

PROJETO DE DESENVOLVIMENTO RÁPIDO DE APLICAÇÕES EM PYTHON

Alunos: Daniel Seixas Tavares, Júlia Teles da Silva Bispo e Pedro Henrique Ribeiro

SISTEMA DE CADASTRO DE LIVROS | Biblioteca Virtual

1ª Etapa:

Requisitos:

Propósito	O Sistema de Cadastro de Livros será útil para uma biblioteca virtual, cujo cliente poderá comprar o seu livro digital.
Escopo	O Sistema de Cadastro de Livros Virtuais será projetado para cadastrar (código, título do livro, autor, editora, ano e preço). Haverá a integração com um banco de dados permitindo o controle do acervo.
Interfaces do Sistema	O sistema é compatível com o Windows, Linux, Android e IOS.
Interfaces de Usuário	O sistema contém botões, caixas de textos para cada dado. O sistema é compatível com mouse e teclado.
Interfaces de Software	O software permite cadastrar em banco de dados e o administrador poderá fazer as devidas alterações. O banco de dados escolhido foi o PostgreSQL.
Requisitos Funcionais	Será possível cadastrar os livros; Ler os dados dos livros; Atualizar os dados dos livros; Excluir os livros.
Requisitos Não-Funcionais	Requisitos do sistema: <ul style="list-style-type: none">· CPU a partir de 1 GHz (gigahertz) ou mais rápido com 2 ou mais núcleos de processamento;· Memória RAM de 8GB (gigabytes);· HD ou SSD de 1TB;· Tela de alta definição (720p) a partir de 4.7 polegadas.

Nosso produto:

Livro

-
- | | |
|----|---------------------|
| 1. | id |
| 2. | Titulo_livro |
| 3. | Autor |
| 4. | Ano (de publicação) |
| 5. | Editora |
| 6. | Genero |
| 7. | Preço |
-

Preço ajustado: + Taxa de 10%

2ª Etapa:

Protótipo

Livraria Virtual
A - Z

Consulta

TÍTULO DO LIVRO:

AUTOR:

ANO:

EDITORIA:

GÊNERO:

PREÇO:

PREÇO COM TAXA (10%):

Cadastrar **Atualizar** **Excluir** **Limpar**

Na segunda fase, criamos um protótipo (sujeito a mudança ao longo do desenvolvimento do projeto avaliativo) tanto em papel quanto na plataforma Canva para uma compreensão visual do projeto. Esses protótipos foram desenvolvidos para permitir que o grupo visualizasse de forma clara e tangível a ideia do projeto, facilitando a compreensão do que seria implementado no código.

3ª Etapa:

Criação do código

- Importação das bibliotecas:

```
import tkinter as tk
import psycopg2
```

- Conexão com o Banco de Dados, Criação da Tabela, Mensagem de confirmação e fechamento da conexão:

```
#Conectando com o BD e criando (CREATE) a tabela
conn = psycopg2.connect(database="postgres",user="postgres",password="1234",port="5432")
print("Conexão com o Banco de Dados aberta com sucesso!")
comando = conn.cursor()
comando.execute(""" CREATE TABLE Livros
(id INT PRIMARY KEY NOT NULL,Autor TEXT NOT NULL,Titulo_livro TEXT NOT
NULL,Editora TEXT NOT NULL,Ano CHAR(4),Genero TEXT NOT NULL, Preco CHAR(5) );""")
conn.commit()
print("Tabela criada com sucesso no BD!!!")
conn.close
```

```
import psycopg2
conn = psycopg2.connect(database="postgres",user="postgres",password="1234",port="5432")
comando = conn.cursor()
comando.execute (""" INSERT INTO LIVROS (id,Autor,Titulo_livro,Ano,Editora,Genero,Preco)
VALUES ()""")
```

Esse bloco de código realiza a conexão com o banco de dados PostgreSQL usando o adaptador psycopg2. Seguidamente, é criada uma tabela chamada "Livros" com as colunas: id, Autor, Titulo_livro, Ano, Editora, e Genero. Também exibe uma mensagem de confirmação e fecha a conexão depois de realizar todas as operações.

- Função cadastrar_livro, lógica para o cadastramento dos livros:

```
# função dos botões
def cadastrar_livro(autor,titulo_livro,ano,editora,genero,preco):
    autor = entry_autor.get()
    titulo_livro = entry_titulo_livro.get()
    editora = entry_editora.get()
    ano = entry_ano.get()
    genero = entry_genero.get()
    preco = float(entry_preco.get()) # Converte o preço para float
    # Calcula o preço com a taxa de 10%
    def acrescimo_dez_por_cento(preco):
        return preco * 1.10
    conn.commit()
    print("Inserção realizada com sucesso!")
    conn.close()
```

Nessa função, são obtidos os valores dos campos de entrada para nome, autor, editora, ano, gênero e preço. Ela obtém os valores usando os métodos `get()` dos widgets `Entry`. O preço é ajustado para inserir e calcular uma taxa de 10%. Após isso, é feita uma nova conexão para realizar as operações no banco de dados, executando comandos SQL para inserir registros na tabela "Livros". E então, realiza o commit da transação e imprime informações do livro cadastrado, incluindo o preço original e o preço com a taxa, são exibidas no console.

