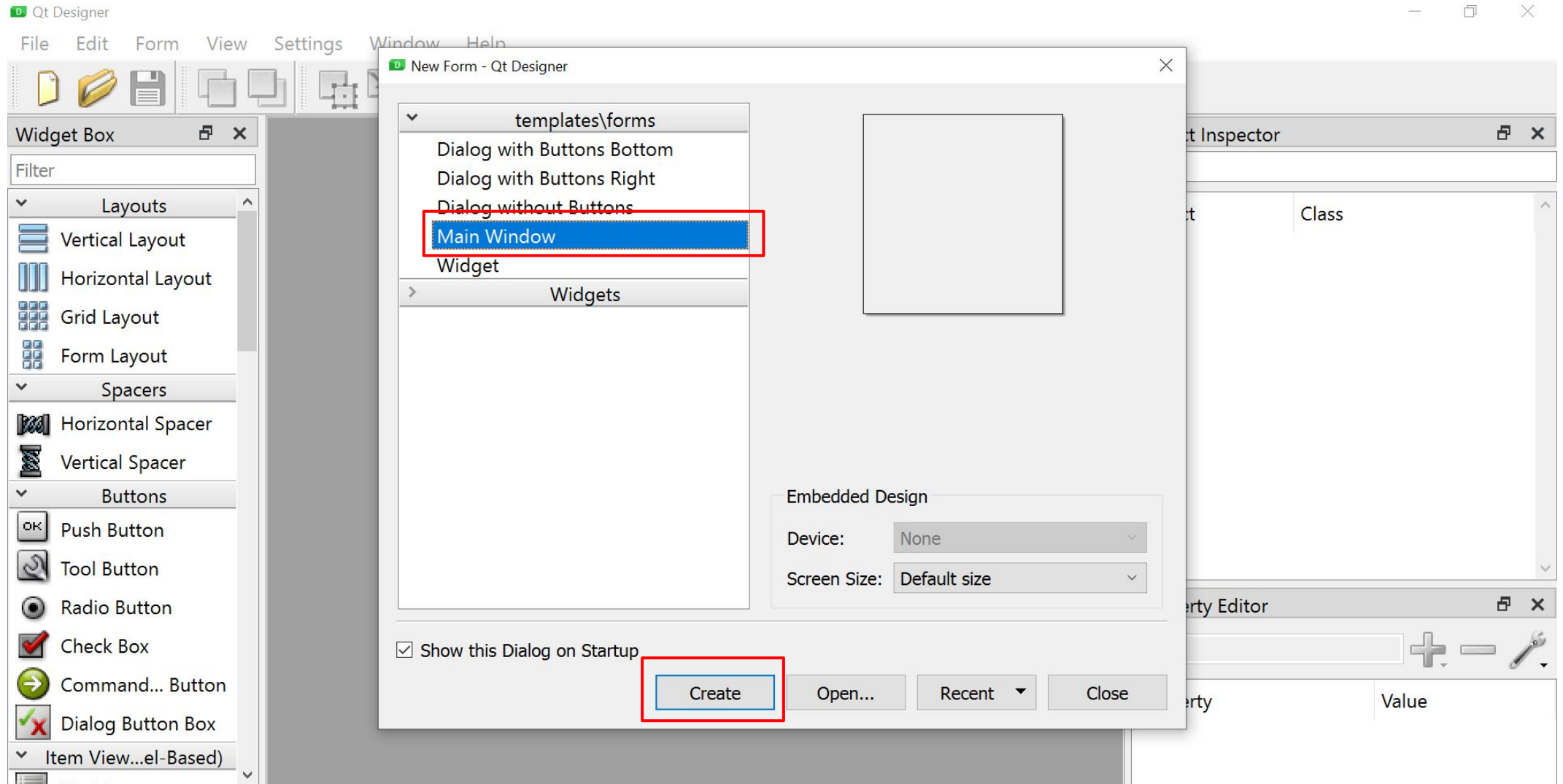


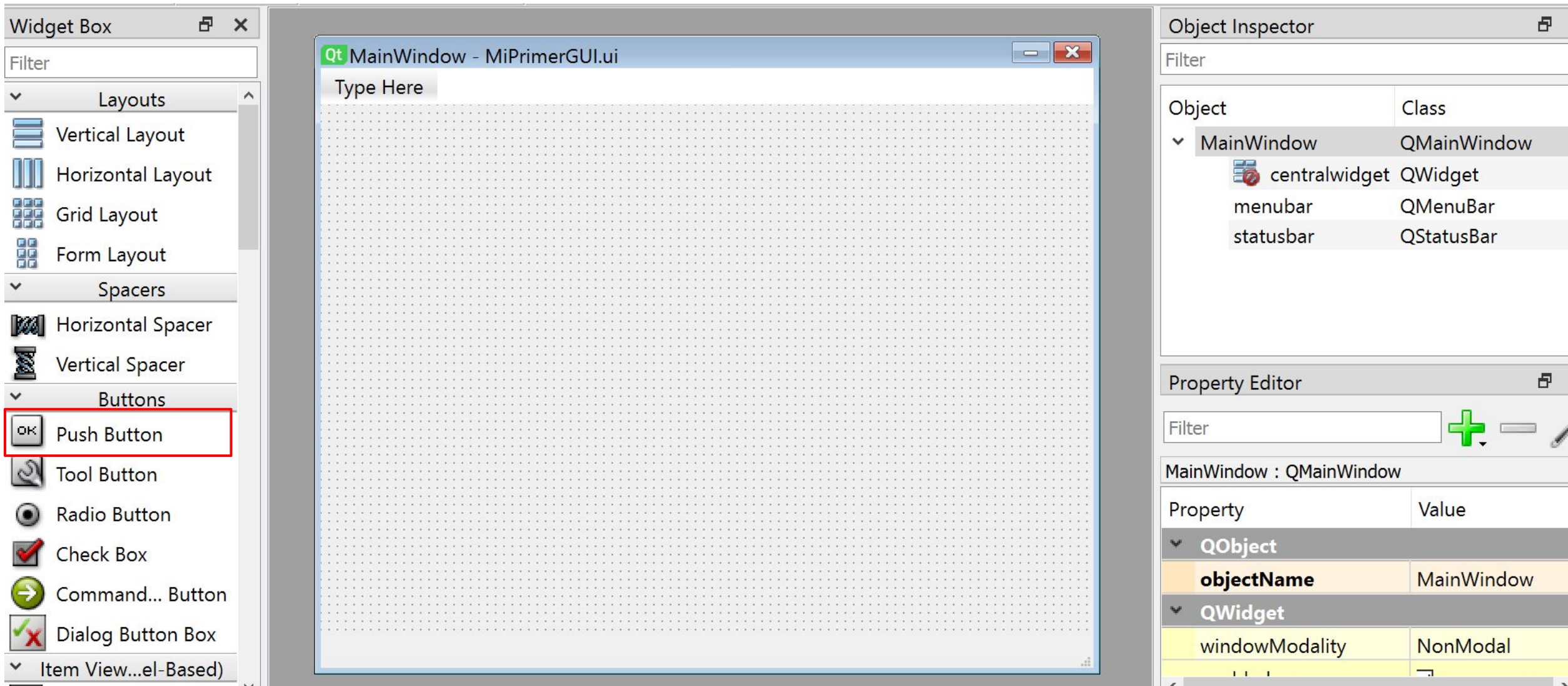
GUI con Qt Designer

Procesamiento de Señales e Imágenes Biomédicas

1- GUI “Hola mundo”

Ventana principal





Arrastrar un botón en la MainWindow

Qt MainWindow - MiPrimerGUI.ui*

Type Here

PushButton

Filter

Object	Class
▼ MainWindow	QMainWindow
▼ centralwidget	QWidget
pushB...ickMe	QPushButton
menubar	QMenuBar
statusbar	QStatusBar

