### Curso:

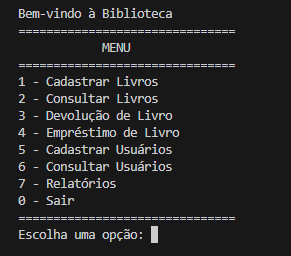
### Disciplina:

Aluno(a):

**Sistema de Biblioteca**

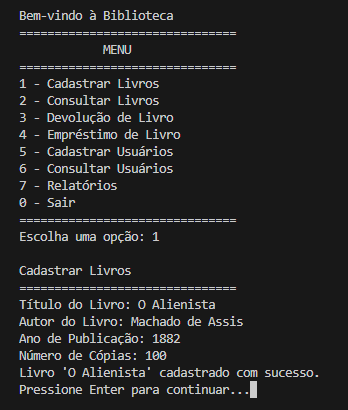
Introdução

O Sistema de Biblioteca é uma aplicação desenvolvida em Python que permite o gerenciamento de livros e usuários em uma biblioteca. Com este sistema, é possível cadastrar novos livros e usuários, consultar informações sobre livros e usuários cadastrados, realizar empréstimos e devoluções de livros, além de gerar relatórios sobre os livros, usuários e empréstimos realizados.



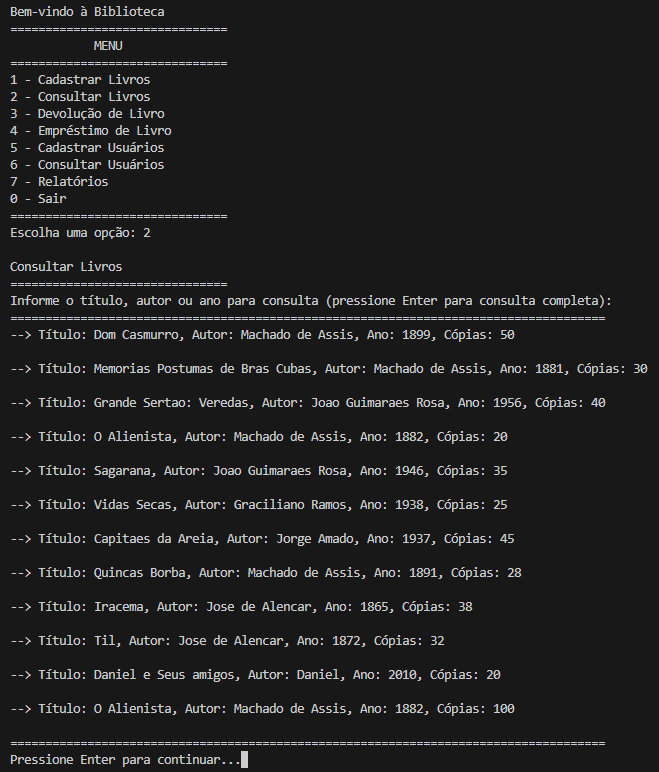
**O Sistema de Biblioteca oferece as seguintes funcionalidades principais:**

1. Cadastro de Livros: Permite cadastrar novos livros na biblioteca, informando título, autor, ano de publicação e número de cópias disponíveis.

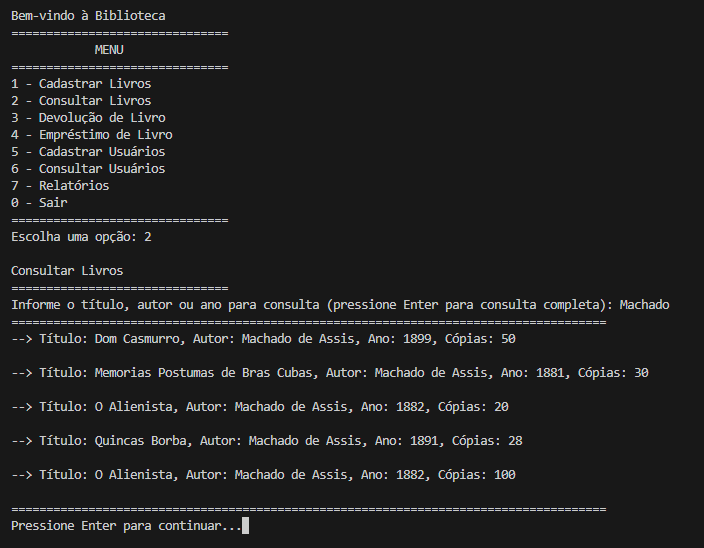


1. Consulta de Livros: Permite consultar informações sobre os livros cadastrados na biblioteca, incluindo título, autor, ano de publicação e número de cópias disponíveis. A consulta pode ser realizada por título, autor ou ano de publicação.

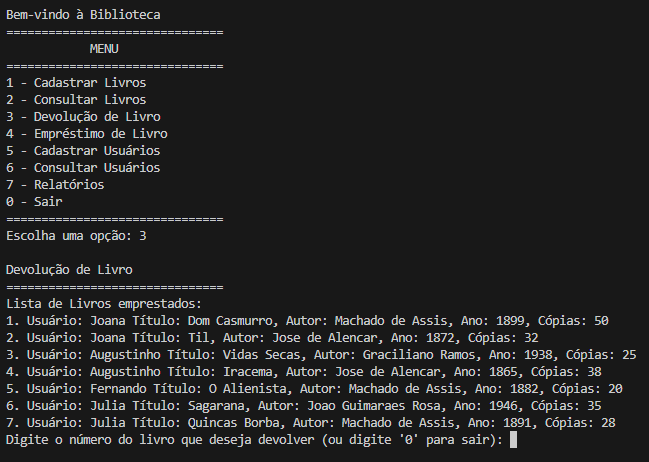
* Se nada for informado na pesquisa irá listar todos os livros cadastrados



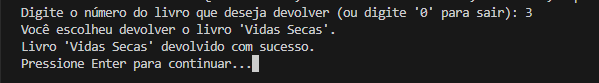
* Se algo for digitado na pesquisa irá filtrar os livros correspondentes



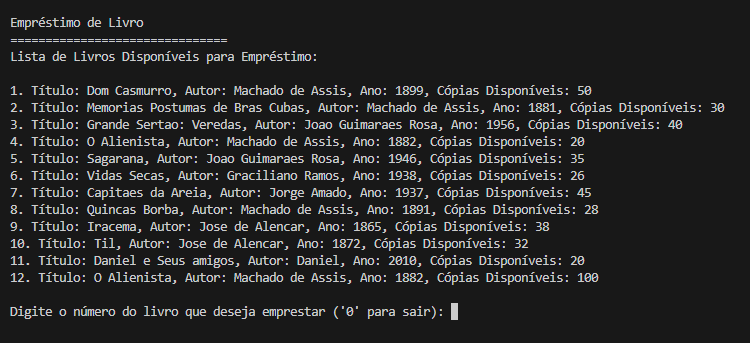
1. Devolução de Livros: Permite devolver um livro que foi emprestado por um usuário. Ao devolver o livro, a quantidade de cópias disponíveis é atualizada e o livro é removido da lista de livros emprestados pelo usuário.

