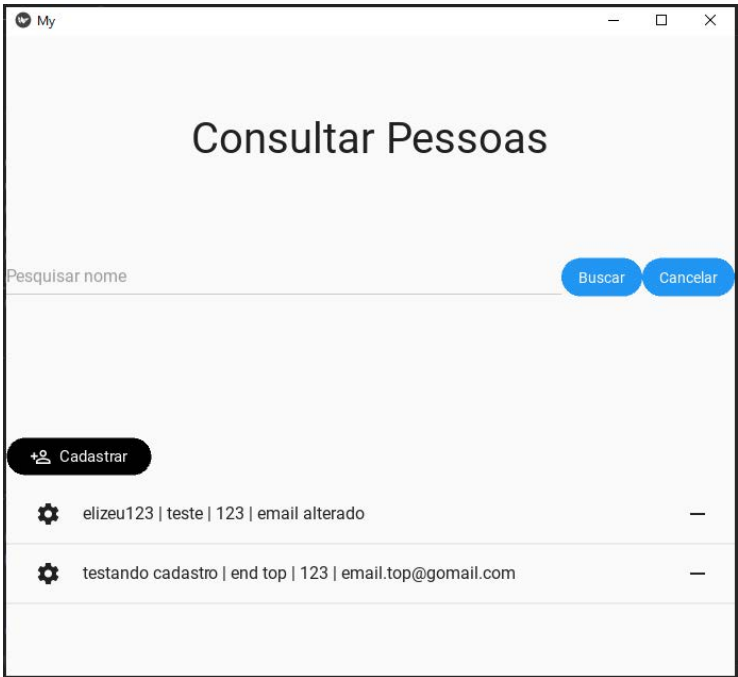


Das 9 tabelas foi feito telas de 4 : PESSOA; DOACAO; USUARIO; ITEM_DOACAO

Tela de consulta



Na tela inicial é feita a consulta no banco de dados pelo nome digitado e retornado uma lista de resultados abaixo do botão de cadastrar, cada linha do registro tem um botão de engrenagem que vai para a tela de edição e um botão de menos que faz a exclusão do registro.

comando pesquisar

```
def pesquisar(self, texto):
    try:
        print(texto)
        lista_pessoas = operar_pessoa('SELECT', dados={'nome': texto.strip()})
        print(lista_pessoas)

        # Limpar a lista de pessoas
        self.ids.pessoa_list.clear_widgets()

        btnBuscar = self.ids.button_buscar

        # Iterar sobre os resultados da consulta
        for row in lista_pessoas:
            id_pessoa, nome, endereco, telefone, email = row
            pessoa_item = PessoaListItem(id_pessoa, nome, endereco, telefone, email, btnBuscar)

            # Adicionar o item à lista
            self.ids.pessoa_list.add_widget(pessoa_item)

    except Exception as e:
        toast(f"Error: {e}", duration=5)
        print(e)
```

Quando o botão buscar é clicado ele ativa a função pesquisar. Ela pega o texto escrito no campo de pesquisa e envia para a função 'operar_pessoa' com o argumento 'SELECT' e recebe de volta uma lista com os resultados. Para cada item da lista é criado uma nova instancia da classe 'PessoaListItem' e enviado como argumentos os dados da resposta.

Função operar pessoa opção SELECT

```
2 usages  Elizeu
def operar_pessoa(operador, dados=None):
    conn, cursor = conectarBancoECursor()

    if dados is None and (operador != "SELECT"):
        return print("dados vazio")

    if operador == "SELECT":
        if not dados['nome']:
            cursor.execute(""" SELECT * FROM PESSOA ORDER BY Nome LIMIT 20""")
            resposta = cursor.fetchall()

            commitEFecharConexao(conn)
            return resposta

        cursor.execute(""" SELECT * FROM PESSOA WHERE Nome LIKE ? ORDER BY Nome LIMIT 20""", ('%' + dados[
            'nome'] + '%',))
        resposta = cursor.fetchall()

        commitEFecharConexao(conn)
        return resposta
```

A função recebe o tipo de operação que deve ser feita na tabela PESSOA e um objeto com os dados necessários. Ela verifica se a opção recebida é igual a 'SELECT' e se veio um nome para filtrar, limitando a resposta em 20 registros, depois retorna a resposta.

