Flask :Estudo de formulário

Objetivo: Tratamento de dados de formulário e envio para API

O que realizamos:

1. Rota principal ("/") e Renderização do Template: Definimos uma rota principal ("/") que renderiza o template HTML "index.html" quando acessada.
2. Rota principal ("/") e Renderização do Template: Definimos uma rota principal ("/") que renderiza o template HTML "index.html" quando acessada.
3. Lista de Usuários: Criamos uma lista vazia chamada "usuarios" para armazenar os dados dos usuários cadastrados.
4. Rota de Cadastro ("/register") para Receber Dados do Formulário HTML: ● Definimos uma rota ("/register") que aceita métodos GET e POST. ● No método POST, verificamos se os campos obrigatórios (username e email) foram preenchidos no formulário HTML. ● Se os campos estiverem preenchidos, obtemos os valores de username e email do formulário e os enviamos para a rota da API ("/api/register") usando o método POST do módulo requests. ● Se ocorrer algum erro durante o envio dos dados para a API, tratamos as exceções e exibimos uma mensagem de erro.
5. Rota da API de Cadastro ("/api/register") para Receber e Processar Dados: ● Definimos uma rota ("/api/register") exclusiva para a API que aceita apenas o método POST. ● Na rota da API, verificamos se os dados foram recebidos corretamente e se são válidos. ● Se os dados forem válidos, adicionamos o novo usuário à lista de usuários. ● Retornamos uma resposta JSON indicando se os dados foram cadastrados com sucesso ou se houve algum erro.
6. Rota da API para Obter Usuários Cadastrados ("/api/usuarios"): ● Definimos uma rota ("/api/usuarios") exclusiva para a API que aceita apenas o método GET. ● Na rota da API, retornamos a lista de usuários cadastrados em formato JSON.
7. Flask App Execution: ● Executamos a aplicação Flask usando app.run(debug=True) para ativar o modo de depuração.

Tratamento de dados do formulário

(**request.form**)Tratamento dos dados do formulário :

Quando recebemos uma solicitação POST do formulário HTML, verificamos se os campos obrigatórios (como username e email) estão presentes no objeto request.form.

```
if request.method == 'POST':
```

Verificamos se os campos obrigatórios foram preenchidos

```
if 'username' in request.form and 'email' in request.form:
```

Se os campos estiverem presentes, obtemos os valores de username e email usando request.form['username'] e request.form['email'].

```
username = request.form['username']
email = request.form['email']
```

(**request.post**)Em seguida, enviamos esses dados para a rota da API usando requests.post. Os dados são enviados como um formulário codificado.

```
Enviar dados para a API como formulário codificado:
api_url = "http://127.0.0.1:5000/api/register"
data = {'username': username, 'email': email}
try:
    response = requests.post(api_url, data=data)  # Enviar dados como formulário codificado.
    if response.status_code == 200:
        flash('Dados cadastrados com sucesso na API.', 'success')
    else:
        flash('Erro ao cadastrar dados na API.', 'error')
except requests.exceptions.RequestException as e:
    flash(f'Erro de conexão com a API: {e}', 'error')

return redirect(url_for('register'))
else:
    flash('Por favor, preencha todos os campos obrigatórios.', 'error')
```

(/api/register) Na rota da API (/api/register), recebemos os dados usando request.form.

```
    if request.method == 'POST':
    # Obtenha os dados do formulário codificado
    data = request.form
```

Verificamos se os dados estão presentes e são válidos, e então os utilizamos para adicionar um novo usuário à lista de usuários.

```
#Verifica se os dados estão presentes e são válidos
    if data and 'username' in data and 'email' in data:
        username = data['username'] obter os dados
        email = data['email']   obter os dados

 Adiciona os dados à lista de usuários
        usuarios.append({'username': username, 'email': email})
```