**UNIVERSIDADE ESTÁCIO DE SÁ**

**Analise e desenvolvimento de sistemas/ ciência da computação**

**UNIDADE: Nova América/Del Castilho**

**Rio de janeira - RJ**

**Maio / 2024**  
  
  
 **Yuri de Oliveira Cabral** - 202203352113   
**Rickson Dellatorre** – 202109061224  
 **Rennan Olímpio** – 202202852881

**Trabalho:** Cursos online

**Disciplina:** Desenvolvimento rápido de aplicações em Python

Trabalho de ...... apresentado a Universidade Estácio de Sá, como exigência para avaliação na disciplina ........................

Orientador:

Prof. Ronaldo Candido dos Santos

SUMÁRIO

[1](#_30j0zll) INTRODUÇÃO 3

[1.1](#_1fob9te) DESCRIÇÃO DO PROBLEMA 3

[1.2](#_3znysh7) OBJETIVOS 3

[2](#_tyjcwt) DESENVOLVIMENTO 4

[2.1](#_3dy6vkm) XXXXXXXXXXXXX 4

[3](#_1t3h5sf) CONCLUSÃO 5

REFERÊNCIAS: https://getbootstrap.com/docs/5.3/getting-started/introduction/ https://www.w3schools.com/sql/ https://www.w3schools.com/html/ 6

# INTRODUÇÃO

(É a apresentação do trabalho, onde são informados os motivos que levaram à pesquisa e os objetivos do trabalho. Fazer uma introdução destacando a motivação/justificativa para a escolha do tema.)

## DESCRIÇÃO DO PROBLEMA

(Descrever as situações-problema que levaram a realização do trabalho.)

## OBJETIVOS

(Definir os resultados a obter na realização do trabalho.)

# DESENVOLVIMENTO

### (É a parte principal do texto, dividido em seções ou subseções. Contém a descrição pormenorizada do assunto e a fundamentação teórica, podendo conter a metodologia (material e método), os resultados e respectivas discussões (quando previstas atividades experimentais/numéricas no Plano de Trabalho). Devem ser feitas as citações e as notas bibliográficas e/ou explicativas, no texto. Discorrer sobre o tema proposto, fundamentando-se nos textos obtidos de livros e artigos encontrados na literatura, discutindo os principais dados e/ou resultados obtidos, destacando pontos que não estão consolidados na ATUALIDADE).

### 1. Configuração Inicial

#### 1.1

A estrutura do

básico

my\_project/

├── app.py # Código principal da aplicação Flask

├── login.db # Banco de dados SQLite

├── templates/ # Diretório para os templates HTML

│ └── login.html # Template HTML para a tela de login

├── requirements.txt # Lista de dependências do projeto

└── README.md # Documentação do projeto

#### 1

O projeto dependerequirements.txtpara facilitar a instalação

texto simples

Flask

werkzeug

Para instalar conforme depende

eh

pip install -r requirements.txt

### 2. Configuração do Banco de Dados

#### 2.1 Criação da

Utilizamos SQLite como usuários para armazenar as informações de login. O script SQL

SQL

CREATE TABLE usuarios (

id INTEGER PRIMARY KEY AUTOINCREMENT,

username TEXT NOT NULL UNIQUE,

password TEXT NOT NULL

);

Este script deve ser executadologin.db.

### 3. Desenvolvimento da Aplicação

#### 3.

No arquivo app.py, configuramos a aplicação Flask e

from flask import Flask, render\_template, request, redirect, url\_for, flash

import sqlite3

from werkzeug.security import generate\_password\_hash, check\_password\_hash

app = Flask(\_\_name\_\_)

app.secret\_key = 'uma\_chave\_secreta\_para\_segurança'

#### 3.2 Conex

Criamos uma funçãoget\_db\_connectionpara estabelecer a conexão

def get\_db\_connection():

conn = sqlite3.connect('login.db')

conn.row\_factory = sqlite3.Row # Permite acesso aos resultados por nome da coluna

return conn

#### 3.3 Rotas de Aplicação

##### 3.3.1/)

Um redireccionamento principal da rota

@app.route('/')

def home():

return redirect(url\_for('login'))

##### 3.3.2 Rota de Login (/login)

A rota /loginpermite que os usuários façam login. Se o método de requisição for POST,

@app.route('/login', methods=['GET', 'POST'])

def login():

if request.method == 'POST':

username = request.form['username']

password = request.form['password']

conn = get\_db\_connection()

cursor = conn.cursor()

cursor.execute("SELECT \* FROM usuarios WHERE username = ?", (username,))

user = cursor.fetchone()

conn.close()

if user and check\_password\_hash(user['password'], password):

return f"Bem-vindo, {username}!"

else:

flash("Nome de usuário ou senha incorretos.", "danger")

return render\_template('login.html')

