

Cadastrar Livros

A função `CadastrarLivro()` tem como objetivo **cadastrar um novo livro em um sistema**, armazenando suas informações em um banco de dados. Vamos detalhar cada parte da função para facilitar o entendimento:

1. Limpar a tela e solicitar dados do livro

```
os.system("cls")
```

```
print("Insira os dados do novo livro:")
```

- `os.system("cls")`: Limpa a tela do terminal (comum em sistemas Windows).
 - `print("Insira os dados do novo livro:")`: Exibe uma mensagem solicitando os dados do livro.
-

2. Coleta dos dados do livro

A função solicita os seguintes dados do livro:

- **Título**: `titulo = input("Título do Livro: ")`
- **Gênero**: `genero = input("Gênero: ")`
- **Sinopse**: `sinopse = input("Sinopse: ")`
- **Autor**: `autor = input("Autor: ")`
- **Data de publicação**: `publicacao = input("Data de publicação (AAAA/MM/DD): ")`
- **Valor de compra**: `valorCompra = input("Valor de Compra: ")`
- **Valor de revenda**: `valorRevenda = input("Valor de revenda: ")`

Após cada entrada, a função verifica se o usuário digitou "sair" (em letras minúsculas). Se sim, a função redireciona para o `Menu()`.

3. Validação dos dados

```
if not titulo or not genero or not sinopse or not autor or not publicacao or not valorCompra or not valorRevenda or len(publicacao) != 10:
```

```
    print("\033[31mDados inválidos. Verifique os dados e tente novamente.\033[m")
```

```
    input()
```

```
    Menu()
```

- Verifica se todos os campos foram preenchidos.

- Confere se a data de publicação (publicacao) tem exatamente 10 caracteres (formato AAAA/MM/DD).
 - Se algum dado estiver inválido, exibe uma mensagem de erro em vermelho (\033[31m) e retorna ao Menu().
-

4. Inserção dos dados no banco de dados

try:

```
cursor.execute(f"INSERT INTO livro (titulo, genero, sinopse, autor, publicacao, valor_compra, valor_revenda) VALUES ('{titulo}', '{genero}', '{sinopse}', '{autor}', '{publicacao}', '{valorCompra}', '{valorRevenda}')
```

```
conexao.commit()
```

- Usa um cursor (cursor) para executar um comando SQL que insere os dados do livro na tabela livro.
 - conexao.commit(): Confirma a inserção dos dados no banco de dados.
-

5. Adicionar o livro ao estoque

```
cursor.execute("SELECT id FROM livro")
```

```
add_estoque = cursor.fetchall()
```

```
for linha in add_estoque:
```

```
    id = linha[0]
```

```
cursor.execute(f"INSERT INTO estoque (id_livro, quantidade) VALUES ('{id}', '{0}')
```

```
conexao.commit()
```

- Recupera o id do livro recém-cadastrado na tabela livro.
 - Insere uma nova linha na tabela estoque, associando o id do livro a uma quantidade inicial de 0.
-

6. Mensagem de sucesso ou erro

```
print("\033[32mCadastro registrado com sucesso.\033[m")
```

```
except:
```

```
    print("\033[31mNão foi possível registrar o cadastro. Verifique os dados e tente novamente.\033[m")
```

- Se tudo der certo, exibe uma mensagem de sucesso em verde (\033[32m).
- Se ocorrer algum erro (por exemplo, dados inválidos ou falha no banco de dados), exibe uma mensagem de erro em vermelho (\033[31m).

7. Retorno ao menu

`input()`

`Menu()`

- Aguarda o usuário pressionar Enter (`input()`).
- Retorna ao `Menu()` principal do sistema.

Resumo da funcionalidade

A função `CadastrarLivro()`:

1. Coleta os dados de um novo livro.
2. Valida se os dados estão corretos.
3. Insere os dados na tabela livro do banco de dados.
4. Adiciona o livro ao estoque com quantidade inicial 0.
5. Exibe mensagens de sucesso ou erro.
6. Retorna ao menu principal.

Essa função é útil para gerenciar o cadastro de livros em um sistema de biblioteca ou loja de livros.

Listar Livros

Aqui está a explicação detalhada da função `ListarLivros()`, que tem como objetivo **listar todos os livros cadastrados no sistema**. Vamos analisar cada parte do código para facilitar o entendimento:

1. Limpar a tela

```
os.system("cls")
```

- `os.system("cls")`: Limpa a tela do terminal (comum em sistemas Windows).
-

2. Consulta ao banco de dados

```
cursor.execute("SELECT * FROM livro")
```

```
resultado = cursor.fetchall()
```

- `cursor.execute("SELECT * FROM livro")`: Executa um comando SQL para selecionar todos os registros da tabela livro.
 - `cursor.fetchall()`: Recupera todos os resultados da consulta e armazena na variável `resultado`.
-

3. Verificação de registros

```
if len(resultado) == 0:
```

```
    print("\033[31mNão há registros.\033[m")
```

```
    input()
```

```
    Menu()
```

- Verifica se a lista de resultados (`resultado`) está vazia.
 - Se não houver registros, exibe uma mensagem de erro em vermelho (`\033[31m`) e retorna ao `Menu()`.
-

4. Exibição dos livros cadastrados

```
print("="*50)
```

```
print("Livros Cadastrados")
```

```
print("="*50)
```

```
for linha in resultado:
```

```
print(f"ID: {linha[0]}")
```

```
print(f"Título: {linha[1]}")
```

```
print(f"Gênero: {linha[2]}")
```

```
print(f"Autor: {linha[3]}")
```

```
print(f"Publicação: {linha[4]}")
```

```
print(f"Sinopse: {linha[5]}\n")
```

```
print("-"*50)
```

- Exibe um cabeçalho formatado com = para destacar o título "Livros Cadastrados".
 - Itera sobre cada linha (linha) da lista resultado e exibe os seguintes dados de cada livro:
 - **ID:** Identificador único do livro.
 - **Título:** Nome do livro.
 - **Gênero:** Categoria do livro.
 - **Autor:** Nome do autor.
 - **Publicação:** Data de publicação.
 - **Sinopse:** Breve descrição do livro.
 - Após exibir os dados de um livro, insere uma linha de separação (-).
-

5. Mensagem de conclusão

```
print("\033[32mListagem Concluída com sucesso.\033[m")
```

- Exibe uma mensagem de sucesso em verde (\033[32m) indicando que a listagem foi concluída.
-

6. Retorno ao menu

```
input()
```

```
Menu()
```

- Aguarda o usuário pressionar Enter (input()).
 - Retorna ao Menu() principal do sistema.
-

Resumo da funcionalidade

A função ListarLivros():

1. Limpa a tela.

2. Consulta todos os livros cadastrados na tabela livro do banco de dados.
3. Verifica se há registros. Se não houver, exibe uma mensagem de erro e retorna ao menu.
4. Exibe os dados de todos os livros cadastrados em um formato organizado.
5. Mostra uma mensagem de sucesso ao concluir a listagem.
6. Retorna ao menu principal.

Essa função é útil para visualizar todos os livros cadastrados no sistema, permitindo ao usuário conferir as informações de forma clara e organizada.

Listar Estoque

Aqui está a explicação detalhada da função `ListarEstoque()`, que tem como objetivo **listar o estoque de livros cadastrados no sistema**. Vamos analisar cada parte do código para facilitar o entendimento:

1. Limpar a tela

```
os.system("cls")
```

- `os.system("cls")`: Limpa a tela do terminal (comum em sistemas Windows).
-

2. Consulta ao banco de dados

```
cursor.execute("select l.id, l.titulo, l.valor_compra, l.valor_revenda, e.quantidade from  
livro as l join estoque as e on l.id = e.id_livro")
```

```
resultado = cursor.fetchall()
```

- `cursor.execute(...)`: Executa um comando SQL que seleciona dados das tabelas **livro** e **estoque**.
 - **Tabela livro (l)**: Recupera o id, titulo, valor_compra e valor_revenda do livro.
 - **Tabela estoque (e)**: Recupera a quantidade disponível no estoque.
 - A junção (JOIN) é feita pelo id do livro (`l.id = e.id_livro`).
 - `cursor.fetchall()`: Recupera todos os resultados da consulta e armazena na variável `resultado`.
-

3. Verificação de registros

```
if len(resultado) == 0:
```

```
    print("\033[31mNão há livros registrados para listar um estoque.\033[m")
```

```
    input()
```

```
    Menu()
```

- Verifica se a lista de resultados (`resultado`) está vazia.
 - Se não houver registros, exibe uma mensagem de erro em vermelho (`\033[31m`) e retorna ao `Menu()`.
-

4. Exibição do estoque

```

print("="*50)

print("ESTOQUE")

print("="*50)

for linha in resultado:

    print(f"livro ID: {linha[0]}")

    print(f"Título: {linha[1]}")

    print(f"Preço de compra: {linha[2]}R$")

    print(f"Valor de revenda: {linha[3]}R$")

    print(f"Quantidade disponível: {linha[4]}\n")

print("-"*50)

```

- Exibe um cabeçalho formatado com = para destacar o título "ESTOQUE".
- Itera sobre cada linha (linha) da lista resultado e exibe os seguintes dados de cada livro:
 - **livro ID:** Identificador único do livro.
 - **Título:** Nome do livro.
 - **Preço de compra:** Valor de compra do livro.
 - **Valor de revenda:** Valor de revenda do livro.
 - **Quantidade disponível:** Quantidade do livro em estoque.
- Após exibir os dados de um livro, insere uma linha de separação (-).

5. Mensagem de conclusão

```
print("\033[32mListagem concluída com sucesso.\033[m")
```

- Exibe uma mensagem de sucesso em verde (\033[32m) indicando que a listagem foi concluída.

6. Retorno ao menu

```
input()
```

```
Menu()
```

- Aguarda o usuário pressionar Enter (input()).
- Retorna ao Menu() principal do sistema.

Resumo da funcionalidade

A função `ListarEstoque()`:

1. Limpa a tela.
2. Consulta os dados de estoque dos livros, combinando informações das tabelas livro e estoque.
3. Verifica se há registros. Se não houver, exibe uma mensagem de erro e retorna ao menu.
4. Exibe os dados de estoque de todos os livros cadastrados em um formato organizado.
5. Mostra uma mensagem de sucesso ao concluir a listagem.
6. Retorna ao menu principal.

Essa função é útil para visualizar o estoque de livros, permitindo ao usuário conferir a quantidade disponível, os preços de compra e revenda, e outras informações relevantes.