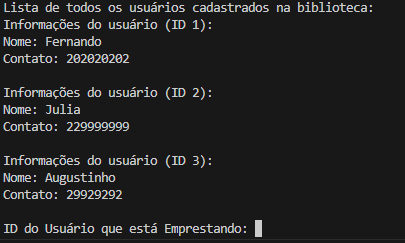


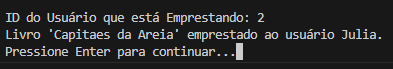
* Ao informar o livro que será devolvido é exibido a mensagem



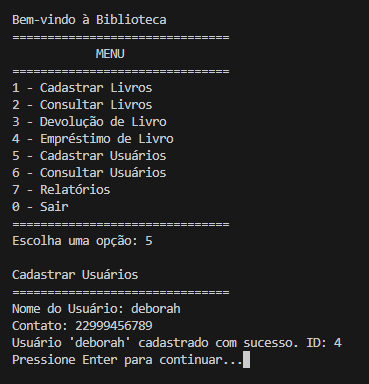
1. Empréstimo de Livros: Permite realizar o empréstimo de um livro para um usuário cadastrado na biblioteca. Ao realizar um empréstimo, a quantidade de cópias disponíveis do livro é atualizada e o livro é associado ao usuário que o emprestou. Aqui será emprestado o livro 7 ‘Capitães da Areia à Julia



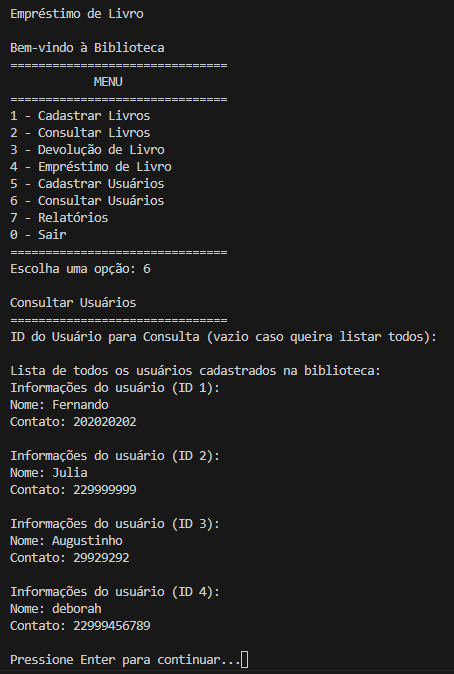
* Ao selecionar o livro irá solicitar o usuário



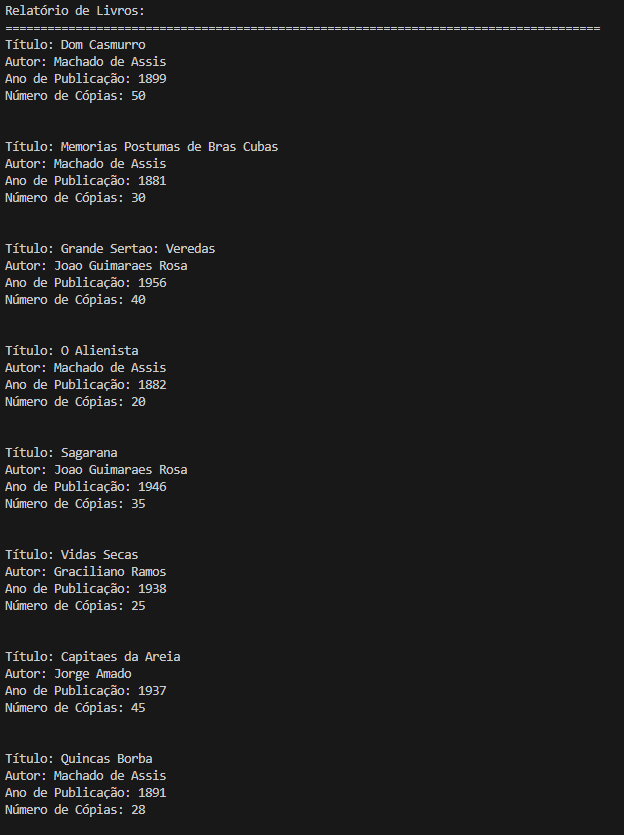
1. Cadastro de Usuários: Permite cadastrar novos usuários da biblioteca, informando nome e contato.

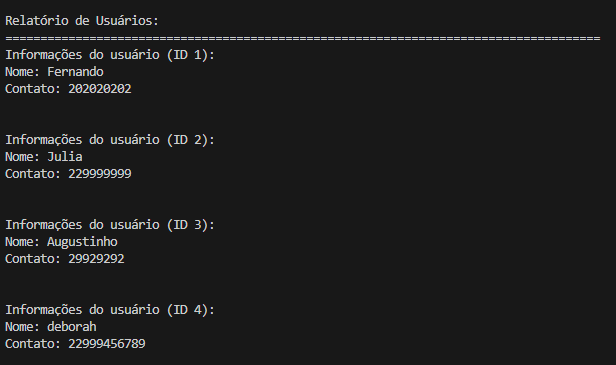


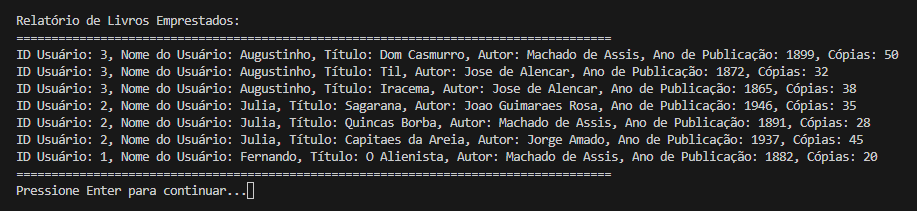
1. Consulta de Usuários: Permite consultar informações sobre os usuários cadastrados na biblioteca, incluindo nome e contato. A consulta pode ser feita informando o ID do usuário ou listando todos os usuários cadastrados.



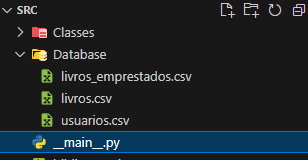
1. Relatórios: Permite gerar relatórios com informações detalhadas sobre os livros cadastrados, os usuários cadastrados e os livros emprestados na biblioteca.







* Todos os dados cadastrados são salvos em arquivos CSV dentro de uma pasta **Database** para que possa ser carregadas as informações mesmo após fechar a aplicação.



**Utilização do Sistema**

Para utilizar o Sistema de Biblioteca, siga as instruções abaixo:

**Menu Principal:** Ao iniciar o sistema, será exibido um menu principal com as opções disponíveis. O usuário pode escolher a opção desejada digitando o número correspondente e pressionando Enter.

**Cadastro de Livros:** Para cadastrar um novo livro, selecione a opção "Cadastrar Livros" no menu e siga as instruções fornecidas. Informe o título, autor, ano de publicação e número de cópias disponíveis do livro.

**Cadastro de Usuários:** Para cadastrar um novo usuário, selecione a opção "Cadastrar Usuários" no menu e siga as instruções fornecidas. Informe o nome e o contato do usuário.