**Explicação do Processo de Login:**

1. **Verificação do Método HTTP**: Se o método for POST, significa que o formulário de login
2. **Recebimento dos Dados** : O nome
3. **Conexão com o Banco de Dados**: Abra uma conexão
4. **Consulta ao Banco**: Verifique se o nome do usuário existe na tabela usuarios.
5. **Validação da Senha** :check\_password\_hashpara verificar se a senha forne
6. **Redirecionamento ou Mensagem de Erro** : Faça login para bem

##### 3.3/add\_user)

Esta rota é usada para adicionar um usuário ao banco de

@app.route('/add\_user')

def add\_user():

conn = get\_db\_connection()

cursor = conn.cursor()

username = 'admin

## Descrição prévia do Processo de Criação do Banco de Dados

### Introdução

A criação e a configuração de um banco de dados são passos cruciais em qualquer aplicação que exija persistência de dados. No contexto de um sistema de login, a segurança e a integridade dos dados dos usuários são fundamentais. Este documento detalha o processo de criação de um banco de dados SQLite, a definição da tabela de usuários e a implementação da prática desse processo em Python utilizando a biblioteca sqlite3.

### Fundação Teórica

SQLite é uma biblioteca C que fornece um banco de dados SQL de nível, alto desempenho e autocontido. É um dos sistemas de gerenciamento de banco de dados mais amplamente implantados no mundo. A simplicidade do SQLite o torna ideal para protótipos, testes ou pequenas aplicações que não são comuns em um servidor de banco de dados completo.

### Metodologia

A metodologia para a criação do banco de dados envolve os seguintes passos:

1. **Conexão com o Banco de Dados** : Estabelecer uma conexão com o banco de dados SQLite. Se o banco de dados não existir, ele será criado.
2. **Criação da Tabela de Usuários** : Definir e criar a tabela usuarioscom colunas destacadas para armazenar os dados de login.
3. **Execução do Script SQL** : Utilizar comandos SQL para criar a tabela, garantindo que ela seja criada se ainda não existir.
4. **Fechamento da Conexão** : Fecha a conexão com o banco de dados após a execução dos comandos para liberar recursos.

### Implementação

A seguir está o código Python para criar o banco de dados e a tabela de usuários. Este código é comentado para fornecer uma compreensão detalhada de cada passo.

#### Código para Criação do Banco de Dados

import sqlite3

def create\_database():

# Estabelece conexão com o banco de dados 'login.db'

conn = sqlite3.connect('login.db')

cursor = conn.cursor() # Cria um cursor para executar comandos SQL

# Comando SQL para criar a tabela 'usuarios' se ela não existir

cursor.execute('''

CREATE TABLE IF NOT EXISTS usuarios (

id INTEGER PRIMARY KEY AUTOINCREMENT, # Coluna 'id' como chave primária autoincrementada

username TEXT NOT NULL UNIQUE, # Coluna 'username' como texto, não nulo e único

password TEXT NOT NULL # Coluna 'password' como texto, não nulo

)

''')

# Confirma as alterações feitas no banco de dados

conn.commit()

# Fecha a conexão com o banco de dados

conn.close()

# Chama a função para criar o banco de dados

create\_database()

### Descrição privilegiada dos Passos

#### 1. Importação da Bibliotecasqlite3

import sqlite3

A biblioteca sqlite3é importada para permitir a interação com o banco de dados SQLite.

#### 2. Definição da Funçãocreate\_database

def create\_database():

A função create\_databaseé definida para encapsular todo o processo de criação do banco de dados e da tabela.

#### 3. Conexão com o Banco de Dados

conn = sqlite3.connect('login.db')

cursor = conn.cursor()

A função sqlite3.connect('login.db')cria uma conexão com o banco de dados chamado login.db. Se o banco de dados não existir, ele será criado. A variável connmantém a conexão aberta. O método cursor()de conncriar um cursor, que é usado para executar comandos SQL.

#### 4. Criação da Tabela de Usuários

cursor.execute('''

CREATE TABLE IF NOT EXISTS usuarios (

id INTEGER PRIMARY KEY AUTOINCREMENT,

username TEXT NOT NULL UNIQUE,

password TEXT NOT NULL

)

''')

O comando CREATE TABLE IF NOT EXISTSé utilizado para criar a tabela usuariosse ela ainda não existir. A tabela possui três colunas:

* id: um inteiro que serve como chave primária e é autoincrementado.
* username: um texto que não pode ser nulo e deve ser único.
* password: um texto que também não pode ser nulo.

#### 5. Confirmação das Alterações

Pitão

Copiar

conn.commit()

O método commit()é connchamado para salvar (ou "comitar") as alterações feitas no banco de dados.

#### 6. Fechamento da Conexão

conn.close()

Finalmente, o método close()é connchamado para fechar a conexão com o banco de dados, liberando todos os recursos que estavam sendo utilizados.

### Introdução

A criação de um site

### Fundação

A construção de um site.  
Essa parte do código é referente ao menu inicial onde o cliente ira navegar nas opções do nosso site.

