Lista 12

Gerado com IA

Exercício: Refatorando a Aplicação com Flask-Login e Cadastro de Receitas

Objetivo: Substituir o sistema de gerenciamento de sessões manuais pelo **Flask-Login** para autenticação e controle de acesso. Manter o recurso de cadastro e listagem de receitas com todos os requisitos anteriores.

Passo 1: Instalando o Flask-Login

1. Certifique-se de instalar o pacote **Flask-Login**:

```
pip install flask-login
```

2. **Objetivo deste passo:** Adicionar uma solução robusta para gerenciamento de login.

Passo 2: Configurando o Flask-Login

1. **Atualize o arquivo `app.py`:**
   - Configure o **Flask-Login** e defina um *user loader*:

```
from flask_login import LoginManager, UserMixin, login_user,
logout_user, login_required, current_user

# Configuração do Flask-Login
login_manager = LoginManager(app)
login_manager.login_view = 'login'

@login_manager.user_loader
def load_user(user_id):
    conn = conectar_banco()
    cursor = conn.cursor()
    cursor.execute('SELECT id, username, nome FROM usuarios WHERE id =
?', (user_id,))
    user = cursor.fetchone()
    conn.close()
    if user:
        return User(id=user[0], username=user[1], nome=user[2])
    return None
```

2. **Objetivo deste passo:** Configurar o **Flask-Login** para gerenciar usuários autenticados.

Passo 3: Atualizando o Modelo `User`

1. **Atualize a classe `User` no arquivo `models.py`:**
   - Torne a classe compatível com o **Flask-Login** implementando o mixin `UserMixin`:

```
from flask_login import UserMixin

class User(UserMixin):
    def __init__(self, id, username, password=None, nome=None):
        self.id = id
        self.username = username
        self.password = password
        self.nome = nome

    def salvar(self):
        conn = conectar_banco()
        cursor = conn.cursor()
        try:
            cursor.execute('''
                INSERT INTO usuarios (username, password, nome)
                VALUES (?, ?, ?)
            ''', (self.username, self.password, self.nome))
            conn.commit()
        except Exception as e:
            raise ValueError(f"Erro ao salvar usuário: {e}")
        finally:
            conn.close()

    @staticmethod
    def buscar_por_username(username):
        conn = conectar_banco()
        cursor = conn.cursor()
        cursor.execute('SELECT id, username, password, nome FROM
usuarios WHERE username = ?', (username,))
        user = cursor.fetchone()
        conn.close()
        if user:
            return User(id=user[0], username=user[1], password=user[2],
nome=user[3])
        return None
```

2. **Objetivo deste passo:** Adaptar o modelo para funcionar com o **Flask-Login**.

Passo 4: Atualizando as Rotas para Login

1. **Atualize as rotas de autenticação em `app.py`:**
   - Substitua o sistema de sessões pelo **Flask-Login**:

```
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']

        user = User.buscar_por_username(username)
        if user and check_password_hash(user.password, password):
            login_user(user)
            return redirect(url_for('receitas'))
        else:
            return render_template('login.html', erro="Usuário ou senha
inválidos.")

    return render_template('login.html')

@app.route('/logout')
@login_required
def logout():
    logout_user()
    return redirect(url_for('login'))
```

2. **Objetivo deste passo:** Migrar as funcionalidades de login e logout para o **Flask-Login**.

Passo 5: Atualizando as Rotas de Receitas

1. **Proteja as rotas de receitas com o decorador `@login_required`:**

```
@app.route('/receitas', methods=['GET', 'POST'])
@login_required
def receitas():
    if request.method == 'POST':
        title = request.form['title']
        description = request.form['description']

        try:
            receita = Recipe(title=title, description=description,
user_id=current_user.id)
            receita.salvar()
            return redirect(url_for('receitas'))
        except ValueError as e:
            return render_template('receitas.html', erro=str(e),
receitas=Recipe.listar_por_usuario(current_user.id))

    receitas = Recipe.listar_por_usuario(current_user.id)
    return render_template('receitas.html', receitas=receitas)
```

2. **Objetivo deste passo:** Proteger o acesso às receitas apenas para usuários autenticados.

Passo 6: Testando a Aplicação

1. **Inicie a aplicação:**

```
python app.py
```

2. **Siga os passos:**
   - Acesse `/cadastro` para criar um usuário.
   - Faça login em `/login`.
   - Acesse `/receitas` para cadastrar e listar receitas.

3. **Verifique:**
   - O login usa o **Flask-Login** para autenticação.
   - As receitas são associadas ao usuário logado.

Conclusão do Exercício

Neste exercício, você:

1. Substituiu o sistema de sessões manual pelo **Flask-Login**.
2. Adaptou o modelo `User` para trabalhar com o **Flask-Login**.
3. Manteve as funcionalidades de cadastro e listagem de receitas protegidas por autenticação.

Desafio Extra:

1. Adicione ao modelo `Recipe` um método para editar e excluir receitas.
2. Implemente uma funcionalidade de *logout automático* após inatividade por tempo determinado.