**Consulta de Livros:** Para consultar informações sobre os livros cadastrados na biblioteca, selecione a opção "Consultar Livros" no menu. Você pode informar o título, autor ou ano de publicação para realizar a consulta.

**Consulta de Usuários**: Para consultar informações sobre os usuários cadastrados na biblioteca, selecione a opção "Consultar Usuários" no menu. Você pode informar o ID do usuário ou listar todos os usuários cadastrados.

**Empréstimo de Livros:** Para realizar o empréstimo de um livro, selecione a opção "Empréstimo de Livro" no menu. Escolha o livro desejado e informe o ID do usuário que está realizando o empréstimo.

**Devolução de Livros:** Para devolver um livro que foi emprestado por um usuário, selecione a opção "Devolução de Livro" no menu. Escolha o livro que deseja devolver e siga as instruções fornecidas.

**Relatórios:** Para gerar relatórios com informações detalhadas sobre os livros, usuários e empréstimos realizados na biblioteca, selecione a opção "Relatórios" no menu.

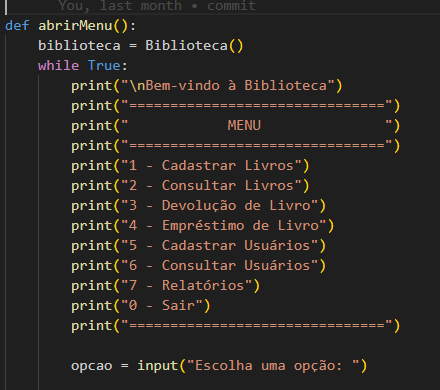
Sair do Sistema: Para sair do sistema, selecione a opção "Sair" no menu principal.

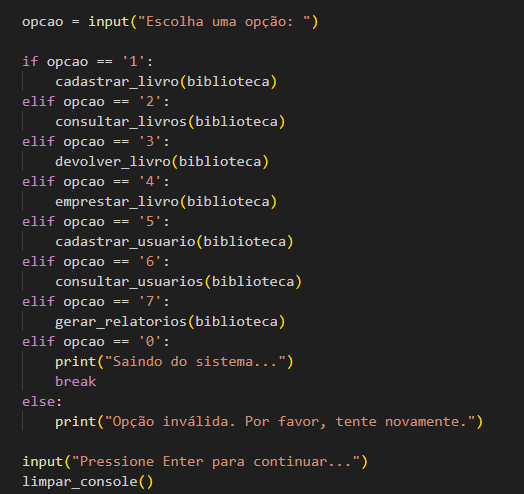
**Como tudo funciona por dentro da aplicação?**

**Função Abrir Menu**

Esta função apresenta o menu principal da biblioteca e gerencia as interações do usuário com base nas opções selecionadas:

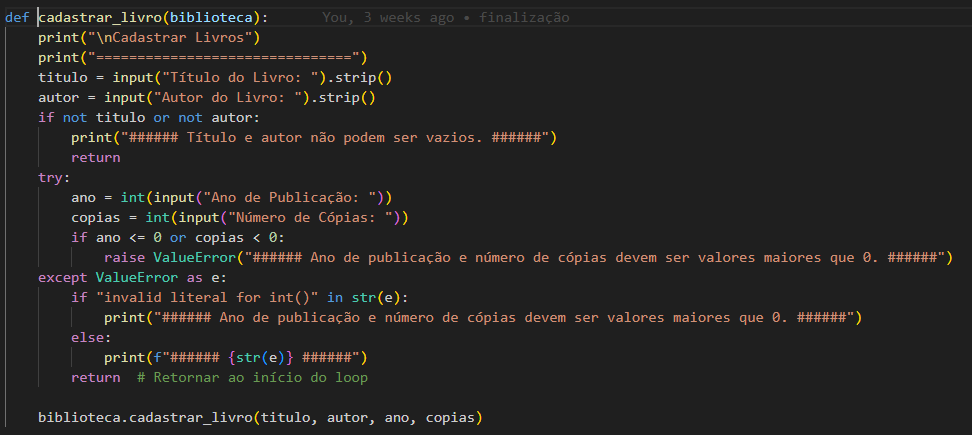
* Cadastrar Livros (1): Chama a função cadastrar\_livro(biblioteca).
* Consultar Livros (2): Chama a função consultar\_livros(biblioteca).
* Devolução de Livro (3): Chama a função devolver\_livro(biblioteca).
* Empréstimo de Livro (4): Chama a função emprestar\_livro(biblioteca).
* Cadastrar Usuários (5): Chama a função cadastrar\_usuario(biblioteca).
* Consultar Usuários (6): Chama a função consultar\_usuarios(biblioteca).
* Relatórios (7): Chama a função gerar\_relatorios(biblioteca).
* Sair (0): Sai do sistema.



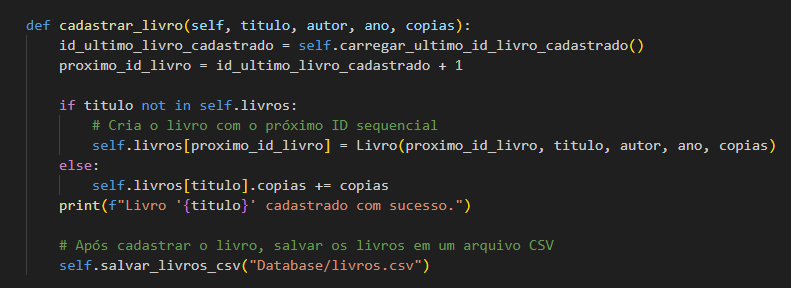


**Função cadastrar\_livro(biblioteca)**

Esta função coleta dados do usuário para cadastrar um novo livro na biblioteca. Ela valida se o título e o autor não estão vazios e se o ano de publicação e o número de cópias são válidos (valores maiores que 0).

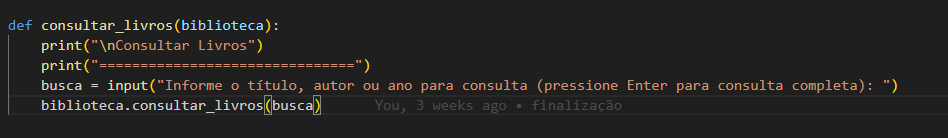


Função da classe Biblioteca

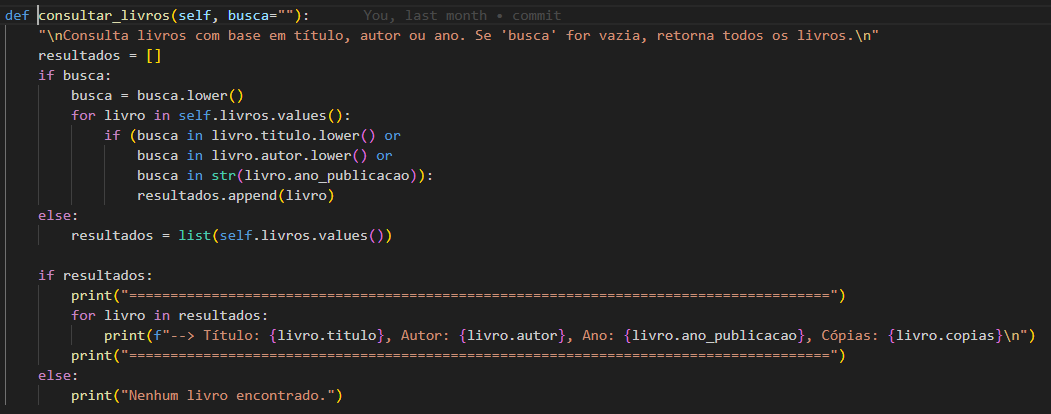


**Função consultar\_livros(biblioteca)**

Esta função permite ao usuário consultar livros na biblioteca, podendo buscar por título, autor ou ano. Se o usuário não fornecer nenhum parâmetro, a consulta retorna todos os livros.



