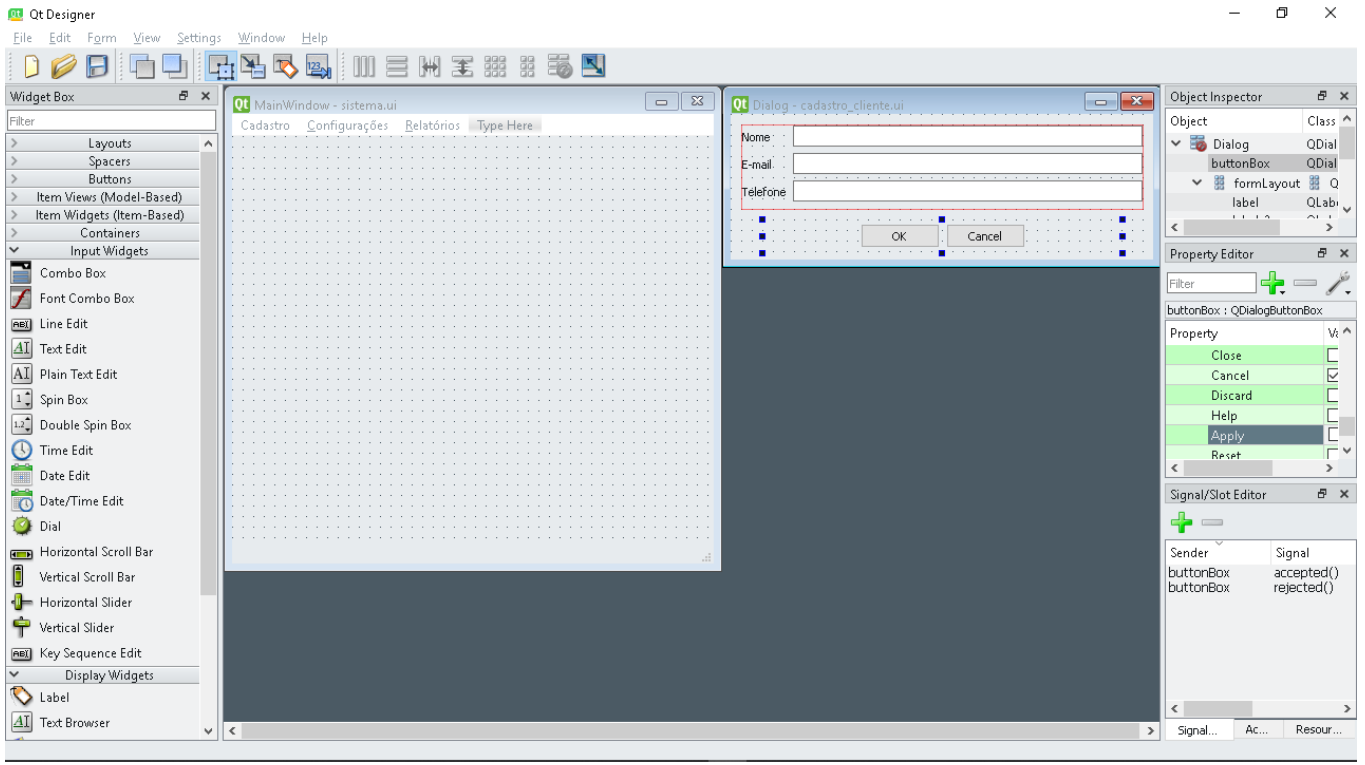
Interface Gráfica
(GUI, Graphic User Interface)