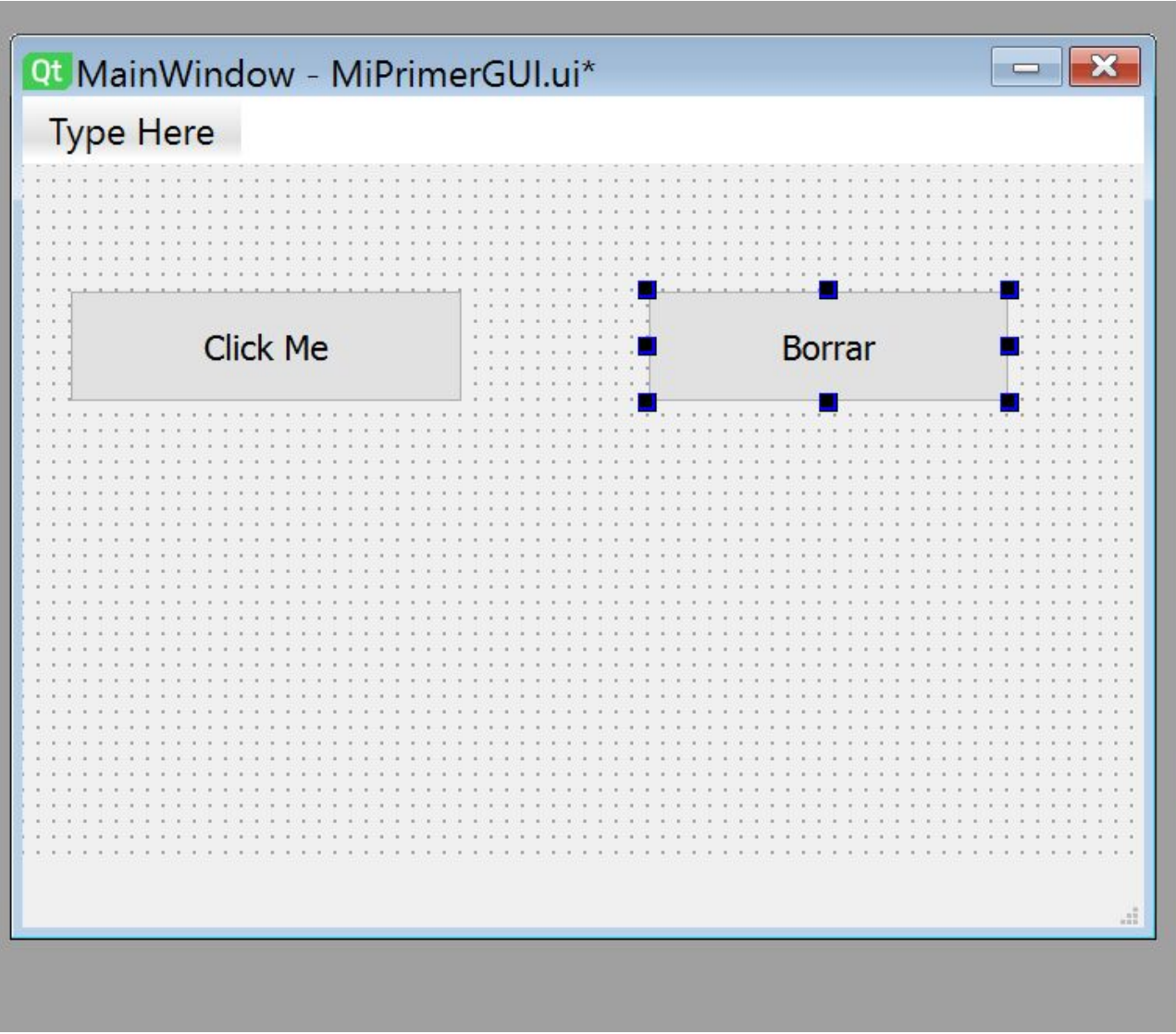
Property Editor

Filter

pushButton_clickMe : QPushButton

Property	Value
▼ QObject	
objectName	pushButton_clickMe
▼ QWidget	
enabled	<input checked="" type="checkbox"/>
▼ geometry	[(15, 40), 176 x 36]
X	15

Cambiar los nombres de los widget.



Object

Class

▼

MainWindowQMainWindow

▼

centralwidgetQWidget

pushB...ickMeQPushButton

pushB...eleteQPushButton

menubarQMenuBar

Property Editor

Filter

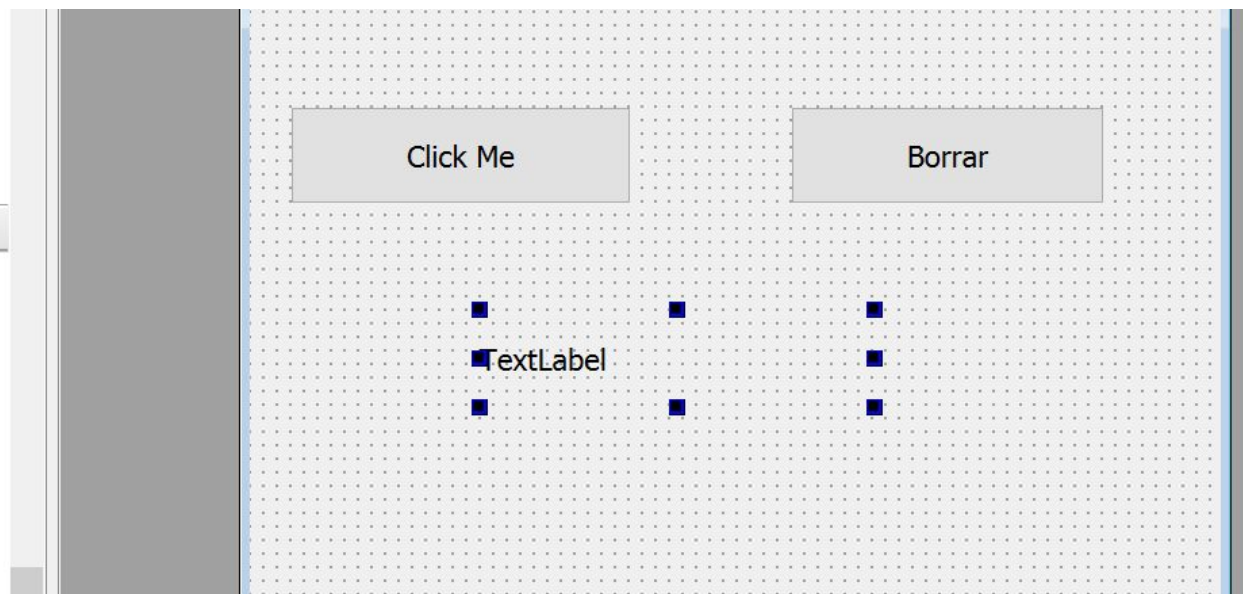
+

-

pushButton_delete : QPushButton

Property	Value
▼ QObject	
objectName	pushButton_delete
▼ QWidget	
enabled	<input checked="" type="checkbox"/>
▼ geometry	[(200, 40), 116 x 36]