Classe da apresentação dos registros

```
2 usages  Elizeu
class PessoaListItem(OneLineAvatarIconListItem, EventDispatcher):
    texto = StringProperty('')

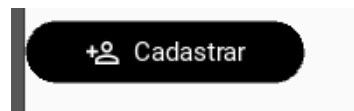
    Elizeu
    def __init__(self, id_pessoa, nome='', endereco='', telefone='', email='', btnBuscar=None, **kwargs):
        super(PessoaListItem, self).__init__(**kwargs)
        self.id_pessoa = id_pessoa
        self.nome = nome
        self.endereco = endereco
        self.telefone = telefone
        self.email = email
        self.texto = f"{nome} | {endereco} | {telefone} | {email}"
        self.btnBuscar = btnBuscar

    2 usages  Elizeu
    def editar(self):...

    Elizeu
    def deletar(self):...
```

Cada item retornado da pesquisa é usado para criar uma instancia dessa classe que possui os metodos de editar no botão de engrenagem e deletar no símbolo de menos.

Botão de cadastrar pessoas



Tela cadastrar pessoa

A interface de uma tela de cadastro de pessoa. No topo, o título 'Cadastrar Pessoa' está centralizado. Abaixo dele, há quatro campos de entrada de texto, cada um com um rótulo à esquerda: 'Nome', 'Endereço', 'Telefone' e 'Email'. Na base da tela, há dois botões: 'Voltar' com um ícone de seta para trás e 'Salvar' com um ícone de disco de salvar.

Quando o botão cadastrar pessoa é clicado aparece a tela de cadastrar pessoa e o atributo 'editar' é mudado para false

Código do botão salvar na tela de cadastro

```
def salvar_dados(self):
    print('Salvando:', self.ids.pessoaNome.text, self.ids.pessoaEndereco.text,
          self.ids.pessoaTelefone.text, self.ids.pessoaEmail.text)

    operar_pessoa("INSERT", dados={
        'nome': self.ids.pessoaNome.text,
        'endereco': self.ids.pessoaEndereco.text,
        'telefone': self.ids.pessoaTelefone.text,
        'email': self.ids.pessoaEmail.text
    })

    self.ids.pessoaNome.text = ''
    self.ids.pessoaEndereco.text = ''
    self.ids.pessoaTelefone.text = ''
    self.ids.pessoaEmail.text = ''

    self.manager.transition.direction = 'right'
    self.manager.current = "lista_pessoa"
```

Quando o botão de salvar é clicado e o atributo 'editar' é marcado como 'false' a função 'salvar_dados' é ativada. Ela envia para a função 'operar_pessoa' os dados com o operador 'INSERT', depois limpa os campos de texto e muda de tela.

Código dentro da função 'operar_pessoa' de insert

```
if operador == "INSERT":
    cursor.execute("""
        INSERT INTO PESSOA (Nome, Endereco, Telefone, Email)
        VALUES (?, ?, ?, ?)""" , (dados['nome'], dados['endereco'], dados['telefone'],
                                     dados['email']))

    print("Insert feito com sucesso")
```

Código KV descrevendo o layout de cada registro na lista

```
<PessoaListItem>:
    id: pessoa
    text: root.texto

    IconLeftWidget:
        icon: 'cog'
        on_release: root.editar()

    MDIconButton:
        icon: "minus"
        pos_hint: {"center_x": .95, "center_y": .5}
        on_release: root.deletar()
```

Código do botão de engrenagem

 elizeu123 | teste | 123 | email alterado

Um registro

```
def editar(self):
    global gt

    # alterando os dados da tela pessoa para os da pessoa a ser editada
    tela_pessoa = gt.get_screen('pessoa')
    tela_pessoa.ids['pessoaLabel'].text = 'Editar\nPessoa'
    tela_pessoa.ids['pessoaNome'].text = self.nome
    tela_pessoa.ids['pessoaEndereco'].text = self.endereco
    tela_pessoa.ids['pessoaTelefone'].text = self.telefone
    tela_pessoa.ids['pessoaEmail'].text = self.email
    tela_pessoa.editar = True
    tela_pessoa.id_pessoa_editar = self.id_pessoa

    gt.transition.direction = 'left'
    gt.current = 'pessoa'
```

Quando a pessoa clica na engrenagem os dados do registro são enviados para a tela de cadastro de pessoa. Os campos de input de texto são preenchidos com os dados do registro, o título da tela é mudado e a função do botão salvar é mudada para editar com o atributo 'editar'.