Compra Livros

Aqui está a explicação detalhada da função `ComprarLivros()`, que tem como objetivo **registrar a compra de livros e atualizar o estoque**. Vamos analisar cada parte do código para facilitar o entendimento:

1. Limpar a tela

```
os.system("cls")
```

- `os.system("cls")`: Limpa a tela do terminal (comum em sistemas Windows).
-

2. Solicitar o ID do livro

```
id_livro = input("Insira o ID do livro a ser comprado: ")
```

```
if not id_livro:
```

```
    ComprarLivros()
```

```
if id_livro.lower() == "sair":
```

```
    Menu()
```

- Solicita ao usuário o **ID do livro** que será comprado.
 - Se o campo estiver vazio (`if not id_livro`), a função se chama novamente (`ComprarLivros()`).
 - Se o usuário digitar "sair", a função retorna ao `Menu()`.
-

3. Verificar se o livro existe

```
cursor.execute(f"SELECT * FROM livro WHERE id = {id_livro}")
```

```
resultado = cursor.fetchall()
```

```
if len(resultado) == 0:
```

```
    print(f"\033[31mNão há livro registrado com o id {id_livro}\033[m")
```

```
    input()
```

```
    Menu()
```

- Executa uma consulta SQL para verificar se o livro com o ID fornecido existe na tabela livro.
 - Se não houver resultados (`len(resultado) == 0`), exibe uma mensagem de erro em vermelho (`\033[31m`) e retorna ao `Menu()`.
-

4. Recuperar informações do livro

```
for linha in resultado:
```

```
    titulo = linha[1]
```

```
    valor_compra = linha[6]
```

- Itera sobre o resultado da consulta (mesmo que haja apenas uma linha) e armazena:
 - **Título do livro:** titulo = linha[1].
 - **Valor de compra:** valor_compra = linha[6].
-

5. Solicitar a quantidade de cópias

```
print(f"Insira quantas cópias do livro {titulo} serão compradas.")
```

```
try:
```

```
    quantidade = int(input())
```

```
except ValueError:
```

```
    print("\033[31mDado inválido.\033[m")
```

```
    input()
```

```
    Menu()
```

- Solicita ao usuário a **quantidade de cópias** do livro que serão compradas.
 - Se o usuário digitar um valor não numérico, exibe uma mensagem de erro em vermelho (\033[31m) e retorna ao Menu().
-

6. Solicitar a data da compra

```
data_compra = input("Registre a data da compra. (AAAA/MM/DD): ")
```

```
if len(data_compra) != 10:
```

```
    print("\033[31m data precisa ter 10 caracteres.\033[m")
```

```
    input()
```

```
    Menu()
```

- Solicita a **data da compra** no formato AAAA/MM/DD.
 - Se a data não tiver exatamente 10 caracteres, exibe uma mensagem de erro em vermelho (\033[31m) e retorna ao Menu().
-

7. Calcular o custo total

```
custo = valor_compra * quantidade
```

- Calcula o **custo total** da compra multiplicando o valor de compra do livro pela quantidade.

8. Atualizar o estoque e registrar a compra

try:

```
cursor.execute("SELECT * from estoque")
```

```
resultado3 = cursor.fetchall()
```

```
for linha in resultado3:
```

```
    quantidadeEstoque = linha[1]
```

```
    atualizarQuantidade = quantidadeEstoque + quantidade
```

```
    cursor.execute(f"INSERT INTO compra (id_livro, quantidade, data_compra, custo)
VALUES ('{id_livro}','{quantidade}','{data_compra}','{custo}')
```

```
    conexao.commit()
```

```
    cursor.execute(f"UPDATE estoque SET quantidade = {atualizarQuantidade} WHERE
id_livro = {id_livro}")
```

```
    conexao.commit()
```

```
    print("\033[32mCompra registrada com sucesso.\033[m")
```

```
    input()
```

except:

```
    print("\033[31mNão foi possível registrar a compra. verifique os dados e tente
novamente.\033[m")
```

```
    input()
```

- **Passo 1:** Recupera a quantidade atual do livro no estoque (quantidadeEstoque) e calcula a nova quantidade (atualizarQuantidade = quantidadeEstoque + quantidade).
- **Passo 2:** Insere os dados da compra na tabela compra, incluindo:
 - ID do livro (id_livro).
 - Quantidade comprada (quantidade).
 - Data da compra (data_compra).
 - Custo total (custo).
- **Passo 3:** Atualiza a quantidade do livro no estoque (UPDATE estoque).
- Se tudo der certo, exibe uma mensagem de sucesso em verde (\033[32m).

- Se ocorrer algum erro (por exemplo, dados inválidos ou falha no banco de dados),
exibe uma mensagem de erro em vermelho (033[31m).
-

9. Retorno ao menu

`Menu()`

- Retorna ao `Menu()` principal do sistema.
-

Resumo da funcionalidade

A função `ComprarLivros()`:

1. Solicita o ID do livro e verifica se ele existe.
2. Recupera o título e o valor de compra do livro.
3. Solicita a quantidade de cópias e a data da compra.
4. Calcula o custo total da compra.
5. Atualiza o estoque e registra a compra no banco de dados.
6. Exibe mensagens de sucesso ou erro.
7. Retorna ao menu principal.

Essa função é útil para gerenciar as compras de livros, atualizando o estoque e mantendo um registro das transações.

Lista Compras

Aqui está a explicação detalhada da função `ListarCompras()`, que tem como objetivo **listar as compras de livros registradas no sistema**. Vamos analisar cada parte do código para facilitar o entendimento:

1. Limpar a tela

```
os.system("cls")
```

- `os.system("cls")`: Limpa a tela do terminal (comum em sistemas Windows).
-

2. Consulta ao banco de dados

```
cursor.execute("SELECT c.id, l.id, l.titulo, c.quantidade, c.data_compra, c.custo FROM compra as c JOIN livro as l ON c.id_livro = l.id")
```

```
resultado = cursor.fetchall()
```

- `cursor.execute(...)`: Executa um comando SQL que seleciona dados das tabelas compra e livro.
 - **Tabela compra (c)**: Recupera o id, quantidade, data_compra e custo da compra.
 - **Tabela livro (l)**: Recupera o id e titulo do livro.
 - A junção (JOIN) é feita pelo id_livro da compra e o id do livro (`c.id_livro = l.id`).
 - `cursor.fetchall()`: Recupera todos os resultados da consulta e armazena na variável resultado.
-

3. Verificação de registros

```
if len(resultado) == 0:
```

```
    print("\033[31mNão há registro.\033[m")
```

```
    input()
```

```
    Menu()
```

- Verifica se a lista de resultados (resultado) está vazia.
 - Se não houver registros, exibe uma mensagem de erro em vermelho (`\033[31m`) e retorna ao `Menu()`.
-

4. Exibição das compras

```
print("="*50)
```

```

print("REGISTRO DE COMPRAS")

print("="*50)

for linha in resultado:

    id = linha[0]

    id_livro = linha[1]

    titulo = linha[2]

    quantidade = linha[3]

    data_compra = linha[4]

    custo = linha[5]

    print(f"ID da compra: {id}")

    print(f"ID do livro: {id_livro}")

    print(f"Livro: {titulo}")

    print(f"Quantidade comprada: {quantidade}")

    print(f>Data da compra: {data_compra}")

    print(f"Custo total: {custo}R$\n")

    print("-"*50)

```

- Exibe um cabeçalho formatado com = para destacar o título "REGISTRO DE COMPRAS".
- Itera sobre cada linha (linha) da lista resultado e exibe os seguintes dados de cada compra:
 - **ID da compra:** Identificador único da compra.
 - **ID do livro:** Identificador único do livro comprado.
 - **Livro:** Título do livro comprado.
 - **Quantidade comprada:** Quantidade de cópias compradas.
 - **Data da compra:** Data em que a compra foi realizada.
 - **Custo total:** Valor total da compra.
- Após exibir os dados de uma compra, insere uma linha de separação (-).

5. Mensagem de conclusão

```

print("\033[32mListagem concluída com sucesso.\033[m")

```

- Exibe uma mensagem de sucesso em verde (\033[32m) indicando que a listagem foi concluída.

6. Retorno ao menu

`input()`

`Menu()`

- Aguarda o usuário pressionar Enter (`input()`).
- Retorna ao `Menu()` principal do sistema.

Resumo da funcionalidade

A função `ListarCompras()`:

1. Limpa a tela.
2. Consulta os dados das compras, combinando informações das tabelas compra e livro.
3. Verifica se há registros. Se não houver, exibe uma mensagem de erro e retorna ao menu.
4. Exibe os dados de todas as compras registradas em um formato organizado.
5. Mostra uma mensagem de sucesso ao concluir a listagem.
6. Retorna ao menu principal.

Essa função é útil para visualizar o histórico de compras de livros, permitindo ao usuário conferir detalhes como o título do livro, a quantidade comprada, a data da compra e o custo total.

Vender Livros

Aqui está a explicação detalhada da função `VenderLivros()`, que tem como objetivo **registrar a venda de livros e atualizar o estoque**. Vamos analisar cada parte do código para facilitar o entendimento:

1. Limpar a tela

```
os.system("cls")
```

- `os.system("cls")`: Limpa a tela do terminal (comum em sistemas Windows).
-

2. Verificar disponibilidade de livros

```
cursor.execute("SELECT * FROM livro")
```

```
disponibilidade = cursor.fetchall()
```

```
if len(disponibilidade) == 0:
```

```
    print("\033[31mNão há livros cadastrados para a venda.\033[m")
```

```
    input()
```

```
    Menu()
```

- Consulta a tabela `livro` para verificar se há livros cadastrados.
 - Se não houver livros (`len(disponibilidade) == 0`), exibe uma mensagem de erro em vermelho (`\033[31m`) e retorna ao `Menu()`.
-

3. Solicitar o ID do livro

```
id_livro = input("Insira o ID do livro a ser vendido: ")
```

```
if id_livro.lower() == "sair":
```

```
    Menu()
```

```
if not id_livro:
```

```
    VenderLivros()
```

- Solicita ao usuário o **ID do livro** que será vendido.
 - Se o usuário digitar "sair", a função retorna ao `Menu()`.
 - Se o campo estiver vazio (`if not id_livro`), a função se chama novamente (`VenderLivros()`).
-

4. Verificar se o livro existe

```
cursor.execute(f"SELECT * FROM livro WHERE id = {id_livro}")
```

```
resultado = cursor.fetchall()
```

```
if len(resultado) == 0:
```

```
    print(f"\033[31mNão há livro registrado com o id {id_livro}\033[m")
```

```
    input()
```

```
    VenderLivros()
```

- Executa uma consulta SQL para verificar se o livro com o ID fornecido existe na tabela livro.
 - Se não houver resultados (`len(resultado) == 0`), exibe uma mensagem de erro em vermelho (`\033[31m`) e chama novamente a função `VenderLivros()`.
-

5. Recuperar informações do livro

```
for linha in resultado:
```

```
    id = linha[0]
```

```
    titulo = linha[1]
```

```
    valor_compra = linha[6]
```

```
    valor_venda = linha[7]
```

- Itera sobre o resultado da consulta (mesmo que haja apenas uma linha) e armazena:
 - **ID do livro:** `id = linha[0]`.
 - **Título do livro:** `titulo = linha[1]`.
 - **Valor de compra:** `valor_compra = linha[6]`.
 - **Valor de venda:** `valor_venda = linha[7]`.
-

6. Verificar estoque do livro

```
cursor.execute(f"SELECT quantidade FROM estoque WHERE id_livro = {id}")
```

```
estoqueBolado = cursor.fetchall()
```

```
for linha in estoqueBolado:
```

```
    quantidadeEstocada = linha[0]
```

```
if quantidadeEstocada == 0:
```

```
    print(f"\033[31mNão há cópias do livro {titulo} disponíveis a serem vendidas.\033[m")
```

```
input()
```

```
Menu()
```

