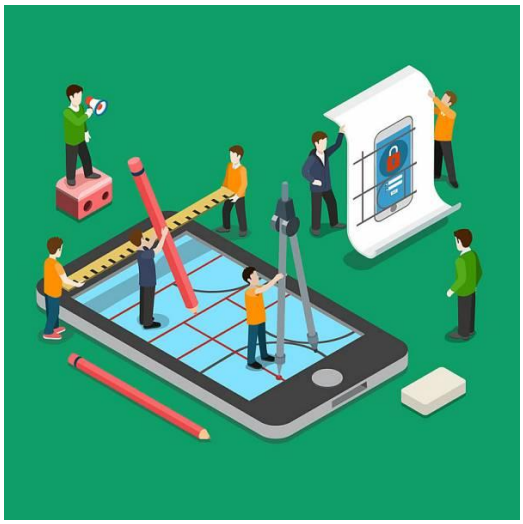
CIP de FP BATOI

DESENROTLLAMENT D'INTERFÍCIES

2N DAM

SA2: Creació i us de components.



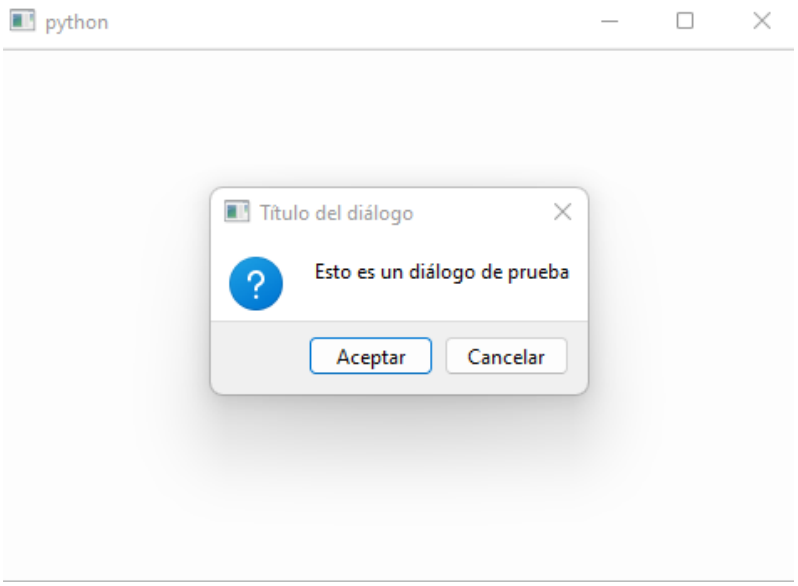
QUADRES DE DIÀLEG EN PYSIDE 6

Contenido

Els diàlegs són finestres emergents temporals que ens permeten comunicar-nos amb l'usuari de l'aplicació i apareixen a causa de la producció d'un determinat esdeveniment.

Son finestres modals, és a dir, bloquegen la interacció amb la resta de l'aplicació fins que finalitza la seua execució, ja siga tancant-les o introduint la informació que se li demane.

1. Diàlegs de misatge QMessageBox



Si el que volem enviar un missatge a l'usuari tenim a la nostra disposició una classe anomenada QMessageBox, que permet traduir els botons i personalitzar els icones.

```
from PySide6.QtWidgets import (
    QApplication, QMainWindow, QPushButton, QMessageBox)
import sys


class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.resize(480, 320)

        boton = QPushButton("Mostrar diálogo")
        boton.clicked.connect(self.boton_clicado)
        self.setCentralWidget(boton)

    def boton_clicado(self):
        # creamos un diálogo de mensaje con un título y un texto
        dialogo = QMessageBox(self)
        dialogo.setWindowTitle("Título del diálogo")
        dialogo.setText("Esto es un diálogo de prueba")
        # añadimos unos botones y los traducimos
        dialogo.setStandardButtons(QMessageBox.Ok |
QMessageBox.Cancel)
        dialogo.button(QMessageBox.Ok).setText("Aceptar")
        dialogo.button(QMessageBox.Cancel).setText("Cancelar")
        # configuramos un icono
        dialogo.setIcon(QMessageBox.Question)

        # ejecutamos el diálogo y capturamos la respuesta
        respuesta = dialogo.exec_()
        # ahora debemos comprobar qué tipo de botón se ha
clicado
        if respuesta == QMessageBox.Ok:
            print("Diálogo aceptado")
        else:
            print("Diálogo denegado")


if __name__ == "__main__":
    app = QApplication(sys.argv)
    window = MainWindow()
    window.show()
    sys.exit(app.exec_())
```