- Função `ler_livro`:

```
def ler_livro():
    import psycopg2
    conn = psycopg2.connect(database="postgres",user="postgres",password="1234",port="5432")
    comando = conn.cursor()
    comando.execute(""" SELECT * FROM LIVROS where id = %s""")
    resultado = comando.fetchone()
    print("Livro encontrado ->", resultado)
    conn.commit()
    print("Seleção realizada com sucesso!")
    conn.close()
```

A função `ler_livro` é chamada quando o botão correspondente é pressionado. Portanto, será estabelecida uma conexão com o banco de dados executando uma consulta SQL para selecionar registros da tabela "Livros" com base no id e imprime o resultado.

- Função `atualizar_livro`:

```
def atualizar_livro():
    import psycopg2
    conn = psycopg2.connect(database="postgres",user="postgres",password="1234",port="5432")
    comando = conn.cursor()
    comando.execute(""" SELECT * FROM LIVROS where id = ?""")
    resultado = comando.fetchone()
    print("Livro encontrado ->", resultado)
    comando.execute(""" UPDATE LIVROS SET id = ?, Autor = ?, Titulo_livro = ?, Ano = ?,
    Editora = ?, Genero = ?, Preco = ?, where id = ?""")
    print("Atualização realizada com sucesso!")
    comando = conn.cursor()
    print("--- Consulta após atualização ---")
    comando.execute(""" SELECT * FROM LIVROS where id = ?""")
    resultado = comando.fetchone()
    print("Dados atualizados ->", resultado)
    conn.commit()
    print("Livro atualizado com sucesso!")
    conn.close()
```

Essa função é responsável por executar uma instrução SQL para atualizar registros na tabela "Livros". Depois da realização do commit da transação, será impresso informações após a atualização e uma mensagem "Livro atualizado com sucesso!".

- Função `excluir_livro`:

```
def excluir_livro():
    import psycopg2
    conn = psycopg2.connect(database="postgres",user="postgres",password="1234",port="5432")
    comando = conn.cursor()
    comando.execute(""" DELETE FROM LIVROS where id = ?""")
    conn.commit()
    cont = comando.rowcount
    print(cont, '-> Registro(s) excluído(s) com sucesso!')
    conn.close()
    print("Livro excluído com sucesso!")
```

Essa função vai exercer uma instrução SQL para excluir registros da tabela "Livros" com base no id e realiza o commit da transação, obtém o número de registros afetados (cont) e imprime a mensagem de sucesso.

- Função `limpar_campos`:

```
def limpar_campos():
    entry_nome.delete(0, tk.END)
    entry_autor.delete(0, tk.END)
    entry_editora.delete(0, tk.END)
    entry_ano.delete(0, tk.END)
    entry_genero.delete(0, tk.END)
```

A função `limpar_campos` usa o método `delete` dos widgets `Entry` para limpar os campos de entrada.

- Configuração da Janela Principal:

```
# configuração da janela principal
root = tk.Tk()
root.title("Sistema de cadastramento de livros virtuais")
# campos de entrada para as informações do livro
label_nome = tk.Label(root, text="Nome do Livro:")
label_nome.pack()
entry_nome = tk.Entry(root)
entry_nome.pack()
label_autor = tk.Label(root, text="Autor:")
label_autor.pack()
entry_autor = tk.Entry(root)
entry_autor.pack()
label_editora = tk.Label(root, text="Editora:")
```

```

label_editora.pack()
entry_editora = tk.Entry(root)
entry_editora.pack()
label_ano = tk.Label(root, text="Ano de Publicação:")
label_ano.pack()
entry_ano = tk.Entry(root)
entry_ano.pack()
label_genero = tk.Label(root, text="Gênero:")
label_genero.pack()
entry_genero = tk.Entry(root)
entry_genero.pack()
label_preco = tk.Label(root, text="Preço:")
label_preco.pack()
entry_preco = tk.Entry(root)
entry_preco.pack()
# botões
btn_cadastrar = tk.Button(root, text="Cadastrar", command=cadastrar_livro)
btn_cadastrar.pack()
btn_atualizar = tk.Button(root, text="Atualizar", command=atualizar_livro)
btn_atualizar.pack()
btn_excluir = tk.Button(root, text="Excluir", command=excluir_livro)
btn_excluir.pack()
btn_limpar = tk.Button(root, text="Limpar", command=limpar_campos)
btn_limpar.pack()
root.mainloop()

```

Nesse bloco de código, temos `tk.Tk()` que cria uma instância da classe `Tk` para representar a janela principal da interface gráfica e `root.title` que define o título da janela, nesse caso o título é: "Sistema de cadastramento de livros virtuais".

Para inserir as informações dos livros (Nome do Livro, Autor, Editora, Ano de Publicação, Gênero, Preço) são criados rótulos (`tk.Label`) e campos de entrada (`tk.Entry`) na janela principal.


O método `pack()` organiza os elementos na janela, empilhando-os verticalmente na ordem em que são chamados.

Também são criados botões para realizar ações como cadastrar, atualizar, excluir e limpar os campos.

`tk.Button(root, text="Cadastrar", command=cadastrar_livro)`: Cria um botão com o texto "Cadastrar" e associa a função `cadastrar_livro` quando clicado. E da mesma forma para os botões de atualizar, excluir e limpar.

Por fim, o `root.mainloop()` inicia o loop principal da interface gráfica, aguardando interações do usuário.

Interface Final:

 Sistema de cadastro de livros virtuais

Nome do Livro:

Código Limpo

Autor:

Robert C. Martin

Editora:

Alta Books

Ano de Publicação:

2009

Gênero:

Tecnologia

Preço:

70.85

Cadastrar

Atualizar

Excluir

Limpar