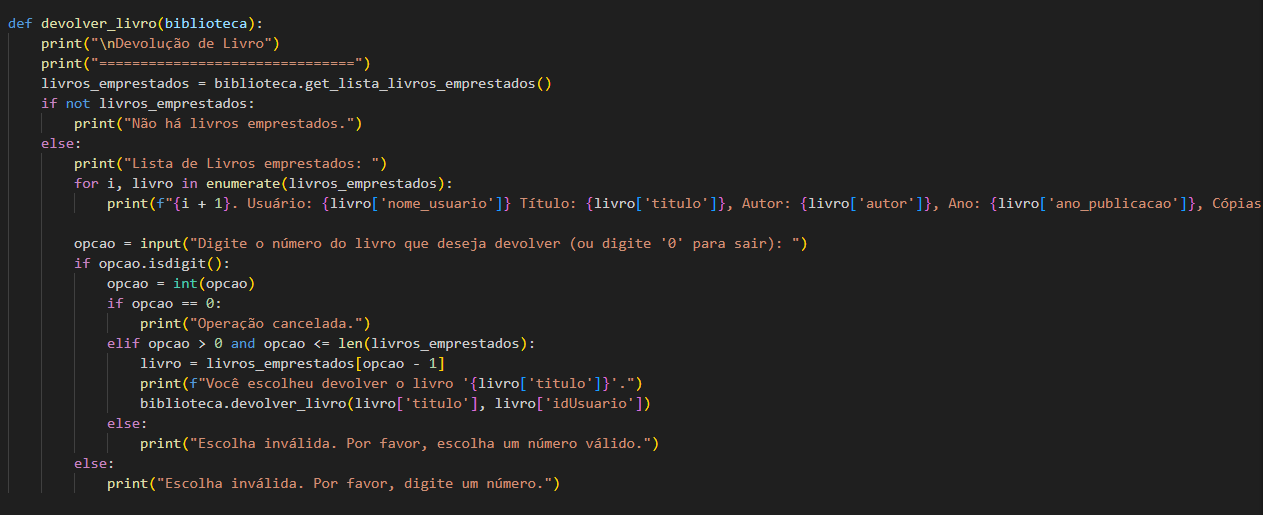
Função da classe Biblioteca



**Função devolver\_livro(biblioteca)**

Esta função permite ao usuário devolver um livro emprestado. Lista os livros emprestados e permite ao usuário escolher qual livro deseja devolver, validando a escolha.

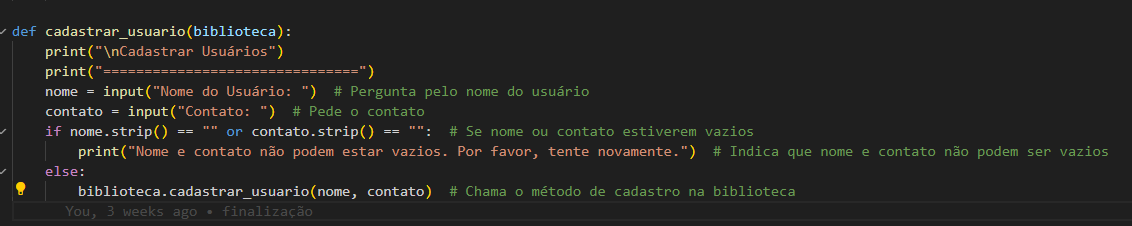
* Obtém a lista de livros disponíveis para empréstimo.
* Exibe essa lista ao usuário.
* Pede ao usuário que escolha um livro pelo número correspondente na lista.
* Caso um livro seja escolhido, lista os usuários cadastrados para que o usuário possa selecionar quem vai pegar o livro emprestado.
* Solicita o ID do usuário que está pegando o livro emprestado.
* Chama o método emprestar\_livro da classe Biblioteca para registrar o empréstimo.

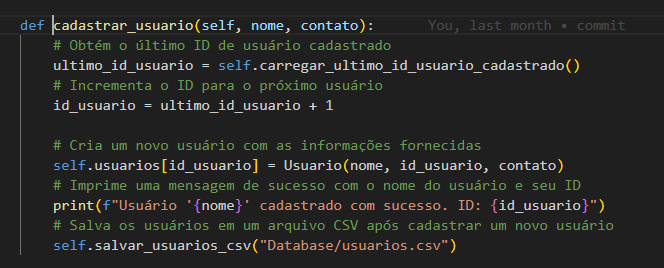


**Função cadastrar\_usuario(biblioteca)**

Esta função cadastra novos usuários na biblioteca:

* Solicita o nome do usuário.
* Solicita o contato do usuário.
* Verifica se o nome e o contato não estão vazios.
* Caso estejam preenchidos, chama o método cadastrar\_usuario da classe Biblioteca para registrar o novo usuário.

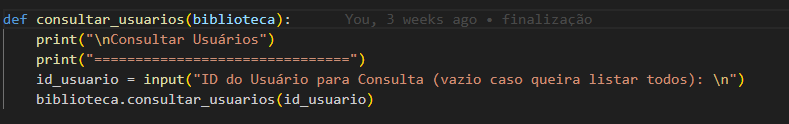
****

Função da classe biblioteca

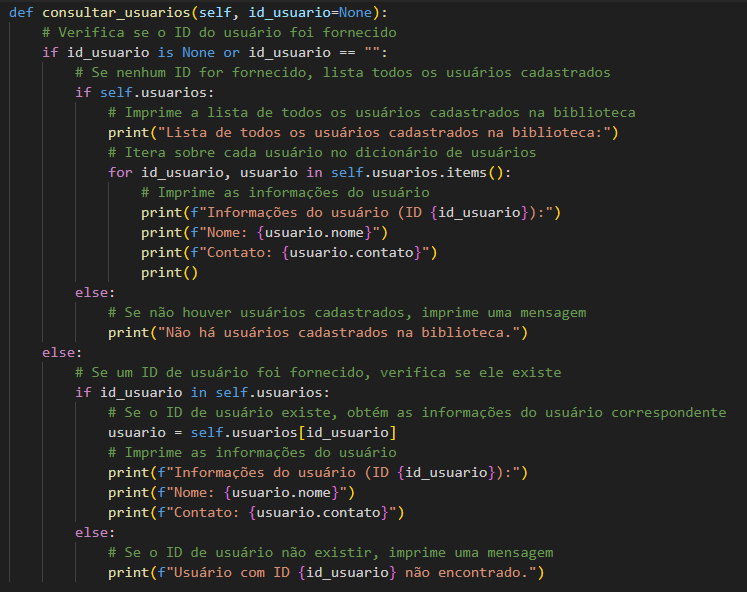
**Função consultar\_usuarios(biblioteca)**

Esta função permite a consulta de usuários cadastrados:

* Solicita o ID do usuário para consulta. Se o usuário não fornecer um ID, todos os usuários serão listados.
* Chama o método consultar\_usuarios da classe Biblioteca para realizar a consulta.