- Consulta a tabela estoque para verificar a quantidade disponível do livro (quantidadeEstocada).
 - Se não houver cópias disponíveis (quantidadeEstocada == 0), exibe uma mensagem de erro em vermelho (\033[31m) e retorna ao Menu().
-

7. Solicitar o ID do cliente

```
id_cliente = input(f"Insira o ID do cliente que irá comprar o livro {titulo}: ")
```

```
if id_cliente.lower() == "sair":
```

```
Menu()
```

```
cursor.execute(f"SELECT * FROM cliente WHERE id = {id_cliente}")
```

```
resultado2 = cursor.fetchall()
```

```
for linha in resultado2:
```

```
nome = linha[1]
```

```
if len(resultado2) == 0:
```

```
print(f"\033[31mNão há cliente registrado com o id {id_cliente}\033[m")
```

```
input()
```

```
VenderLivros()
```

- Solicita ao usuário o **ID do cliente** que está comprando o livro.
 - Se o usuário digitar "sair", a função retorna ao Menu().
 - Verifica se o cliente existe na tabela cliente.
 - Se não houver cliente (len(resultado2) == 0), exibe uma mensagem de erro em vermelho (\033[31m) e chama novamente a função VenderLivros().
-

8. Solicitar a quantidade de cópias

```
print(f"Insira quantas cópias do livro {titulo} o senhor(a) {nome} irá comprar:")
```

```
try:
```

```
quantidade = int(input())
```

```
except ValueError:
```

```
print("\033[31mDado inválido.\033[m")
```

```
input()
```

```
VenderLivros()
```

```
if quantidade > quantidadeEstocada:
```

```
    quantidade = quantidadeEstocada
```

- Solicita ao usuário a **quantidade de cópias** do livro que serão vendidas.
 - Se o usuário digitar um valor não numérico, exibe uma mensagem de erro em vermelho (\033[31m) e chama novamente a função VenderLivros().
 - Se a quantidade solicitada for maior que a quantidade em estoque, ajusta a quantidade para o valor disponível.
-

9. Solicitar a data da venda

```
data_venda = input("Registre a data da venda. (AAAA/MM/DD): ")
```

```
if len(data_venda) != 10:
```

```
    print("\033[31m data precisa ter 10 caracteres.\033[m")
```

```
    input()
```

```
    VenderLivros()
```

- Solicita a **data da venda** no formato AAAA/MM/DD.
 - Se a data não tiver exatamente 10 caracteres, exibe uma mensagem de erro em vermelho (\033[31m) e chama novamente a função VenderLivros().
-

10. Calcular o lucro

```
lucro = (valor_venda - valor_compra) * quantidade
```

- Calcula o **lucro** da venda subtraindo o valor de compra do valor de venda e multiplicando pela quantidade vendida.
-

11. Atualizar o estoque e registrar a venda

```
novaQuantidade = quantidadeEstocada - quantidade
```

```
try:
```

```
    cursor.execute(f"INSERT INTO venda (data_venda, quantidade, id_cliente, id_livro, lucro)  
VALUES ('{data_venda}','{quantidade}','{id_cliente}','{id_livro}','{lucro}')
```

```
    conexao.commit()
```

```
    cursor.execute(f"UPDATE estoque SET quantidade = {novaQuantidade} WHERE id_livro =  
{id_livro}")
```

```
    conexao.commit()
```

```
print("\033[32mVenda registrada com sucesso.\033[m")
```

```
input()
```

```
except:
```

```
print("\033[32mNão foi possível registrar a venda. verifique os dados e tente novamente.\033[m")
```

```
input()
```

- **Passo 1:** Calcula a nova quantidade em estoque ($\text{novaQuantidade} = \text{quantidadeEstocada} - \text{quantidade}$).
- **Passo 2:** Insere os dados da venda na tabela venda, incluindo:
 - Data da venda (`data_venda`).
 - Quantidade vendida (`quantidade`).
 - ID do cliente (`id_cliente`).
 - ID do livro (`id_livro`).
 - Lucro (`lucro`).
- **Passo 3:** Atualiza a quantidade do livro no estoque (`UPDATE estoque`).
- Se tudo der certo, exibe uma mensagem de sucesso em verde (`\033[32m`).
- Se ocorrer algum erro, exibe uma mensagem de erro em vermelho (`\033[31m`).

12. Retorno ao menu

```
Menu()
```

- Retorna ao `Menu()` principal do sistema.

Resumo da funcionalidade

A função `VenderLivros()`:

1. Verifica se há livros cadastrados.
2. Solicita o ID do livro e verifica se ele existe.
3. Recupera o título, valor de compra e valor de venda do livro.
4. Verifica se há cópias disponíveis em estoque.
5. Solicita o ID do cliente e verifica se ele existe.
6. Solicita a quantidade de cópias e ajusta se necessário.
7. Solicita a data da venda.
8. Calcula o lucro da venda.

9. Atualiza o estoque e registra a venda no banco de dados.

10. Exibe mensagens de sucesso ou erro.

11. Retorna ao menu principal.

Essa função é útil para gerenciar as vendas de livros, atualizando o estoque e mantendo um registro das transações, incluindo o lucro obtido.

Listar Vendas

Aqui está a explicação detalhada da função ListarVendas(), que tem como objetivo **listar todas as vendas registradas no sistema**. Vamos analisar cada parte do código para facilitar o entendimento:

1. Limpar a tela

```
os.system("cls")
```

- `os.system("cls")`: Limpa a tela do terminal (comum em sistemas Windows).

2. Consulta ao banco de dados

```
cursor.execute("SELECT v.id, l.id, c.id, l.titulo, c.nome, v.quantidade, v.lucro FROM venda AS v JOIN livro AS l ON v.id_livro = l.id JOIN cliente AS c ON v.id_cliente = c.id")
```

```
resultado = cursor.fetchall()
```

- Executa uma consulta SQL que seleciona dados das tabelas venda, livro e cliente:
 - **Tabela venda (v)**: Recupera o id, quantidade e lucro da venda.
 - **Tabela livro (l)**: Recupera o id e titulo do livro.
 - **Tabela cliente (c)**: Recupera o id e nome do cliente.
 - A junção (JOIN) é feita pelo id_livro da venda e o id do livro (`v.id_livro = l.id`), e pelo id_cliente da venda e o id do cliente (`v.id_cliente = c.id`).
- `cursor.fetchall()`: Recupera todos os resultados da consulta e armazena na variável resultado.

3. Verificação de registros

```
if len(resultado) == 0:
```

```
    print("\033[31mNão há registro.\033[m")
```

```
    input()
```

```
    Menu()
```

- Verifica se a lista de resultados (resultado) está vazia.
- Se não houver registros, exibe uma mensagem de erro em vermelho (`\033[31m`) e retorna ao Menu().

4. Exibição das vendas

```

print("="*50)

print("LISTA DE VENDAS")

print("="*50)

for linha in resultado:

    v_id = linha[0]

    l_id = linha[1]

    c_id = linha[2]

    l_titulo = linha[3]

    c_nome = linha[4]

    v_quantidade = linha[5]

    v_lucro = linha[6]

    print(f"ID da venda: {v_id}")

    print(f"ID do livro: {l_id}")

    print(f"ID do cliente: {c_id}")

    print(f"Livro: {l_titulo}")

    print(f"Cliente: {c_nome}")

    print(f"Cópias vendidas: {v_quantidade}")

    print(f"Lucro: {v_lucro}R$\n")

    print("-"*50)

```

- Exibe um cabeçalho formatado com = para destacar o título "LISTA DE VENDAS".
 - Itera sobre cada linha (linha) da lista resultado e exibe os seguintes dados de cada venda:
 - **ID da venda:** Identificador único da venda.
 - **ID do livro:** Identificador único do livro vendido.
 - **ID do cliente:** Identificador único do cliente que comprou o livro.
 - **Livro:** Título do livro vendido.
 - **Cliente:** Nome do cliente que comprou o livro.
 - **Cópias vendidas:** Quantidade de cópias vendidas.
 - **Lucro:** Lucro obtido com a venda.
 - Após exibir os dados de uma venda, insere uma linha de separação (-).
-

5. Mensagem de conclusão

```
print("\033[32mListagem concluída com sucesso.\033[m")
```

- Exibe uma mensagem de sucesso em verde (\033[32m) indicando que a listagem foi concluída.
-

6. Retorno ao menu

```
input()
```

```
Menu()
```

- Aguarda o usuário pressionar Enter (input()).
 - Retorna ao Menu() principal do sistema.
-

Resumo da funcionalidade

A função ListarVendas():

1. Limpa a tela.
2. Consulta os dados das vendas, combinando informações das tabelas venda, livro e cliente.
3. Verifica se há registros. Se não houver, exibe uma mensagem de erro e retorna ao menu.
4. Exibe os dados de todas as vendas registradas em um formato organizado.
5. Mostra uma mensagem de sucesso ao concluir a listagem.
6. Retorna ao menu principal.

Essa função é útil para visualizar o histórico de vendas, permitindo ao usuário conferir detalhes como o livro vendido, o cliente que realizou a compra, a quantidade de cópias vendidas e o lucro obtido.

Remover Livros

Aqui está a explicação detalhada da função `RemoverLivro()`, que tem como objetivo **remover um livro do banco de dados**. Vamos analisar cada parte do código para facilitar o entendimento:

1. Limpar a tela

```
os.system("cls")
```

- `os.system("cls")`: Limpa a tela do terminal (comum em sistemas Windows).

2. Solicitar o ID do livro

```
print("Insira o ID do livro a ser removido:")
```

```
id = input()
```

- Solicita ao usuário o **ID do livro** que será removido.

3. Verificar se o campo está vazio

```
if not id:
```

```
    RemoverLivro()
```

- Se o campo estiver vazio (`if not id`), a função se chama novamente (`RemoverLivro()`).

4. Verificar se o usuário deseja sair

```
if id.lower() == "sair":
```

```
    Menu()
```

- Se o usuário digitar "sair", a função retorna ao `Menu()`.

