Modulo 5

Sumário

1 Introdução ao PyQT5

```
      # PySide6 para GUI (interface gráfica) com Qt em
         Python - Instalação
# - A seção original desse curso usou PyQt5 (estamos
   atualizando para PySide6)
# - Essas bibliotecas (PySide e PyQt) usam a biblioteca
   Qt
# - Qt e uma biblioteca usada para a criacao de GUI (
   interface grafica
#   do usuário) escrita em C++.
#   - PySide e PyQt conseguem fazer a ponte (binding)
   entre o Python e a
#   biblioteca para a criação de interfaces gráficas sem
   ter que usar outra
#   linguagem de programação.
# - PySide6 e uma referencia a versão 6 da Qt (Qt 6)
# - Qt e multiplataforma, ou seja, deve funcionar em
   Windows, Linux e Mac.

# Por que mudei de PyQt para PySide na atualização?
# - PySide foi desenvolvido pela The Qt Company (da Nokia
   ), como parte do
#   projeto Qt for Python project - https://doc.qt.io/
   qtforpython/
# - Por usarem a mesma biblioteca (Qt), PySide e PyQt são
    extremamente
#   similares, muitas vezes os codigos são identicos.
   Portanto, mesmo que voce
#   ainda queira usar PyQt, sera muito simples portar os
   codigos. Muitas vezes
#   basta trocar o nome de PySide para PyQt e vice-versa.
# - A maior diferença entre PyQt e PySide está na licença
   :
#   PyQt usa GPL ou commercial e PySide usa LGPL.
#   Em resumo: com PySide, voce tem a permissao uso da
   biblioteca para fins
#   comerciais, sem ter que compartilhar o codigo escrito
    por voce para o
#   publico.
#   Licenças sao topicos complexos, portanto, se oriente
   sobre elas
#   antes de usar qualquer software de terceiros.
#   https://tldrlegal.com/license/gnu-lesser-general-
   public-license-v3-(lgpl-3)
```

2 Instalando o PySide6 no seu ambiente virtual

```
#   Licenças sao topicos complexos, portanto, se
   oriente sobre elas
#   antes de usar qualquer software de terceiros.
#   https://tldrlegal.com/license/gnu-lesser-general-
   public-license-v3-(lgpl-3)
import PySide6.QtCore

# Prints PySide6 version
print(PySide6.__version__)  # type: ignore

# Prints the Qt version used to compile PySide6
print(PySide6.QtCore.__version__)  # type: ignore
```

3 QApplication e QPushButton de PySide6.QtWidgets

```
# QApplication e QPushButton de PySide6.QtWidgets
# QApplication -> O Widget principal da aplicação
# QPushButton -> Um botão
# PySide6.QtWidgets -> Onde estão os widgets do
   PySide6
import sys

from PySide6.QtWidgets import QApplication,
   QPushButton

app = QApplication(sys.argv)

botao = QPushButton('Texto do botão')
botao.setStyleSheet('font-size: 40px;')
botao.show()  # Adiciona o widget na hierarquia e
    exibe a janela

app.exec()  # O loop da aplicação
```

4 QWidget e QLayout de PySide6.QtWidgets

```
# QWidget e QLayout de PySide6.QtWidgets
# QWidget -> generico
# QLayout -> Um widget de layout que recebe
   outros widgets
import sys
```

```
from PySide6.QtWidgets import QApplication,
   QGridLayout, QPushButton, QWidget

app = QApplication(sys.argv)

botao = QPushButton('Texto do botão')
botao.setStyleSheet('font-size: 80px;')

botao2 = QPushButton('Botão 2')
botao2.setStyleSheet('font-size: 40px;')

botao3 = QPushButton('Botão 3')
botao3.setStyleSheet('font-size: 40px;')

central_widget = QWidget()

layout = QGridLayout()
central_widget.setLayout(layout)

layout.addWidget(botao, 1, 1, 1, 1)
layout.addWidget(botao2, 1, 2, 1, 1)
layout.addWidget(botao3, 3, 1, 1, 2)

central_widget.show()  # Central widget entre na
   hierarquia e mostre sua janela
app.exec()  # O loop da aplicação
```

5 QMainWindow e centralWidget

```
# QMainWindow e centralWidget
# -> QApplication (app)
#   -> QMainWindow (window->setCentralWidget)
#       -> CentralWidget (central_widget)
#           -> Layout (layout)
#               -> Widget 1 (botao1)
#               -> Widget 2 (botao2)
#               -> Widget 3 (botao3)
#   -> show
# -> exec
import sys

from PySide6.QtWidgets import (QApplication,
   QGridLayout, QMainWindow,
                              QPushButton, QWidget)
```

```
app = QApplication(sys.argv)
window = QMainWindow()
central_widget = QWidget()
window.setCentralWidget(central_widget)
window.setWindowTitle('Minha janela bonita')

botao1 = QPushButton('Texto do botão')
botao1.setStyleSheet('font-size: 80px;')

botao2 = QPushButton('Botão 2')
botao2.setStyleSheet('font-size: 40px;')

botao3 = QPushButton('Botão 3')
botao3.setStyleSheet('font-size: 40px;')

layout = QGridLayout()
central_widget.setLayout(layout)

layout.addWidget(botao1, 1, 1, 1, 1)
layout.addWidget(botao2, 1, 2, 1, 1)
layout.addWidget(botao3, 3, 1, 1, 2)


def slot_example(status_bar):
    status_bar.showMessage('O meu slot foi
       executado')


# statusBar
status_bar = window.statusBar()
status_bar.showMessage('Mostrar mensagem na barra
   ')

# menuBar
menu = window.menuBar()
primeiro_menu = menu.addMenu('Primeiro menu')
primeira_acao = primeiro_menu.addAction('Primeira
    ação')
primeira_acao.triggered.connect(  # type:ignore
    lambda: slot_example(status_bar)
)


window.show()
app.exec()  # O loop da aplicação
```

6 O básico sobre Signal e Slots (eventos e documentação)

```python
# O básico sobre Signal e Slots (eventos e
   documentação)
import sys

from PySide6.QtCore import Slot
from PySide6.QtWidgets import (QApplication,
   QGridLayout, QMainWindow,
                               QPushButton, QWidget)

app = QApplication(sys.argv)
window = QMainWindow()
central_widget = QWidget()
window.setCentralWidget(central_widget)
window.setWindowTitle('Minha janela bonita')

botao1 = QPushButton('Texto do botão')
botao1.setStyleSheet('font-size: 80px;')

botao2 = QPushButton('Botão 2')
botao2.setStyleSheet('font-size: 40px;')

botao3 = QPushButton('Botão 3')
botao3.setStyleSheet('font-size: 40px;')

layout = QGridLayout()
central_widget.setLayout(layout)

layout.addWidget(botao1, 1, 1, 1, 1)
layout.addWidget(botao2, 1, 2, 1, 1)
layout.addWidget(botao3, 3, 1, 1, 2)


@Slot()
def slot_example(status_bar):
    def inner():
        status_bar.showMessage('O meu slot foi
           executado')
    return inner


@Slot()
def outro_slot(checked):
    print('Está marcado?', checked)
```

```
@Slot()
def terceiro_slot(action):
    def inner():
        outro_slot(action.isChecked())
    return inner


# statusBar
status_bar = window.statusBar()
status_bar.showMessage('Mostrar mensagem na barra
   ')

# menuBar
menu = window.menuBar()
primeiro_menu = menu.addMenu('Primeiro menu')
primeira_acao = primeiro_menu.addAction('Primeira
    ação')
primeira_acao.triggered.connect(slot_example(
   status_bar)) # type:ignore

segunda_action = primeiro_menu.addAction('Segunda
    ação')
segunda_action.setCheckable(True)
segunda_action.toggled.connect(outro_slot) #
   type:ignore
segunda_action.hovered.connect(terceiro_slot(
   segunda_action)) # type:ignore

botao1.clicked.connect(terceiro_slot(
   segunda_action)) # type:ignore

window.show()
app.exec() # O loop da aplicação
```