PyQt5 – Aula 7

Qt Designer Tools – Parte 2

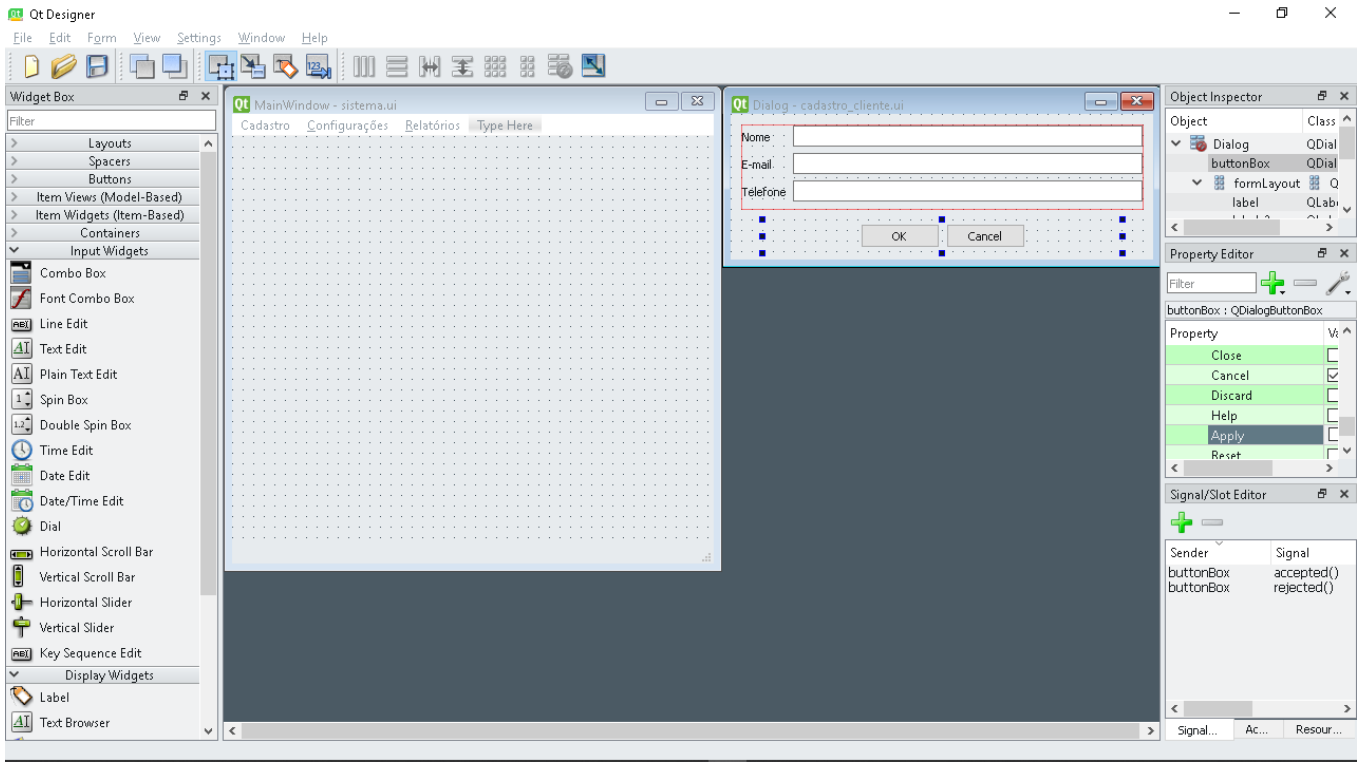
Criando e chamando uma janela de diálogo

Criaremos dois arquivos, chamados sistema.ui (MainWindows) e cadastro_cliente.ui (Janela de diálogo).



Criando e chamando uma janela de diálogo

Na tela principal teremos o menu Cadastro/Clientes abrindo a segunda janela que terá caixas de texto, label e botões.



Criando e chamando uma janela de diálogo

Vamos exportar as duas janelas para .py.

$ pyuic5 sistema.ui -x -o sistema.py

$ pyuic5 cadastro_cliente.ui -o cadastro_cliente.py

Lembre-se:

Se fizer alterações nos arquivos py e depois exportar novamente o arquivo ui, criando o py, as alterações realizadas serão perdidas, pois será gerado um novo arquivo. Esta informação está descrita no início do arquivo gerado.

```
# WARNING! All changes made in this file will be lost!
```

Criando e chamando uma janela de diálogo

No arquivo sistema.py vamos importar a classe Ui_Dialog do arquivo cadastro_cliente.py.

from cadastro_cliente import Ui_Dialog

Criando e chamando uma janela de diálogo

Agora, no arquivo sistema.py vamos implementar o método abrirCadastroCliente, que irá chamar a janela de cadastro do cliente.

```
def abrirCadastroCliente(self):
    JanelaCadastroCliente = QtWidgets.QDialog()
    ui = Ui_Dialog()
    ui.setupUi(JanelaCadastroCliente)
    JanelaCadastroCliente.show()
```