2. Diàlegs predeterminats QMessageBox

Per sort no necessitem crear diàlegs tot el temps, Qt inclou diàlegs predeterminats per realitzar diferents tasques:

a. Missatge de qüestió QMessageBox.question

```
from PySide6.QtWidgets import (
    QApplication, QMainWindow, QPushButton, QMessageBox)
import sys

class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.resize(480, 320)

        boton = QPushButton("Mostrar diálogo")
        boton.clicked.connect(self.boton_clicado)
        self.setCentralWidget(boton)

    def boton_clicado(self):
        # creamos un diálogo de tipo cuestión
        dialogo = QMessageBox.question(
            self, "Diálogo de cuestión", "Esta es una pregunta
de prueba")

        # ahora podemos comprobar qué tipo de botón se devuelve
        print(dialogo)
        if dialogo == QMessageBox.Yes:
            print("Ha respondido sí")

if __name__ == "__main__":
    app = QApplication(sys.argv)
    window = MainWindow()
    window.show()
    sys.exit(app.exec_())
```

Traduir els diàlegs predeterminats no és una cosa trivial, a la següent lliçó us ensenyaré com activar el traductor, per ara vegem altres exemples.

b. Missatge crític QMessageBox.critical

Aquest diàleg mostra un error mentre mostra la finestra:

```
def boton_clicado(self):
    dialogo = QMessageBox.critical(
        self, "Diálogo de error", "Ha ocurrido algo malo")
    print(dialogo)
```

c. Mensaje informativo QMessageBox.information

```
def boton_clicado(self):
     dialogo = QMessageBox.information(
          self, "Diálogo informativo", "Esto es un texto
informativo")
```

d. Mensaje de aviso QMessageBox.warning

```
def boton_clicado(self):
    dialogo = QMessageBox.warning(
        self, "Diálogo de aviso", "Cuidado con este diálogo")
```

Si volem modificar els botons d'un missatge predeterminat, podem fer-ho:

```
def boton_clicado(self):
    dialogo = QMessageBox.warning(
        self, "Diálogo de aviso", "¿Estás seguro de aplicar los
cambios?",
        buttons=QMessageBox.Apply | QMessageBox.Cancel,
        defaultButton=QMessageBox.Cancel)

    if dialogo == QMessageBox.Apply:
        print("Aplicamos los cambios")
```

3. Diàlegs modals i no modals

Els diàlegs (QDialog) poden funcionar de dues maneres segons si bloquegen o no la interacció amb la finestra principal:

Tipus	Descripció	Mètode
Modal	Bloqueja la resta de la interfície fins que es tanque.	exec()
No modal	Permet seguir treballant amb la finestra principal mentre el diàleg està obert.	show()

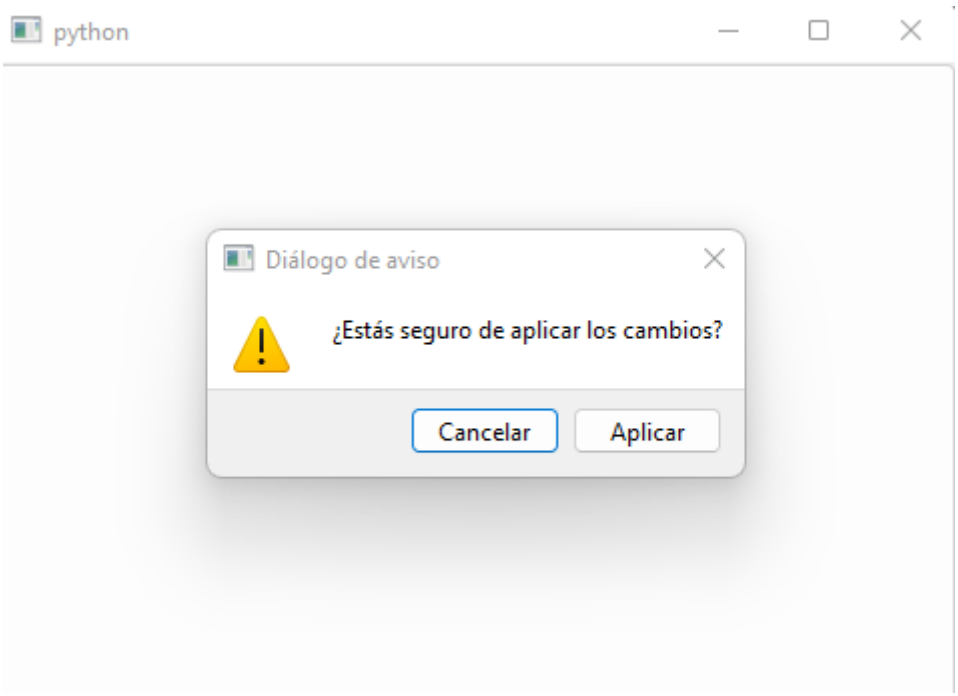
El mode **modal** és útil quan cal una resposta immediata de l'usuari (confirmar, acceptar, etc.). El mode **no modal** és adequat per a mostrar informació o panells auxiliars que no interrompen el flux de treball.