7 Trabalhando com classes e herança no PySide6

```
# Trabalhando com classes e herança no PySide6
import sys

from PySide6.QtCore import Slot
from PySide6.QtWidgets import (QApplication,
   QGridLayout, QMainWindow,
                               QPushButton,
                                    QWidget)


class MyWindow(QMainWindow):
    def __init__(self, parent=None):
        super().__init__(parent)

        self.central_widget = QWidget()

        self.setCentralWidget(self.central_widget
           )
        self.setWindowTitle('Minha janela bonita'
           )

        # Botão
        self.botao1 = self.make_button('Texto do
           botão')
        self.botao1.clicked.connect(self.
           segunda_acao_marcada)  # type: ignore

        self.botao2 = self.make_button('Botão 2')

        self.botao3 = self.make_button('Terceiro'
           )

        self.grid_layout = QGridLayout()
        self.central_widget.setLayout(self.
           grid_layout)

        self.grid_layout.addWidget(self.botao1,
           1, 1, 1, 1)
        self.grid_layout.addWidget(self.botao2,
           1, 2, 1, 1)
        self.grid_layout.addWidget(self.botao3,
           3, 1, 1, 2)

        # statusBar
        self.status_bar = self.statusBar()
        self.status_bar.showMessage('Mostrar
           mensagem na barra')

        # menuBar
        self.menu = self.menuBar()
        self.primeiro_menu = self.menu.addMenu('
           Primeiro menu')
        self.primeira_acao = self.primeiro_menu.
```

```
                addAction('Primeira ação')
            self.primeira_acao.triggered.connect(  #
                type:ignore
                 self.muda_mensagem_da_status_bar)

            self.segunda_action = self.primeiro_menu.
                addAction('Segunda ação')
            self.segunda_action.setCheckable(True)
            self.segunda_action.toggled.connect(  #
                type:ignore
                 self.segunda_acao_marcada)
            self.segunda_action.hovered.connect(  #
                type:ignore
                 self.segunda_acao_marcada)

        @Slot()
        def muda_mensagem_da_status_bar(self):
            self.status_bar.showMessage('O meu slot
                foi executado')

        @Slot()
        def segunda_acao_marcada(self):
            print('Está marcado?', self.
                segunda_action.isChecked())

        def make_button(self, text):
            btn = QPushButton(text)
            btn.setStyleSheet('font-size: 80px;')
            return btn


    if __name__ == '__main__':
        app = QApplication(sys.argv)
        window = MyWindow()
        window.show()
        app.exec()  # O loop da aplicação
```

8 calculadora

8.1 Criando a janela principal

main.py

```
    import sys

from main_window import MainWindow
from PySide6.QtWidgets import QApplication, QLabel

if __name__ == '__main__':
    app = QApplication(sys.argv)
    window = MainWindow()

    label1 = QLabel('O meu texto')
    label1.setStyleSheet('font-size: 150px;')
    window.v_layout.addWidget(label1)
    window.adjustFixedSize()

    window.show()
    app.exec()
```

mainWindow.py

```
    from PySide6.QtWidgets import QMainWindow,
       QVBoxLayout, QWidget


class MainWindow(QMainWindow):
def __init__(self, parent: QWidget | None = None, *
   args, **kwargs) -> None:
    super().__init__(parent, *args, **kwargs)

    # Configurando o layout básico
    self.cw = QWidget()
    self.v_layout = QVBoxLayout()
    self.cw.setLayout(self.v_layout)
    self.setCentralWidget(self.cw)

    # Titulo da janela
    self.setWindowTitle('Calculadora')

def adjustFixedSize(self):
    # ultima coisa a ser feita
    self.adjustSize()
    self.setFixedSize(self.width(), self.height())
```

8.2 variáveis e método p/ adicionar widgets no vlayout

```
        import sys

from main_window import MainWindow
from PySide6.QtWidgets import QApplication, QLabel
from PySide6.QtGui import QIcon
from PySide6.QtWidgets import QApplication
from variables import WINDOW_ICON_PATH

if __name__ == '__main__':
    # Cria a aplicação
    app = QApplication(sys.argv)
    window = MainWindow()

    label1 = QLabel('O meu texto')
    label1.setStyleSheet('font-size: 150px;')
    window.v_layout.addWidget(label1)
    window.adjustFixedSize()
    # Define o icone
    icon = QIcon(str(WINDOW_ICON_PATH))
    window.setWindowIcon(icon)
    app.setWindowIcon(icon)

    # Executa tudo
    window.adjustFixedSize()
    window.show()
    app.exec()
```

MainWindow

```
        # ultima coisa a ser feita
        self.adjustSize()
        self.setFixedSize(self.width(), self.height())

    def addWidgetToVLayout(self, widget: QWidget):
        self.v_layout.addWidget(widget)
```

variables.py

```
    from pathlib import Path

    ROOT_DIR = Path(__file__).parent
    FILES_DIR = ROOT_DIR / 'files'
    WINDOW_ICON_PATH = FILES_DIR / 'icon.png'
```

8.3 Configurando o layout básico

MainWindow.py

```
        self.cw = QWidget()
        self.v_layout = QVBoxLayout()
        self.cw.setLayout(self.v_layout)
        self.vLayout = QVBoxLayout()
        self.cw.setLayout(self.vLayout)
        self.setCentralWidget(self.cw)
        # Titulo da janela
        def adjustFixedSize(self):
        self.setFixedSize(self.width(), self.height())

    def addWidgetToVLayout(self, widget: QWidget):
        self.v_layout.addWidget(widget)
        self.vLayout.addWidget(widget)
```

8.4 QLineEdit e o display

display.py

```
        from PySide6.QtCore import Qt
        from PySide6.QtWidgets import QLineEdit
        from variables import BIG_FONT_SIZE,
           MINIMUM_WIDTH, TEXT_MARGIN


class Display(QLineEdit):
    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)
        self.configStyle()

    def configStyle(self):
        margins = [TEXT_MARGIN for _ in range(4)]
        self.setStyleSheet(f'font-size: {BIG_FONT_SIZE}px
           ;')
        self.setMinimumHeight(BIG_FONT_SIZE * 2)
        self.setMinimumWidth(MINIMUM_WIDTH)
        self.setAlignment(Qt.AlignmentFlag.AlignRight)
        self.setTextMargins(*margins)
```

main.py

```
        import sys

    from display import Display
    from main_window import MainWindow
    from PySide6.QtGui import QIcon
    from PySide6.QtWidgets import QApplication
    @@ -15,6 +16,10 @@
        window.setWindowIcon(icon)
        app.setWindowIcon(icon)

        # Display adicionado
        display = Display()
        window.addToVLayout(display)
        # Executa tudo
        window.adjustFixedSize()
        window.show()
```

mainWindow.py

```
        self.adjustSize()
        self.setFixedSize(self.width(), self.height())

    def addWidgetToVLayout(self, widget: QWidget):
    def addToVLayout(self, widget: QWidget):
        self.vLayout.addWidget(widget)# new commit
```

variables.py

```
        ROOT_DIR = Path(__file__).parent
        FILES_DIR = ROOT_DIR / 'files'
        WINDOW_ICON_PATH = FILES_DIR / 'icon.png'

        # Sizing
        BIG_FONT_SIZE = 40
        MEDIUM_FONT_SIZE = 24
        SMALL_FONT_SIZE = 18
        TEXT_MARGIN = 15
        MINIMUM_WIDTH = 500
```