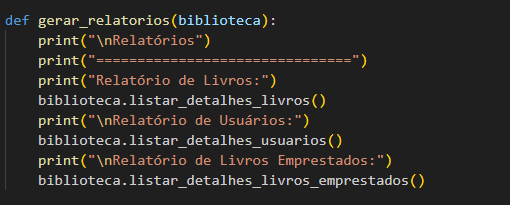
Função da classe biblioteca



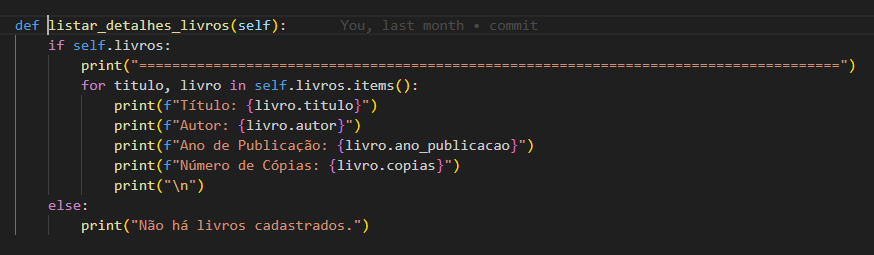
**Função gerar\_relatorios(biblioteca)**

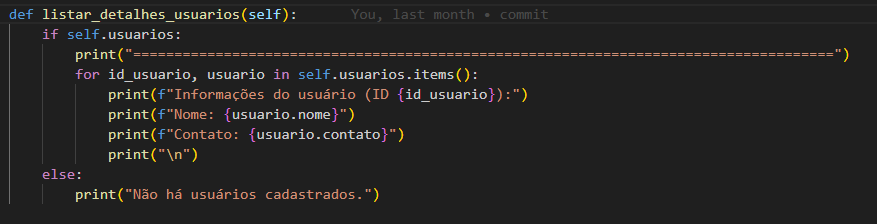
Esta função gera e exibe relatórios da biblioteca:

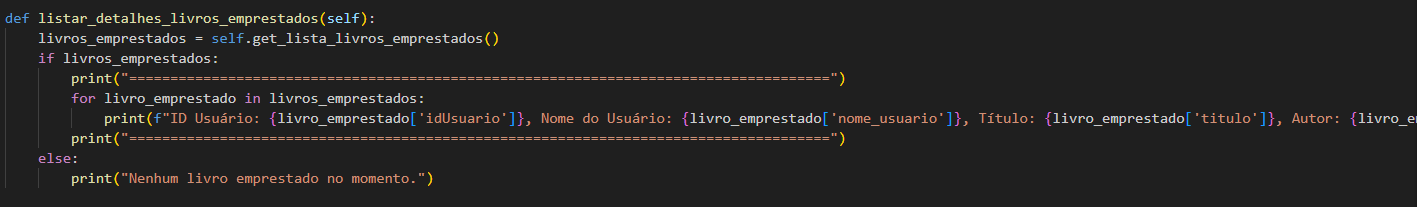
* Exibe o relatório de livros cadastrados chamando listar\_detalhes\_livros da classe Biblioteca.
* Exibe o relatório de usuários cadastrados chamando listar\_detalhes\_usuarios da classe Biblioteca.
* Exibe o relatório de livros emprestados chamando listar\_detalhes\_livros\_emprestados da classe Biblioteca.

****

Funções da classe biblioteca







**Considerações Finais**

O Sistema de Biblioteca é uma ferramenta poderosa para o gerenciamento eficiente de livros e usuários em uma biblioteca. Com uma interface simples e intuitiva, os usuários podem realizar diversas operações de forma rápida e fácil. Este sistema é uma solução ideal para bibliotecas de todos os tamanhos, proporcionando uma experiência agradável tanto para os usuários quanto para os administradores.

**Código completo**

**Arquivo \_\_main\_\_.py**

import os

from Classes.Biblioteca import Biblioteca

def limpar\_console():

    os.system("cls")

def abrirMenu():

    biblioteca = Biblioteca()

    while True:

        print("\nBem-vindo à Biblioteca")

        print("===============================")

        print("            MENU               ")

        print("===============================")

        print("1 - Cadastrar Livros")

        print("2 - Consultar Livros")

        print("3 - Devolução de Livro")

        print("4 - Empréstimo de Livro")

        print("5 - Cadastrar Usuários")

        print("6 - Consultar Usuários")

        print("7 - Relatórios")

        print("0 - Sair")

        print("===============================")

        opcao = input("Escolha uma opção: ")

        if opcao == '1':

            cadastrar\_livro(biblioteca)

        elif opcao == '2':

            consultar\_livros(biblioteca)

        elif opcao == '3':

            devolver\_livro(biblioteca)

        elif opcao == '4':

            emprestar\_livro(biblioteca)

        elif opcao == '5':

            cadastrar\_usuario(biblioteca)

        elif opcao == '6':

            consultar\_usuarios(biblioteca)

        elif opcao == '7':

            gerar\_relatorios(biblioteca)

        elif opcao == '0':

            print("Saindo do sistema...")

            break

        else:

            print("Opção inválida. Por favor, tente novamente.")

        input("Pressione Enter para continuar...")

        limpar\_console()

def cadastrar\_livro(biblioteca):

    print("\nCadastrar Livros")

    print("===============================")

    titulo = input("Título do Livro: ").strip()

    autor = input("Autor do Livro: ").strip()

    if not titulo or not autor:

        print("###### Título e autor não podem ser vazios. ######")

        return

    try:

        ano = int(input("Ano de Publicação: "))

        copias = int(input("Número de Cópias: "))

        if ano <= 0 or copias < 0:

            raise ValueError("###### Ano de publicação e número de cópias devem ser valores maiores que 0. ######")

    except ValueError as e:

        if "invalid literal for int()" in str(e):

            print("###### Ano de publicação e número de cópias devem ser valores maiores que 0. ######")

        else:

            print(f"###### {str(e)} ######")

        return  # Retornar ao início do loop

    biblioteca.cadastrar\_livro(titulo, autor, ano, copias)

def consultar\_livros(biblioteca):

    print("\nConsultar Livros")

    print("===============================")

    busca = input("Informe o título, autor ou ano para consulta (pressione Enter para consulta completa): ")

    biblioteca.consultar\_livros(busca)

def devolver\_livro(biblioteca):

    print("\nDevolução de Livro")

    print("===============================")

    livros\_emprestados = biblioteca.get\_lista\_livros\_emprestados()

    if not livros\_emprestados:

        print("Não há livros emprestados.")