5. Verificar se o livro existe

```
cursor.execute(f"SELECT id, titulo, autor FROM livro WHERE id = {id}")
```

```
resultado = cursor.fetchall()
```

```
if len(resultado) == 0:
```

```
    print(f"\033[31mERRO. Nenhum registro encontrado com o id {id}\033[m")
```

```
    input()
```

```
return RemoverLivro()
```

- Executa uma consulta SQL para verificar se o livro com o ID fornecido existe na tabela livro.
 - Se não houver resultados (`len(resultado) == 0`), exibe uma mensagem de erro em vermelho (`\033[31m`) e chama novamente a função `RemoverLivro()`.
-

6. Exibir os dados do livro

```
print("\033[32mRegistro encontrado !\033[m")
```

```
print("-"*100)
```

```
print(f'{'ID':5}{ 'Título':30}{ 'Autor'}')
```

```
print("-"*100)
```

```
for linha in resultado:
```

```
    id = linha[0]
```

```
    titulo = linha[1]
```

```
    autor = linha[2]
```

```
    print(f"{'id':<5}{ 'titulo':20}{ 'autor':21}")
```

```
    print("-"*100)
```

- Exibe uma mensagem de sucesso em verde (`\033[32m`) indicando que o registro foi encontrado.
 - Formata e exibe os dados do livro em uma tabela:
 - **ID:** Identificador único do livro.
 - **Título:** Título do livro.
 - **Autor:** Autor do livro.
 - Insere uma linha de separação (-) após exibir os dados.
-

7. Confirmar a exclusão

```
print("Deseja realmente \033[31mexcluir\033[m esse registro ? S / N:")
```

```
opcao = input()
```

- Pergunta ao usuário se ele deseja realmente excluir o registro, destacando a palavra "excluir" em vermelho (`\033[31m`).
-

8. Excluir o livro

```
if opcao.lower() == "s":
```

```
    try:
```

```
        cursor.execute(f"DELETE FROM livro WHERE id = {id}")
```

```
        conexao.commit()
```

```
        print("\033[32mRegistro deletado com sucesso.\033[m")
```

```
    input()
```

```
    Menu()
```

```
    except:
```

```
        print("\031[32mNão foi possível deletar o cadastro por um erro sinistro.\031[m")
```

```
        input("Tecle Enter para retornar ao menu: ")
```

```
    Menu()
```

- Se o usuário confirmar a exclusão (opcao.lower() == "s"):
 - Tenta executar o comando SQL para excluir o livro da tabela livro.
 - Se a exclusão for bem-sucedida, exibe uma mensagem de sucesso em verde (\033[32m).
 - Se ocorrer um erro, exibe uma mensagem de erro em vermelho (\033[31m) e retorna ao Menu().
-

9. Retorno ao menu

```
else:
```

```
    Menu()
```

- Se o usuário não confirmar a exclusão (opcao.lower() != "s"), a função retorna ao Menu().
-

Resumo da funcionalidade

A função `RemoverLivro()`:

1. Limpa a tela.
2. Solicita o ID do livro a ser removido.
3. Verifica se o campo está vazio ou se o usuário deseja sair.
4. Verifica se o livro existe no banco de dados.
5. Exibe os dados do livro em um formato organizado.
6. Pergunta ao usuário se ele deseja realmente excluir o registro.

7. Tenta excluir o livro do banco de dados e exibe mensagens de sucesso ou erro.
8. Retorna ao menu principal.

Essa função é útil para gerenciar o catálogo de livros, permitindo ao usuário remover registros de forma segura e confirmada.

Atualizar Livros

A função `AtualizarLivro()` é responsável por atualizar as informações de um livro em um banco de dados. Vamos detalhar cada parte da função para facilitar o entendimento:

1. Limpeza da Tela e Entrada do ID do Livro

```
os.system("cls")
```

```
print("Insira o ID do livro a ser atualizado:")
```

```
id = input()
```

- **`os.system("cls")`**: Limpa a tela do console para uma nova interação.
- **`print("Insira o ID do livro a ser atualizado:")`**: Solicita ao usuário que insira o ID do livro que deseja atualizar.
- **`id = input()`**: Captura o ID inserido pelo usuário.

2. Verificação do ID

```
if not id:
```

```
    AtualizarLivro()
```

```
if id.lower() == "sair":
```

```
    Menu()
```

- **`if not id::`** Se o usuário não inserir nenhum ID, a função chama a si mesma novamente (`AtualizarLivro()`), solicitando o ID novamente.
- **`if id.lower() == "sair":`** Se o usuário digitar "sair", a função redireciona para o menu principal (`Menu()`).

3. Busca do Livro no Banco de Dados

```
cursor.execute(f"SELECT * FROM livro WHERE id = {id}")
```

```
resultado = cursor.fetchall()
```

- **`cursor.execute(...)`**: Executa uma consulta SQL para buscar o livro com o ID fornecido.
- **`resultado = cursor.fetchall()`**: Armazena o resultado da consulta.

4. Verificação do Resultado da Busca

```
if len(resultado) == 0:
```

```
    print(f"\033[mERRO. Nenhum registro encontrado com o id {id}\033[m")
```

```
    input()
```

```
    return AtualizarLivro()
```

- **if len(resultado) == 0::** Se nenhum livro for encontrado com o ID fornecido, exibe uma mensagem de erro e chama a função novamente.

5. Exibição dos Dados do Livro

```
print("="*50)

print("\033[32mRegistro encontrado !\033[m")

print("="*50)

for linha in resultado:

    print(f"ID: {linha[0]}")

    print(f"Título: {linha[1]}")

    print(f"Gênero: {linha[2]}")

    print(f"Autor: {linha[3]}")

    print(f"Publicação: {linha[4]}")

    print(f"Sinopse: {linha[5]}")

    print(f"Preço de compra: {linha[6]:.2f}R$")

    print(f"Preço de Venda: {linha[7]:.2f}R$")

print("-"*50)
```

- **for linha in resultado::** Itera sobre os dados do livro encontrado e exibe cada campo (ID, Título, Gênero, Autor, etc.).

6. Entrada dos Novos Dados

```
novoTitulo = input("Novo título: ")

novoGenero = input("Novo Gênero: ")

novoAutor = input("Novo autor: ")

novaPublicacao = input("Nova data de publicação (AAAA/MM/DD): ")

novaSinopse = input("Nova sinopse: ")

try:

    novoValorCompra = float(input("Novo preço de compra: "))

    novoValorVenda = float(input("Novo preço de revenda: "))

except ValueError:

    print(f"\033[31mNenhum número inserido.\033[m")

    input()

return AtualizarLivro()
```

- **novoTitulo = input("Novo título: ")**: Solicita e captura o novo título do livro.

- **novoGenero = input("Novo Gênero: ")**: Solicita e captura o novo gênero do livro.
- **novoAutor = input("Novo autor: ")**: Solicita e captura o novo autor do livro.
- **novaPublicacao = input("Nova data de publicação (AAAA/MM/DD): ")**: Solicita e captura a nova data de publicação.
- **novaSinopse = input("Nova sinopse: ")**: Solicita e captura a nova sinopse do livro.
- **try::** Tenta capturar os novos preços de compra e revenda como números decimais.
- **except ValueError::** Se o usuário não inserir números válidos, exibe uma mensagem de erro e chama a função novamente.

7. Validação dos Dados

```
if not novoTitulo or not novoGenero or not novoAutor or not novaPublicacao or not
novaSinopse or not novoValorCompra or not novoValorVenda or len(novaPublicacao) != 10
or novoValorCompra < 0 or novoValorVenda < 0:
```

```
os.system("cls")
```

```
print("\033[31mDados inválidos. Verifique os dados e tente novamente.\033[m")
```

```
input()
```

```
Menu()
```

- **if not ...:** Verifica se todos os campos foram preenchidos corretamente e se os valores são válidos (por exemplo, datas no formato correto e preços positivos).
- **os.system("cls"):** Limpa a tela.
- **print("\033[31mDados inválidos. Verifique os dados e tente novamente.\033[m"):** Exibe uma mensagem de erro se os dados forem inválidos.
- **Menu():** Redireciona para o menu principal.

8. Confirmação da Atualização

```
print("\033[33mCONFIRMAÇÃO:\033[m Deseja realmente Atualizar o registro atual ? S/N")
```

```
confirmacao = input()
```

```
if confirmacao.lower() == "s":
```

```
try:
```

```
cursor.execute(f"UPDATE livro SET titulo = '{novoTitulo}', genero = '{novoGenero}', autor
= '{novoAutor}', publicacao = '{novaPublicacao}', sinopse = '{novaSinopse}', valor_Compra
= '{novoValorCompra}', valor_revenda = '{novoValorVenda}' WHERE (id = {id})")
```

```
conexao.commit()
```

```
print("\033[32mRegistro Atualizado com sucesso.\033[m")
```

```
except:
```



```
print("\033[31mNão foi possível atualizar o registro devido a um erro sinistro.\033[m")
```

```
else:
```

```
os.system("cls")
```

```
print("\033[32mAtualização cancelada.\033[m")
```

- **print("\033[33mCONFIRMAÇÃO:\033[m Deseja realmente Atualizar o registro atual ? S/N"):** Solicita confirmação do usuário para atualizar o livro.
- **if confirmacao.lower() == "s":** Se o usuário confirmar (digitando "S"), a função tenta atualizar o livro no banco de dados.
- **cursor.execute(...):** Executa o comando SQL para atualizar os dados do livro.
- **conexao.commit():** Confirma a transação no banco de dados.
- **print("\033[32mRegistro Atualizado com sucesso.\033[m"):** Exibe uma mensagem de sucesso.
- **except::** Se ocorrer um erro durante a atualização, exibe uma mensagem de erro.
- **else::** Se o usuário não confirmar a atualização, exibe uma mensagem de cancelamento.

9. Retorno ao Menu Principal

```
input()
```

```
Menu()
```

- **input():** Aguarda uma entrada do usuário (para que ele possa ver a mensagem de sucesso ou erro antes de continuar).
- **Menu():** Retorna ao menu principal após a atualização ou cancelamento.

Resumo

A função `AtualizarLivro()` permite ao usuário atualizar as informações de um livro no banco de dados. Ela solicita o ID do livro, busca os dados atuais, permite a entrada de novos dados, valida essas entradas e, após confirmação, atualiza o registro no banco de dados. Se algo der errado, a função informa o erro e permite ao usuário tentar novamente ou cancelar a operação.

Cadastrar Cliente

A função `CadastrarCliente()` é responsável por cadastrar um novo cliente em um banco de dados. Vamos detalhar cada parte da função para facilitar o entendimento:

1. Limpeza da Tela e Entrada dos Dados do Cliente

```
os.system("cls")
```

```
print("Insira os dados do novo cliente:")
```

```
nome = input("Nome: ")
```

- **`os.system("cls")`**: Limpa a tela do console para uma nova interação.
- **`print("Insira os dados do novo cliente:")`**: Solicita ao usuário que insira os dados do novo cliente.
- **`nome = input("Nome: ")`**: Captura o nome do cliente inserido pelo usuário.

2. Verificação de Saída

```
if nome.lower() == "sair":
```

```
    Menu()
```

```
telefone = input("Telefone (11 dígitos): ")
```

```
if telefone.lower() == "sair":
```

```
    Menu()
```

```
email = input("E-mail: ")
```

```
if email.lower() == "sair":
```

```
    Menu()
```

- **`if nome.lower() == "sair":`**: Se o usuário digitar "sair" no campo do nome, a função redireciona para o menu principal (`Menu()`).
- **`telefone = input("Telefone (11 dígitos): ")`**: Captura o número de telefone do cliente inserido pelo usuário.
- **`if telefone.lower() == "sair":`**: Se o usuário digitar "sair" no campo do telefone, a função redireciona para o menu principal (`Menu()`).
- **`email = input("E-mail: ")`**: Captura o e-mail do cliente inserido pelo usuário.
- **`if email.lower() == "sair":`**: Se o usuário digitar "sair" no campo do e-mail, a função redireciona para o menu principal (`Menu()`).