8.5 criando um QLabel para mostrar informações main

info.py

```
        from PySide6.QtCore import Qt
from PySide6.QtWidgets import QLabel, QWidget
from variables import SMALL_FONT_SIZE


class Info(QLabel):
    def __init__(self, text: str, parent: QWidget | None
       = None) -> None:
        super().__init__(text, parent)
        self.configStyle()

    def configStyle(self):
        self.setStyleSheet(f'font-size: {SMALL_FONT_SIZE}
           px;')
        self.setAlignment(Qt.AlignmentFlag.AlignRight)
```

main.py

```
        import sys
```

```
from display import Display
from info import Info
from main_window import MainWindow
from PySide6.QtGui import QIcon
from PySide6.QtWidgets import QApplication
@@ -16,6 +17,10 @@
    window.setWindowIcon(icon)
    app.setWindowIcon(icon)

    # Info
    info = Info('2.0 ^ 10.0 = 1024')
    window.addToVLayout(info)

    # Display
    display = Display()
    window.addToVLayout(display)
```

8.6 configurando o PyQt Dark Theme (qdarktheme) no PySide6

main.py

```
        from main_window import MainWindow
from PySide6.QtGui import QIcon
from PySide6.QtWidgets import QApplication
from styles import setupTheme
from variables import WINDOW_ICON_PATH

if __name__ == '__main__':
    # Cria a aplicação
    app = QApplication(sys.argv)
    setupTheme()
    window = MainWindow()

    # Define o icone
```

styles.py

```
        # QSS - Estilos do QT for Python
# https://doc.qt.io/qtforpython/tutorials/basictutorial/
   widgetstyling.html
# Dark Theme
# https://pyqtdarktheme.readthedocs.io/en/latest/
   how_to_use.html
import qdarktheme
from variables import (DARKER_PRIMARY_COLOR,
   DARKEST_PRIMARY_COLOR,
                       PRIMARY_COLOR)
```

```
qss = f"""
    PushButton[cssClass="specialButton"] {{
        color: #fff;
        background: {PRIMARY_COLOR};
    }}
    PushButton[cssClass="specialButton"]:hover {{
        color: #fff;
        background: {DARKER_PRIMARY_COLOR};
    }}
    PushButton[cssClass="specialButton"]:pressed {{
        color: #fff;
        background: {DARKEST_PRIMARY_COLOR};
    }}
"""


def setupTheme():
    qdarktheme.setup_theme(
        theme='dark',
        corner_shape='rounded',
        custom_colors={
            "[dark]": {
                "primary": f"{PRIMARY_COLOR}",
            },
            "[light]": {
                "primary": f"{PRIMARY_COLOR}",
            },
        },
        additional_qss=qss
    )
```

variables.py

```
        FILES_DIR = ROOT_DIR / 'files'
WINDOW_ICON_PATH = FILES_DIR / 'icon.png'

# Colors
(PRIMARY_COLOR = '#1e81b0'
DARKER_PRIMARY_COLOR = '#16658a'
DARKEST_PRIMARY_COLOR = '#115270'
)
# Sizing
BIG_FONT_SIZE = 40
MEDIUM_FONT_SIZE = 24
```

8.7 criando um botão com QPushButton no PySide6

criando um botão com QPushButton no PySide6

```
        from PySide6.QtWidgets import QPushButton
from variables import MEDIUM_FONT_SIZE


class Button(QPushButton):
    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)
        self.configStyle()

    def configStyle(self):
        font = self.font()
        font.setPixelSize(MEDIUM_FONT_SIZE)
        self.setFont(font)
        self.setMinimumSize(75, 75)
        self.setProperty('cssClass', 'specialButton')
```

main.py

```
        import sys

from buttons import Button
from display import Display
from info import Info
from main_window import MainWindow
@@ -27,6 +28,12 @@
    display = Display()
    window.addToVLayout(display)

    button = Button('Texto do botão')
    window.addToVLayout(button)

    button2 = Button('Texto do botão')
    window.addToVLayout(button2)

    # Executa tudo
    window.adjustFixedSize()
    window.show()
```

styles.py

```
        qss = f"""
    PushButton[cssClass="specialButton"] {{
    QPushButton[cssClass="specialButton"] {{
        color: #fff;
        background: {PRIMARY_COLOR};
    }}
    PushButton[cssClass="specialButton"]:hover {{
    QPushButton[cssClass="specialButton"]:hover {{
        color: #fff;
        background: {DARKER_PRIMARY_COLOR};
    }}
    PushButton[cssClass="specialButton"]:pressed {{
    QPushButton[cssClass="specialButton"]:pressed {{
        color: #fff;
        background: {DARKEST_PRIMARY_COLOR};
    }}
```

8.8 grid de botões com QGridLayout

buttons.py

```
        from PySide6.QtWidgets import QPushButton
from PySide6.QtWidgets import QGridLayout, QPushButton
from variables import MEDIUM_FONT_SIZE


@@ -13,3 +13,16 @@ def configStyle(self):
        self.setFont(font)
        self.setMinimumSize(75, 75)
        self.setProperty('cssClass', 'specialButton')


class ButtonsGrid(QGridLayout):
    def __init__(self, *args, **kwargs) -> None:
        super().__init__(*args, **kwargs)

        self._grid_mask = [
            ['C', '', '^', '/'],
            ['7', '8', '9', '*'],
            ['4', '5', '6', '-'],
            ['1', '2', '3', '+'],
            ['',  '0', '.', '='],
        ]
```

main.py

```
        import sys

from buttons import Button
from buttons import Button, ButtonsGrid
from display import Display
from info import Info
from main_window import MainWindow
@@ -22,17 +22,15 @@
```

```
    # Info
    info = Info('2.0 ^ 10.0 = 1024')
    window.addToVLayout(info)
    window.addWidgetToVLayout(info)

    # Display
    display = Display()
    window.addToVLayout(display)
    window.addWidgetToVLayout(display)

    button = Button('Texto do botão')
    window.addToVLayout(button)

    button2 = Button('Texto do botão')
    window.addToVLayout(button2)
    # Grid
    buttonsGrid = ButtonsGrid()
    window.vLayout.addLayout(buttonsGrid)

    # Executa tudo
    window.adjustFixedSize()
```

mainWindow.py

```
        self.adjustSize()
        self.setFixedSize(self.width(), self.height())

    def addToVLayout(self, widget: QWidget):
    def addWidgetToVLayout(self, widget: QWidget):
        self.vLayout.addWidget(widget)
```

8.9 criando a grid de botões

buttons.py

```
        from PySide6.QtWidgets import QGridLayout,
           QPushButton
from utils import isEmpty, isNumOrDot
from variables import MEDIUM_FONT_SIZE


@@ -12,17 +13,27 @@ def configStyle(self):
        font.setPixelSize(MEDIUM_FONT_SIZE)
        self.setFont(font)
        self.setMinimumSize(75, 75)
        self.setProperty('cssClass', 'specialButton')
```

```
class ButtonsGrid(QGridLayout):
    def __init__(self, *args, **kwargs) -> None:
        super().__init__(*args, **kwargs)

        self._grid_mask = [
        self._gridMask = [
            ['C', '', '^', '/'],
            ['7', '8', '9', '*'],
            ['4', '5', '6', '-'],
            ['1', '2', '3', '+'],
            ['',  '0', '.', '='],
        ]
        self._makeGrid()

    def _makeGrid(self):
        for rowNumber, rowData in enumerate(self.
           _gridMask):
            for colNumber, buttonText in enumerate(
               rowData):
                button = Button(buttonText)

                if not isNumOrDot(buttonText) and not
                   isEmpty(buttonText):
                    button.setProperty('cssClass', '
                       specialButton')

                self.addWidget(button, rowNumber,
                   colNumber)
```

main.py

```
        import sys

from buttons import Button, ButtonsGrid
from buttons import ButtonsGrid
from display import Display
from info import Info
from main_window import MainWindow
```

utils.py

```
        import re

NUM_OR_DOT_REGEX = re.compile(r'^[0-9.]$')


def isNumOrDot(string: str):
    return bool(NUM_OR_DOT_REGEX.search(string))
```

```
def isEmpty(string: str):
    return len(string) == 0
```