Ainda no sistema.py, no método setupUi, vincule o método abrirCadastroCliente à opção Cadastro/Clientes.

```
self.action_Clientes.triggered.connect(self.abrirCadastroCliente)
```

Criando e chamando uma janela de diálogo

Ao executar o programa e clicar na opção “Clientes” do menu cadastro, você verá a janela abrir e fechar rapidamente, precisamos adicionar uma chamada ao método exec_() da janela de cadastro de clientes, como segue:

```
def abrirCadastroCliente(self):
    JanelaCadastroCliente = QtWidgets.QDialog()
    ui = Ui_Dialog()
    ui.setupUi(JanelaCadastroCliente)
    JanelaCadastroCliente.show()
    JanelaCadastroCliente.exec_()
```

Ao executar o sistema novamente e clicar em Cadastro/Clientes, a janela será aberta.

Criando e chamando uma janela de diálogo

Uma caixa de diálogo como a que usamos retorna uma resposta quando finalizada. Observe no código abaixo, retirado do cadastro_cliente.py que por padrão foi vinculado Dialog.accept ao botão “OK” e Dialog.reject ao botão “Cancel”.

```
self.buttonBox.accepted.connect(Dialog.accept)
self.buttonBox.rejected.connect(Dialog.reject)
```

Desta forma podemos capturar o retorno para podermos validar o botão pressionado.

```
retorno = JanelaCadastroCliente.exec_()
```

Criando e chamando uma janela de diálogo

Podemos usar QtWidgets.QDialog.Accepted para validar um clique no "OK" ou QtWidgets.QDialog.Rejected para o "Cancel".

```
if retorno == QtWidgets.QDialog.Accepted:
    print("Você pressionou OK!!!")
else:
    print("Você pressionou Cancelar")
```

Criando e chamando uma janela de diálogo

Podemos conectar o botão “OK” a outro evento, se necessário. Por exemplo, podemos implementar um método para salvar os dados do cliente (somente como exemplo).

```
...
    self.retranslateUi(Dialog)
    #self.buttonBox.accepted.connect(Dialog.accept)
    self.buttonBox.accepted.connect(self.cadastrarCliente)
    self.buttonBox.rejected.connect(Dialog.reject)
    QtCore.QMetaObject.connectSlotsByName(Dialog)

def cadastrarCliente(self):
    # Aqui ficaria todo código para salvar o cliente
    print("O cliente foi cadastrado")
...
```

Porém, se você mudar a interface no QtDesigner e gerar novamente o arquivo cadastro_cliente.py, Irá perder o método cadastrarCliente.

Criando e chamando uma janela de diálogo

Para não perder o código, temos que separá-lo da interface.

Crie um arquivo chamado metodos_cliente.py e coloque o método cadastrarCliente neste arquivo.

```
def cadastrarCliente():
    # Aqui ficaria todo código para salvar o cliente
    print("O cliente foi cadastrado")
```

Em seguida, no arquivo sistema.py faça a importação.

```
import metodos_cliente
```

Criando e chamando uma janela de diálogo

Agora, vincule o método ao botão “OK” na criação da janela.

```
def abrirCadastroCliente(self):
    JanelaCadastroCliente = QtWidgets.QDialog()
    ui = Ui_Dialog()
    ui.setupUi(JanelaCadastroCliente)
    ui.buttonBox.accepted.connect(metodos_cliente.cadastrarCliente)
    JanelaCadastroCliente.show()
    retorno = JanelaCadastroCliente.exec_()
```

Ao gerarmos uma nova janela do cliente, os métodos relacionados ao cliente não serão apagados. Ao gerar um novo arquivo sistema.py, teremos que adicionar novamente as chamadas, neste caso, salve o arquivo com outro nome e depois cole o que for necessário no novo arquivo gerado. No nosso exemplo vamos precisar guardar o método abrirCadastroCliente e a chamada a este método antes de gerar um novo arquivo sistema.py.

FIM