- Horizontal Slider
- Vertical Slider
- Key Sequence Edit
- Display Widgets
 - Label**
 - Text Browser
 - Graphics View
 - Calendar Widget
 - LCD Number



label_terminal QLabel
pushB...ickMe QPushButton
pushB...elete QPushButton

Property Editor

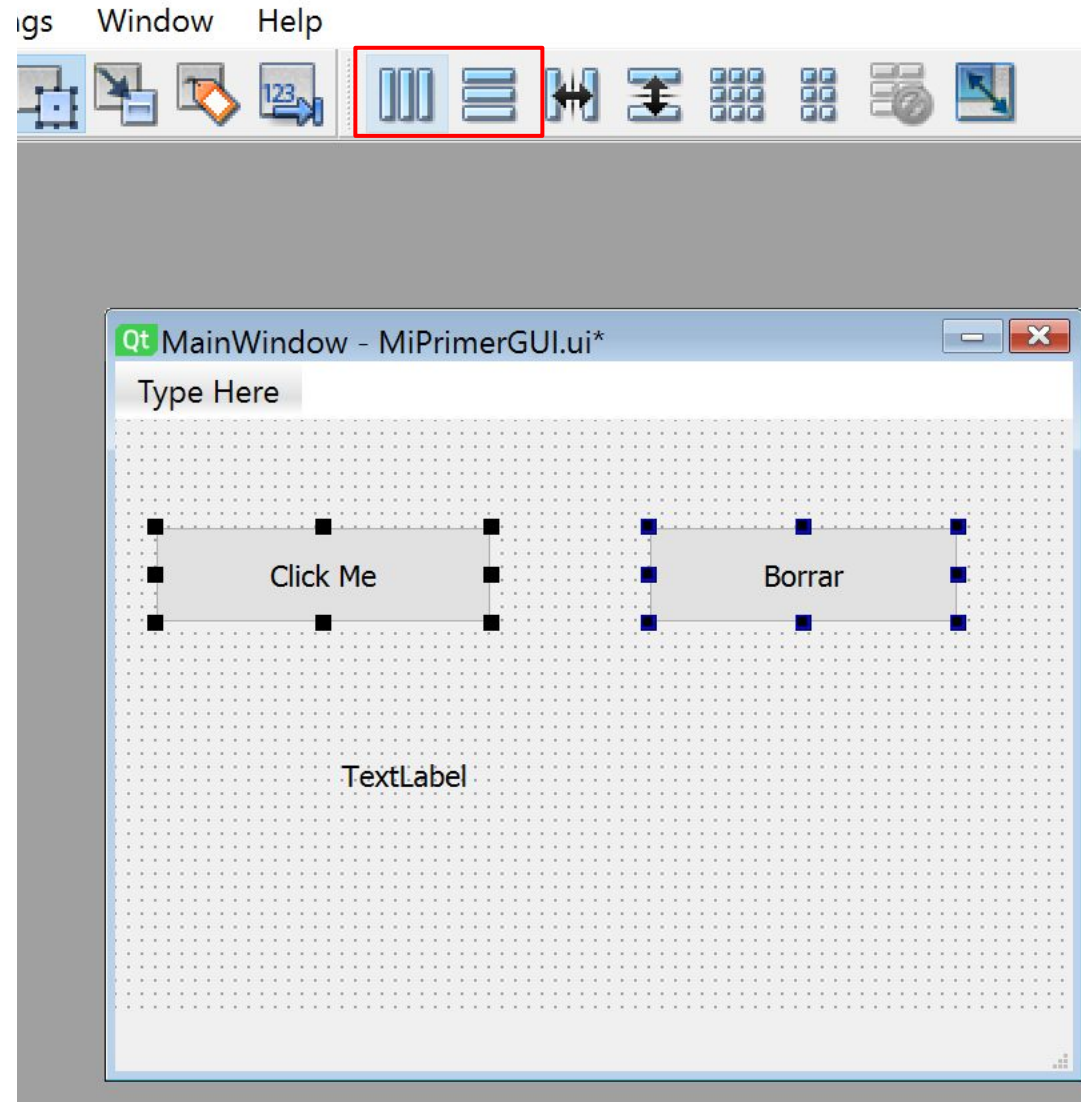
Filter

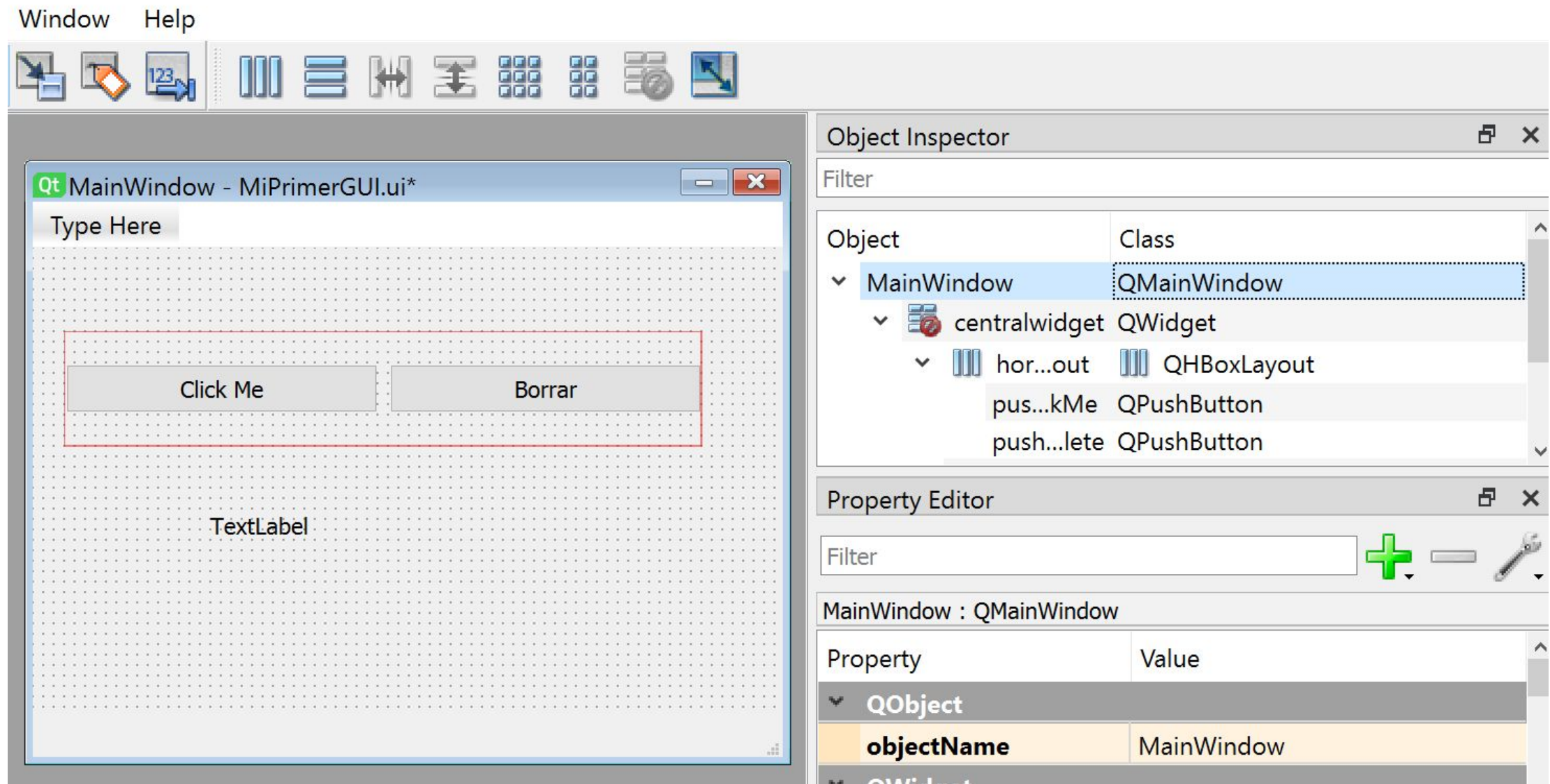
label_terminal : QLabel

Property	Value
objectName	label_terminal

Ahora nos queda alinear los widget:

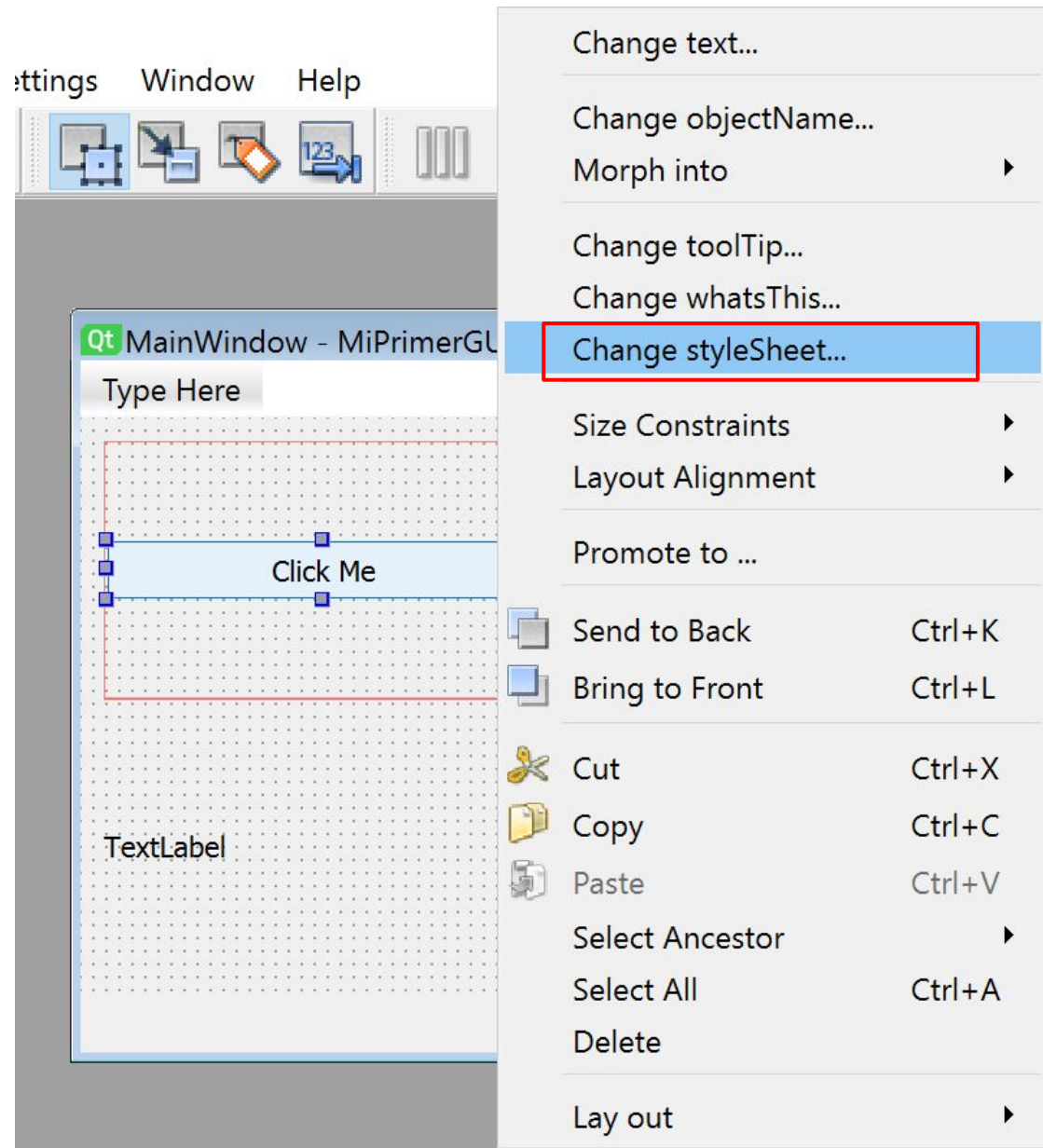
Ver que se seleccionan los dos botones.