8.10 criando um Slot com dados para o Signal clicked

buttons.py

```
        self.setCheckable(True)


class ButtonsGrid(QGridLayout):
    def __init__(self, *args, **kwargs) -> None:
    def __init__(self, display: Display, *args,
       **kwargs) -> None:
        super().__init__(*args, **kwargs)

        self._gridMask = [
@@ -26,6 +29,7 @@ def __init__(self, *args, **
   kwargs) -> None:
            ['1', '2', '3', '+'],
            ['',  '0', '.', '='],
        ]
        self.display = display
        self._makeGrid()

    def _makeGrid(self):
@@ -37,3 +41,18 @@ def _makeGrid(self):
                button.setProperty('cssClass'
                    , 'specialButton')

            self.addWidget(button, rowNumber,
                colNumber)
            buttonSlot = self.
               _makeButtonDisplaySlot(
                self.
                   _insertButtonTextToDisplay
                   ,
                button,
            )
            button.clicked.connect(buttonSlot
               )  # type: ignore

    def _makeButtonDisplaySlot(self, func, *args,
        **kwargs):
        @Slot(bool)
```

```
                def realSlot(_):
                    func(*args, **kwargs)
                return realSlot

            def _insertButtonTextToDisplay(self, button):
                button_text = button.text()
                self.display.insert(button_text)
```

main.py

```
        window.addWidgetToVLayout(display)

    # Grid
    buttonsGrid = ButtonsGrid()
    buttonsGrid = ButtonsGrid(display)
    window.vLayout.addLayout(buttonsGrid)

    # Executa tudo
```

8.11 permitindo apenas números válidos no display

buttons.py

```
        from display import Display
from PySide6.QtCore import Slot
from PySide6.QtWidgets import QGridLayout, QPushButton
from utils import isEmpty, isNumOrDot
from utils import isEmpty, isNumOrDot, isValidNumber
from variables import MEDIUM_FONT_SIZE


@@ -15,7 +15,6 @@ def configStyle(self):
        font.setPixelSize(MEDIUM_FONT_SIZE)
        self.setFont(font)
        self.setMinimumSize(75, 75)
        self.setCheckable(True)


class ButtonsGrid(QGridLayout):
@@ -54,5 +53,10 @@ def realSlot(_):
        return realSlot

    def _insertButtonTextToDisplay(self, button):
        button_text = button.text()
        self.display.insert(button_text)
        buttonText = button.text()
        newDisplayValue = self.display.text() +
           buttonText
        if not isValidNumber(newDisplayValue):
            return

        self.display.insert(buttonText)
```

utils.py

```
        return bool(NUM_OR_DOT_REGEX.search(string))


def isValidNumber(string: str):
    valid = False
    try:
        float(string)
        valid = True
    except ValueError:
        valid = False
    return valid


def isEmpty(string: str):
    return len(string) == 0
```

8.12 Info (QLabel), TYPECHECKING, getter e setter

buttons.py

```
        from display import Display
from typing import TYPE_CHECKING

from PySide6.QtCore import Slot
from PySide6.QtWidgets import QGridLayout, QPushButton
from utils import isEmpty, isNumOrDot, isValidNumber
from variables import MEDIUM_FONT_SIZE

if TYPE_CHECKING:
    from display import Display
    from info import Info


class Button(QPushButton):
    def __init__(self, *args, **kwargs):
@@ -18,7 +23,9 @@ def configStyle(self):


class ButtonsGrid(QGridLayout):
    def __init__(self, display: Display, *args, **kwargs)
        -> None:
    def __init__(
            self, display: 'Display', info: 'Info', *args
               , **kwargs
    ) -> None:
        super().__init__(*args, **kwargs)

        self._gridMask = [
@@ -29,8 +36,19 @@ def __init__(self, display: Display, *
   args, **kwargs) -> None:
            ['', '0', '.', '='],
        ]
        self.display = display
        self.info = info
        self._equation = ''
        self._makeGrid()

    @property
    def equation(self):
        return self._equation

    @equation.setter
    def equation(self, value):
        self._equation = value
        self.info.setText(value)

    def _makeGrid(self):
        for rowNumber, rowData in enumerate(self.
           _gridMask):
            for colNumber, buttonText in enumerate(
               rowData):
```

main.py

```
        app.setWindowIcon(icon)

    # Info
    info = Info('2.0 ^ 10.0 = 1024')
    info = Info('Sua conta')
    window.addWidgetToVLayout(info)

    # Display
    display = Display()
    window.addWidgetToVLayout(display)

    # Grid
    buttonsGrid = ButtonsGrid(display)
    buttonsGrid = ButtonsGrid(display, info)
    window.vLayout.addLayout(buttonsGrid)

    # Executa tudo
```

8.13 iniciando a configuração dos botões especiais

buttons.py

```
        if not isNumOrDot(buttonText) and not isEmpty(
           buttonText):
        button.setProperty('cssClass', 'specialButton')
        self._configSpecialButton(button)

    self.addWidget(button, rowNumber, colNumber)
    buttonSlot = self._makeButtonDisplaySlot(
        self._insertButtonTextToDisplay,
        button,
    )
    button.clicked.connect(buttonSlot)  # type: ignore
    slot = self._makeSlot(self._insertButtonTextToDisplay
       , button)
    self._connectButtonClicked(button, slot)

def _makeButtonDisplaySlot(self, func, *args, **kwargs):
def _connectButtonClicked(self, button, slot):
button.clicked.connect(slot)  # type: ignore

def _configSpecialButton(self, button):
text = button.text()

if text == 'C':
self._connectButtonClicked(button, self._clear)

def _makeSlot(self, func, *args, **kwargs):
@Slot(bool)
def realSlot(_):
func(*args, **kwargs)
@@ -78,3 +85,7 @@ def _insertButtonTextToDisplay(self,
   button):
return

self.display.insert(buttonText)

def _clear(self):
print('Vou fazer outra coisa aqui')
self.display.clear()
```

