picture containing graphical user interface

Description automatically generated