3. Validação dos Dados

```
if not nome or not telefone or not email or len(telefone) != 11:
```

```
    print("\033[31mDados inválidos. Verifique os dados e tente novamente.\033[m")
```

```
input()
```

```
return CadastrarCliente()
```

- **if not nome or not telefone or not email or len(telefone) != 11::** Verifica se todos os campos foram preenchidos e se o telefone tem exatamente 11 dígitos.
- **print("\033[31mDados inválidos. Verifique os dados e tente novamente.\033[m"):** Exibe uma mensagem de erro se os dados forem inválidos.
- **input():** Aguarda uma entrada do usuário (para que ele possa ver a mensagem de erro antes de continuar).
- **return CadastrarCliente():** Chama a função novamente para permitir que o usuário insira os dados corretamente.

4. Inserção dos Dados no Banco de Dados

```
try:
```

```
cursor.execute(f"INSERT INTO cliente (nome, telefone, email) VALUES ('{nome}',  
'{telefone}', '{email}')
```

```
conexao.commit()
```

```
print("\033[32mCadastro registrado com sucesso.\033[m")
```

```
except:
```

```
print("\033[31mNão foi possível registrar o cadastro. Verifique os dados e tente  
novamente.\033[m")
```

- **try::** Tenta inserir os dados do cliente no banco de dados.
- **cursor.execute(...):** Executa o comando SQL para inserir os dados do cliente na tabela cliente.
- **conexao.commit():** Confirma a transação no banco de dados.
- **print("\033[32mCadastro registrado com sucesso.\033[m"):** Exibe uma mensagem de sucesso se o cadastro for realizado com sucesso.
- **except::** Se ocorrer um erro durante a inserção, exibe uma mensagem de erro.

5. Retorno ao Menu Principal

```
input()
```

```
Menu()
```

- **input():** Aguarda uma entrada do usuário (para que ele possa ver a mensagem de sucesso ou erro antes de continuar).
- **Menu():** Retorna ao menu principal após o cadastro ou erro.

Resumo

A função `CadastrarCliente()` permite ao usuário cadastrar um novo cliente no banco de dados. Ela solicita o nome, telefone e e-mail do cliente, valida esses dados e, se

estiverem corretos, insere o novo cliente no banco de dados. Se algo der errado, a função informa o erro e permite ao usuário tentar novamente. A função também permite ao usuário sair para o menu principal a qualquer momento digitando "sair".

Listar Clientes

A função `ListarClientes()` é responsável por exibir todos os clientes cadastrados em um banco de dados. Vamos detalhar cada parte da função para facilitar o entendimento:

1. Limpeza da Tela

```
os.system("cls")
```

- `os.system("cls")`: Limpa a tela do console para uma nova interação.

2. Consulta ao Banco de Dados

```
cursor.execute("SELECT * FROM cliente")
```

```
resultado = cursor.fetchall()
```

- `cursor.execute("SELECT * FROM cliente")`: Executa uma consulta SQL para buscar todos os registros da tabela cliente.
- `resultado = cursor.fetchall()`: Armazena o resultado da consulta em uma variável chamada resultado.

3. Verificação de Registros

```
if len(resultado) == 0:
```

```
    print("\033[31mNão há registros.\033[m")
```

```
    input()
```

```
    Menu()
```

- `if len(resultado) == 0`:: Verifica se a lista de resultados está vazia, ou seja, se não há clientes cadastrados.
- `print("\033[31mNão há registros.\033[m")`: Exibe uma mensagem informando que não há registros.
- `input()`: Aguarda uma entrada do usuário (para que ele possa ver a mensagem antes de continuar).
- `Menu()`: Redireciona para o menu principal.

4. Exibição dos Dados dos Clientes

```
print("="*50)
```

```
print("Clientes cadastrados")
```

```
print("="*50)
```

```
for linha in resultado:
```

```
    print(f"ID: {linha[0]}")
```

```
    print(f"Nome: {linha[1]}")
```

```
print(f"Telefone: {linha[2]}")
```

```
print(f"E_mail: {linha[3]}\n")
```

```
print("-"*50)
```

- **print("-"*50):** Exibe uma linha de separação para organizar a visualização.
- **print("Clientes cadastrados"):** Exibe um título indicando que os clientes cadastrados serão listados.
- **for linha in resultado::** Itera sobre cada registro (linha) da lista de resultados.
- **print(f"ID: {linha[0]}"):** Exibe o ID do cliente.
- **print(f"Nome: {linha[1]}"):** Exibe o nome do cliente.
- **print(f"Telefone: {linha[2]}"):** Exibe o telefone do cliente.
- **print(f"E_mail: {linha[3]}\n"):** Exibe o e-mail do cliente.
- **print("-"*50):** Exibe uma linha de separação entre os clientes.

5. Mensagem de Conclusão

```
print("\033[32mListagem Concluída com sucesso.\033[m")
```

- **print("\033[32mListagem Concluída com sucesso.\033[m"):** Exibe uma mensagem indicando que a listagem foi concluída com sucesso.

6. Retorno ao Menu Principal

```
input()
```

```
Menu()
```

- **input():** Aguarda uma entrada do usuário (para que ele possa ver a listagem e a mensagem de conclusão antes de continuar).
- **Menu():** Retorna ao menu principal após a listagem.

Resumo

A função `ListarClientes()` busca todos os clientes cadastrados no banco de dados e os exibe de forma organizada na tela. Se não houver clientes cadastrados, a função informa isso ao usuário e retorna ao menu principal. Após exibir a listagem, a função informa que a operação foi concluída com sucesso e retorna ao menu principal. Essa função é útil para visualizar rapidamente todos os clientes registrados no sistema.

Atualizar Cliente

A função `AtualizarCliente()` é responsável por atualizar as informações de um cliente já cadastrado no banco de dados. Vamos detalhar cada parte da função para facilitar o entendimento:

1. Limpeza da Tela e Entrada do ID do Cliente

```
os.system("cls")
```

```
print("Insira o ID do cliente a ser atualizado:")
```

```
id = input()
```

- **`os.system("cls")`**: Limpa a tela do console para uma nova interação.
- **`print("Insira o ID do cliente a ser atualizado:")`**: Solicita ao usuário que insira o ID do cliente que deseja atualizar.
- **`id = input()`**: Captura o ID inserido pelo usuário.

2. Verificação de Entrada Vazia ou Saída

```
if not id:
```

```
    AtualizarCliente()
```

```
if id.lower == "sair":
```

```
    Menu()
```

- **`if not id::`** Se o ID estiver vazio, a função chama a si mesma novamente para solicitar o ID.
- **`if id.lower == "sair":`** Se o usuário digitar "sair", a função redireciona para o menu principal (`Menu()`).

3. Busca do Cliente no Banco de Dados

```
cursor.execute(f"SELECT * FROM cliente WHERE id = {id}")
```

```
resultado = cursor.fetchall()
```

- **`cursor.execute(f"SELECT * FROM cliente WHERE id = {id}")`**: Executa uma consulta SQL para buscar o cliente com o ID especificado.
- **`resultado = cursor.fetchall()`**: Armazena o resultado da consulta em uma variável chamada `resultado`.

4. Verificação de Registro Encontrado

```
if len(resultado) == 0:
```

```
    print(f"\033[31mERRO. Nenhum registro encontrado com o id {id}\033[m")
```

```
    input()
```

AtualizarCliente()

- **if len(resultado) == 0::** Verifica se a lista de resultados está vazia, ou seja, se nenhum cliente foi encontrado com o ID especificado.
- **print(f"\033[31mERRO. Nenhum registro encontrado com o id {id}\033[m"):** Exibe uma mensagem de erro informando que nenhum registro foi encontrado.
- **input():** Aguarda uma entrada do usuário (para que ele possa ver a mensagem de erro antes de continuar).
- **AtualizarCliente():** Chama a função novamente para permitir que o usuário insira um ID válido.

5. Exibição dos Dados do Cliente Encontrado

```
print("\033[32mRegistro encontrado !\033[m")
```

```
print("-"*100)
```

```
print(f"{'ID':5}{ 'Nome':20}{ 'Telefone':21}{ 'E_mail'}")
```

```
print("-"*100)
```

```
for linha in resultado:
```

```
    id = linha[0]
```

```
    nome = linha[1]
```

```
    telefone = linha[2]
```

```
    email = linha[3]
```

```
    print(f"{'id':<5}{nome:20}{telefone:21}{email}")
```

```
    print("-"*100)
```

- **print("\033[32mRegistro encontrado !\033[m"):** Exibe uma mensagem informando que o registro foi encontrado.
- **print("-"*100):** Exibe uma linha de separação para organizar a visualização.
- **print(f"{'ID':5}{ 'Nome':20}{ 'Telefone':21}{ 'E_mail'}"):** Exibe os cabeçalhos das colunas (ID, Nome, Telefone, E-mail).
- **for linha in resultado::** Itera sobre os dados do cliente encontrado.
- **id = linha[0]:** Armazena o ID do cliente.
- **nome = linha[1]:** Armazena o nome do cliente.
- **telefone = linha[2]:** Armazena o telefone do cliente.
- **email = linha[3]:** Armazena o e-mail do cliente.
- **print(f"{'id':<5}{nome:20}{telefone:21}{email}"):** Exibe os dados do cliente formatados.
- **print("-"*100):** Exibe uma linha de separação entre os dados.

6. Entrada dos Novos Dados

```
newNome = input("Novo nome: ")
```

```
newTelefone = input("Novo Telefone (11 dígitos): ")
```

```
newEmail = input("Novo E-mail: ")
```

- **newNome = input("Novo nome: ")**: Solicita e captura o novo nome do cliente.
- **newTelefone = input("Novo Telefone (11 dígitos): ")**: Solicita e captura o novo telefone do cliente.
- **newEmail = input("Novo E-mail: ")**: Solicita e captura o novo e-mail do cliente.

7. Validação dos Dados

```
if not newNome or not newTelefone or not newEmail or len(newTelefone) != 11:
```

```
    print("\033[31mDados inválidos. O registro não foi atualizado.\033[m")
```

```
    input()
```

```
    Menu()
```

- **if not newNome or not newTelefone or not newEmail or len(newTelefone) != 11::** Verifica se todos os campos foram preenchidos corretamente e se o telefone tem exatamente 11 dígitos.
- **print("\033[31mDados inválidos. O registro não foi atualizado.\033[m")**: Exibe uma mensagem de erro se os dados forem inválidos.
- **input()**: Aguarda uma entrada do usuário (para que ele possa ver a mensagem de erro antes de continuar).
- **Menu()**: Redireciona para o menu principal.