8.14 botões especiais de operadores, clear e equation

buttons.py

```
        self.display = display
        self.info = info
        self._equation = ''
        self._equationInitialValue = 'Sua conta'
        self._left = None
        self._right = None
        self._op = None

        self.equation = self._equationInitialValue
        self._makeGrid()

    @property
@@ -71,8 +77,14 @@ def _configSpecialButton(self, button)
   :
        if text == 'C':
            self._connectButtonClicked(button, self.
               _clear)

        if text in '+-/*':
            self._connectButtonClicked(
                button,
                self._makeSlot(self._operatorClicked,
                   button)
            )

    def _makeSlot(self, func, *args, **kwargs):
        @Slot(bool)
        @ Slot(bool)
        def realSlot(_):
            func(*args, **kwargs)
        return realSlot
@@ -87,5 +99,27 @@ def _insertButtonTextToDisplay(self,
   button):
        self.display.insert(buttonText)

    def _clear(self):
        print('Vou fazer outra coisa aqui')
        self._left = None
        self._right = None
        self._op = None
        self.equation = self._equationInitialValue
        self.display.clear()
    def _operatorClicked(self, button):
        buttonText = button.text()  # +-/* (etc...)
        displayText = self.display.text()  # Devera ser
           meu numero left
        self.display.clear()  # Limpa o display

        # Se a pessoa clicou no operador sem
        # configurar qualquer numero
        if not isValidNumber(displayText) and self._left
           is None:
            print('Não tem nada para colocar no valor da
               esquerda')
            return

        # Se houver algo no numero da esquerda,
        # nao fazemos nada. Aguardaremos o numero da
           direita.
        if self._left is None:
            self._left = float(displayText)

        self._op = buttonText
        self.equation = f'{self._left} {self._op} ??'
```

8.15 configurando o botão de igual e o número da direita

buttons.py

```
        self._makeSlot(self._operatorClicked, button)
        )

    if text in '=':
        self._connectButtonClicked(button, self._eq)

def _makeSlot(self, func, *args, **kwargs):
    @ Slot(bool)
    def realSlot(_):
@@ -123,3 +126,24 @@ def _operatorClicked(self, button):

    self._op = buttonText
    self.equation = f'{self._left} {self._op} ??'

def _eq(self):
    displayText = self.display.text()

    if not isValidNumber(displayText):
        print('Sem nada para a direita')
        return
    self._right = float(displayText)
    self.equation = f'{self._left} {self._op} {self.
       _right}'
    result = 0.0
        import math
from typing import TYPE_CHECKING

from PySide6.QtCore import Slot
@@ -77,7 +78,7 @@ def _configSpecialButton(self, button):
        if text == 'C':
            self._connectButtonClicked(button, self.
               _clear)

        if text in '+-/*':
        if text in '+-/*^':
            self._connectButtonClicked(
                button,
                self._makeSlot(self._operatorClicked,
                   button)
@@ -136,14 +137,22 @@ def _eq(self):

        self._right = float(displayText)
        self.equation = f'{self._left} {self._op} {self.
           _right}'
        result = 0.0
        result = 'error'

        try:
            result = eval(self.equation)
            if '^' in self.equation and isinstance(self.
               _left, float):
                result = math.pow(self._left, self._right
                   )
            else:
                result = eval(self.equation)
        except ZeroDivisionError:
            print('Zero Division Error')
        except OverflowError:
            print('Numero muito grande')

        self.display.clear()
        self.info.setText(f'{self.equation} = {result}')
        self._left = result
        self._right = None
        if result == 'error':
            self._left = None
```

8.16 configurando o backspace do display no botão back

buttons.py

```
        super().__init__(*args, **kwargs)

        self._gridMask = [
            ['C', 'space', '^', '/'],
            ['C', 'D', '^', '/'],
            ['7', '8', '9', '*'],
            ['4', '5', '6', '-'],
            ['1', '2', '3', '+'],
@@ -78,6 +78,9 @@ def _configSpecialButton(self, button):
        if text == 'C':
            self._connectButtonClicked(button, self.
               _clear)

        if text in 'D':
            self._connectButtonClicked(button, self.
               display.backspace)

        if text in '+-/*^':
            self._connectButtonClicked(
                button,
```

8.17 diálogos com o usuário com QMessageBox

buttons.py

```
        if TYPE_CHECKING:
    from display import Display
    from info import Info
    from main_window import MainWindow


class Button(QPushButton):
@@ -25,7 +26,8 @@ def configStyle(self):

class ButtonsGrid(QGridLayout):
    def __init__(
            self, display: 'Display', info: 'Info', *args
               , **kwargs
            self, display: 'Display', info: 'Info',
               window: 'MainWindow',
            *args, **kwargs
    ) -> None:
        super().__init__(*args, **kwargs)

@@ -38,6 +40,7 @@ def __init__(
        ]
        self.display = display
        self.info = info
        self.window = window
        self._equation = ''
        self._equationInitialValue = 'Sua conta'
        self._left = None
@@ -120,7 +123,7 @@ def _operatorClicked(self, button):
        # Se a pessoa clicou no operador sem
        # configurar qualquer numero
        if not isValidNumber(displayText) and self._left
           is None:
            print('Não tem nada para colocar no valor da
               esquerda')
            self._showError('Voce nao digitou nada.')
            return

        # Se houver algo no numero da esquerda,
@@ -135,7 +138,7 @@ def _eq(self):
        displayText = self.display.text()

        if not isValidNumber(displayText):
            print('Sem nada para a direita')
            self._showError('Conta incompleta.')
            return

        self._right = float(displayText)
@@ -148,9 +151,9 @@ def _eq(self):
            else:
                result = eval(self.equation)
        except ZeroDivisionError:
            print('Zero Division Error')
            self._showError('Divisão por zero.')
        except OverflowError:
            print('Numero muito grande')
            self._showError('Essa conta não pode ser
               realizada.')

        self.display.clear()
        self.info.setText(f'{self.equation} = {result}')
@@ -159,3 +162,18 @@ def _eq(self):
        if result == 'error':
            self._left = None

    def _makeDialog(self, text):
        msgBox = self.window.makeMsgBox()
        msgBox.setText(text)
        return msgBox

    def _showError(self, text):
        msgBox = self._makeDialog(text)
        msgBox.setIcon(msgBox.Icon.Critical)
        msgBox.exec()

    def _showInfo(self, text):
        msgBox = self._makeDialog(text)
        msgBox.setIcon(msgBox.Icon.Information)
        msgBox.exec()
```

main.py

```
window.addWidgetToVLayout(display)

# Grid
buttonsGrid = ButtonsGrid(display, info)
buttonsGrid = ButtonsGrid(display, info, window)
window.vLayout.addLayout(buttonsGrid)

# Executa tudo
```

mainWindow.py

```
from PySide6.QtWidgets import QMainWindow,
   QVBoxLayout, QWidget
from PySide6.QtWidgets import QMainWindow,
   QMessageBox, QVBoxLayout, QWidget


class MainWindow(QMainWindow):
@@ -21,3 +21,6 @@ def adjustFixedSize(self):

def addWidgetToVLayout(self, widget: QWidget):
    self.vLayout.addWidget(widget)

def makeMsgBox(self):
    return QMessageBox(self)
```