    else:

        print("Lista de Livros emprestados: ")

        for i, livro in enumerate(livros\_emprestados):

            print(f"{i + 1}. Usuário: {livro['nome\_usuario']} Título: {livro['titulo']}, Autor: {livro['autor']}, Ano: {livro['ano\_publicacao']}, Cópias: {livro['copias']}")

        opcao = input("Digite o número do livro que deseja devolver (ou digite '0' para sair): ")

        if opcao.isdigit():

            opcao = int(opcao)

            if opcao == 0:

                print("Operação cancelada.")

            elif opcao > 0 and opcao <= len(livros\_emprestados):

                livro = livros\_emprestados[opcao - 1]

                print(f"Você escolheu devolver o livro '{livro['titulo']}'.")

                biblioteca.devolver\_livro(livro['titulo'], livro['idUsuario'])

            else:

                print("Escolha inválida. Por favor, escolha um número válido.")

        else:

            print("Escolha inválida. Por favor, digite um número.")

def emprestar\_livro(biblioteca):

    print("\nEmpréstimo de Livro")

    print("===============================")

    livros\_disponiveis = biblioteca.get\_lista\_livros()

    if not livros\_disponiveis:

        print("Não há livros disponíveis para empréstimo.")

    else:

        print("Lista de Livros Disponíveis para Empréstimo: \n")

        for i, livro in enumerate(livros\_disponiveis):

            print(f"{i + 1}. Título: {livro.titulo}, Autor: {livro.autor}, Ano: {livro.ano\_publicacao}, Cópias Disponíveis: {livro.copias}")

    opcao = input("\nDigite o número do livro que deseja emprestar ('0' para sair): ")

    if opcao.isdigit():

        opcao = int(opcao)

        if opcao == 0:

            print("Operação cancelada.")

        elif opcao > 0 and opcao <= len(livros\_disponiveis):

            livro\_escolhido = livros\_disponiveis[opcao - 1]

            biblioteca.consultar\_usuarios()

            id\_usuario = input("ID do Usuário que está Emprestando: ")

            biblioteca.emprestar\_livro(livro\_escolhido, id\_usuario)

        else:

            print("Escolha inválida. Por favor, escolha um número válido.")

    else:

        print("Escolha inválida. Por favor, digite um número.")

def cadastrar\_usuario(biblioteca):

    print("\nCadastrar Usuários")

    print("===============================")

    nome = input("Nome do Usuário: ")  # Pergunta pelo nome do usuário

    contato = input("Contato: ")  # Pede o contato

    if nome.strip() == "" or contato.strip() == "":  # Se nome ou contato estiverem vazios

        print("Nome e contato não podem estar vazios. Por favor, tente novamente.")  # Indica que nome e contato não podem ser vazios

    else:

        biblioteca.cadastrar\_usuario(nome, contato)  # Chama o método de cadastro na biblioteca

def consultar\_usuarios(biblioteca):

    print("\nConsultar Usuários")

    print("===============================")

    id\_usuario = input("ID do Usuário para Consulta (vazio caso queira listar todos): \n")

    biblioteca.consultar\_usuarios(id\_usuario)

def gerar\_relatorios(biblioteca):

    print("\nRelatórios")

    print("===============================")

    print("Relatório de Livros:")

    biblioteca.listar\_detalhes\_livros()

    print("\nRelatório de Usuários:")

    biblioteca.listar\_detalhes\_usuarios()

    print("\nRelatório de Livros Emprestados:")

    biblioteca.listar\_detalhes\_livros\_emprestados()

if \_\_name\_\_ == "\_\_main\_\_":

    abrirMenu()

**Código completo**

**Arquivo Biblioteca.py**

import csv  # Importa a bibioteca csv para trabalhar com arquivos CSV

from Classes.Livro import Livro  # Importa a classe Livro do módulo Livro

from Classes.Usuario import Usuario  # Importa a classe Usuario do módulo Usuario

class Biblioteca:

    def \_\_init\_\_(self):

        self.livros = {}

        self.livros\_emprestados = {}

        self.usuarios = {}

        # Carregar dados do CSV ao iniciar o programa

        self.carregar\_livros\_csv("Database/livros.csv")

        self.carregar\_usuarios\_csv("Database/usuarios.csv")

        self.carregar\_livros\_emprestados\_csv("Database/livros\_emprestados.csv")

    def cadastrar\_livro(self, titulo, autor, ano, copias):

        id\_ultimo\_livro\_cadastrado = self.carregar\_ultimo\_id\_livro\_cadastrado()

        proximo\_id\_livro = id\_ultimo\_livro\_cadastrado + 1

        if titulo not in self.livros:

            # Cria o livro com o próximo ID sequencial

            self.livros[proximo\_id\_livro] = Livro(proximo\_id\_livro, titulo, autor, ano, copias)

        else:

            self.livros[titulo].copias += copias

        print(f"Livro '{titulo}' cadastrado com sucesso.")

        # Após cadastrar o livro, salvar os livros em um arquivo CSV

        self.salvar\_livros\_csv("Database/livros.csv")

    def cadastrar\_usuario(self, nome, contato):

        # Obtém o último ID de usuário cadastrado

        ultimo\_id\_usuario = self.carregar\_ultimo\_id\_usuario\_cadastrado()

        # Incrementa o ID para o próximo usuário

        id\_usuario = ultimo\_id\_usuario + 1

        # Cria um novo usuário com as informações fornecidas

        self.usuarios[id\_usuario] = Usuario(nome, id\_usuario, contato)

        # Imprime uma mensagem de sucesso com o nome do usuário e seu ID

        print(f"Usuário '{nome}' cadastrado com sucesso. ID: {id\_usuario}")

        # Salva os usuários em um arquivo CSV após cadastrar um novo usuário

        self.salvar\_usuarios\_csv("Database/usuarios.csv")

    def consultar\_usuarios(self, id\_usuario=None):

        # Verifica se o ID do usuário foi fornecido

        if id\_usuario is None or id\_usuario == "":

            # Se nenhum ID for fornecido, lista todos os usuários cadastrados

            if self.usuarios:

                # Imprime a lista de todos os usuários cadastrados na biblioteca

                print("Lista de todos os usuários cadastrados na biblioteca:")

                # Itera sobre cada usuário no dicionário de usuários

                for id\_usuario, usuario in self.usuarios.items():

                    # Imprime as informações do usuário

                    print(f"Informações do usuário (ID {id\_usuario}):")

                    print(f"Nome: {usuario.nome}")

                    print(f"Contato: {usuario.contato}")

                    print()

            else:

                # Se não houver usuários cadastrados, imprime uma mensagem

                print("Não há usuários cadastrados na biblioteca.")

        else:

            # Se um ID de usuário foi fornecido, verifica se ele existe

            if id\_usuario in self.usuarios:

                # Se o ID de usuário existe, obtém as informações do usuário correspondente

                usuario = self.usuarios[id\_usuario]

                # Imprime as informações do usuário

                print(f"Informações do usuário (ID {id\_usuario}):")

                print(f"Nome: {usuario.nome}")

                print(f"Contato: {usuario.contato}")

            else:

                # Se o ID de usuário não existir, imprime uma mensagem

                print(f"Usuário com ID {id\_usuario} não encontrado.")

    def carregar\_ultimo\_id\_usuario\_cadastrado(self):