8. Confirmação da Atualização

```
print("\033[33mCONFIRMAÇÃO:\033[m Deseja realmente Atualizar o registro atual ? S/N")
```

```
confirmacao = input()
```

```
if confirmacao.lower() == "s":
```

```
    try:
```

```
        cursor.execute(f"UPDATE cliente SET nome = '{newNome}', telefone = '{newTelefone}',  
email = '{newEmail}' WHERE (id = {id})")
```

```
        conexao.commit()
```

```
        print("\033[32mRegistro Atualizado com sucesso.\033[m")
```

```
    except:
```

```
        print("\033[31mNão foi possível atualizar o registro devido a um erro sinistro.\033[m")
```

```
    else:
```

```
os.system("cls")
```

```
print("\033[32mAtualização cancelada.\033[m")
```

- **print("\033[33mCONFIRMAÇÃO:\033[m Deseja realmente Atualizar o registro atual ? S/N"):** Solicita confirmação do usuário para atualizar o cliente.
- **if confirmacao.lower() == "s":** Se o usuário confirmar (digitando "S"), a função tenta atualizar o cliente no banco de dados.
- **cursor.execute(...):** Executa o comando SQL para atualizar os dados do cliente.
- **conexao.commit():** Confirma a transação no banco de dados.
- **print("\033[32mRegistro Atualizado com sucesso.\033[m"):** Exibe uma mensagem de sucesso.
- **except::** Se ocorrer um erro durante a atualização, exibe uma mensagem de erro.
- **else::** Se o usuário não confirmar a atualização, exibe uma mensagem de cancelamento.

9. Retorno ao Menu Principal

```
input()
```

```
Menu()
```

- **input():** Aguarda uma entrada do usuário (para que ele possa ver a mensagem de sucesso ou erro antes de continuar).
- **Menu():** Retorna ao menu principal após a atualização ou cancelamento.

Resumo

A função `AtualizarCliente()` permite ao usuário atualizar as informações de um cliente já cadastrado no banco de dados. Ela solicita o ID do cliente, busca os dados atuais, permite a entrada de novos dados, valida essas entradas e, após confirmação, atualiza o registro no banco de dados. Se algo der errado, a função informa o erro e permite ao usuário tentar novamente ou cancelar a operação.

Adicionar Avaliações

A função `AdicionarAvaliacoes()` permite que um cliente cadastrado avalie um livro, adicionando uma nota e um comentário. Vamos detalhar cada parte da função para facilitar o entendimento:

1. Limpeza da Tela e Entrada do ID do Cliente

```
os.system("cls")
```

```
id_cliente = input("ID do cliente que irá avaliar: ")
```

- **`os.system("cls")`**: Limpa a tela do console para uma nova interação.
- **`id_cliente = input("ID do cliente que irá avaliar: ")`**: Solicita e captura o ID do cliente que fará a avaliação.

2. Verificação de Entrada Vazia ou Saída

```
if not id_cliente:
```

```
    AdicionarAvaliacoes()
```

```
if id_cliente.lower() == "sair":
```

```
    Menu()
```

- **`if not id_cliente::`** Se o ID do cliente estiver vazio, a função chama a si mesma novamente para solicitar o ID.
- **`if id_cliente.lower() == "sair":`** Se o usuário digitar "sair", a função redireciona para o menu principal (`Menu()`).

3. Busca do Cliente no Banco de Dados

```
cursor.execute(f"select nome from cliente where id = {id_cliente}")
```

```
resultado_cliente = cursor.fetchall()
```

- **`cursor.execute(f"select nome from cliente where id = {id_cliente}")`**: Executa uma consulta SQL para buscar o nome do cliente com o ID especificado.
- **`resultado_cliente = cursor.fetchall()`**: Armazena o resultado da consulta em uma variável chamada `resultado_cliente`.

4. Verificação de Cliente Encontrado

```
if len(resultado_cliente) == 0:
```

```
    print(f"Não há cliente cadastrado com o id {id_cliente}")
```

```
    input()
```

```
    return AdicionarAvaliacoes()
```

- **`if len(resultado_cliente) == 0::`** Verifica se a lista de resultados está vazia, ou seja, se nenhum cliente foi encontrado com o ID especificado.

- **print(f"Não há cliente cadastrado com o id {id_cliente}")**: Exibe uma mensagem informando que nenhum cliente foi encontrado.
- **input()**: Aguarda uma entrada do usuário (para que ele possa ver a mensagem antes de continuar).
- **return AdicionarAvaliacoes()**: Chama a função novamente para permitir que o usuário insira um ID válido.

5. Captura do Nome do Cliente

```
for linha in resultado_cliente:
```

```
    nome = linha[0]
```

- **for linha in resultado_cliente::** Itera sobre os dados do cliente encontrado.
- **nome = linha[0]**: Armazena o nome do cliente.

6. Entrada do ID do Livro

```
id_livro = input("ID do livro que será avaliado: ")
```

```
if id_livro.lower() == "sair":
```

```
    Menu()
```

- **id_livro = input("ID do livro que será avaliado: ")**: Solicita e captura o ID do livro que será avaliado.
- **if id_livro.lower() == "sair"::** Se o usuário digitar "sair", a função redireciona para o menu principal (Menu()).

7. Busca do Livro no Banco de Dados

```
cursor.execute(f"select titulo from livro where id = {id_livro}")
```

```
resultado_livro = cursor.fetchall()
```

- **cursor.execute(f"select titulo from livro where id = {id_livro}")**: Executa uma consulta SQL para buscar o título do livro com o ID especificado.
- **resultado_livro = cursor.fetchall()**: Armazena o resultado da consulta em uma variável chamada resultado_livro.

8. Verificação de Livro Encontrado

```
if len(resultado_livro) == 0:
```

```
    print(f"Não há livro registrado com o ID {id_livro}")
```

```
    input()
```

```
    return AdicionarAvaliacoes()
```

- **if len(resultado_livro) == 0::** Verifica se a lista de resultados está vazia, ou seja, se nenhum livro foi encontrado com o ID especificado.

- **print(f"Não há livro registrado com o ID {id_livro}"):** Exibe uma mensagem informando que nenhum livro foi encontrado.
- **input():** Aguarda uma entrada do usuário (para que ele possa ver a mensagem antes de continuar).
- **return AdicionarAvaliacoes():** Chama a função novamente para permitir que o usuário insira um ID válido.

9. Captura do Título do Livro

```
for linha in resultado_livro:
```

```
    titulo = linha[0]
```

- **for linha in resultado_livro::** Itera sobre os dados do livro encontrado.
- **titulo = linha[0]:** Armazena o título do livro.

10. Entrada da Nota

```
os.system("cls")
```

```
print(f"Qual a nota que o cliente {nome} dará ao livro {titulo} de 1 a 10 ?")
```

```
try:
```

```
    nota = int(input())
```

```
except ValueError:
```

```
    print("\033[31mA nota precisa ser um número entre 1 e 10.\033[m")
```

```
    input()
```

```
    return AdicionarAvaliacoes()
```

```
if not nota or nota > 10 or nota < 1:
```

```
    print("\033[31mA nota precisa ser um número entre 1 e 10.\033[m")
```

```
    input()
```

```
    return AdicionarAvaliacoes()
```

- **os.system("cls"):** Limpa a tela do console para uma nova interação.
- **print(f"Qual a nota que o cliente {nome} dará ao livro {titulo} de 1 a 10 ?"):** Solicita a nota que o cliente dará ao livro.
- **try::** Tenta capturar a nota como um número inteiro.
- **except ValueError::** Se a nota não for um número, exibe uma mensagem de erro.
- **if not nota or nota > 10 or nota < 1::** Verifica se a nota está dentro do intervalo válido (1 a 10).
- **print("\033[31mA nota precisa ser um número entre 1 e 10.\033[m"):** Exibe uma mensagem de erro se a nota for inválida.

- **input():** Aguarda uma entrada do usuário (para que ele possa ver a mensagem de erro antes de continuar).
- **return AdicionarAvaliacoes():** Chama a função novamente para permitir que o usuário insira uma nota válida.

11. Entrada do Comentário

```
os.system("cls")
```

```
print(f"Insira o comentário da avaliação:")
```

```
comentario = input()
```

- **os.system("cls"):** Limpa a tela do console para uma nova interação.
- **print(f"Insira o comentário da avaliação:"):** Solicita o comentário da avaliação.
- **comentario = input():** Captura o comentário inserido pelo usuário.

12. Inserção da Avaliação no Banco de Dados

```
try:
```

```
cursor.execute(f"insert into avaliacao (comentario, nota, id_livro, id_cliente) values  
( '{comentario}', '{nota}', '{id_livro}', '{id_cliente}')
```

```
conexao.commit()
```

```
print("\033[32mComentário adicionado com sucesso.\033[m")
```

```
except:
```

```
print("\033[31mNão foi possível adicionar o comentário em questão.\033[m")
```

- **try::** Tenta inserir a avaliação no banco de dados.
- **cursor.execute(...):** Executa o comando SQL para inserir o comentário, a nota, o ID do livro e o ID do cliente na tabela avaliacao.
- **conexao.commit():** Confirma a transação no banco de dados.
- **print("\033[32mComentário adicionado com sucesso.\033[m"):** Exibe uma mensagem de sucesso.
- **except::** Se ocorrer um erro durante a inserção, exibe uma mensagem de erro.

13. Retorno ao Menu Principal

```
input()
```

```
Menu()
```

- **input():** Aguarda uma entrada do usuário (para que ele possa ver a mensagem de sucesso ou erro antes de continuar).
- **Menu():** Retorna ao menu principal após a inserção da avaliação.

Resumo

A função `AdicionarAvaliacoes()` permite que um cliente cadastrado avalie um livro, adicionando uma nota e um comentário. Ela solicita o ID do cliente e do livro, verifica se ambos existem no banco de dados, permite a entrada da nota e do comentário, valida essas entradas e, após confirmação, insere a avaliação no banco de dados. Se algo der errado, a função informa o erro e permite ao usuário tentar novamente. Essa função é útil para coletar feedback dos clientes sobre os livros disponíveis.

Verificar Atualizações

A função `VerificarAvaliacoes()` é responsável por exibir todas as avaliações registradas no banco de dados, mostrando informações como o ID da avaliação, o nome do cliente, o título do livro, a nota e o comentário. Vamos detalhar cada parte da função para facilitar o entendimento:

1. Limpeza da Tela

```
os.system("cls")
```

- **`os.system("cls")`**: Limpa a tela do console para uma nova interação. Isso garante que a tela esteja limpa antes de exibir as avaliações.

2. Consulta no Banco de Dados

```
cursor.execute("select a.id, c.nome, l.titulo, a.nota, a.comentario from avaliacao as a JOIN livro as l on l.id = a.id_livro JOIN cliente as c on c.id = a.id_cliente")
```

```
resultado = cursor.fetchall()
```

- **`cursor.execute(...)`**: Executa uma consulta SQL que busca informações das avaliações, juntando as tabelas `avaliacao`, `livro` e `cliente`. A consulta retorna o ID da avaliação (`a.id`), o nome do cliente (`c.nome`), o título do livro (`l.titulo`), a nota (`a.nota`) e o comentário (`a.comentario`).
- **`resultado = cursor.fetchall()`**: Armazena o resultado da consulta em uma variável chamada `resultado`. Esse resultado é uma lista de tuplas, onde cada tupla representa uma avaliação.