9 Criando e compilando um arquivo UI com o Qt Designer

aula203-qtdesigner/src/window.py

```
          # -*- coding: utf-8 -*-

## Form generated from reading UI file 'window.ui'
##
## Created by: Qt User Interface Compiler version 6.4.2
##
## WARNING! All changes made in this file will be lost
   when recompiling UI file!


from PySide6.QtCore import (QCoreApplication, QDate,
   QDateTime, QLocale,
    QMetaObject, QObject, QPoint, QRect,
    QSize, QTime, QUrl, Qt)
from PySide6.QtGui import (QBrush, QColor,
   QConicalGradient, QCursor,
    QFont, QFontDatabase, QGradient, QIcon,
    QImage, QKeySequence, QLinearGradient, QPainter,
    QPalette, QPixmap, QRadialGradient, QTransform)
from PySide6.QtWidgets import (QApplication, QGridLayout,
    QHBoxLayout, QLabel,
    QLineEdit, QMainWindow, QMenuBar, QPushButton,
    QSizePolicy, QStatusBar, QWidget)

class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        if not MainWindow.objectName():
            MainWindow.setObjectName(u"MainWindow")
        MainWindow.resize(800, 600)
        self.centralwidget = QWidget(MainWindow)
        self.centralwidget.setObjectName(u"centralwidget"
           )
        self.horizontalLayout = QHBoxLayout(self.
           centralwidget)
        self.horizontalLayout.setObjectName(u"
           horizontalLayout")
        self.gridLayout = QGridLayout()
        self.gridLayout.setObjectName(u"gridLayout")
        self.labelResult = QLabel(self.centralwidget)
        self.labelResult.setObjectName(u"labelResult")
        font = QFont()
```

```
font.setPointSize(40)
self.labelResult.setFont(font)
self.labelResult.setAlignment(Qt.AlignCenter)

self.gridLayout.addWidget(self.labelResult, 0, 0,
    1, 1)

self.gridLayout_2 = QGridLayout()
self.gridLayout_2.setObjectName(u"gridLayout_2")
self.labelName = QLabel(self.centralwidget)
self.labelName.setObjectName(u"labelName")
font1 = QFont()
font1.setPointSize(30)
self.labelName.setFont(font1)

self.gridLayout_2.addWidget(self.labelName, 0, 0,
    1, 1)

self.lineName = QLineEdit(self.centralwidget)
self.lineName.setObjectName(u"lineName")
self.lineName.setFont(font1)

self.gridLayout_2.addWidget(self.lineName, 0, 1,
   1, 1)

self.buttonSend = QPushButton(self.centralwidget)
self.buttonSend.setObjectName(u"buttonSend")
self.buttonSend.setFont(font1)

self.gridLayout_2.addWidget(self.buttonSend, 0,
   2, 1, 1)


self.gridLayout.addLayout(self.gridLayout_2, 1,
   0, 1, 1)


self.horizontalLayout.addLayout(self.gridLayout)

MainWindow.setCentralWidget(self.centralwidget)
self.menubar = QMenuBar(MainWindow)
self.menubar.setObjectName(u"menubar")
self.menubar.setGeometry(QRect(0, 0, 800, 22))
MainWindow.setMenuBar(self.menubar)
self.statusbar = QStatusBar(MainWindow)
self.statusbar.setObjectName(u"statusbar")
```

```
        MainWindow.setStatusBar(self.statusbar)

        self.retranslateUi(MainWindow)

        QMetaObject.connectSlotsByName(MainWindow)
    # setupUi

    def retranslateUi(self, MainWindow):
        MainWindow.setWindowTitle(QCoreApplication.
           translate("MainWindow", u"MainWindow", None))
        self.labelResult.setText(QCoreApplication.
           translate("MainWindow", u"Voltei!", None))
        self.labelName.setText(QCoreApplication.translate
           ("MainWindow", u"Seu nome:", None))
        self.lineName.setPlaceholderText(QCoreApplication
           .translate("MainWindow", u"Digite seu nome",
           None))
        self.buttonSend.setText(QCoreApplication.
           translate("MainWindow", u"Enviar", None))
    # retranslateUi
```

aula203qtdesigner/ui/uiwindow.py

```
            # -*- coding: utf-8 -*-


from PySide6.QtCore import (QCoreApplication, QDate,
   QDateTime, QLocale,
    QMetaObject, QObject, QPoint, QRect,
    QSize, QTime, QUrl, Qt)
from PySide6.QtGui import (QBrush, QColor,
   QConicalGradient, QCursor,
    QFont, QFontDatabase, QGradient, QIcon,
    QImage, QKeySequence, QLinearGradient, QPainter,
    QPalette, QPixmap, QRadialGradient, QTransform)
from PySide6.QtWidgets import (QApplication, QGridLayout,
    QHBoxLayout, QLabel,
    QLineEdit, QMainWindow, QMenuBar, QPushButton,
    QSizePolicy, QStatusBar, QWidget)

class UiMainWindow(object):
    def setupUi(self, MainWindow):
        if not MainWindow.objectName():
            MainWindow.setObjectName(u"MainWindow")
        MainWindow.resize(800, 600)
        self.centralwidget = QWidget(MainWindow)
        self.centralwidget.setObjectName(u"centralwidget"
           )
```

```
self.horizontalLayout = QHBoxLayout(self.
   centralwidget)
self.horizontalLayout.setObjectName(u"
   horizontalLayout")
self.gridLayout = QGridLayout()
self.gridLayout.setObjectName(u"gridLayout")
self.labelResult = QLabel(self.centralwidget)
self.labelResult.setObjectName(u"labelResult")
font = QFont()
font.setPointSize(40)
self.labelResult.setFont(font)
self.labelResult.setAlignment(Qt.AlignCenter)

self.gridLayout.addWidget(self.labelResult, 0, 0,
    1, 1)

self.gridLayout2 = QGridLayout()
self.gridLayout2.setObjectName(u"gridLayout2")
self.labelName = QLabel(self.centralwidget)
self.labelName.setObjectName(u"labelName")
font1 = QFont()
font1.setPointSize(30)
self.labelName.setFont(font1)

self.gridLayout2.addWidget(self.labelName, 0, 0,
   1, 1)

self.lineName = QLineEdit(self.centralwidget)
self.lineName.setObjectName(u"lineName")
self.lineName.setFont(font1)

self.gridLayout2.addWidget(self.lineName, 0, 1,
   1, 1)

self.buttonSend = QPushButton(self.centralwidget)
self.buttonSend.setObjectName(u"buttonSend")
self.buttonSend.setFont(font1)

self.gridLayout2.addWidget(self.buttonSend, 0, 2,
    1, 1)


self.gridLayout.addLayout(self.gridLayout2, 1, 0,
    1, 1)
```

```
        self.horizontalLayout.addLayout(self.gridLayout)

        MainWindow.setCentralWidget(self.centralwidget)
        self.menubar = QMenuBar(MainWindow)
        self.menubar.setObjectName(u"menubar")
        self.menubar.setGeometry(QRect(0, 0, 800, 22))
        MainWindow.setMenuBar(self.menubar)
        self.statusbar = QStatusBar(MainWindow)
        self.statusbar.setObjectName(u"statusbar")
        MainWindow.setStatusBar(self.statusbar)

        self.retranslateUi(MainWindow)

        QMetaObject.connectSlotsByName(MainWindow)
     setupUi

    def retranslateUi(self, MainWindow):
        MainWindow.setWindowTitle(QCoreApplication.
           translate("MainWindow", u"MainWindow", None))
        self.labelResult.setText(QCoreApplication.
           translate("MainWindow", u"Voltei!", None))
        self.labelName.setText(QCoreApplication.translate
           ("MainWindow", u"Seu nome:", None))
        self.lineName.setPlaceholderText(QCoreApplication
           .translate("MainWindow", u"Digite seu nome",
           None))
        self.buttonSend.setText(QCoreApplication.
           translate("MainWindow", u"Enviar", None))
     retranslateUi
```