Exemple pràctic:

```
from PySide6.QtWidgets import QApplication, QMainWindow,
QPushButton, QDialog, QLabel, QVBoxLayout

class InfoDialog(QDialog):
    def __init__(self, titol, text, parent=None):
        super().__init__(parent)
        self.setWindowTitle(titol)
        layout = QVBoxLayout(self)
        layout.addWidget(QLabel(text))


class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Modal vs No modal")

        boto_modal = QPushButton("Obrir diàleg MODAL")
        boto_no_modal = QPushButton("Obrir diàleg NO MODAL")

        boto_modal.clicked.connect(self.obre_modal)
        boto_no_modal.clicked.connect(self.obre_no_modal)

        self.addToolBar("Diàlegs").addWidget(boto_modal)
        self.addToolBar("Diàlegs").addWidget(boto_no_modal)

    def obre_modal(self):
        dlg = InfoDialog("Diàleg modal", "Bloqueja la finestra
principal", self)
        dlg.exec()  # Modal → bloqueja fins que es tanque

    def obre_no_modal(self):
```

```
        self.dlg = InfoDialog("Diàleg no modal", "Pots seguir
usant la finestra principal", self)
        self.dlg.show()  # No modal → no bloqueja

app = QApplication()
finestra = MainWindow()
finestra.resize(400, 200)
finestra.show()
app.exec()
```

4. Activant les traduccions



Traduir els diàlegs predeterminats a mà fa que se'n perda la simplicitat, per això és recomanable activar les traduccions. Això implica que cal distribuir els fitxers de traducció junt als executables, però això és una cosa que veurem més endavant.

Per activar la localització de l'idioma del sistema operatiu hem de traduir l'aplicació de la manera següent:

```
from PySide6.QtWidgets import (
    QApplication, QMainWindow, QPushButton, QMessageBox)
from PySide6.QtCore import QTranslator, QLibraryInfo  # nuevo
import sys


class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.resize(480, 320)
```

```
        boton = QPushButton("Mostrar diálogo")
        boton.clicked.connect(self.boton_clicado)
        self.setCentralWidget(boton)

    def boton_clicado(self):
        dialogo = QMessageBox.warning(
            self, "Diálogo de aviso", "¿Estás seguro de aplicar
los cambios?",
            buttons=QMessageBox.Apply | QMessageBox.Cancel,
            defaultButton=QMessageBox.Cancel)

        if dialogo == QMessageBox.Apply:
            print("Aplicamos los cambios")


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

    # envolvemos la aplicación con el traductor
    translator = QTranslator(app)
    # recuperamos el directorio de traducciones
    translations =
QLibraryInfo.location(QLibraryInfo.TranslationsPath)
    # cargamos la traducción en el traductor
    translator.load("qt_es", translations)
    # la aplicamos
    app.installTranslator(translator)

    window = MainWindow()
    window.show()
    sys.exit(app.exec_())
```

5. Creació de diàlegs personalitzats amb QdialogButtonBox

Quan necessitem un diàleg amb camps de dades (formularis), podem **heretar de QDialog** i construir-lo a mida.
Per a gestionar els botons d'acceptar/cancel·lar s'utilitza la classe QDialogButtonBox, que proporciona una interfície estandarditzada.