#### Código HTML Completo

<!DOCTYPE html>

<html lang="pt-br">

<head>

<meta charset="UTF-8">

<title>Cursos Online</title>

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/5.1.3/css/bootstrap.min.css">

<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.10.3/font/bootstrap-icons.css">

</head>

<body>

<nav class="navbar navbar-expand-lg navbar-dark bg-dark">

<div class="container-fluid">

<a class="navbar-brand" href="#">Cursos Online</a>

<button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav"

aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarNav">

<ul class="navbar-nav">

<li class="nav-item">

<a class="nav-link active" aria-current="page" href="index.html">Home</a>

</li>

<li class="nav-item">

<a class="nav-link" href="index4.html">Cursos</a>

</li>

<li class="nav-item">

<a class="nav-link" href="index3.html">Opinião de nossos Clientes</a>

</li>

<li class="nav-item">

<a class="nav-link" href="index5.html">Login</a>

</li>

</ul>

</div>

</div>

</nav>

### Implementação

#### Estruturação do HTML

Essas foram as bibliotecas utilizadas para a criação desse site

<!DOCTYPE html>

<html lang="pt-br">

<head>

<meta charset="UTF-8">

<title>Cursos Online</title>

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/5.1.3/css/bootstrap.min.css">

</head>

<body>

## 

## Utilização para a opinião pelo os usuários

Esse código foi criado com intuito de obter informações a respeito da nossa plataforma e o que podemos melhorar

<!DOCTYPE html>

<html lang="pt-br">

<head>

<meta charset="UTF-8">

<title>Cursos Online</title>

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/5.1.3/css/bootstrap.min.css">

<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.10.3/font/bootstrap-icons.css">

</head>

<style>

.gold {

color: #ffc107;

}

</style>

<body>

<nav class="navbar navbar-expand-lg navbar-dark bg-dark">

<div class="container-fluid">

<a class="navbar-brand" href="#">Cursos Online</a>

<button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav"

aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarNav">

<ul class="navbar-nav">

<li class="nav-item">

<a class="nav-link" href="index.html">Home</a>

</li>

<li class="nav-item">

<a class="nav-link" href="index4.html">Cursos</a>

</li>

## Criação do Css

Esta parte foi responsável pela estilização de nosso site, siga o código abaixo:

<!DOCTYPE html>

<html lang="pt-br">

<head>

<meta charset="UTF-8">

<title>Cursos Online</title>

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/5.1.3/css/bootstrap.min.css">

<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.10.3/font/bootstrap-icons.css">

<style>

html, body {

height: 100%;

margin: 0;

display: flex;

flex-direction: column;

}

body {

background-color: #f8f9fa;

}

.container {

flex: 1;

display: flex;

justify-content: center;

align-items: center;

}

.card {

margin-top: 50px;

}

.footer {

width: 100%;

position: relative;

bottom: 0;

}

</style>

</head>

<body>

<nav class="navbar navbar-expand-lg navbar-dark bg-dark">

<div class="container-fluid">

<a class="navbar-brand" href="#">Cursos Online</a>

# CONCLUSÃO

O nosso trabalho foi desenvolvido com intuito de melhorar a administração e fornecimento de melhores conteúdo para os alunos que possuem acesso nessa plataforma

# REFERÊNCIAS

(Sigam os conteúdos qu utilizamos para a criação de nosso site:)

ASCENCIO, A. F. G.; ARAÚJO, G. S. de. Estrutura de Dados: Algoritmos, Análise da Complexidade e implementações em Java e C/C++. São Paulo: Pearson Prentice Hall, 2010. Capítulo 2: Algoritmos de ordenação e busca. Páginas 21-102.

BACKES, A. Vídeo [ED] Aula 52 - Ordenação - QuickSort. Disponível em: <https://www.youtube.com/watch?v=RZbg5oT5Fgw>. Acesso em: 22 mai. 2024.

BLOODSHED DEV C++: Download do software. Disponível em: <https://sourceforge.net/projects/orwelldevcpp/>. Acesso em: 23 mai. 2024.

CORMEN, T. Desmistificando algoritmos. Rio de Janeiro: Elsevier, 2014. Capítulo 3: Algoritmos para ordenar e buscar. Páginas 20-49.

KOFFMAN, E. B.; WOLFGANG, P. A. T. Abstração, Estruturas de Dados e Projeto Usando C++. Rio de Janeiro: LTC, 2008. Capítulo 10: Ordenação.

NORMAS ABNT. Normas para elaboração de trabalhos acadêmicos. Disponível em: <https://www.normasabnt.org/>. Acesso em: 26 mai. 2024.

REGRAS DE UTILIZAÇÃO DO BOOTSTRAP: <https://getbootstrap.com/docs/5.3/getting-started/introduction/> Acesso em: 22 mai. 2024

MANUAL DE UTILIZAÇÃO DO SQL: <https://www.w3schools.com/sql/> Acesso em: 20 mai. 2024

MANUAL DE UTILIZAÇÃO HTML 5: <https://www.w3schools.com/html/> Acesso em: 21 mai. 2024