Lenguaje de Programación Visual

Ingeniería Mecatrónica

Semana 2: Diseño UI y Layouts con PyQt6

PyQt es una libreria de Python para crear aplicaciones GUI con Qt toolkit, fue creada por **Riverbank Computing Limited** desarrollada desde el 1999. La ultima versión basada en Qt6 es **PyQt6** publicada en 2021 con actualizaciones continuas.

PyQt6 puedes crear ventanas, botones, etiquetas de texto, campos de entrada, layouts (diseños), menús, eventos interactivos y aplicaciones completas con interfaz gráfica.

Funciona mediante una arquitectura basada en:

- **App** → controla la aplicación
- **Siganls, Slots and Events** → comunicación entre eventos y funciones
- **Widgets** → elementos visuales
- **Layouts** → organización de widgets
- **Menus** → menús
- **Dialogs** → ventanas emergentes

Instalación de dependencias

Agrega PyQt6 al proyecto usando poetry:

```
poetry add pyqt6
```

Verifica la instalación:

```
poetry run python -c "import PyQt6"
```

QtApplication

La clase `QApplication` es la encargada de gestionar todos los recursos de la aplicación como el loop permanente de escucha de eventos (pulsar una tecla, mover el mouse, etc.).

Solo debe haber una y solo una instancia de `QApplication` por aplicación.

Ejemplo basico de `app.py`:

```
from PyQt6.QtWidgets import QApplication, QWidget, QPushButton

app = QApplication([])
```

```
window = QWidget()
window.show()

app.exec()
```

> Nota: Todos los Qt Wigets pueden ser "ventas", por ejemplo cambiando QWidget por QPushButton("Click") crea un botón, esto es porque todos las clases Qt Widgets son hijas de QWidget.

Widgets

Un Widgets son elementos visuales de la aplicación, por ejemplo un botón, un campo de texto, una etiqueta, etc.

1. Widgets básicos

Todos los widgets de Qt son clases hijas de QWidget, de entre los cuales tenemos los widgets elementales o basicos:

- QLabel: texto

```
window = QWidget()
label = QLabel("Hola Mecatrónica", parent=window)
```

- QPushButton: botón

```
def on_click():
    print("Botón presionado")
window = QWidget()
button = QPushButton("Presionar", parent=window)
button.clicked.connect(on_click)
```

- QLineEdit: campo de texto

```
window = QWidget()
input = QLineEdit("Hola Mecatrónica", parent=window)
```

- QCheckBox: checkbox

```
window = QWidget()
check = QCheckBox("Checkbox", parent=window)
```

- QComboBox: combobox

```
window = QWidget()
combo = QComboBox(parent=window)
combo.addItems(["Opción 1", "Opción 2", "Opción 3"])
```

- QSlider: slider

```
window = QWidget()
slider = QSlider(parent=window)
slider.setRange(0, 100)
slider.setValue(50)
```

Ejemplo unificado de widgets basicos:

```
import sys
from PyQt6.QtWidgets import QApplication, QWidget, QLabel, QPushButton, QLineEdit,
QCheckBox, QComboBox, QSlider, QMessageBox

def mostrar_mensaje():
    msg = QMessageBox(window)
    msg.setText("Interacción detectada")
    msg.exec()

app = QApplication(sys.argv)

window = QWidget()
window.setWindowTitle("Ejemplo Widgets")
window.resize(400, 450)

# QLabel
label = QLabel("Hola Mecatrónica", parent=window)
label.move(30, 20)

# QPushButton
button = QPushButton("Presionar", parent=window)
button.move(30, 60)
button.clicked.connect(mostrar_mensaje)

# QLineEdit
input_field = QLineEdit("Hola Mecatrónica", parent=window)
input_field.move(30, 110)
input_field.returnPressed.connect(mostrar_mensaje)

# QCheckBox
check = QCheckBox("Checkbox", parent=window)
check.move(30, 160)
check.clicked.connect(mostrar_mensaje)

# QComboBox
combo = QComboBox(parent=window)
```

```
combo.addItems(["Opción 1", "Opción 2", "Opción 3"])
combo.move(30, 210)
combo.activated.connect(mostrar_mensaje)

# QSlider
slider = QSlider(parent=window)
slider.setRange(0, 100)
slider.move(30, 260)
slider.sliderReleased.connect(mostrar_mensaje)

window.show()

sys.exit(app.exec())
```

2. QMainWindow

QMainWindow es una clase que contiene un set de widgets modificables por herencia como la barra de menú, la barra de herramientas, la barra de estado, entre otros. Además, pueden incorporarse widgets básicos que interactúan dentro de la clase preprogramada modulando asi el codigo entre app.py y demas modulos.

Ejemplo básico

En `myWindow.py`:

```
from PyQt6.QtWidgets import QMainWindow, QPushButton

class MyWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Mi primera ventana PyQt6")

        button = QPushButton("Click")
        self.setCentralWidget(button)
        button.clicked.connect(self.on_click)

    def on_click(self):
        print("Evento de boton presionado")
```

En `app.py`:

```
from PyQt6.QtWidgets import QApplication
from myWindow import MyWindow
import sys

app = QApplication(sys.argv)
window = MyWindow()
window.show()
app.exec()
```

Events, Signals and Slots

Los `Signals` son notificaciones emitidas por un widget como respuesta a un `Event`, es decir, cuando algo ocurre como: como presionar un boton o cuando el texto de un inputbox cambia

Los `Slots` son funciones que se ejecutan cuando se emite un `Signal`, son equivalente de los `EventListeners` o `handler` de otros lenguajes.

```
import sys
from PyQt6.QtWidgets import QApplication, QMainWindow, QPushButton


class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()

        self.setWindowTitle("My App")

        button = QPushButton("Press Me!")
        button.setCheckable(True)

        # Conectamos el slot al signal
        button.clicked.connect(self.the_button_was_clicked)
        self.setCentralWidget(button)

    # Este es el slot
    def the_button_was_clicked(self, checked):
        print(f"The button was clicked! Checked: {checked}")


app = QApplication(sys.argv)

window = MainWindow()
window.show()

app.exec()
```

Layouts

...

Toolbars and Menu

...

Qt Designer

...

Ejercicio

...