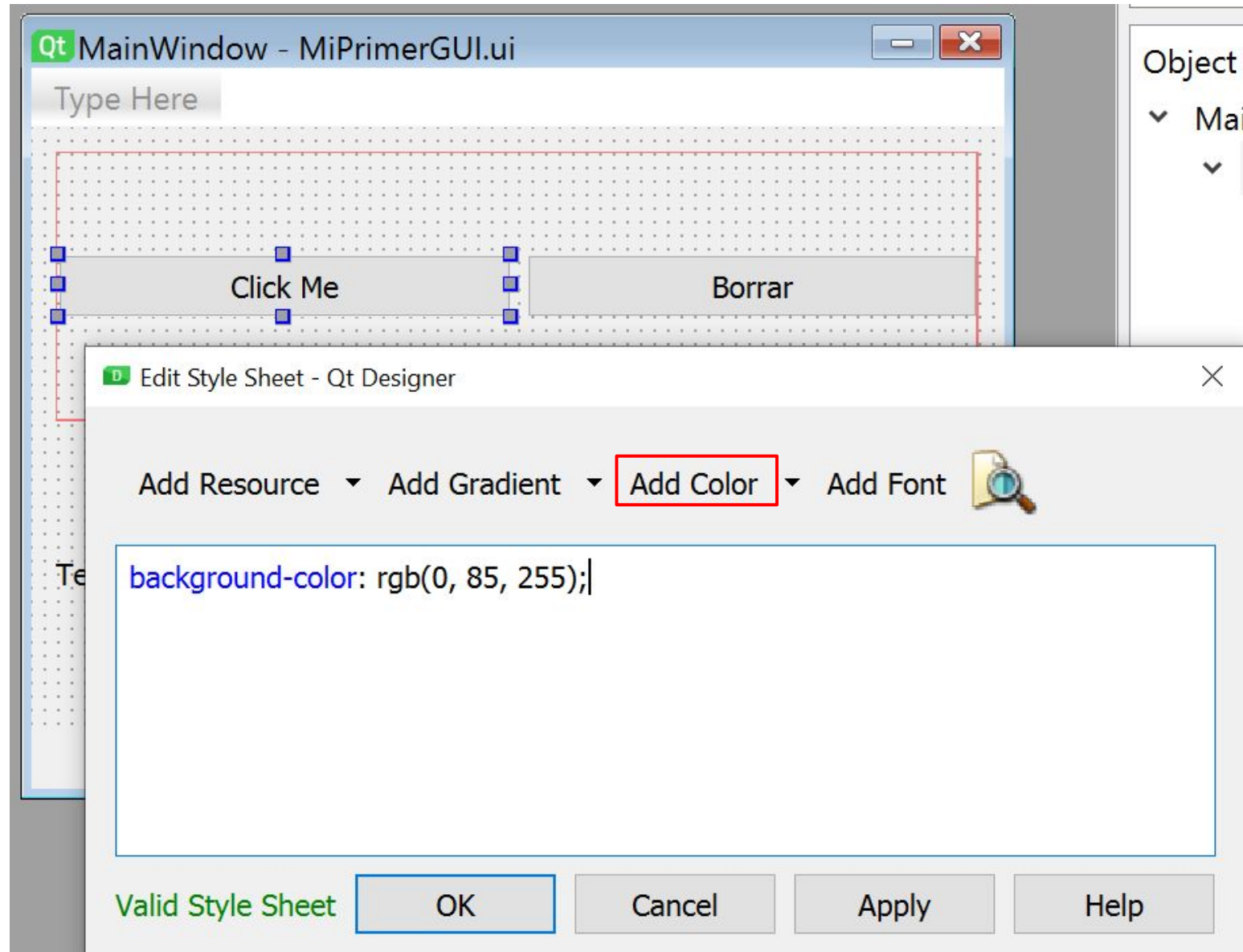




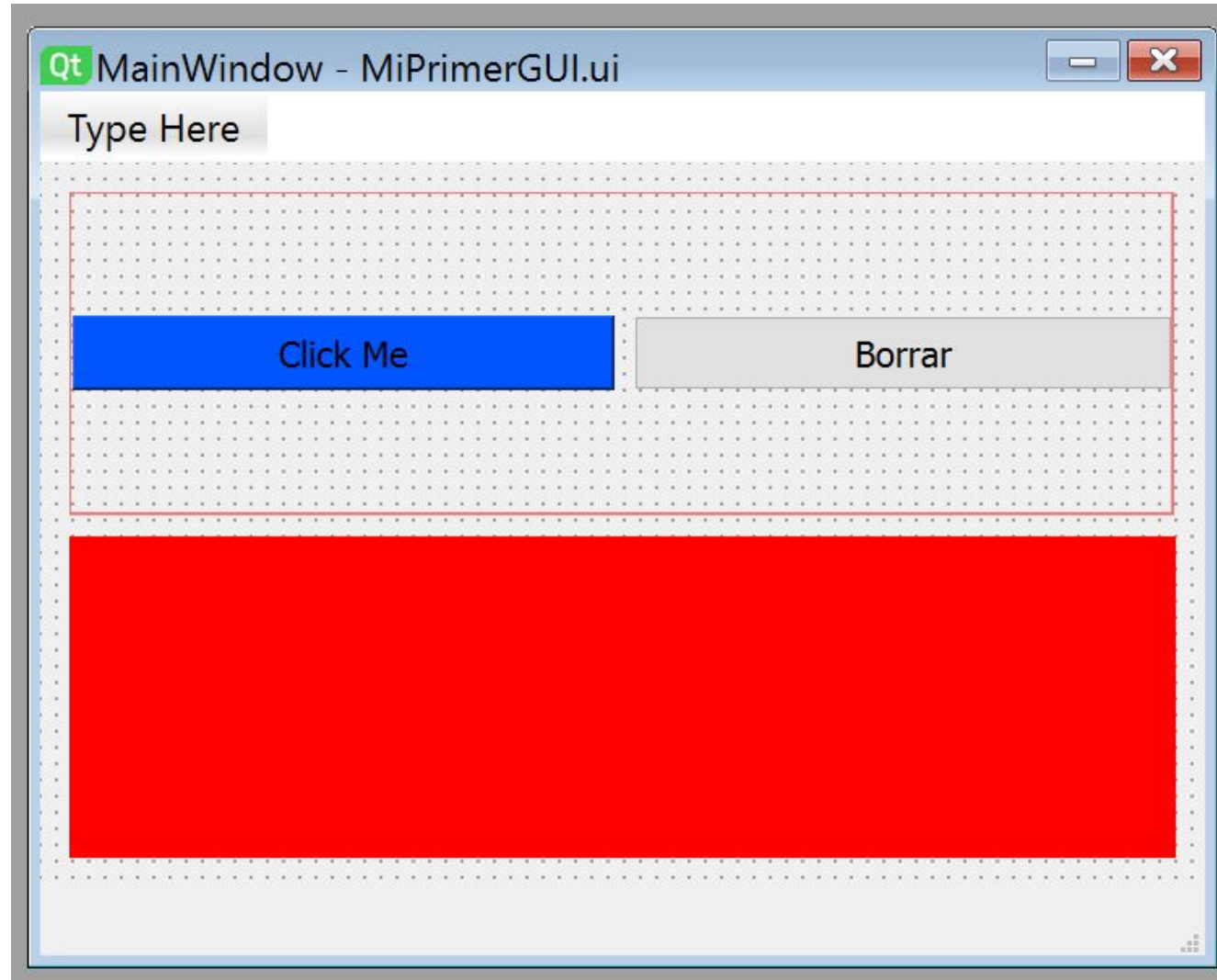
Si seleccionamos la MainWindow le podemos dar un layout a toda la ventana.

Con el botón derecho sobre un widget podemos cambiar su styleSheet..





Vista final de la GUI



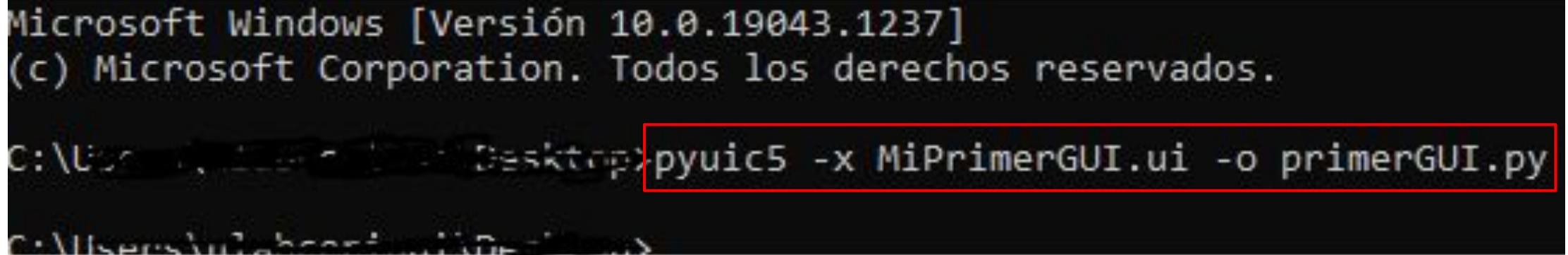
Ahora tenemos que guardar el archivo .ui

Luego pasamos a Python, donde existen dos caminos posibles:

- 1) Convertir el archivo .ui a .py
- 2) Cargar directamente el archivo .ui desde el script en .py

Convertir el archivo .ui a .py

Este enfoque consiste en transformar el archivo .ui (que es XML) en un archivo de código Python que define la interfaz gráfica. Para hacerlo, se usa una herramienta como pyuic5 (si usás PyQt5) o pyuic6 (para PyQt6), por ejemplo:



```
Microsoft Windows [Versión 10.0.19043.1237]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\jorge\Desktop>pyuic5 -x MiPrimerGUI.ui -o primerGUI.py

C:\Users\jorge\Desktop>
```

Esto genera un archivo interfaz.py con todas las clases y widgets definidos en Python. Luego, simplemente lo importas en tu script y lo usás como cualquier módulo de Python.

- Ventaja: No se necesita el archivo .ui en tiempo de ejecución.
- Desventaja: Si haces cambios en el archivo .ui, tenés que volver a convertirlo manualmente.

