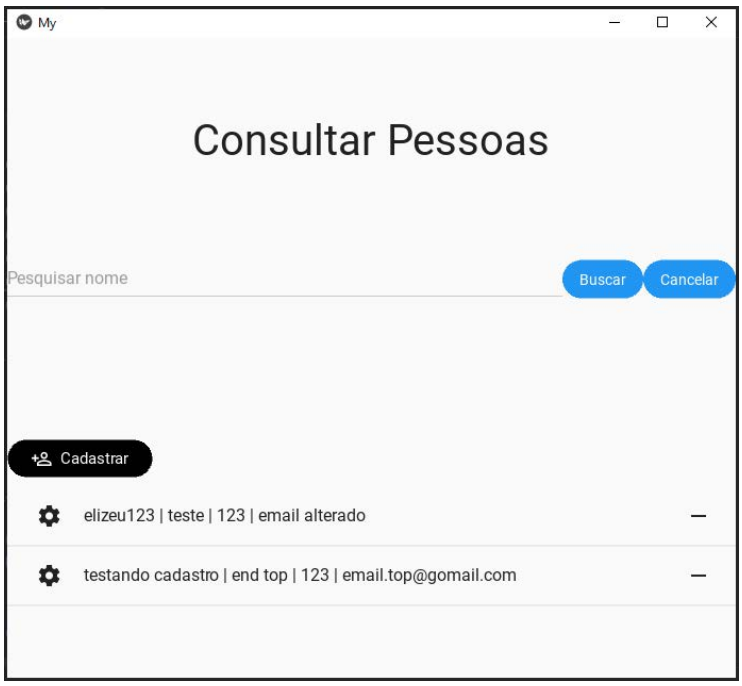
Elizeu Cordeiro do Amaral Vailant : Projeto BD Etapa 5
link Github: https://github.com/movingGreen/projeto_app_ONG

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

```python
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

```
5 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
```

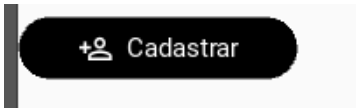
A função recebe o tipo de operação que deve ser feita na tabela PESSOA e um objeto com os dados necessários. Ela verifica se a opção recebida é igual a 'SELECT' e se veio um nome para filtrar, limitando a resposta em 20 registros, depois retorna a resposta.

Classe da apresentação dos registros

```
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

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

Codigo 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.

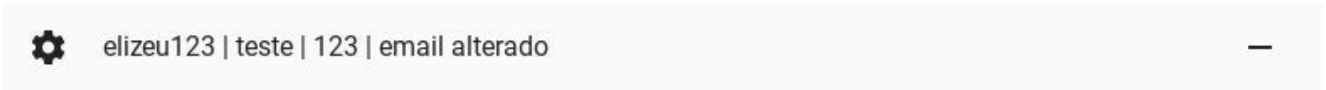
Código dentro da função 'operar_pessoa' de insert

```
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



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 titulo da tela é mudado e a função do botão salvar é mudada para editar com o atribúto 'editar'.

Tela de editar

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.

Codigo 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"
```

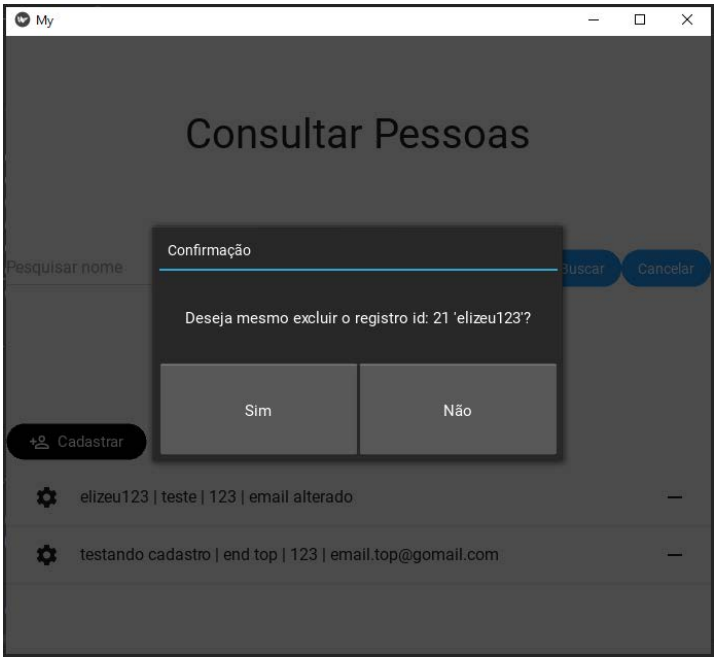
Quando a pessoa clica no botão salvar com a função editar selecionada os dados dos campos de input de texto são enviados para a função 'operar_pessoa' com o operador 'UPDATE' e a 'id' do registro, após isso os campos são limpos e ele volta para a tela de consulta.

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")
```