Tela de editar



My

Editar Pessoa

Nome
elizeu123 9/25

Endereço
teste 5/25

Telefone
123 3/25

Email
email alterado 14/25

Voltar Salvar

Quando a engrenagem é clicada aparece a tela de editar com os campos preenchidos e quando a pessoa clica em salvar a edição do registro é salva.

Código função editar do botão salvar

```
def editar_dados(self):  
    print('Salvando:', self.ids.pessoaNome.text, self.ids.pessoaEndereco.text,  
          self.ids.pessoaTelefone.text,  
          self.ids.pessoaEmail.text)  
  
    operar_pessoa("UPDATE", dados={  
        'nome': self.ids.pessoaNome.text,  
        'endereco': self.ids.pessoaEndereco.text,  
        'telefone': self.ids.pessoaTelefone.text,  
        'email': self.ids.pessoaEmail.text,  
        'id_pessoa': self.id_pessoa_editar  
    })  
  
    self.ids.pessoaNome.text = ''  
    self.ids.pessoaEndereco.text = ''  
    self.ids.pessoaTelefone.text = ''  
    self.ids.pessoaEmail.text = ''  
    self.id_pessoa_editar = 0  
  
    self.manager.transition.direction = 'right'  
    self.manager.current = "lista_pessoa"
```

Quando a pessoa clica no botão salvar com a função editar selecionada os dados dos campos de input de texto são enviados para a função 'operar_pessoa' com o operador 'UPDATE' e a 'id' do registro, após isso os campos são limpos e ele volta para a tela de consulta.

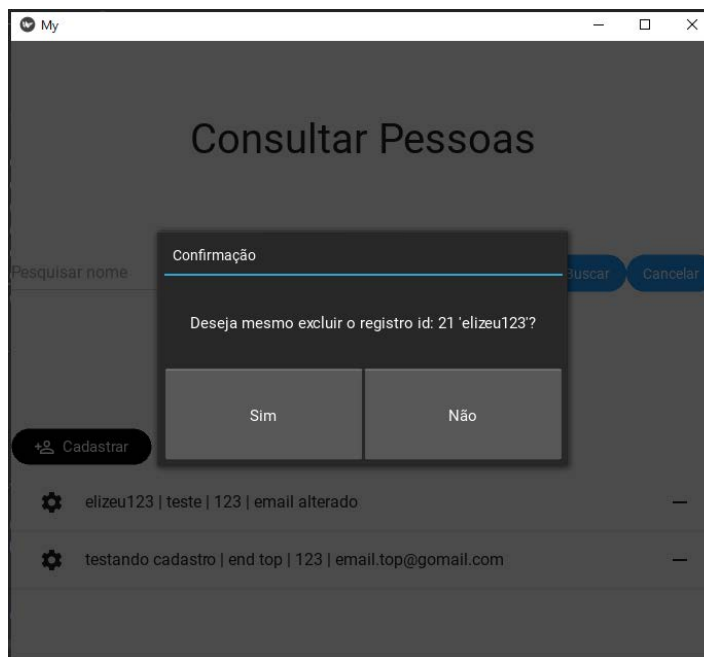
Código dentro da função 'operar_pessoa' de atualização de um registro

```
if operador == "UPDATE":
    cursor.execute("""
        UPDATE PESSOA set Nome = ?, Endereco = ?, Telefone = ?, Email = ?
        WHERE ID_Pessoa = ?""",
        (dados['nome'], dados['endereco'], dados['telefone'], dados['email'],
        dados['id_pessoa']))

    print("UPDATE feito com sucesso")
```

Se o operador for igual a 'UPDATE' a query de update é executada com os dados recebidos