Cargar directamente el archivo .ui

En este caso, el archivo .ui se mantiene sin convertir, y lo cargás dinámicamente usando una función como **uic.loadUi** de PyQt5:

```
from PyQt5 import uic
from PyQt5.QtWidgets import QApplication, QMainWindow

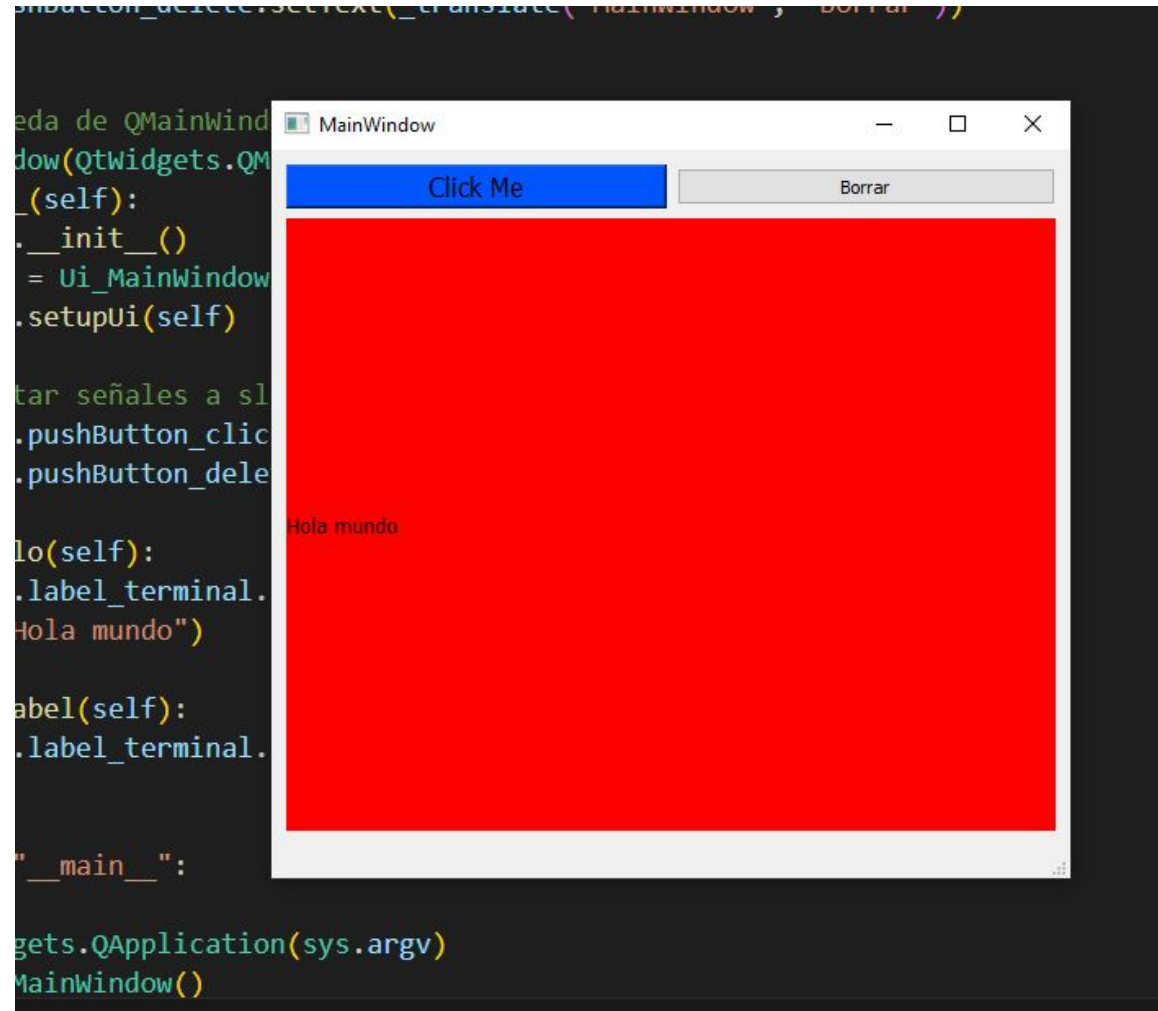
class MiVentana(QMainWindow):
    def __init__(self):
        super().__init__()
        uic.loadUi("interfaz.ui", self)

app = QApplication([])
ventana = MiVentana()
ventana.show()
app.exec_()
```

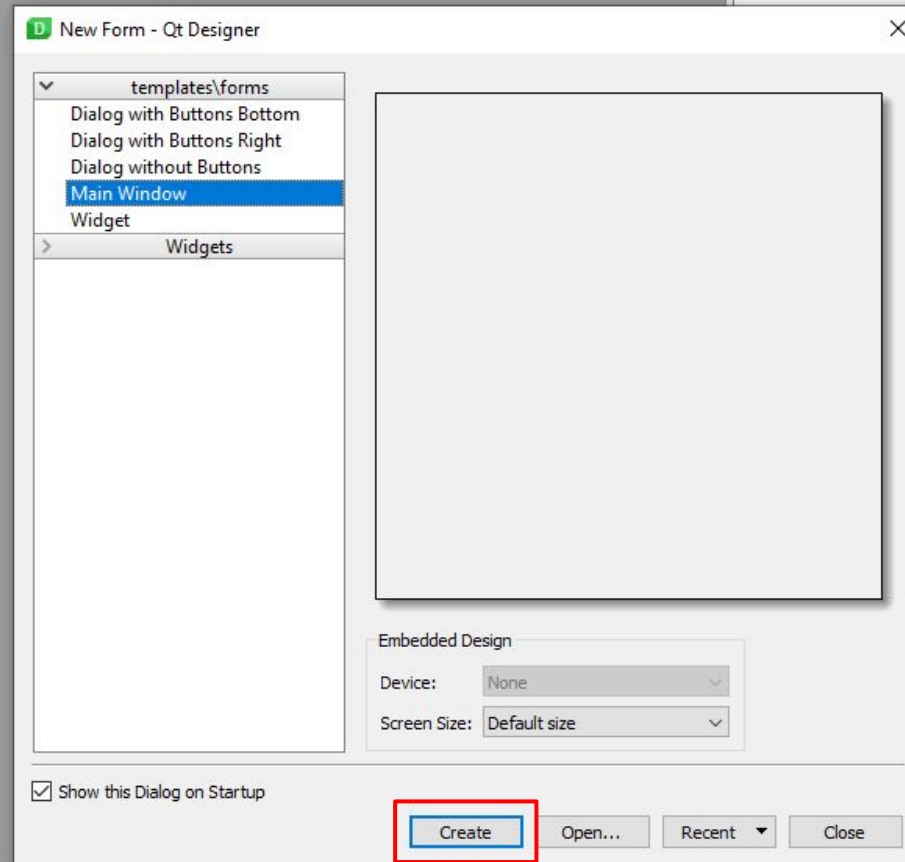
Ventaja: Podés modificar el archivo .ui sin necesidad de convertirlo cada vez.

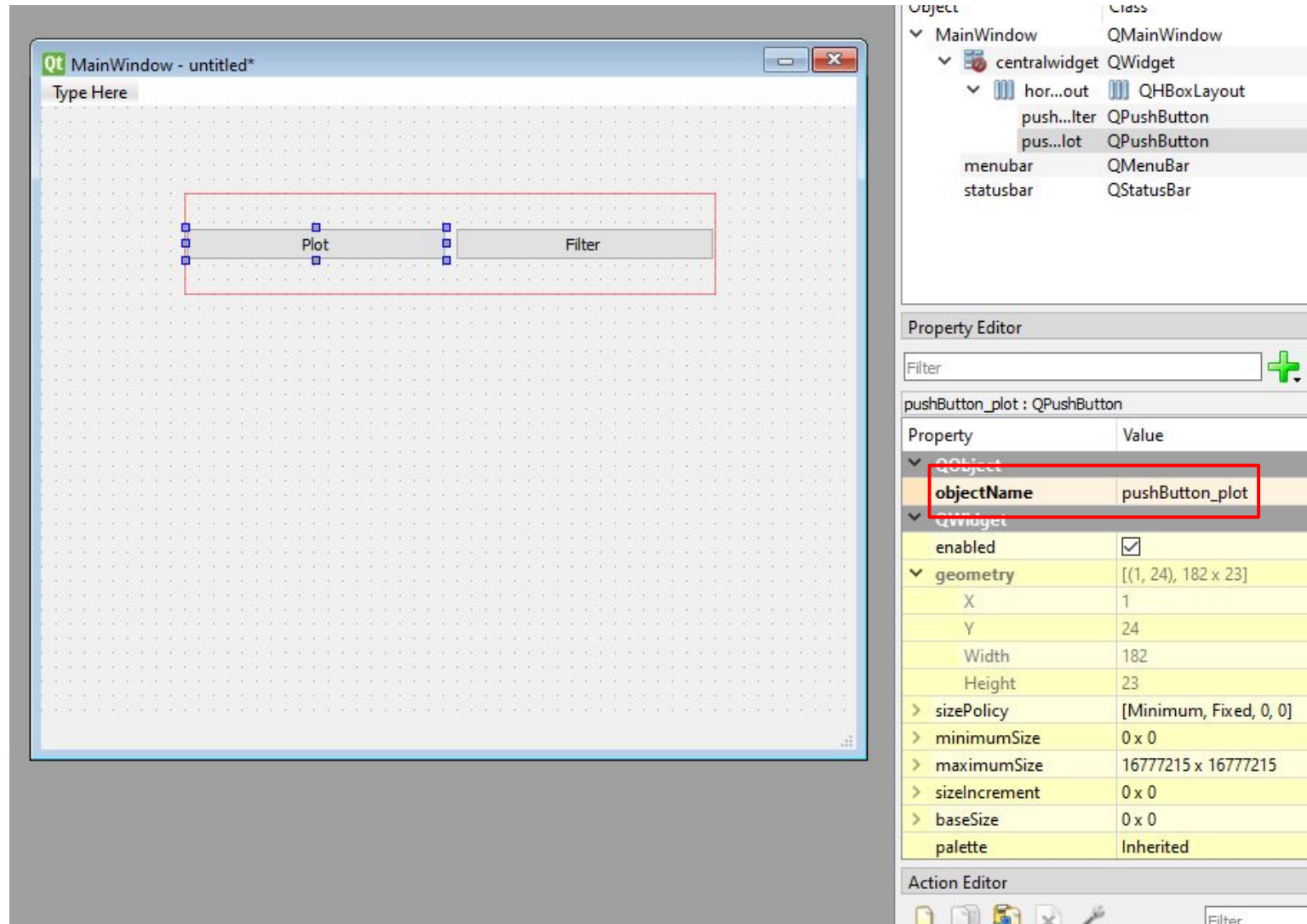
Desventaja: El archivo .ui debe estar disponible en el mismo directorio o ruta definida al ejecutar el script.