aula203-qtdesigner/ui/window.ui

```
            <?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>MainWindow</class>
 <widget class="QMainWindow" name="MainWindow">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>800</width>
    <height>600</height>
   </rect>
  </property>
  <property name="windowTitle">
   <string>MainWindow</string>
  </property>
  <widget class="QWidget" name="centralwidget">
```

```
<layout class="QHBoxLayout" name="horizontalLayout">
 <item>
  <layout class="QGridLayout" name="gridLayout">
   <item row="0" column="0">
    <widget class="QLabel" name="labelResult">
     <property name="font">
      <font>
       <pointsize>40</pointsize>
      </font>
     </property>
     <property name="text">
      <string>Voltei!</string>
     </property>
     <property name="alignment">
      <set>Qt::AlignCenter</set>
     </property>
    </widget>
   </item>
   <item row="1" column="0">
    <layout class="QGridLayout" name="gridLayout_2">
     <item row="0" column="0">
      <widget class="QLabel" name="labelName">
       <property name="font">
        <font>
         <pointsize>30</pointsize>
        </font>
       </property>
       <property name="text">
        <string>Seu nome:</string>
       </property>
      </widget>
     </item>
     <item row="0" column="1">
      <widget class="QLineEdit" name="lineName">
       <property name="font">
        <font>
         <pointsize>30</pointsize>
        </font>
       </property>
       <property name="placeholderText">
        <string>Digite seu nome</string>
       </property>
      </widget>
     </item>
     <item row="0" column="2">
      <widget class="QPushButton" name="buttonSend">
```

```
          <property name="font">
           <font>
            <pointsize>30</pointsize>
           </font>
          </property>
          <property name="text">
           <string>Enviar</string>
          </property>
         </widget>
        </item>
       </layout>
      </item>
     </layout>
    </item>
   </layout>
  </widget>
  <widget class="QMenuBar" name="menubar">
   <property name="geometry">
    <rect>
     <x>0</x>
     <y>0</y>
     <width>800</width>
     <height>22</height>
    </rect>
   </property>
  </widget>
  <widget class="QStatusBar" name="statusbar"/>
 </widget>
 <resources/>
 <connections/>
</ui>
```

10 Usando um arquivo UI do Qt Designer com seu código Python

aula203qtdesigner/src/mainwindow.py

```
        import sys

from PySide6.QtWidgets import QApplication, QMainWindow
from window import Ui_MainWindow


class MainWindow(QMainWindow, Ui_MainWindow):
    def __init__(self, parent=None):
        super().__init__(parent)
        self.setupUi(self)

        self.buttonSend.clicked.connect(self.
           changeLabelResult)  # type: ignore

    def changeLabelResult(self):
        text = self.lineName.text()
        self.labelResult.setText(text)


if __name__ == '__main__':
    app = QApplication(sys.argv)
    mainWindow = MainWindow()
    mainWindow.show()
    app.exec()
```

11 QObject e QThread

11.1 criando a janela inicial com Qt Designer

main.py

```
        import sys

from PySide6.QtWidgets import QApplication, QWidget
from ui_workerui import Ui_myWidget


class MyWidget(QWidget, Ui_myWidget):
    def __init__(self, parent=None):
        super().__init__(parent)
        self.setupUi(self)


if __name__ == '__main__':
    app = QApplication(sys.argv)
    myWidget = MyWidget()
    myWidget.show()
    app.exec()
```

uiworkerui.py

```
        from PySide6.QtCore import (QCoreApplication,
           QDate, QDateTime, QLocale,
    QMetaObject, QObject, QPoint, QRect,
    QSize, QTime, QUrl, Qt)
from PySide6.QtGui import (QBrush, QColor,
   QConicalGradient, QCursor,
    QFont, QFontDatabase, QGradient, QIcon,
    QImage, QKeySequence, QLinearGradient, QPainter,
    QPalette, QPixmap, QRadialGradient, QTransform)
from PySide6.QtWidgets import (QApplication, QGridLayout,
    QHBoxLayout, QLabel,
    QPushButton, QSizePolicy, QWidget)

class Ui_myWidget(object):
    def setupUi(self, myWidget):
        if not myWidget.objectName():
            myWidget.setObjectName(u"myWidget")
        myWidget.resize(400, 300)
        font = QFont()
        font.setPointSize(40)
        myWidget.setFont(font)
        self.horizontalLayout = QHBoxLayout(myWidget)
        self.horizontalLayout.setObjectName(u"
           horizontalLayout")
        self.gridLayout = QGridLayout()
        self.gridLayout.setObjectName(u"gridLayout")
        self.label2 = QLabel(myWidget)
        self.label2.setObjectName(u"label2")

        self.gridLayout.addWidget(self.label2, 0, 1, 1,
           1)

        self.label1 = QLabel(myWidget)
        self.label1.setObjectName(u"label1")

        self.gridLayout.addWidget(self.label1, 0, 0, 1,
           1)

        self.button1 = QPushButton(myWidget)
        self.button1.setObjectName(u"button1")

        self.gridLayout.addWidget(self.button1, 1, 0, 1,
           1)

        self.button2 = QPushButton(myWidget)
        self.button2.setObjectName(u"button2")

        self.gridLayout.addWidget(self.button2, 1, 1, 1,
           1)


        self.horizontalLayout.addLayout(self.gridLayout)
        self.retranslateUi(myWidget)

        QMetaObject.connectSlotsByName(myWidget)
    # setupUi

    def retranslateUi(self, myWidget):
        myWidget.setWindowTitle(QCoreApplication.
           translate("myWidget", u"Form", None))
        self.label2.setText(QCoreApplication.translate("
           myWidget", u"L2", None))
        self.label1.setText(QCoreApplication.translate("
           myWidget", u"L1", None))
        self.button1.setText(QCoreApplication.translate("
           myWidget", u"B1", None))
        self.button2.setText(QCoreApplication.translate("
           myWidget", u"B2", None))
    # retranslateUi
```

workerui.ui

```
        <?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>myWidget</class>
 <widget class="QWidget" name="myWidget">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>400</width>
    <height>300</height>
   </rect>
  </property>
  <property name="font">
   <font>
    <pointsize>40</pointsize>
   </font>
  </property>
  <property name="windowTitle">
   <string>Form</string>
  </property>
  <layout class="QHBoxLayout" name="horizontalLayout">
   <item>
    <layout class="QGridLayout" name="gridLayout">
     <item row="0" column="1">
      <widget class="QLabel" name="label2">
       <property name="text">
```

```
        <string>L2</string>
       </property>
      </widget>
     </item>
     <item row="0" column="0">
      <widget class="QLabel" name="label1">
       <property name="text">
        <string>L1</string>
       </property>
      </widget>
     </item>
     <item row="1" column="0">
      <widget class="QPushButton" name="button1">
       <property name="text">
        <string>B1</string>
       </property>
      </widget>
     </item>
     <item row="1" column="1">
      <widget class="QPushButton" name="button2">
       <property name="text">
        <string>B2</string>
       </property>
      </widget>
     </item>
    </layout>
   </item>
  </layout>
 </widget>
 <resources/>
 <connections/>
</ui>
```

11.2 criando o Worker

main.py

```
        import sys
import time

from PySide6.QtCore import QObject, Signal, Slot
from PySide6.QtWidgets import QApplication, QWidget
from uiworkerui import UimyWidget


class Worker1(QObject):
    started = Signal(str)
    progressed = Signal(str)
    finished = Signal(str)

    def run(self):
        value = '0'
        self.started.emit(value)
        for i in range(5):
            value = str(i)
            self.progressed.emit(value)
            time.sleep(1)
        self.finished.emit(value)


class MyWidget(QWidget, Ui_myWidget):
    def __init__(self, parent=None):
        super().__init__(parent)
        self.setupUi(self)

        self.button1.clicked.connect(self.hardWork1)
        self.button2.clicked.connect(self.hardWork2)

    def hardWork1(self):
        self.label1.setText('1 terminado')

    def hardWork2(self):
        for i in range(5):
            print(i)
            time.sleep(1)
        self.label2.setText('2 terminado')


if __name__ == '__main__':
    app = QApplication(sys.argv)
```