3. Verificação de Avaliações Encontradas

```
if len(resultado) == 0:
```

```
    print("\033[31mNão há registro.\033[m")
```

```
    input()
```

```
    Menu()
```

- **`if len(resultado) == 0:`**: Verifica se a lista de resultados está vazia, ou seja, se nenhuma avaliação foi encontrada.
- **`print("\033[31mNão há registro.\033[m")`**: Exibe uma mensagem informando que não há avaliações registradas.
- **`input()`**: Aguarda uma entrada do usuário (para que ele possa ver a mensagem antes de continuar).
- **`Menu()`**: Redireciona para o menu principal (`Menu()`).

4. Exibição das Avaliações

```
print("="*50)
```



```

print("AVALIAÇÕES")

print("="*50)

for linha in resultado:

    print(f"ID: {linha[0]}")

    print(f"Cliente: {linha[1]}")

    print(f"Livro: {linha[2]}")

    print(f"Nota: {linha[3]}")

    print(f"Comentário: {linha[4]}\n")

print("-"*50)

```

- **print("="*50):** Exibe uma linha de separação para melhorar a visualização.
- **print("AVALIAÇÕES"):** Exibe o título "AVALIAÇÕES".
- **print("="*50):** Exibe outra linha de separação.
- **for linha in resultado::** Itera sobre cada avaliação encontrada no banco de dados.
- **print(f"ID: {linha[0]}"):** Exibe o ID da avaliação.
- **print(f"Cliente: {linha[1]}"):** Exibe o nome do cliente que fez a avaliação.
- **print(f"Livro: {linha[2]}"):** Exibe o título do livro que foi avaliado.
- **print(f"Nota: {linha[3]}"):** Exibe a nota dada pelo cliente.
- **print(f"Comentário: {linha[4]}\n"):** Exibe o comentário feito pelo cliente.
- **print("-"*50):** Exibe uma linha de separação entre as avaliações.

5. Mensagem de Conclusão

```
print("\033[32mConsulta concluída com sucesso.\033[m")
```

- **print("\033[32mConsulta concluída com sucesso.\033[m"):** Exibe uma mensagem de sucesso indicando que a consulta foi realizada com êxito.

6. Retorno ao Menu Principal

```
input()
```

```
Menu()
```

- **input():** Aguarda uma entrada do usuário (para que ele possa ver as avaliações e a mensagem de sucesso antes de continuar).
- **Menu():** Retorna ao menu principal após a exibição das avaliações.

Resumo

A função `VerificarAvaliacoes()` busca e exibe todas as avaliações registradas no banco de dados, mostrando detalhes como o ID da avaliação, o nome do cliente, o título do livro, a

nota e o comentário. Se não houver avaliações, a função informa o usuário e redireciona para o menu principal. Essa função é útil para visualizar o feedback dos clientes sobre os livros de forma organizada e clara.

Remover Cliente

A função `RemoverCliente()` permite que um cliente seja removido do banco de dados com base no seu ID. Vamos detalhar cada parte da função para facilitar o entendimento:

1. Limpeza da Tela

```
os.system("cls")
```

- **`os.system("cls")`**: Limpa a tela do console para uma nova interação. Isso garante que a tela esteja limpa antes de solicitar o ID do cliente.

2. Entrada do ID do Cliente

```
print("Insira o ID do cliente a ser removido:")
```

```
id = input()
```

- **`print("Insira o ID do cliente a ser removido:")`**: Solicita o ID do cliente que será removido.
- **`id = input()`**: Captura o ID inserido pelo usuário.

3. Verificação de Entrada Vazia ou Saída

```
if not id:
```

```
    RemoverCliente()
```

```
if id.lower() == "sair":
```

```
    Menu()
```

- **`if not id:`**: Se o ID estiver vazio, a função chama a si mesma novamente para solicitar o ID.
- **`if id.lower() == "sair":`**: Se o usuário digitar "sair", a função redireciona para o menu principal (`Menu()`).

4. Busca do Cliente no Banco de Dados

```
cursor.execute(f"SELECT * FROM cliente WHERE id = {id}")
```

```
resultado = cursor.fetchall()
```

- **`cursor.execute(f"SELECT * FROM cliente WHERE id = {id}")`**: Executa uma consulta SQL para buscar o cliente com o ID especificado.
- **`resultado = cursor.fetchall()`**: Armazena o resultado da consulta em uma variável chamada `resultado`.

5. Verificação de Cliente Encontrado

```
if len(resultado) == 0:
```

```
    print(f"\033[31mERRO. Nenhum registro encontrado com o id {id}\033[m")
```

```
input()
```

```
return RemoverCliente()
```

- **if len(resultado) == 0::** Verifica se a lista de resultados está vazia, ou seja, se nenhum cliente foi encontrado com o ID especificado.
- **print(f"\033[31mERRO. Nenhum registro encontrado com o id {id}\033[m"]:** Exibe uma mensagem de erro informando que nenhum cliente foi encontrado.
- **input():** Aguarda uma entrada do usuário (para que ele possa ver a mensagem de erro antes de continuar).
- **return RemoverCliente():** Chama a função novamente para permitir que o usuário insira um ID válido.

6. Exibição do Cliente Encontrado

```
print("\033[32mRegistro encontrado !\033[m")
```

```
print("-"*100)
```

```
print(f'{'ID':5}{'Nome':20}{'Telefone':21}{'E_mail'}')
```

```
print("-"*100)
```

```
for linha in resultado:
```

```
    id = linha[0]
```

```
    nome = linha[1]
```

```
    telefone = linha[2]
```

```
    email = linha[3]
```

```
    print(f'{'id':<5}{'nome':20}{'telefone':21}{'email'}')
```

```
    print("-"*100)
```

- **print("\033[32mRegistro encontrado !\033[m"]:** Exibe uma mensagem de sucesso indicando que o cliente foi encontrado.
- **print("-"*100):** Exibe uma linha de separação para melhorar a visualização.
- **print(f'{'ID':5}{'Nome':20}{'Telefone':21}{'E_mail'}')**: Exibe o cabeçalho da tabela com os campos ID, Nome, Telefone e E-mail.
- **print("-"*100):** Exibe outra linha de separação.
- **for linha in resultado::** Itera sobre os dados do cliente encontrado.
- **id = linha[0]:** Armazena o ID do cliente.
- **nome = linha[1]:** Armazena o nome do cliente.
- **telefone = linha[2]:** Armazena o telefone do cliente.
- **email = linha[3]:** Armazena o e-mail do cliente.

- **print(f"{id:<5}{nome:20}{telefone:21}{email}"): Exibe os dados do cliente formatados em uma tabela.**
- **print("-"*100): Exibe uma linha de separação entre os registros.**

7. Confirmação de Exclusão

```
print("Deseja realmente \033[31mexcluir\033[m esse registro ? S / N:")
```

```
opcao = input()
```

- **print("Deseja realmente \033[31mexcluir\033[m esse registro ? S / N:"):** Solicita confirmação do usuário para excluir o registro.
- **opcao = input():** Captura a opção escolhida pelo usuário.

8. Exclusão do Cliente

```
if opcao.lower() == "s":
```

```
try:
```

```
    cursor.execute(f"DELETE FROM cliente WHERE id = {id}")
```

```
    conexao.commit()
```

```
    print("\033[32mRegistro deletado com sucesso.\033[m")
```

```
    input()
```

```
    Menu()
```

```
except:
```

```
    print("\031[32mNão foi possível deletar o cadastro por um erro sinistro.\031[m")
```

```
    input("Tecle Enter para retornar ao menu: ")
```

```
    Menu()
```

```
else:
```

```
    Menu()
```

- **if opcao.lower() == "s":** Se o usuário confirmar a exclusão (digitando "S" ou "s"), a função tenta deletar o cliente.
- **try::** Tenta executar o comando SQL para deletar o cliente.
- **cursor.execute(f"DELETE FROM cliente WHERE id = {id}"): Executa o comando SQL para deletar o cliente com o ID especificado.**
- **conexao.commit():** Confirma a transação no banco de dados.
- **print("\033[32mRegistro deletado com sucesso.\033[m): Exibe uma mensagem de sucesso indicando que o cliente foi deletado.**
- **input():** Aguarda uma entrada do usuário (para que ele possa ver a mensagem de sucesso antes de continuar).

- **Menu():** Retorna ao menu principal após a exclusão.
- **except::** Se ocorrer um erro durante a exclusão, exibe uma mensagem de erro.
- **print("\031[32mNão foi possível deletar o cadastro por um erro sinistro.\031[m"):** Exibe uma mensagem de erro.
- **input("Tecle Enter para retornar ao menu: "):** Aguarda uma entrada do usuário (para que ele possa ver a mensagem de erro antes de continuar).
- **Menu():** Retorna ao menu principal após o erro.
- **else::** Se o usuário não confirmar a exclusão (digitando qualquer coisa que não seja "S" ou "s"), a função redireciona para o menu principal (Menu()).

Resumo

A função `RemoverCliente()` permite que um cliente seja removido do banco de dados com base no seu ID. Ela solicita o ID do cliente, verifica se o cliente existe, exibe os detalhes do cliente e solicita confirmação para a exclusão. Se o usuário confirmar, o cliente é deletado do banco de dados. Se algo der errado, a função informa o erro e permite ao usuário tentar novamente. Essa função é útil para manter o banco de dados atualizado, removendo clientes que não são mais necessários.

Deletar Avaliação

A função `deletarAvaliacao()` permite que uma avaliação seja removida do banco de dados com base no seu ID. Vamos detalhar cada parte da função para facilitar o entendimento:

1. Limpeza da Tela

```
os.system("cls")
```

- **`os.system("cls")`**: Limpa a tela do console para uma nova interação. Isso garante que a tela esteja limpa antes de solicitar o ID da avaliação.

2. Entrada do ID da Avaliação

```
id = input("Insira o ID da avaliação a ser deletada: ")
```

- **`input("Insira o ID da avaliação a ser deletada: ")`**: Solicita o ID da avaliação que será deletada.
- **`id = input()`**: Captura o ID inserido pelo usuário.

3. Verificação de Entrada Vazia ou Saída

```
if not id:
```

```
    deletarAvaliacao()
```

```
if id.lower() == "sair":
```

```
    Menu()
```

- **`if not id:`**: Se o ID estiver vazio, a função chama a si mesma novamente para solicitar o ID.
- **`if id.lower() == "sair":`**: Se o usuário digitar "sair", a função redireciona para o menu principal (`Menu()`).

4. Busca da Avaliação no Banco de Dados

```
cursor.execute(f"SELECT * FROM avaliacao WHERE id = {id}")
```

```
resultado = cursor.fetchall()
```

- **`cursor.execute(f"SELECT * FROM avaliacao WHERE id = {id}")`**: Executa uma consulta SQL para buscar a avaliação com o ID especificado.
- **`resultado = cursor.fetchall()`**: Armazena o resultado da consulta em uma variável chamada `resultado`.