Vista desde python

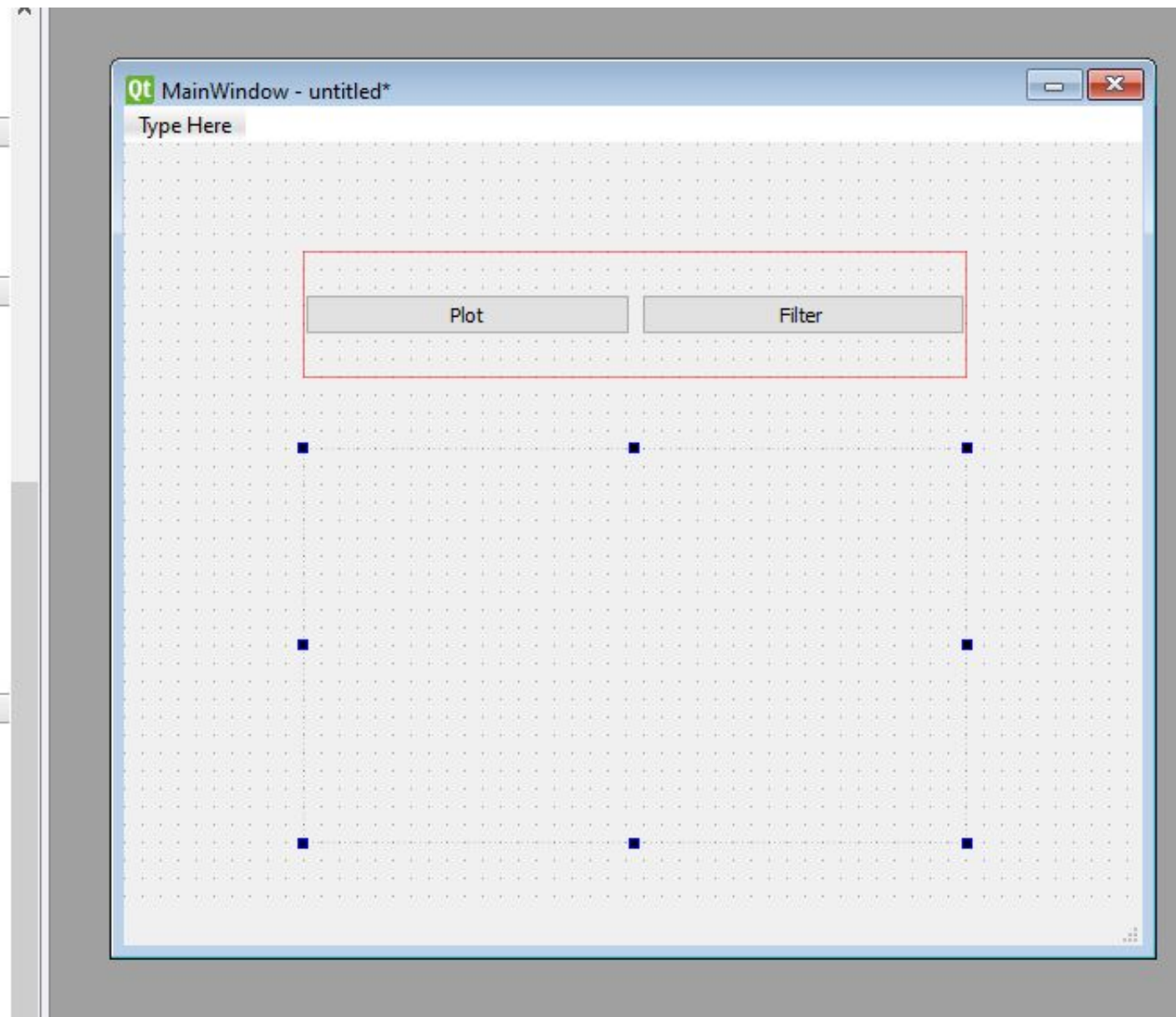
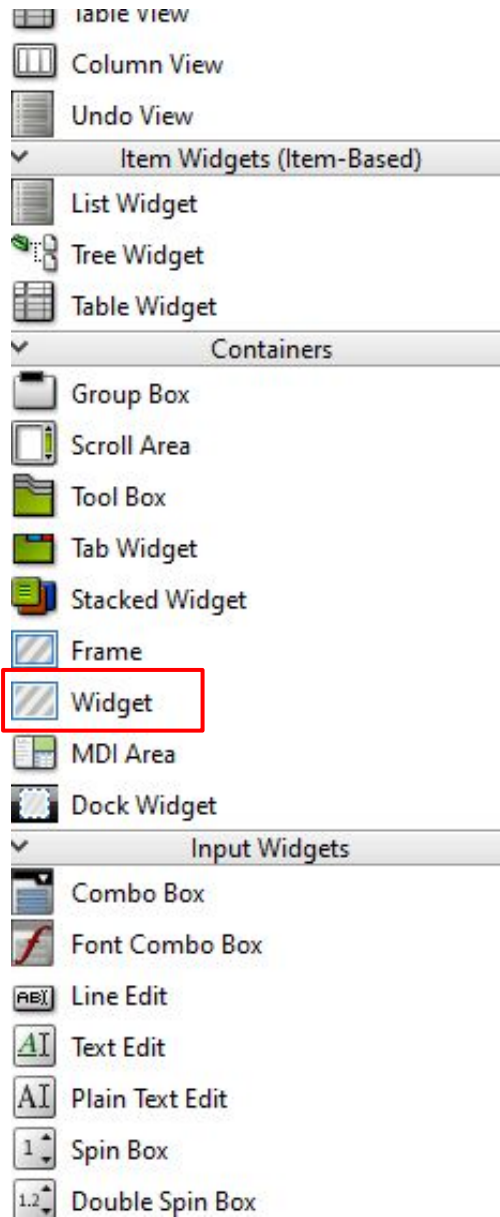


