# Lista 11

*Gerado com IA*

**Exercício: Implementando o Cadastro de Receitas com Banco de Dados e Classes**

**Objetivo:** Adicionar um novo recurso à aplicação para cadastrar e listar receitas. Para isso, você criará um modelo `Recipe`, configurará as rotas e templates necessários, e integrará as receitas ao banco de dados SQLite.

---

## Passo 1: Configurando o Banco de Dados para Receitas

1. **Atualize o arquivo `database.py`:**
   - Adicione a tabela `recipes` para armazenar as receitas:

```
def criar_tabela():
    conn = conectar_banco()
    cursor = conn.cursor()
    # Tabela de usuários
    cursor.execute('''
        CREATE TABLE IF NOT EXISTS usuarios (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            username TEXT UNIQUE NOT NULL,
            password TEXT NOT NULL,
            nome TEXT NOT NULL
        )
    ''')
    # Tabela de receitas
    cursor.execute('''
        CREATE TABLE IF NOT EXISTS recipes (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            title TEXT NOT NULL,
            description TEXT NOT NULL,
            user_id INTEGER NOT NULL,
            FOREIGN KEY(user_id) REFERENCES usuarios(id)
        )
    ''')
    conn.commit()
    conn.close()

# Executa a criação das tabelas ao importar o módulo
criar_tabela()
```

2. **Objetivo deste passo:** Configurar o banco de dados para suportar o cadastro de receitas.

---

## Passo 2: Criando o Modelo `Recipe`

1. **Atualize o arquivo `models.py`:**
    - Adicione a classe `Recipe` para gerenciar receitas:

```
class Recipe:
    def __init__(self, title, description, user_id):
        self.title = title
        self.description = description
        self.user_id = user_id

    def salvar(self):
        conn = conectar_banco()
        cursor = conn.cursor()
        try:
            cursor.execute('''
                INSERT INTO recipes (title, description, user_id)
                VALUES (?, ?, ?)
            ''', (self.title, self.description, self.user_id))
            conn.commit()
        except Exception as e:
            raise ValueError(f"Erro ao salvar receita: {e}")
        finally:
            conn.close()

    @staticmethod
    def listar_por_usuario(user_id):
        conn = conectar_banco()
        cursor = conn.cursor()
        cursor.execute('SELECT title, description FROM recipes WHERE
user_id = ?', (user_id,))
        receitas = cursor.fetchall()
        conn.close()
        return receitas
```

2. **Objetivo deste passo:** Criar um modelo para gerenciar as receitas.

---

## Passo 3: Adicionando Rotas para Receitas

1. **Atualize o arquivo `app.py`:**
    - Adicione as rotas para cadastrar e listar receitas:

```
@app.route('/receitas', methods=['GET', 'POST'])
def receitas():
    if 'username' not in session:
        return redirect(url_for('login'))

    # Obtém o usuário logado
    conn = conectar_banco()
    cursor = conn.cursor()
```

```
    cursor.execute('SELECT id FROM usuarios WHERE username = ?',
(session['username'],))
    user = cursor.fetchone()
    conn.close()

    if not user:
        return redirect(url_for('login'))

    user_id = user[0]

    if request.method == 'POST':
        title = request.form['title']
        description = request.form['description']

        try:
            receita = Recipe(title=title, description=description,
user_id=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(user_id))

    # Lista receitas do usuário logado
    receitas = Recipe.listar_por_usuario(user_id)
    return render_template('receitas.html', receitas=receitas)
```

2. **Objetivo deste passo:** Adicionar funcionalidades para gerenciar receitas.

---

## Passo 4: Criando os Templates

1. **Crie o template `receitas.html`:**
    - Adicione o seguinte conteúdo:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Receitas</title>
</head>
<body>
    <h1>Minhas Receitas</h1>

    {% if erro %}
    <p style="color: red;">{{ erro }}</p>
    {% endif %}

    <form method="POST">
```

```
        <label for="title">Título:</label>
        <input type="text" id="title" name="title" required>
        <br>
        <label for="description">Descrição:</label>
        <textarea id="description" name="description" required>
</textarea>
        <br>
        <button type="submit">Cadastrar Receita</button>
    </form>

    <h2>Receitas Cadastradas</h2>
    <ul>
        {% for receita in receitas %}
        <li><strong>{{ receita[0] }}</strong>: {{ receita[1] }}</li>
        {% endfor %}
    </ul>
</body>
</html>
```

2. **Objetivo deste passo:** Criar a interface para gerenciar receitas.

---

## Passo 5: Testando a Aplicação

1. **Inicie a aplicação:**
   - Execute:

```
python app.py
```

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

3. **Verifique:**
   - Apenas usuários logados podem acessar `/receitas`.
   - As receitas cadastradas pertencem ao usuário logado.

---

**Conclusão do Exercício**

Neste exercício, você:

1. Configurou o banco de dados para suportar receitas.
2. Criou o modelo `Recipe` para gerenciar receitas.
3. Implementou rotas e templates para cadastro e listagem de receitas.

**Desafio Extra:**

1. Adicione uma funcionalidade para editar ou excluir receitas.
2. Implemente uma busca por receitas com base no título.