Criando uma aplicação

✅ O que vamos criar:

Um sistema web com:

- Registro de usuários
- Login e logout
- Página protegida (dashboard)
- Senhas criptografadas
- Armazenamento em `usuarios.json`
- Templates organizados
- Flash messages para avisos

🛠️ PASSO A PASSO:

1. Estrutura de pastas

```
projeto/
│
├── app.py
├── usuarios.json
├── templates/
│   ├── base.html
│   ├── login.html
│   ├── register.html
│   ├── dash.html
│   └── erro.html
```

2. Criando o app.py

```
from flask import Flask, render_template, request, redirect, url_for,
session, flash
```

```
from flask_login import LoginManager, UserMixin, login_user,
logout_user, login_required, current_user
from werkzeug.security import generate_password_hash,
check_password_hash
import json
import os

app = Flask(__name__) #Cria a aplicação Flask.
app.secret_key = 'chave_super_dificil' #secret_key é uma chave de
segurança usada para criptografar sessões, cookies e mensagens flash.


# Configurando Flask-Login
login_manager = LoginManager() # Cria o gerenciador de login.
login_manager.init_app(app) #Liga o Flask-Login com o Flask.
login_manager.login_view = 'login' #Define qual é a página de login
(/login) se o usuário tentar acessar algo protegido.


def carregar_usuarios(): #carregar_usuarios(): abre e lê o arquivo
usuarios.json.
    if os.path.exists("usuarios.json"): #Verifica se o arquivo
usuarios.json existe.
        with open("usuarios.json", "r") as f: #Abre o arquivo
usuarios.json no modo leitura ("r").
            return json.load(f) #Converte o conteúdo do arquivo (que
está em texto JSON) para um dicionário Python.
    return {} #Se o arquivo ainda não existir, a função retorna um
dicionário vazio.



def salvar_usuarios(usuarios): #salvar_usuarios(): salva os dados
atualizados no usuarios.json.
    with open("usuarios.json", "w") as f: #Abre o arquivo no modo
escrita ("w").  isso grava todos os usuários juntos de novo, não só o
novo.
        json.dump(usuarios, f) #Pega o dicionário Python e transforma
em texto JSON, salvando no arquivo.

# Classe do usuário
class User(UserMixin):
    ## Cria um usuário com id, nome e senha criptografada.
```

```
    def __init__(self, id, nome, senha_hash):
        self.id = id
        self.nome = nome
        self.senha_hash = senha_hash

    @classmethod
    def get(cls, user_id):
        usuarios = carregar_usuarios() #vai pegar lá fo arquivo json
        if user_id in usuarios: #compara a chave do dicionário com o id
            dados = usuarios[user_id]
            return User(id=user_id, nome=dados['nome'],
senha_hash=dados['senha']) # se houver id igual em ambos, pega o nome e
a senha
        return None

@login_manager.user_loader #Toda vez que um usuário volta para o site
(com cookie ativo), essa função recarrega os dados dele.
def load_user(user_id):
    return User.get(user_id)

# Rota inicial
@app.route('/')
def index():
    return render_template('index.html')

# Rota de registro
@app.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST': #se o usuário digitou e enviou
        #pega os dados passados no formulário
        nome = request.form['name']
        senha = request.form['password']

        usuarios = carregar_usuarios() #lê o que já tem no JSOn

        for dados in usuarios.values(): #pegar apenas os dados
            if dados['nome'] == nome: #verificar se já tem alguém
cadastrado com esse nome
                flash('Nome de usuário já existe', category='error')
#mostra a mensagem de erro
                return redirect(url_for('register')) #volta para o
registro
```

```
        id = str(len(usuarios) + 1) #gera um id (que vai ser a chve do
dicionário)
        senha_hash = generate_password_hash(senha) #Criptografa a senha
do usuário antes de salvar. (função do flask-login)
        usuarios[id] = {'nome': nome, 'senha': senha_hash} # organiza
os dados no dicionário
        salvar_usuarios(usuarios) #salva no JSon

        user = User(id=id, nome=nome, senha_hash=senha_hash) #cria o
usuário com os dados que acabou de passar
        login_user(user) #faz o login automático

        return redirect(url_for('dashboard'))

    return render_template('register.html') #se nada foi enviado

# Rota de login
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        nome = request.form['name']
        senha = request.form['password']

        usuarios = carregar_usuarios() #lê o que tem no JSON

        for id, dados in usuarios.items(): # percorre os dicionários
            if dados['nome'] == nome and
check_password_hash(dados['senha'], senha): #se tiver o nome no
dicionário e a senha criptografada for igual
                #check_password_hash() compara uma senha "normal" com
uma criptografada.
                user = User(id=id, nome=nome,
senha_hash=dados['senha']) #se for verdadeiro, cria o usuário e faz o
login
                login_user(user)
                return redirect(url_for('dashboard'))

        flash('Dados incorretos', category='error') #se não for, mostra
mensagem de erro com flash e volta para o login
        return redirect(url_for('login'))

    return render_template('login.html')
```

```
# Rota protegida
@app.route('/dashboard')
@login_required #se o usuário tiver logado
def dashboard():
    return render_template('dash.html', nome=current_user.nome) #passa
para o html o nome de quem está logado.

# Logout
@app.route('/logout')
@login_required #se estiver logado
def logout():
    logout_user() #desfaz o login
    return redirect(url_for('index')) #volta para a inicial
```

3. HTMLs usados

base.html

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <title>{% block title %}{% endblock %}</title>
</head>
<body style="background-color: pink">
    {% block conteudo %}{% endblock %}
</body>
</html>
```

index.html

```
{% extends 'base.html' %}

{% block title %}Início{% endblock %}

{% block conteudo %}
  <h1>Bem-vinda ao sistema, {{ current_user.nome if
current_user.is_authenticated else 'Visitante' }}!</h1>
  <a href="{{ url_for('login') }}">Login</a> |
  <a href="{{ url_for('register') }}">Cadastrar</a> |
  {% if current_user.is_authenticated %}
    <a href="{{ url_for('dashboard') }}">Dashboard</a> |
    <a href="{{ url_for('logout') }}">Logout</a>
```

```
  {% endif %}
{% endblock %}
```

login.html

```
{% extends 'base.html' %}
{% block title %}Login{% endblock %}
{% block conteudo %}
<h1>Login</h1>
{% include 'erro.html' %}
<form method="post">
    <input name="name" placeholder="Nome">
    <input type="password" name="password" placeholder="Senha">
    <button>Entrar</button>
</form>
{% endblock %}
```

erro.html

```
{% with errors = get_flashed_messages(category_filter=['error']) %}
  {% if errors %}
    {% for msg in errors %}
      <p style="color: red">{{ msg }}</p>
    {% endfor %}
  {% endif %}
{% endwith %}
```

register.html

```
{% extends 'base.html' %}
{% block title %}Registrar{% endblock %}
{% block conteudo %}
<h1>Cadastro</h1>
{% include 'erro.html' %}
<form method="post">
    <input name="name" placeholder="Nome">
    <input type="password" name="password" placeholder="Senha">
    <button>Cadastrar</button>
</form>
{% endblock %}
```