11.3 movendo workers para threads separadas

main.py

```
        import sys
import time

from PySide6.QtCore import QObject, Signal, Slot
from PySide6.QtCore import QObject, QThread, Signal
from PySide6.QtWidgets import QApplication, QWidget
from ui_workerui import Ui_myWidget

@@ -11,7 +11,7 @@ class Worker1(QObject):
    progressed = Signal(str)
    finished = Signal(str)

    def run(self):
    def doWork(self):
        value = '0'
        self.started.emit(value)
        for i in range(5):
@@ -30,13 +30,78 @@ def __init__(self, parent=None):
        self.button2.clicked.connect(self.hardWork2)

    def hardWork1(self):
        self.label1.setText('1 terminado')
        self._worker = Worker1()
        self._thread = QThread()

        worker = self._worker
        thread = self._thread

        # Mover o worker para a thread
        worker.moveToThread(thread)

        # Run
        thread.started.connect(worker.doWork)
        worker.finished.connect(thread.quit)

        thread.finished.connect(thread.deleteLater)
        worker.finished.connect(worker.deleteLater)

        worker.started.connect(self.worker1Started)
        worker.progressed.connect(self.worker1Progressed)
        worker.finished.connect(self.worker1Finished)

        thread.start()

    def worker1Started(self, value):
        self.button1.setDisabled(True)
        self.label1.setText(value)
        print('worker iniciado')

    def worker1Progressed(self, value):
        self.label1.setText(value)
        print('em progresso')

    def worker1Finished(self, value):
        self.label1.setText(value)
        self.button1.setDisabled(False)
        print('worker finalizado')

    def hardWork2(self):
        for i in range(5):
            print(i)
            time.sleep(1)
        self.label2.setText('2 terminado')
        self._worker2 = Worker1()
        self._thread2 = QThread()

        worker = self._worker2
        thread = self._thread2

        # Mover o worker para a thread
        worker.moveToThread(thread)

        # Run
        thread.started.connect(worker.doWork)
        worker.finished.connect(thread.quit)

        thread.finished.connect(thread.deleteLater)
        worker.finished.connect(worker.deleteLater)

        worker.started.connect(self.worker2Started)
        worker.progressed.connect(self.worker2Progressed)
        worker.finished.connect(self.worker2Finished)

        thread.start()

    def worker2Started(self, value):
        self.button2.setDisabled(True)
        self.label2.setText(value)
        print('worker 2 iniciado')

    def worker2Progressed(self, value):
        self.label2.setText(value)
        print('2 em progresso')

    def worker2Finished(self, value):
        self.label2.setText(value)
        self.button2.setDisabled(False)
        print('2 worker finalizado')


if __name__ == '__main__':
```

11.4 código comentado

main.py

```
        self.finished.emit(value)


class Worker2(QObject):
    started = Signal(str)
    progressed = Signal(str)
    finished = Signal(str)

    def executeMe(self):
        value = '0'
        self.started.emit(value)
        for i in range(50, 100, 5):
            value = str(i)
            self.progressed.emit(value)
            time.sleep(0.3)
        self.finished.emit(value)


class MyWidget(QWidget, Ui_myWidget):
    def __init__(self, parent=None):
        super().__init__(parent)
        self.setupUi(self)

        self.button1.clicked.connect(self.
           hardWork1)
        self.button2.clicked.connect(self.
           hardWork2)
        self.button1.clicked.connect(self.
           hardWork1)  # type: ignore
        self.button2.clicked.connect(self.
           hardWork2)  # type: ignore

    def hardWork1(self):
        self._worker = Worker1()
        self._thread = QThread()
        self._worker1 = Worker1()
        self._thread1 = QThread()

        worker = self._worker
        thread = self._thread
         Isso garante que o widget vai ter uma
            referencia para worker e thread
        worker = self._worker1
```

```
thread = self._thread1

# Mover o worker para a thread
# Worker e movido para a thread. Todas as
    funçoes e metodos do
# objeto de worker serão executados na
   thread criado pela QThread.
worker.moveToThread(thread)

# Run
thread.started.connect(worker.doWork)
# Quando uma QThread e iniciada, emite o
   sinal started automaticamente.
thread.started.connect(worker.doWork)  #
   type: ignore

# O sinal finished e emitido pelo objeto
   worker quando o trabalho que
# ele esta executando e concluido. Isso
   aciona o metodo quit da qthread
# que interrompe o loop de eventos dela.
worker.finished.connect(thread.quit)

thread.finished.connect(thread.
   deleteLater)
# deleteLater solicita a exclusão do
   objeto worker do sistema de
# gerenciamento de memoria do Python.
   Quando o worker finaliza, ele
# emite um sinal finished que vai
   executar o metodo deleteLater.
# Isso garante que objetos sejam
   removidos da memoria corretamente.
thread.finished.connect(thread.
   deleteLater)  # type: ignore
worker.finished.connect(worker.
   deleteLater)

# Aqui estao seus metodos e inicio, meio
   e fim
# execute o que quiser
worker.started.connect(self.
   worker1Started)
worker.progressed.connect(self.
   worker1Progressed)
```

```
        worker.finished.connect(self.
           worker1Finished)

        # Inicie a thread
        thread.start()

    def worker1Started(self, value):
        self.button1.setDisabled(True)
        self.label1.setText(value)
        print('worker iniciado')
        print('worker 1 iniciado', value)

    def worker1Progressed(self, value):
        self.label1.setText(value)
        print('em progresso')
        print('1 em progresso', value)

    def worker1Finished(self, value):
        self.label1.setText(value)
        self.button1.setDisabled(False)
        print('worker finalizado')
        print('worker 1 finalizado', value)

    def hardWork2(self):
        self._worker2 = Worker1()
        self._worker2 = Worker2()
        self._thread2 = QThread()

        # Isso garante que o widget vai ter uma
           referencia para worker e thread
        worker = self._worker2
        thread = self._thread2

        # Mover o worker para a thread
        # Worker e movido para a thread. Todas as
            funçoes e metodos do
        # objeto de worker serão executados na
           thread criado pela QThread.
        worker.moveToThread(thread)

        # Run
        thread.started.connect(worker.doWork)
        # Quando uma QThread e iniciada, emite o
           sinal started automaticamente.
        # Nome do metodo "doWork" modificado para
            "executeMe" (p/ exemplo)
        thread.started.connect(worker.executeMe)
             # type: ignore

        # O sinal finished e emitido pelo objeto
           worker quando o trabalho que
        # ele esta executando e concluido. Isso
           aciona o metodo quit da qthread
        # que interrompe o loop de eventos dela.
        worker.finished.connect(thread.quit)

        thread.finished.connect(thread.
           deleteLater)
        # deleteLater solicita a exclusão do
           objeto worker do sistema de
        # gerenciamento de memoria do Python.
           Quando o worker finaliza, ele
        # emite um sinal finished que vai
           executar o metodo deleteLater.
        # Isso garante que objetos sejam
           removidos da memoria corretamente.
        thread.finished.connect(thread.
           deleteLater)  # type: ignore
        worker.finished.connect(worker.
           deleteLater)

        # Aqui estão seus metodos e inicio, meio
           e fim
        # execute o que quiser
        worker.started.connect(self.
           worker2Started)
        worker.progressed.connect(self.
           worker2Progressed)
        worker.finished.connect(self.
           worker2Finished)

        # Inicie a thread
        thread.start()

    def worker2Started(self, value):
        self.button2.setDisabled(True)
        self.label2.setText(value)
        print('worker 2 iniciado')
        print('worker 2 iniciado', value)

    def worker2Progressed(self, value):
        self.label2.setText(value)
        print('2 em progresso')
        print('2 em progresso', value)

    def worker2Finished(self, value):
        self.label2.setText(value)
        self.button2.setDisabled(False)
        print('2 worker finalizado')
        print('worker 2 finalizado', value)


if __name__ == '__main__':
```