Tela de confirmação de exclusão de um registro



Código de exclusão do botão de menos

```
def deletar(self):
    Elizeu
    def confirmar_exclusao():
        try:
            operar_pessoa('DELETE', dados={'nome': self.nome})
            self.btnBuscar.trigger_action()
            toast("Registro deletado", duration=5)

        except Exception as e:
            toast(f"Error: {e}", duration=5)
            print(e)

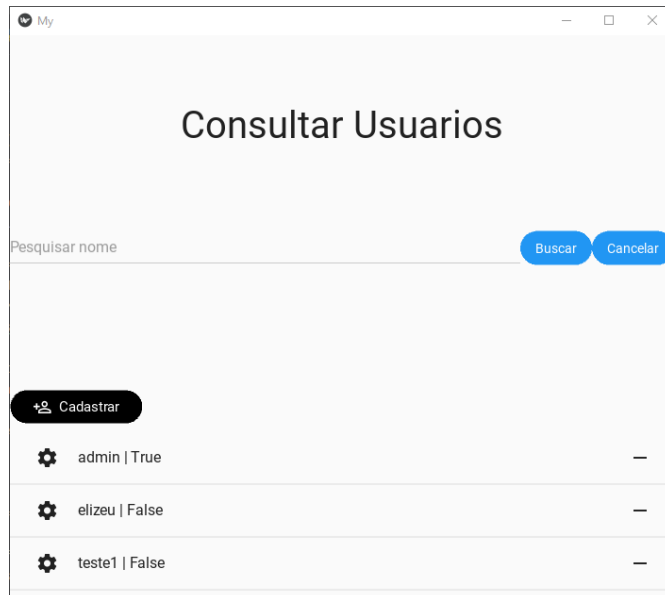
    popup = ConfirmationPopup(callback=confirmar_exclusao, nome_registro=f"id: {self.id_pessoa} '{self.nome}'")
    popup.open()
```

Quando a pessoa clica no botão de menos é chamado um popup de confirmação, se confirmado é chamada a função 'operar_pessoa' com o operador 'DELETE' e os dados do registro

Código dentro da função 'operar_pessoa' de exclusão

```
if operador == "DELETE":
    cursor.execute("""
        DELETE FROM PESSOA WHERE Nome = ?""", (dados['nome'],))
    print("DELETE feito com sucesso")
```

CRUD Usuarios



Consultar Usuarios	
Pesquisar nome	
<button>Buscar</button> <button>Cancelar</button>	
<button>+ Cadastrar</button>	
⚙	admin True
⚙	elizeu False
⚙	teste1 False

```
lista_usuario = operar_usuario('SELECT', dados={'login': texto.strip()})
print(lista_usuario)

# Limpar a lista de usuarios
self.ids.usuario_list.clear_widgets()

btn_buscar = self.ids.button_buscar

# Iterar sobre os resultados da consulta
for row in lista_usuario:
    id_usuario, login, senha, is_admin = row

    if is_admin == 1:
        is_admin = True
    else:
        is_admin = False

    usuario_item = UsuarioListItem(id_usuario, login, senha, is_admin, btn_buscar)

    # Adicionar o item à lista
    self.ids.usuario_list.add_widget(usuario_item)
```

quando clicam no botão de buscar é enviado o texto da barra de busca para a função `operar_usuario` que retorna uma lista de registros mostrada na parte de baixo da tela

```
def operar_usuario(operador, dados):
    conn, cursor = conectar_banco_e_cursor()

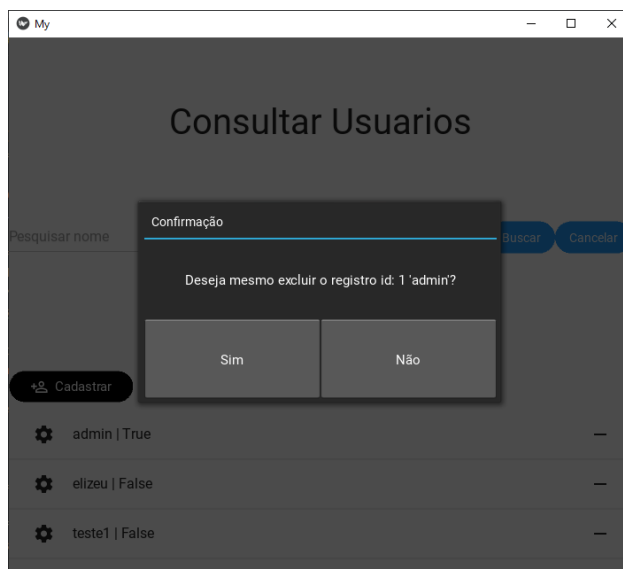
    if dados is None and (operador != "SELECT"):
        return print("dados vazio")

    if operador == "SELECT":
        cursor.execute(
            """ SELECT * FROM USUARIO
                WHERE login LIKE ?
                ORDER BY login LIMIT 20""",
            ('%' + dados['login'] + '%',))

        resposta = cursor.fetchall()

        commit_e_fechar_conexao(conn)
        return resposta
```