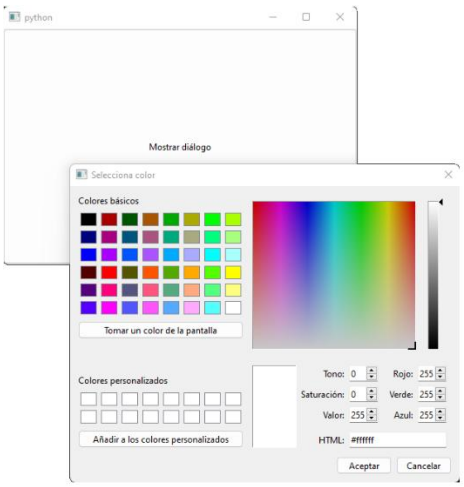
Exemple pràctic:

```
from PySide6.QtWidgets import (QApplication, QMainWindow,
QDialog,
                               QLineEdit, QSpinBox, QFormLayout,
                               QDialogButtonBox, QMessageBox,
QPushButton)
import sys
```

```
class UsuariDialog(QDialog):
    def __init__(self, parent=None):
        super().__init__(parent)
        self.setWindowTitle("Nou usuari")

        # Camps del formulari
        self.nom = QLineEdit()
        self.edat = QSpinBox()
        self.edat.setRange(0, 120)

        # Botons d'acceptar/cancel·lar
        botons = QDialogButtonBox(QDialogButtonBox.Ok |
QDialogButtonBox.Cancel)
        botons.accepted.connect(self.valida_i_accepta)
        botons.rejected.connect(self.reject)

        # Disseny del formulari
        layout = QFormLayout(self)
        layout.addRow("Nom:", self.nom)
        layout.addRow("Edat:", self.edat)
        layout.addRow(botons)

    def valida_i_accepta(self):
        if not self.nom.text().strip():
            QMessageBox.warning(self, "Validació", "El nom no
pot estar buit.")
            return
        if self.edat.value() < 18:
            QMessageBox.warning(self, "Validació", "Cal ser
major de 18 anys.")
            return
        self.accept()  # Dades vàlides → tancar amb èxit

class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        boto = QPushButton("Afegir usuari")
        boto.clicked.connect(self.crea_usuari)
        self.setCentralWidget(boto)

    def crea_usuari(self):
        dlg = UsuariDialog(self)
        if dlg.exec():  # Si l'usuari accepta
            QMessageBox.information(self, "Acceptat",
                f"Usuari: {dlg.nom.text()} ({dlg.edat.value()}
anys)")
        else:
```

```
            QMessageBox.information(self, "Cancel·lat",
"Operació cancel·lada")

app = QApplication(sys.argv)
finestra = MainWindow()
finestra.resize(320, 160)
finestra.show()
sys.exit(app.exec())
```

6. Diàlegs específics



Acabem aquesta unitat veient altres exemples de diàlegs per a usos específics. És important tenir en compte que cal activar les traduccions o els textos apareixeran en anglès.

a. Diàlegs de fitxer QFileDialog

S'utilitzen per generar la ruta a un fitxer usant l'explorador, és a dir, no afecten el fitxer en si i només serveixen per saber on es troba un fitxer, ja siga per obrir-lo o per desar-lo:

```
from PySide6.QtWidgets import (
    QApplication, QMainWindow, QPushButton, QFileDialog)  #
editado

def boton_clicado(self):
    fichero, _ = QFileDialog.getOpenFileName(self, "Abrir
archivo", ".")
    print(fichero)
```

També es pot fer servir en el mode per desar un fitxer:

```
fichero, _ = QFileDialog.getSaveFileName(self, "Guardar
archivo", ".")
```

b. Diàlegs d'entrada de dades QinputDialog

Pensats per demanar una dada concreta a l'usuari:

```
from PySide6.QtWidgets import (
```

```
    QApplication, QMainWindow, QPushButton, QInputDialog)  # editado

def boton_clicado(self):
    dialogo = QInputDialog.getText(self, "Título", "Texto")
    dialogo = QInputDialog.getInt(self, "Título", "Entero")
    dialogo = QInputDialog.getDouble(self, "Título", "Decimal")
    dialogo = QInputDialog.getItem(
        self, "Título",  "Colores", ["Rojo", "Azul", "Blanco",
"Verde"])
```

Aquest diàleg torna una tupla, primer el valor i després si s'ha confirmat el diàleg. Això serveix per saber si es cancel·la la captura de dades, per això una manera de tractar la informació és en dues variables:

```
color, confirmado = QInputDialog.getItem(
    self, "Título",  "Colores", ["Rojo", "Azul", "Blanco", "Verde"])

if confirmado:
    print(color)
```

c. Diàlegs de font QFontDialog i color QcolorDialog

Aquests tenen l'objectiu de seleccionar fonts del sistema i colors.

Vegem com obrir una font per utilitzar-la en un botó:

```
from PySide6.QtWidgets import (
    QApplication, QMainWindow, QPushButton, QFontDialog)  # new

class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.resize(480, 320)

        boton = QPushButton("Mostrar diálogo")
        boton.clicked.connect(self.boton_clicado)
        self.setCentralWidget(boton)

        self.boton = boton

    def boton_clicado(self):
        confirmado, fuente = QFontDialog.getFont(self)
        if confirmado:
            # fuente es un objeto QFont
            self.boton.setFont(fuente)
```

7. Credits

Aquests apunts són un material de suport per als alumnes de DAM i han sigut desenvolupats per Herctor Docs i publicats sota una Llicència CC BY 4.0. Estos apunts han sigut modificats i adaptats per Iván Martos.