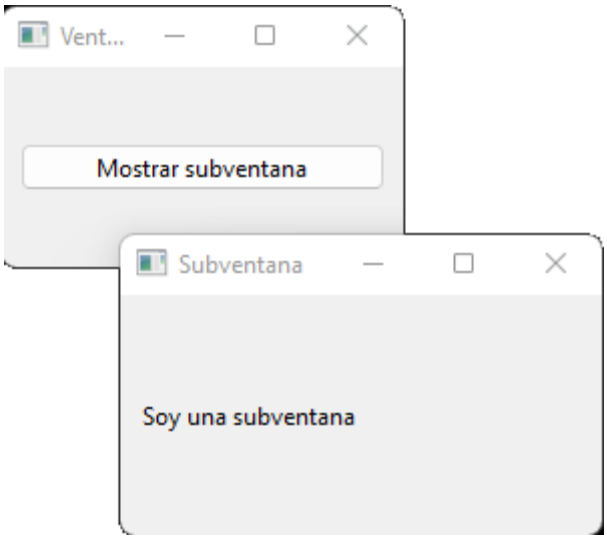
Creación de subventanas



En esta lección vamos a ver cómo manejar ventanas secundarias. Para ello partiremos del siguiente programa:

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


class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        # Le damos un tamaño y un título
        self.setWindowTitle("Ventana principal")
        # dummy widget para un layout
        layout = QVBoxLayout()
        widget = QWidget()
        widget.setLayout(layout)
        self.setCentralWidget(widget)
        # botón para abrir la subventana
        boton_mostrar = QPushButton("Mostrar subventana")
        boton_mostrar.clicked.connect(self.mostrar_subventana)
        layout.addWidget(boton_mostrar)

    def mostrar_subventana(self):
        print("Subventana abierta")


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

Para nuestra subventana vamos a partir de una clase heredada de `QWidget` que instanciaremos en el método `mostrar_subventana`:

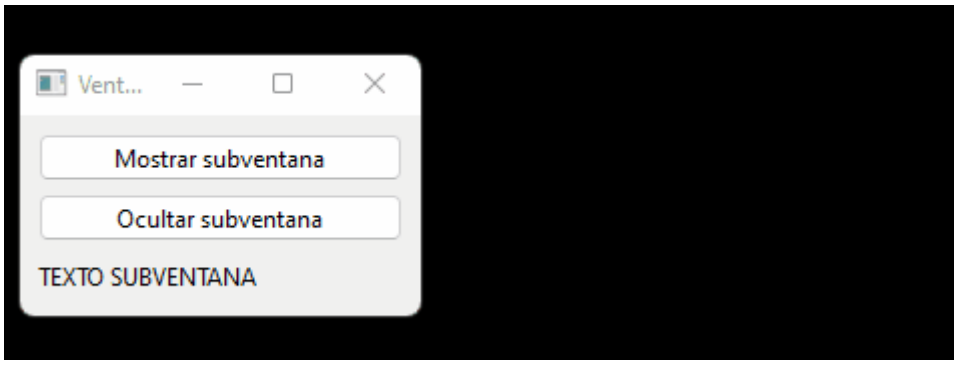
```
from PySide6.QtWidgets import (
    QApplication, QMainWindow, QVBoxLayout, QWidget, QPushButton,
QLabel)  # editado
```

```
import sys

class Subventana(QWidget):
    def __init__(self):
        super().__init__()
        # Le damos un tamaño y un título
        self.resize(240, 120)
        self.setWindowTitle("Subventana")
        # creamos una etiqueta
        etiqueta = QLabel("Soy una subventana")
        # creamos un layout y añadimos la etiqueta
        layout = QVBoxLayout()
        layout.addWidget(etiqueta)
        # asignamos el layout al widget
        self.setLayout(layout)

class MainWindow(QMainWindow):
    def mostrar_subventana(self):
        subventana = Subventana()
        subventana.show()
```

Si ejecutamos el código la subventana se creará pero desaparecerá en un instante. Esto ocurre porque la instancia existe en el ámbito de la función, cuando la función finaliza la subventana se borra de la memoria. Si queremos conservar la instancia debemos crear la instancia en la clase:

```
def mostrar_subventana(self):
    self.subventana = Subventana()
    self.subventana.show()
```

Subventanas persistentes



En este punto es importante dejar claro que cada vez que presionamos el botón "mostrar subventana" se crea una nueva subventana que sobreescribe a la anterior, esto podemos comprobarlo generando un número aleatorio en la etiqueta:

```
import random  # new

# creamos una etiqueta con texto aleatorio
etiqueta = QLabel(f"Soy una subventana... {random.randint(0, 100)}")
```

A veces generar una nueva instancia todo el rato no es un problema, pero si queremos interactuar con la subventana y que los cambios se almacenen en ella, entonces no podemos tomarnos el lujo de sobreescribirla.

Para evitar sobreescribirla guardaremos la instancia solamente si la variable de clase es nula:

```
class MainWindow(QMainWindow):
    def __init__(self):
        # iniciamos la variable de la subventana nula por defecto
        self.subventana = None

    def mostrar_subventana(self):
        # si no hay ninguna instancia la guardamos
        if not self.subventana:
            self.subventana = Subventana()
        # y la mostramos
        self.subventana.show()
```

Con esto nos aseguramos de tener almacenada la primera instancia de la subventana para no perder su contenido.

Al tenerla guardada podemos interactuar con ella, por ejemplo para esconderla desde la ventana principal:

```
# botón para cerrar la subventana
boton_cerrar = QPushButton("Ocultar subventana")
boton_cerrar.clicked.connect(self.ocultar_subventana)
layout.addWidget(boton_cerrar)

def ocultar_subventana(self):
    # si la subventana existe y es visible la escondemos
    if self.subventana and self.subventana.isVisible():
        self.subventana.hide()
```

Ahora bien, si os lo paráis a pensar, dado que solo necesitamos una instancia de la subventana, podríamos crearla desde el principio en la ventana principal:

```
# creamos una instancia de la subventana al principio
self.subventana = Subventana()
```

Al tener la subventana desde el principio podemos acceder a sus métodos `show` y `hide`, pero deberemos llamarlos como funciones anónimas:

```
# botón para mostrar la subventana
boton_abrir = QPushButton("Mostrar subventana")
boton_abrir.clicked.connect(lambda: self.subventana.show())
layout.addWidget(boton_abrir)
# botón para ocultar la subventana
boton_cerrar = QPushButton("Ocultar subventana")
boton_cerrar.clicked.connect(lambda: self.subventana.hide())
layout.addWidget(boton_cerrar)

##### ===> Borramos los métodos mostrar_ventana() y ocultar_ventana()
```

Las funciones anónimas se crean en tiempo de ejecución y nos aseguran de que la subventana existe antes de llamar a sus métodos.

Vamos a crear un tercer botón alternador en única línea que pueda mostrar y ocultar la subventana aprovechando el potencial de estas funciones:

```
# botón para alternar la subventana
boton_alternador = QPushButton("Alternar subventana")
boton_alternador.clicked.connect(
    lambda: self.subventana.hide()
    if self.subventana.isVisible()
    else self.subventana.show())
layout.addWidget(boton_alternador)
```

Ahora ya sabéis como trabajar con subventanas y comunicaros con ellas, ahora es cuestión de echarle imaginación.