função select de operar pessoa



quando é clicado no botão de menos aparece o popup de confirmação e se clicado em sim o registro é excluído

```
def deletar(self):
    @Elizeu
    def confirmar_exclusao_usuario():
        try:
            operar_usuario('DELETE', dados={'id_usuario': self.id_usuario})
            self.btn_buscar.trigger_action()
            toast("Registro deletado", duration=5)

        except Exception as e:
            toast(f"Error: {e}", duration=5)
            print(e)

    popup = ConfirmationPopup(
        callback=confirmar_exclusao_usuario,
        nome_registro=f'id: {self.id_usuario} {self.login}'
    )
    popup.open()
```

o click chama a função operar usuario com o método delete e o id do usuario

```
if operador == "DELETE":
    cursor.execute("""
        DELETE FROM USUARIO WHERE id_usuario = ?""",
        (dados['id_usuario'],))
    print("DELETE feito com sucesso")

    commit_e_fechar_conexao(conn)
```

método delete da função operar usuario

quando é clicado no botão de cadastrar na tela de consulta a tela muda para a de cadastro


```

1 usage 1 Elizeu
def salvar_dados(self):
    if not self.validar_senha(self.ids.usuario_senha.text, self.ids.usuario_senha_confirmar.text):
        return

    print(
        'Salvando:',
        self.ids.usuario_login.text,
        self.ids.usuario_senha.text
    )

    if self.ids.is_admin_switch.active:
        self.admin = 1
    else:
        self.admin = 0

    operar_usuario("INSERT", dados={
        'login': self.ids.usuario_login.text,
        'senha': self.ids.usuario_senha.text,
        'admin': self.admin
    })

    self.manager.transition.direction = 'right'
    self.manager.current = "consultar_usuario"

```

para salvar é antes feito uma validação da senha verificando se o texto da senha e confirmar senha são iguais depois é chamada a função operar_usuario com 'insert' e os dados do usuario

```

if operador == "INSERT":
    cursor.execute("""
        INSERT INTO USUARIO (login, senha, admin)
        VALUES (?, ?, ?)"""
        (dados['login'], dados['senha'], dados['admin']))

    print("Insert feito com sucesso")

```

código insert na função operar_usuario

quando clicam na engrenagem a tela vai para a de edição com o login preenchido, para completar a edição deve ser criada uma nova senha

```

def editar_dados(self):
    if not self.validar_senha(self.ids.usuario_senha.text, self.ids.usuario_senha_confirmar.text):
        return

    print(
        'Atualizando:',
        self.ids.usuario_login.text,
        self.ids.usuario_senha.text
    )

    if self.ids.is_admin_switch.active:
        self.admin = 1
    else:
        self.admin = 0

    operar_usuario("UPDATE", dados={
        'login': self.ids.usuario_login.text,
        'senha': self.ids.usuario_senha.text,
        'admin': self.admin,
        'id_usuario': self.id_usuario_editar
    })

    self.manager.transition.direction = 'right'
    self.manager.current = "consultar_usuario"

```

no modo de editar é feita a validação da senha e chamada função operar_usuario com o 'update' e os dados e id do usuario para atualizar

```
if operador == "UPDATE":
    cursor.execute("""
        UPDATE USUARIO set login = ?, senha = ?, admin = ?
        WHERE ID_USUARIO = ?""",
        (dados['login'], dados['senha'], dados['admin'], dados['id_usuario']))

    print("UPDATE feito com sucesso")
```

