# Lista 10

*Gerado com IA*

**Exercício: Modularizando a Aplicação com Banco de Dados e Classes**

**Objetivo:** Refatorar a aplicação para organizar melhor o código, usando arquivos dedicados para o banco de dados (`database.py`) e os modelos (`models.py`). O modelo `User` será responsável por salvar usuários no banco e retornar uma lista de todos os usuários.

---

## Passo 1: Configurando o Banco de Dados no Arquivo `database.py`

1. **Crie o arquivo `database.py`:**
    - Este arquivo será responsável por configurar e gerenciar a conexão com o banco de dados.
    - Adicione o seguinte conteúdo:

```
import sqlite3

DATABASE = 'usuarios.db'

def conectar_banco():
    return sqlite3.connect(DATABASE)

def criar_tabela():
    conn = conectar_banco()
    cursor = conn.cursor()
    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
        )
    ''')
    conn.commit()
    conn.close()

# Executa a criação da tabela ao importar o módulo
criar_tabela()
```

2. **Objetivo deste passo:** Centralizar a configuração do banco de dados em um único arquivo.

---

## Passo 2: Criando o Modelo no Arquivo `models.py`

1. **Crie o arquivo `models.py`:**

- Este arquivo conterá a classe `User` para gerenciar os dados dos usuários.
- Adicione o seguinte conteúdo:

```
from database import conectar_banco

class User:
    def __init__(self, username, password, nome):
        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 listar_todos():
        conn = conectar_banco()
        cursor = conn.cursor()
        cursor.execute('SELECT nome, username FROM usuarios')
        usuarios = cursor.fetchall()
        conn.close()
        return usuarios
```

2. **Objetivo deste passo:** Definir um modelo para encapsular a lógica de manipulação de dados.

---

## Passo 3: Atualizando o Arquivo `app.py`

1. **Atualize o arquivo principal para usar os novos módulos:**

- No início de `app.py`, importe os novos módulos:

```
from models import User
from werkzeug.security import generate_password_hash,
check_password_hash
from flask import Flask, render_template, request, redirect, url_for,
session, make_response
```

2. **Atualize a rota `/cadastro`:**

   - Substitua a lógica de salvar o usuário para utilizar o método `salvar` da classe `User`:

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

        # Criptografa a senha
        senha_criptografada = generate_password_hash(password)

        try:
            usuario = User(username=username,
password=senha_criptografada, nome=nome)
            usuario.salvar()
            return redirect(url_for('login'))
        except ValueError as e:
            return render_template('cadastro.html', erro=str(e))

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

3. **Atualize a rota `/usuarios`:**

   - Substitua a lógica para listar usuários usando o método `listar_todos` da classe `User`:

```
@app.route('/usuarios')
def listar_usuarios():
    if 'username' not in session:
        return redirect(url_for('login'))

    usuarios = User.listar_todos()
    return render_template('usuarios.html', usuarios=usuarios)
```

4. **Objetivo deste passo:** Garantir que o código principal utilize os métodos definidos no modelo.

---

## Passo 4: Testando a Aplicação

1. **Inicie a aplicação:**

   - Execute o servidor:

```
python app.py
```

2. **Verifique os seguintes pontos:**
    - O cadastro de usuários deve utilizar o método `salvar` do modelo `User`.
    - O login deve validar as credenciais com a senha criptografada.
    - A listagem de usuários em `/usuarios` deve buscar os dados diretamente pelo método `listar_todos` do modelo `User`.
3. **Teste de proteção:**
    - Confirme que a listagem de usuários só é acessível por usuários logados.

---

**Conclusão do Exercício**

Neste exercício, você:

1. Refatorou a aplicação para separar as responsabilidades entre os arquivos `database.py` e `models.py`.
2. Implementou um modelo de usuário com métodos para salvar e listar dados do banco.
3. Manteve as funcionalidades de autenticação e listagem protegidas por login.

**Desafio Extra:**

1. Adicione ao modelo `User` um método para buscar um usuário pelo `username`.
2. Implemente uma funcionalidade para editar os dados do usuário logado.