5. Verificação de Avaliação Encontrada

```
if len(resultado) == 0:
```

```
    print(f"\033[31mNão há avaliação registrada com o ID {id}\033[m")
```

```
    input()
```

```
deletarAvaliacao()
```

- **if len(resultado) == 0::** Verifica se a lista de resultados está vazia, ou seja, se nenhuma avaliação foi encontrada com o ID especificado.
- **print(f"\033[31mNão há avaliação registrada com o ID {id}\033[m"):** Exibe uma mensagem de erro informando que nenhuma avaliação foi encontrada.
- **input():** Aguarda uma entrada do usuário (para que ele possa ver a mensagem de erro antes de continuar).
- **deletarAvaliacao():** Chama a função novamente para permitir que o usuário insira um ID válido.

6. Exibição da Avaliação Encontrada

```
for linha in resultado:
```

```
    print("-"*50)
```

```
    print(f"ID: {linha[0]}")
```

```
    print(f"Cliente: {linha[1]}")
```

```
    print(f"Livro: {linha[2]}")
```

```
    print(f"Nota: {linha[3]}")
```

```
    print(f"Comentário: {linha[4]}\n")
```

```
    print("-"*50)
```

- **for linha in resultado::** Itera sobre os dados da avaliação encontrada.
- **print("-"*50):** Exibe uma linha de separação para melhorar a visualização.
- **print(f"ID: {linha[0]}"):** Exibe o ID da avaliação.
- **print(f"Cliente: {linha[1]}"):** Exibe o nome do cliente que fez a avaliação.
- **print(f"Livro: {linha[2]}"):** Exibe o título do livro que foi avaliado.
- **print(f"Nota: {linha[3]}"):** Exibe a nota dada pelo cliente.
- **print(f"Comentário: {linha[4]}\n"):** Exibe o comentário feito pelo cliente.
- **print("-"*50):** Exibe uma linha de separação entre as avaliações.

7. Confirmação de Exclusão

```
deletar = input("\033[33mCONFIRMAÇÃO\033[mVocê realmente deseja apagar esse  
registro ? (S/N): ")
```

- **input("\033[33mCONFIRMAÇÃO\033[mVocê realmente deseja apagar esse registro ? (S/N): "):** Solicita confirmação do usuário para excluir a avaliação.
- **deletar = input():** Captura a opção escolhida pelo usuário.

8. Exclusão da Avaliação


```

if deletar.lower() == "s":
    try:
        cursor.execute(f"DELETE FROM avaliacao WHERE id = {id}")
        conexao.commit()
        print("\033[32mRegistro deletado com sucesso.\033[m")
        input()
    except:
        print("\033[31mNão foi possível deletar o registro.\033[m")
        input()
    Menu()

```

- **if deletar.lower() == "s":** Se o usuário confirmar a exclusão (digitando "S" ou "s"), a função tenta deletar a avaliação.
- **try::** Tenta executar o comando SQL para deletar a avaliação.
- **cursor.execute(f"DELETE FROM avaliacao WHERE id = {id}"):** Executa o comando SQL para deletar a avaliação com o ID especificado.
- **conexao.commit():** Confirma a transação no banco de dados.
- **print("\033[32mRegistro deletado com sucesso.\033[m"):** Exibe uma mensagem de sucesso indicando que a avaliação foi deletada.
- **input():** Aguarda uma entrada do usuário (para que ele possa ver a mensagem de sucesso antes de continuar).
- **Menu():** Retorna ao menu principal após a exclusão.
- **except::** Se ocorrer um erro durante a exclusão, exibe uma mensagem de erro.
- **print("\033[31mNão foi possível deletar o registro.\033[m"):** Exibe uma mensagem de erro.
- **input():** Aguarda uma entrada do usuário (para que ele possa ver a mensagem de erro antes de continuar).
- **Menu():** Retorna ao menu principal após o erro.

Resumo

A função `deletarAvaliacao()` permite que uma avaliação seja removida do banco de dados com base no seu ID. Ela solicita o ID da avaliação, verifica se a avaliação existe, exibe os detalhes da avaliação e solicita confirmação para a exclusão. Se o usuário confirmar, a avaliação é deletada do banco de dados. Se algo der errado, a função informa o erro e permite ao usuário tentar novamente. Essa função é útil para manter o banco de dados atualizado, removendo avaliações que não são mais necessárias.

Encerrar Conexão

A função `EncerrarConexao()` tem como objetivo encerrar de forma segura uma conexão com um banco de dados, limpar a tela do terminal e finalizar o programa. Abaixo está uma explicação detalhada de cada parte da função:

1. Limpando a tela do terminal

```
os.system("cls")
```

O que faz: Executa o comando `cls`, que limpa a tela no terminal do Windows.

Motivo: Deixa a interface mais limpa para o usuário antes de finalizar ou exibir mensagens.

2. Encerrando a conexão com o banco de dados

```
try:
```

```
    cursor.close()
```

```
    conexao.close()
```

O que faz:

`cursor.close()`: Fecha o cursor do banco de dados. O cursor é usado para executar comandos SQL e recuperar dados.

`conexao.close()`: Encerra a conexão com o banco de dados para liberar os recursos.

Motivo: É uma boa prática fechar as conexões com o banco de dados para evitar vazamento de recursos ou erros futuros.

3. Exibindo mensagem de sucesso

```
print("\033[32mConexão Encerrada com segurança.\033[m")
```

O que faz: Imprime uma mensagem informando que a conexão foi encerrada com sucesso.

Detalhes:

`\033[32m`: Código ANSI que muda a cor do texto para verde.

`\033[m`: Reseta as cores do texto após a mensagem.

Motivo: Fornece um feedback visual ao usuário de que o processo foi bem-sucedido.

4. Finalizando o programa

```
os._exit(0)
```

O que faz: Encerra imediatamente a execução do programa com o código de saída 0 (indica término bem-sucedido).

Motivo: Garante que o programa seja encerrado após a mensagem de sucesso.

5. Tratando falhas ao encerrar a conexão

```
except:
```

```
    print("\033[31mNão foi possível encerrar a conexão. Tente novamente.\033[m")
```

```
    input()
```

```
    Menu()
```

O que faz:

Se ocorrer um erro ao tentar fechar o cursor ou a conexão, uma mensagem de erro é exibida.

\033[31m: Muda a cor do texto para vermelho para indicar falha.

input(): Pausa o programa para que o usuário veja a mensagem.

Menu(): Retorna o usuário para o menu principal do programa.

Motivo: Fornece um feedback em caso de erro e dá ao usuário a oportunidade de tentar novamente.

Resumo do comportamento

Limpa a tela para uma interface mais limpa.

Tenta fechar o cursor e a conexão com o banco de dados:

Se der certo, exibe uma mensagem verde e encerra o programa.

Se falhar, exibe uma mensagem vermelha, pausa para o usuário ler e retorna ao menu principal.

Menu

A função Menu() é o coração de um **sistema de gerenciamento** que permite ao usuário realizar diversas operações relacionadas a livros, clientes e avaliações. Vamos analisar a função em detalhes para entender todas as suas funcionalidades:

1. Limpando a tela

```
os.system("cls")
```

- **O que faz:** Limpa a tela do terminal, deixando o sistema com uma aparência mais limpa antes de exibir o menu.
-

2. Estrutura principal do menu

```
while True:
```

- **O que faz:** Cria um loop infinito para manter o menu sempre ativo até que o usuário escolha sair do sistema.
 - **Motivo:** O menu precisa estar continuamente disponível enquanto o programa estiver em execução.
-

3. Exibição do menu

```
print(" " * 28 + f"{Formatação_cores.GREEN}SISTEMA DE  
GERENCIAMENTO{Formatação_cores.RESET}\n")
```

- **O que faz:** Exibe o título do sistema formatado em verde (Formatação_cores.GREEN) e centralizado.
 - **Motivo:** Dá uma introdução clara e organizada para o sistema.
-

3.1. Opções do menu

As opções são divididas em duas colunas dentro de caixas delimitadas:

```
print('┌'+ '=' * 35 + '┐'+ " "* 4 + '┌'+ '=' * 35 + '┐')
```

```
print('│'+ " 1. CADASTRAR LIVROS " + " "* 5 + '│'+ " "* 4 + '│'+ "10. CADASTRAR  
CLIENTES │")
```

... (continua para todas as opções)

- **O que faz:**
 - Organiza as opções do menu em duas colunas, separadas por um pequeno espaço.

- Cada coluna possui uma borda decorativa para melhor apresentação.
 - **Motivo:** Melhora a legibilidade e organiza as opções de forma visualmente agradável.
-

4. Coletando a escolha do usuário

```
option = input()
```

- **O que faz:** Aguarda o usuário digitar um número correspondente a uma das opções do menu.
 - **Motivo:** Permite que o sistema saiba qual operação o usuário deseja realizar.
-

5. Verificando a escolha e executando ações

Cada opção é mapeada para uma funcionalidade específica usando if/elif:

```
if option == "1":
```

```
    CadastrarLivro()
```

```
elif option == "2":
```

```
    ListarLivros()
```

... (continua para todas as opções)

5.1. Funcionalidades das opções

1. Opções relacionadas a livros:

- **CadastrarLivro():** Adiciona novos livros ao sistema.
- **ListarLivros():** Mostra os livros cadastrados.
- **ListarEstoque():** Exibe os livros disponíveis em estoque.
- **ComprarLivros():** Permite registrar a compra de livros.
- **ListarCompras():** Lista todas as compras realizadas.
- **VenderLivros():** Permite registrar a venda de livros.
- **ListarVendas():** Mostra todas as vendas realizadas.
- **RemoverLivro():** Remove um livro cadastrado.
- **AtualizarLivro():** Atualiza os dados de um livro.

2. Opções relacionadas a clientes:

- **CadastrarCliente():** Adiciona novos clientes ao sistema.
- **ListarClientes():** Exibe os clientes cadastrados.
- **AtualizarCliente():** Atualiza as informações de um cliente.

- **RemoverCliente():** Remove um cliente do sistema.

3. Opções relacionadas a avaliações:

- **AdicionarAvaliacoes():** Permite registrar avaliações de usuários.
- **VerificarAvaliacoes():** Exibe avaliações feitas pelos usuários.
- **deletarAvaliacao():** Remove uma avaliação existente.

4. Opção para encerrar a conexão

- **Opção 17:**
 - Pergunta ao usuário se deseja encerrar a conexão com o banco de dados:

```
print("Deseja realmente encerrar a conexão? S/N")
```

```
opcao = input()
```

- Se o usuário digitar S (sim), executa:

```
os.system("cls")
```

```
Final.run_asciimatics_demo()
```

- Limpa a tela e exibe uma animação de encerramento.
- Caso contrário, retorna ao menu.

5. Opção inválida

- Se o usuário inserir algo que não está listado:

```
else:
```

```
continue
```

- Volta ao início do menu sem fazer nada.

6. Início do programa

Antes de exibir o menu, o sistema executa:

```
Apresentação.animacao_intro()
```

```
Menu()
```

- **O que faz:**
 - `Apresentação.animacao_intro():` Provavelmente exibe uma introdução animada para o sistema.
 - `Menu():` Inicia o menu principal.

Resumo do comportamento

1. Limpa a tela e exibe o menu principal.
 2. Aguarda o usuário escolher uma das 17 opções.
 3. Executa a funcionalidade correspondente ou volta ao menu em caso de erro ou opção inválida.
 4. Permite encerrar o sistema com confirmação.
-