código de update na função operar_usuario

CRUD Doação

Doação	Data	Pessoa	Ações
jobs: Doacao de uma janela	20/04/2012	peessoa: pessoa 123	—
jobs: doacao 14	12/05/2014	peessoa: 123	—
jobs: doacao teste	12/03/2001	peessoa: eliseu	—

```
def pesquisar(self, texto):
    try:
        print(texto)

        doacao_resposta_bd = operar_doacao('SELECT', dados={'observacao': texto.strip()})
        print(doacao_resposta_bd)

        # Limpar a lista de doacao
        self.ids.doacao_list.clear_widgets()

        btn_buscar = self.ids.button_buscar

        # Iterar sobre os resultados da consulta
        for row in doacao_resposta_bd:
            id_doacao, dt_doacao, observacao, nome, login, id_pessoa, id_usuario = row
            doacao_item = DoacaoListItem(id_doacao, dt_doacao, observacao,
                                         nome, login, id_pessoa, id_usuario, btn_buscar)

            # Adicionar o item à lista
            self.ids.doacao_list.add_widget(doacao_item)

    except Exception as e:
```

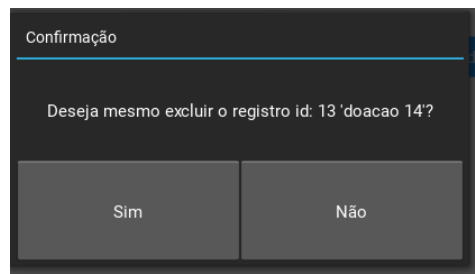
para pesquisar é chamada a função operar_doacao com 'select' e o texto do campo de pesquisa

```
cursor.execute(
    """ SELECT
        id_doacao, dt_doacao, observacao, nome,
        login, D.id_pessoa, D.id_usuario
        FROM DOACAO D, PESSOA P, USUARIO U
        WHERE observacao LIKE ?
        AND D.id_pessoa = P.id_pessoa
        AND D.id_usuario = U.id_usuario
        ORDER BY observacao LIMIT 20""",
    ('%' + dados['observacao'] + '%',))

resposta = cursor.fetchall()
commit_e_fechar_conexao(conn)

return resposta
```

código de select, ele faz a pesquisa na tabela de doacao, usuario e pessoa para trazer alem dos dados da tabela de doacao o nome da pessoa que fez a doacao e o usuario que cadastrou ela



```
if operador == "DELETE":
    cursor.execute("""
        DELETE FROM DOACAO
        WHERE id_doacao = ?""",
        (dados['id_doacao'],))
    print("DELETE feito com sucesso")

    commit_e_fechar_conexao(conn)
```

no momento de excluir aparece um popup com a confirmação de exclusão, se sim é chamada a função operar_doacao com o 'delete' e o id da doação

```
def salvar_dados(self):
    global usuario_atual

    print(
        'Salvando:',
        self.ids.doacaoObservacao.text,
        self.ids.doacaoData.text
    )

    operar_doacao("INSERT", dados={
        'dt_doacao': self.ids['doacaoData'].text,
        'observacao': self.ids['doacaoObservacao'].text,
        'id_pessoa': self.id_pessoa,
        'id_usuario': usuario_atual[0]
    })

    self.manager.transition.direction = 'right'
    self.manager.current = "consultar_doacao"
```

```
def editar_dados(self):
    print(
        'Atualizando:',
        self.ids.doacaoObservacao.text
    )

    operar_doacao("UPDATE", dados={
        'dt_doacao': self.ids['doacaoData'].text,
        'observacao': self.ids['doacaoObservacao'].text,
        'id_doacao': self.id_doacao_editar,
        'id_pessoa': self.id_pessoa
    })

    self.manager.transition.direction = 'right'
    self.manager.current = "consultar_doacao"
```

com o botão cadastrar aparece a tela de cadastro que salva chamando o operar_doacao com 'insert' e os dados preenchidos e o usuario atual