            # Verifica se existem usuários cadastrados para determinar o último ID

            if self.usuarios:

                # Se houver usuários cadastrados, cria uma lista dos IDs de usuários

                lista\_ids\_usuarios = list(self.usuarios.keys())

                # Retorna o maior ID de usuário na lista

                return max(lista\_ids\_usuarios)

            else:

                # Se não houver usuários cadastrados, retorna 0

                return 0

    def carregar\_ultimo\_id\_livro\_cadastrado(self):

            # Verifica se existem livros cadastrados para determinar o último ID

            if self.livros:

                # Se houver livros cadastrados, cria uma lista dos IDs de livros

                lista\_ids\_livros = list(self.livros.keys())

                # Retorna o maior ID de livro na lista

                return max(lista\_ids\_livros)

            else:

                # Se não houver livros cadastrados, retorna 0

                return 0

    def obter\_usuario\_por\_id(self, id\_usuario):

        # Inicializa a variável 'usuario' como None

        usuario = None

        # Converte o ID do usuário para um número inteiro

        id\_usuario = int(id\_usuario)

        # Itera sobre cada item no dicionário de usuários

        for i in self.usuarios:

            # Verifica se o ID do usuário atual corresponde ao ID fornecido

            if(self.usuarios[i].id\_usuario == id\_usuario):

                # Se encontrar o usuário correspondente, atribui-o à variável 'usuario'

                usuario = self.usuarios[i]

                # Interrompe o loop

                break

        # Retorna o usuário encontrado (ou None se não for encontrado)

        return usuario

    def emprestar\_livro(self, livro\_escolhido, id\_usuario):

        # Inicializa a variável 'disponivel' como False

        disponivel = False

        # Inicializa o índice do livro como -1

        idxLivro = -1

        # Itera sobre cada índice no dicionário de livros

        for idx in self.livros:

            # Obtém o livro correspondente ao índice atual

            livro\_encontrado = self.livros[idx]

            # Verifica se o livro atual é o mesmo que o livro escolhido e se ainda há cópias disponíveis

            if (livro\_encontrado == livro\_escolhido and livro\_escolhido.copias > 0):

                # Se o livro estiver disponível, define 'disponivel' como True e armazena o índice do livro

                disponivel = True

                idxLivro = idx

                # Interrompe o loop

                break

        if disponivel:

            usuario = self.obter\_usuario\_por\_id(id\_usuario)

            if usuario is None:

                print(f"Usuário com ID {id\_usuario} não encontrado.")

                return

            # Subtrair uma cópia disponível

            self.livros[idxLivro].copias -= 1

            # Adicionar livro à lista de livros emprestados pelo usuário

            if id\_usuario not in self.livros\_emprestados:

                self.livros\_emprestados[id\_usuario] = [livro\_escolhido.titulo]

            else:

                self.livros\_emprestados[id\_usuario].append(livro\_escolhido.titulo)

            print(f"Livro '{livro\_escolhido.titulo}' emprestado ao usuário {usuario.nome}.")

            # Salvar os livros emprestados em um arquivo CSV após emprestar um livro

            self.salvar\_livros\_emprestados\_csv("Database/livros\_emprestados.csv")

        else:

            print("Livro não disponível para empréstimo.")

    def listar\_detalhes\_livros\_emprestados(self):

        livros\_emprestados = self.get\_lista\_livros\_emprestados()

        if livros\_emprestados:

            print("=====================================================================================")

            for livro\_emprestado in livros\_emprestados:

                print(f"ID Usuário: {livro\_emprestado['idUsuario']}, Nome do Usuário: {livro\_emprestado['nome\_usuario']}, Título: {livro\_emprestado['titulo']}, Autor: {livro\_emprestado['autor']}, Ano de Publicação: {livro\_emprestado['ano\_publicacao']}, Cópias: {livro\_emprestado['copias']}")

            print("=====================================================================================")

        else:

            print("Nenhum livro emprestado no momento.")

    def devolver\_livro(self, titulo, id\_usuario):

        if id\_usuario in self.livros\_emprestados and titulo in self.livros\_emprestados[id\_usuario]:

            idxLivro = -1

            for i in self.livros:

                if(self.livros[i].titulo == titulo):

                  idxLivro = i

                  break

            # Incrementar uma cópia disponível

            self.livros[idxLivro].copias += 1

            # Remover o livro da lista de livros emprestados pelo usuário

            self.livros\_emprestados[id\_usuario].remove(titulo)

            self.salvar\_livros\_emprestados\_csv('Database/livros\_emprestados.csv')

            print(f"Livro '{titulo}' devolvido com sucesso.")

        else:

            print(f"O usuário {id\_usuario} não tem este livro emprestado.")

    def consultar\_livros(self, busca=""):

        "\nConsulta livros com base em título, autor ou ano. Se 'busca' for vazia, retorna todos os livros.\n"

        resultados = []

        if busca:

            busca = busca.lower()

            for livro in self.livros.values():

                if (busca in livro.titulo.lower() or

                    busca in livro.autor.lower() or

                    busca in str(livro.ano\_publicacao)):

                    resultados.append(livro)

        else:

            resultados = list(self.livros.values())

        if resultados:

            print("=====================================================================================")

            for livro in resultados:

                print(f"--> Título: {livro.titulo}, Autor: {livro.autor}, Ano: {livro.ano\_publicacao}, Cópias: {livro.copias}\n")

            print("=====================================================================================")

        else:

            print("Nenhum livro encontrado.")

    def get\_lista\_livros(self):

        return list(self.livros.values())

    def obter\_livro\_por\_titulo(self, titulo):

        for livro in self.livros.values():

            if livro.titulo == titulo:

                return livro

        return None

    def get\_lista\_livros\_emprestados(self):

        lista\_emprestimos = []

        for id\_usuario, livros\_emprestados in self.livros\_emprestados.items():

            usuario = self.obter\_usuario\_por\_id(id\_usuario)

            if usuario:

                for titulo in livros\_emprestados:

                    livro = self.obter\_livro\_por\_titulo(titulo)

                    if livro:

                        lista\_emprestimos.append({

                            'idUsuario': id\_usuario,

                            'nome\_usuario': usuario.nome,

                            'titulo': titulo,

                            'autor': livro.autor,

                            'ano\_publicacao': livro.ano\_publicacao,

                            'copias': livro.copias

                        })

        return lista\_emprestimos

    def gerar\_relatorios(self):

        print("Relatório de Livros:")

        self.listar\_detalhes\_livros()

        print("\nRelatório de Usuários:")

        self.listar\_detalhes\_usuarios()

        print("\nRelatório de Livros Emprestados:")

        self.listar\_detalhes\_livros\_emprestados()

    def listar\_detalhes\_livros(self):

        if self.livros:

            print("=====================================================================================")

            for titulo, livro in self.livros.items():

                print(f"Título: {livro.titulo}")

                print(f"Autor: {livro.autor}")

                print(f"Ano de Publicação: {livro.ano\_publicacao}")

                print(f"Número de Cópias: {livro.copias}")

                print("\n")

        else:

            print("Não há livros cadastrados.")