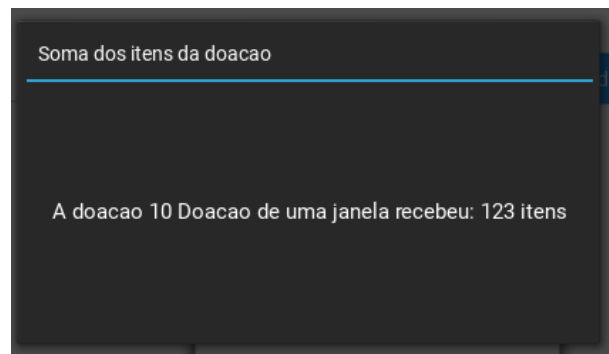
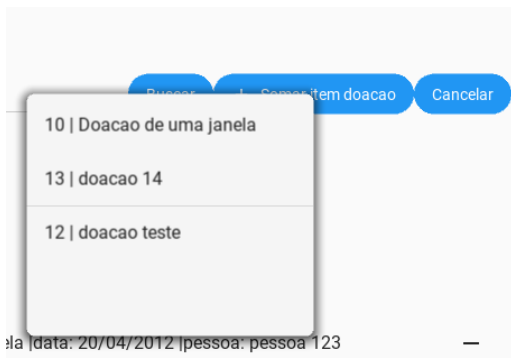
com o botão de engrenagem aparece a tela de edição com os campos preenchidos e ao salvar chama o operar_doacao com 'update' e os dados preenchidos

```
if operador == "INSERT":
    cursor.execute("""
        INSERT INTO DOACAO (dt_doacao, observacao, id_pessoa, id_usuario)
        VALUES (?, ?, ?, ?)""",
        (dados['dt_doacao'], dados['observacao'], dados['id_pessoa'], dados['id_usuario']))

    print("Insert feito com sucesso")
```

```
if operador == "UPDATE":
    cursor.execute("""
        UPDATE DOACAO set dt_doacao = ?, observacao = ?, id_pessoa = ?
        WHERE id_doacao = ?""",
        (dados['dt_doacao'], dados['observacao'], dados['id_doacao'], dados['id_pessoa']))

    print("UPDATE feito com sucesso")
```

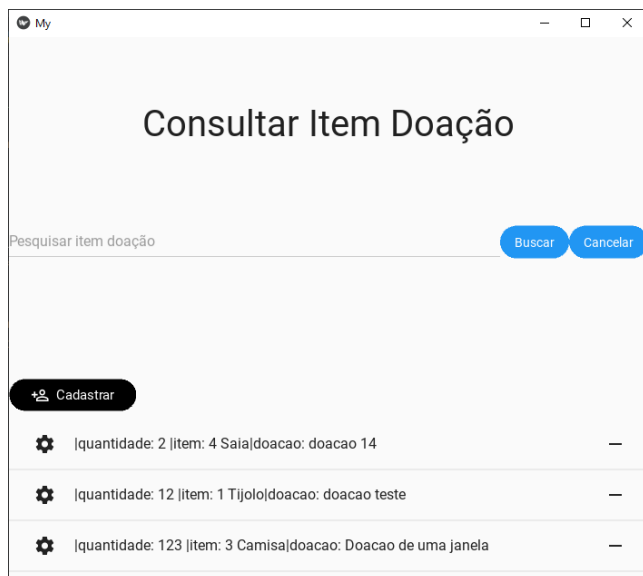


como função especial eu montei um botão que mostra uma lista das doações e ao clicar nelas mostra a quantidade total de itens doados por aquele registro

```
# Funcao no banco de dados sqlite
2 usages 4 Elzeu
def soma_qt_item(id_doacao):
    conn, cursor = conectar_banco_e_cursor()
    resposta_soma = cursor.execute("SELECT SUM(qt_item) AS 'total' FROM item_doacao WHERE id_doacao = ?",
                                   (id_doacao,)).fetchall()
    commit_e_fechar_conexao(conn)
    return resposta_soma
```

por não ter como criar uma função no sqlite com o CREATE FUNCTION o jeito que eu achei foi montar uma função em python que chama o SUM com o argumento enviado

CRUD item doação



```
def pesquisar(self, texto):
    try:
        print(texto)

        item_doacao_resposta_bd = operar_item_doacao('SELECT', {})
        print(item_doacao_resposta_bd)

        # Limpar a lista de doacao
        self.ids.scroll_list_registros.clear_widgets()

        btn_buscar = self.ids.button_buscar

        # Iterar sobre os resultados da consulta
        for row in item_doacao_resposta_bd:
            qt_item, id_item, id_doacao = row

            item = operar_item('SELECT', {'id_item': id_item})
            doacao = operar_doacao('SELECT', {'id_doacao': id_doacao})

            item_registro = ItemDoacaoListItem(qt_item, id_item, item[0][1], id_doacao, doacao[0][2], btn_buscar)

            # Adicionar o item à lista
            self.ids.scroll_list_registros.add_widget(item_registro)
```

```
if operador == "SELECT":

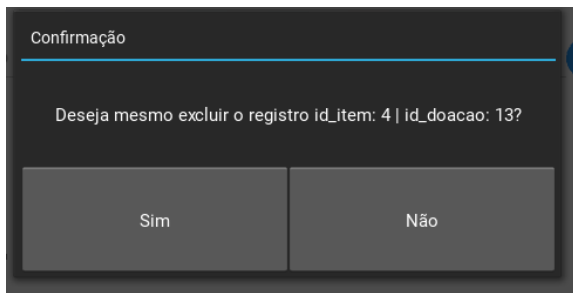
    if 'id_item' not in dados and 'id_doacao' not in dados:
        cursor.execute(
            """ SELECT * FROM ITEM_DOACAO
                ORDER BY qt_item LIMIT 20""",
        )

        resposta = cursor.fetchall()
        commit_e_fechar_conexao(conn)

        return resposta
```

quando clicado no botão de buscar é chamado o operar_item_doacao com 'select' e sempre trazido todos os registros da tabela limitando aos primeiros 20.

Para montar a lista de registros são feitas duas pesquisas, uma na tabela item para pegar o nome do item e outra na tabela doacao para pegar a observacao.



```
def deletar(self):
    @Elizeu
    def confirmar_exclusao_registro():
        try:
            operar_item_doacao('DELETE', dados={'id_doacao': self.id_doacao, 'id_item': self.id_item})
            self.btn_buscar.trigger_action()
            toast("Registro deletado", duration=5)

        except Exception as e:
            toast(f"Error: {e}", duration=5)
            print(e)

    popup = ConfirmationPopup(
        callback=confirmar_exclusao_registro,
        nome_registro=f"id_item: {self.id_item} | id_doacao: {self.id_doacao}"
    )
    popup.open()
```

clicando no botão de menos aparece a tela de confirmação de exclusão e se confirmada é chamada a função `operar_item_doacao` com o 'delete' e os id da doação e o do item

```
if operador == "DELETE":
    cursor.execute("""
        DELETE FROM ITEM_DOACAO
        WHERE id_item = ?
        AND id_doacao = ?""",
        (dados['id_item'], dados['id_doacao']))
    print("DELETE feito com sucesso")

    commit_e_fechar_conexao(conn)
```



```
usage: 1 elizeu
def salvar_dados(self):
    print(
        'Salvando:',
        self.id_item,
        self.id_doacao
    )

    operar_item_doacao("INSERT", dados={
        'qt_item': self.ids['quantidadeItem'].text,
        'id_item': self.id_item,
        'id_doacao': self.id_doacao
    })

    self.manager.transition.direction = 'right'
    self.manager.current = "consultar_item_doacao"
```

```
if operador == "INSERT":
    cursor.execute("""
        INSERT INTO ITEM_DOACAO (qt_item, id_item, id_doacao)
        VALUES (?, ?, ?)""",
        (dados['qt_item'], dados['id_item'], dados['id_doacao']))

    print("Insert feito com sucesso")
```



```
def editar_dados(self):
    print(
        'Atualizando:',
        self.id_item,
        self.id_doacao
    )

    operar_item_doacao("UPDATE", dados={
        'qt_item': self.ids['quantidadeItem'].text,
        'id_item': self.id_item,
        'id_doacao': self.id_doacao,
        'id_item_antes_edicao': self.id_item_antes_edicao,
        'id_doacao_antes_edicao': self.id_doacao_antes_edicao
    })

    self.manager.transition.direction = 'right'
    self.manager.current = "consultar_item_doacao"
```

```
if operador == "UPDATE":
    cursor.execute("""
        UPDATE ITEM_DOACAO set qt_item = ?, id_item = ?, id_doacao = ?
        WHERE id_item = ?
        AND id_doacao = ?""",
        (dados['qt_item'], dados['id_item'], dados['id_doacao'],
        dados['id_item_antes_edicao'], dados['id_doacao_antes_edicao']))

    print("UPDATE feito com sucesso")
```

clicando no botão de cadastrar vai para a tela de cadastro e clicando em salvar chama a função `operar_item_doacao` com o 'insert' e os dados preenchidos

clicando na engrenagem vai para a tela de editar com os campos preenchidos e ao salvar ele chama o `operar_item_doacao` com o 'update' com os dados preenchidos e os ids do item e doacao antes da edição