    def listar\_detalhes\_usuarios(self):

        if self.usuarios:

            print("=====================================================================================")

            for id\_usuario, usuario in self.usuarios.items():

                print(f"Informações do usuário (ID {id\_usuario}):")

                print(f"Nome: {usuario.nome}")

                print(f"Contato: {usuario.contato}")

                print("\n")

        else:

            print("Não há usuários cadastrados.")

    def salvar\_livros\_csv(self, nome\_arquivo):

        # Abrir o arquivo CSV para escrita

        with open(nome\_arquivo, 'w', newline='') as arquivo\_csv:

            # Definir os nomes das colunas

            colunas = ['ID', 'Titulo', 'Autor', 'Ano', 'Copias']

            # Criar o escritor CSV

            escritor\_csv = csv.DictWriter(arquivo\_csv, fieldnames=colunas)

            # Escrever o cabeçalho

            escritor\_csv.writeheader()

            # Escrever os detalhes de cada livro

            for id\_livro, livro in self.livros.items():

                escritor\_csv.writerow({

                    'ID': id\_livro,

                    'Titulo': livro.titulo,

                    'Autor': livro.autor,

                    'Ano': livro.ano\_publicacao,

                    'Copias': livro.copias

                })

    def salvar\_usuarios\_csv(self, nome\_arquivo):

        # Abrir o arquivo CSV para escrita

        with open(nome\_arquivo, 'w', newline='') as arquivo\_csv:

            # Definir os nomes das colunas

            colunas = ['ID', 'Nome', 'Contato']

            # Criar o escritor CSV

            escritor\_csv = csv.DictWriter(arquivo\_csv, fieldnames=colunas)

            # Escrever o cabeçalho

            escritor\_csv.writeheader()

            # Escrever os detalhes de cada usuário

            for usuario in self.usuarios.values():

                escritor\_csv.writerow({

                    'ID': usuario.id\_usuario,

                    'Nome': usuario.nome,

                    'Contato': usuario.contato

                })

    def carregar\_livros\_csv(self, nome\_arquivo):

        # Tentar abrir o arquivo CSV para leitura

        try:

            with open(nome\_arquivo, 'r', newline='') as arquivo\_csv:

                # Criar um leitor CSV para ler o arquivo linha por linha

                leitor\_csv = csv.DictReader(arquivo\_csv)

                # Iterar sobre cada linha do arquivo CSV

                for linha in leitor\_csv:

                    # Extrair informações de cada linha

                    id\_livro = int(linha['ID'])

                    titulo = linha['Titulo']

                    autor = linha['Autor']

                    ano = int(linha['Ano'])

                    copias = int(linha['Copias'])

                    # Adicionar o livro ao dicionário de livros da biblioteca

                    self.livros[id\_livro] = Livro(id\_livro, titulo, autor, ano, copias)

        # Lidar com a exceção se o arquivo não for encontrado

        except FileNotFoundError:

            # Imprimir uma mensagem informando que o arquivo não foi encontrado

            print(f"Arquivo '{nome\_arquivo}' não encontrado. Nenhum livro carregado.")

    def carregar\_usuarios\_csv(self, nome\_arquivo):

        # Tentar abrir o arquivo CSV para leitura

        try:

            with open(nome\_arquivo, 'r', newline='') as arquivo\_csv:

                # Criar um leitor CSV para ler o arquivo linha por linha

                leitor\_csv = csv.DictReader(arquivo\_csv)

                # Iterar sobre cada linha do arquivo CSV

                for linha in leitor\_csv:

                    # Extrair informações de cada linha

                    id\_usuario = int(linha['ID'])

                    nome = linha['Nome']

                    contato = linha['Contato']

                    # Adicionar o usuário ao dicionário de usuários da biblioteca

                    self.usuarios[id\_usuario] = Usuario(nome, id\_usuario, contato)

        # Lidar com a exceção se o arquivo não for encontrado

        except FileNotFoundError:

            # Imprimir uma mensagem informando que o arquivo não foi encontrado

            print(f"Arquivo '{nome\_arquivo}' não encontrado. Nenhum usuário carregado.")

    def salvar\_livros\_emprestados\_csv(self, nome\_arquivo):

        # Abrir o arquivo CSV para adição

        with open(nome\_arquivo, 'w', newline='') as arquivo\_csv:

            # Definir os nomes das colunas

            colunas = ['ID\_Usuario', 'titulo']

            # Criar o escritor CSV

            escritor\_csv = csv.DictWriter(arquivo\_csv, fieldnames=colunas)

            # Escrever o cabeçalho

            escritor\_csv.writeheader()

            # Escrever os detalhes de cada usuário e os livros emprestados

            for id\_usuario, livros\_emprestados in self.livros\_emprestados.items():

                for id\_livro in livros\_emprestados:

                    escritor\_csv.writerow({

                        'ID\_Usuario': id\_usuario,

                        'titulo': id\_livro

                    })

    def carregar\_livros\_emprestados\_csv(self, nome\_arquivo):

        try:

            with open(nome\_arquivo, 'r', newline='') as arquivo\_csv:

                leitor\_csv = csv.DictReader(arquivo\_csv)

                for linha in leitor\_csv:

                    id\_usuario = int(linha['ID\_Usuario'])

                    id\_livro = linha['titulo']

                    # Adiciona o livro emprestado ao usuário correspondente

                    if id\_usuario not in self.livros\_emprestados:

                        self.livros\_emprestados[id\_usuario] = [id\_livro]

                    else:

                        self.livros\_emprestados[id\_usuario].append(id\_livro)

        except FileNotFoundError:

            print(f"Arquivo '{nome\_arquivo}' não encontrado. Nenhum livro emprestado carregado.")

**Código completo**

**Arquivo Livro.py**

class Livro:

    def \_\_init\_\_(self, id\_livro, titulo, autor, ano\_publicacao, copias):

        self.id\_livro = id\_livro

        self.titulo = titulo

        self.autor = autor

        self.ano\_publicacao = ano\_publicacao

        self.copias = copias

**Código completo**

**Arquivo Usuario.py**

class Usuario:

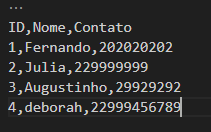
    def \_\_init\_\_(self, nome, id\_usuario, contato):

        self.nome = nome

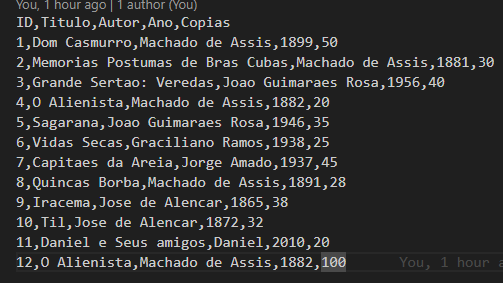
        self.id\_usuario = id\_usuario

        self.contato = contato

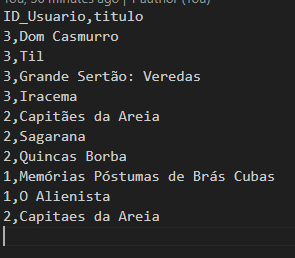
**Arquivo CSV de usuários**

****

**Arquivo CSV de livros**

****

**Arquivo CSV de livros emprestados**



**Organização das pastas**