2- GUI Image Plot

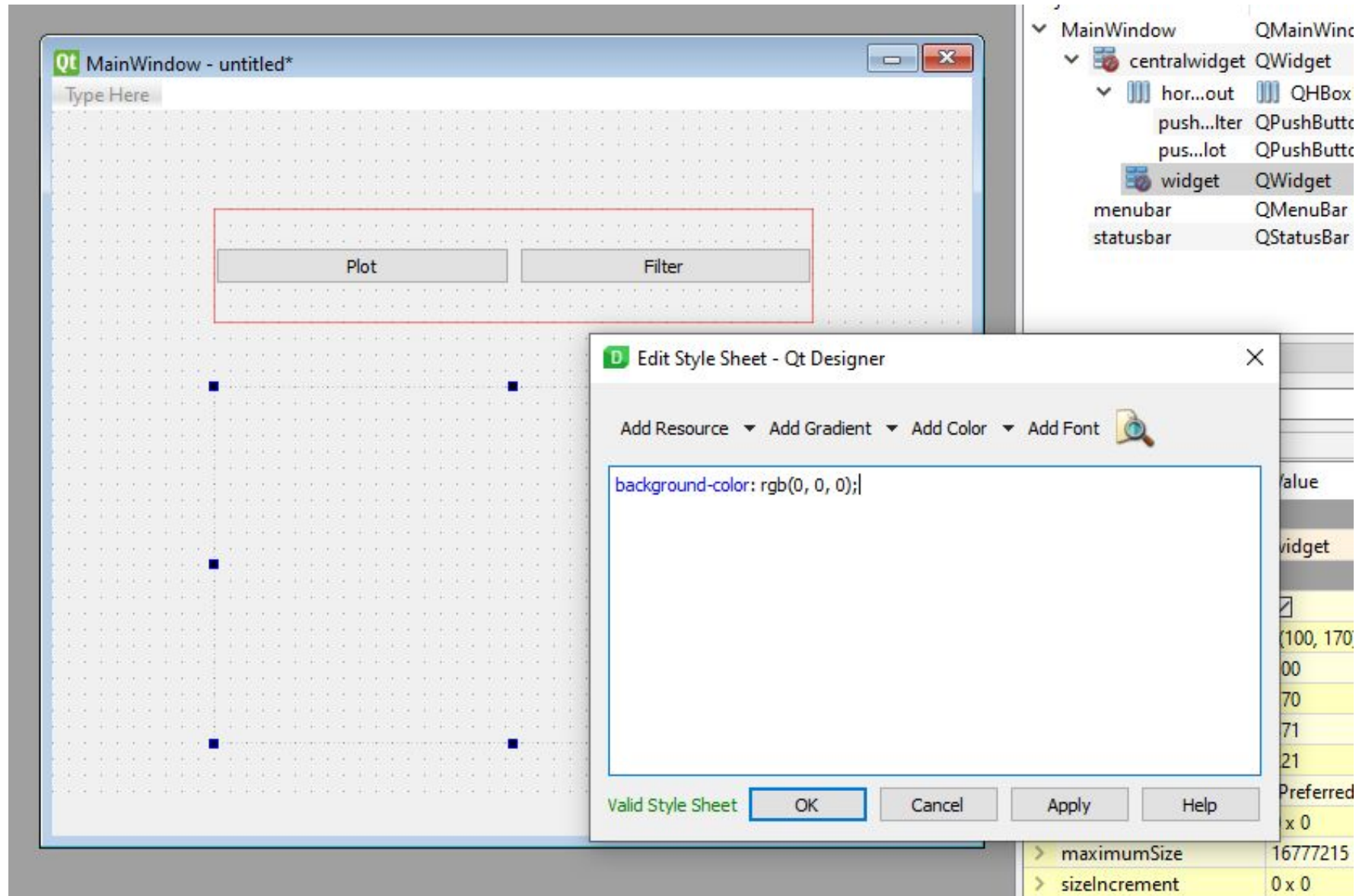




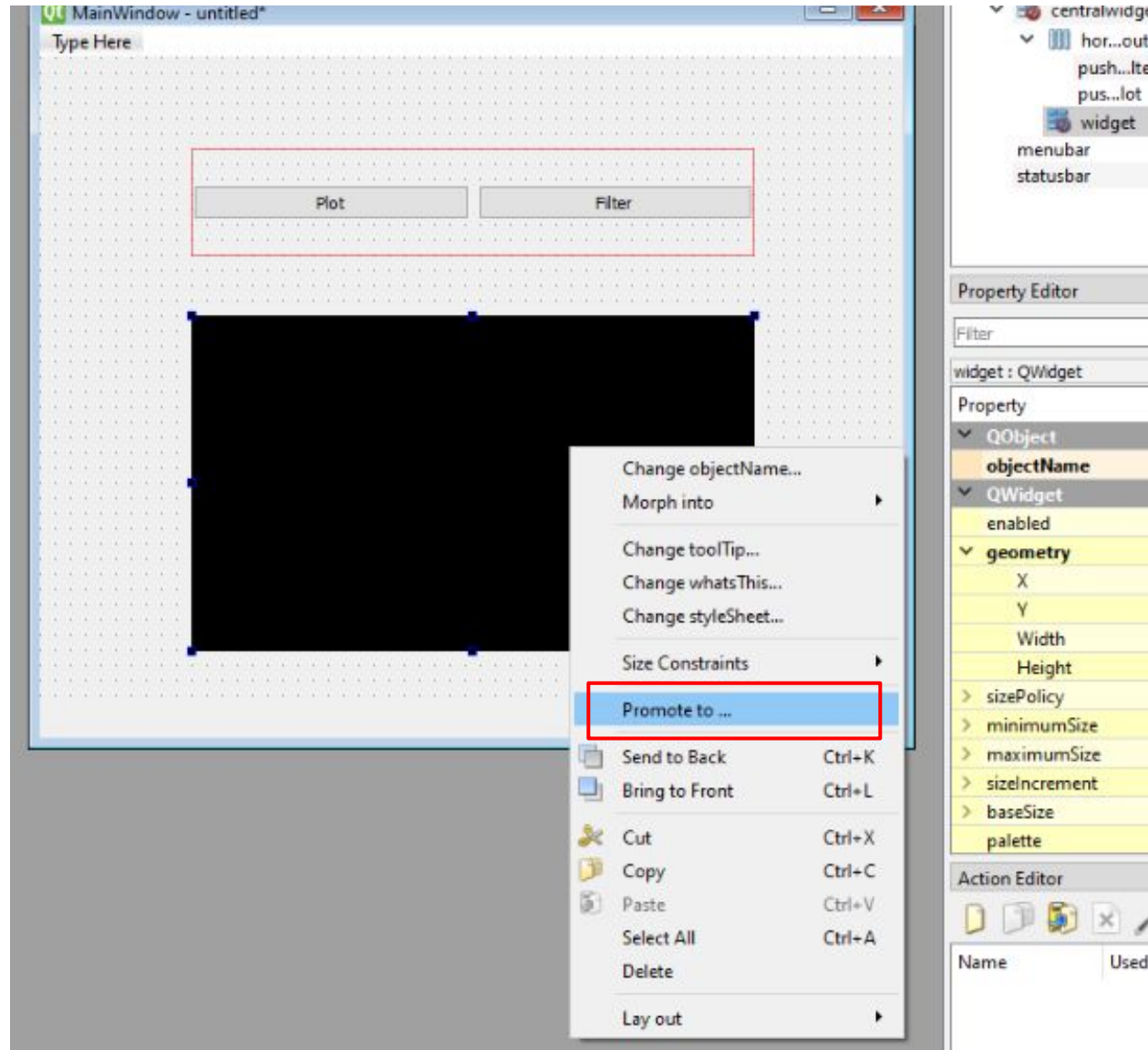
Agregamos dos botones, les cambiamos los nombres y los alineamos.



Agregamos un widget



Botón derecho → Change StyleSheet → Add Color → Background color



Sobre el widget, botón derecho → Promote to...

Promoted Classes

Name	Header file	Global include	Usage



New Promoted Class

Base class name:

QWidget



Add

Promoted class name:

ImageView

Reset

Header file:

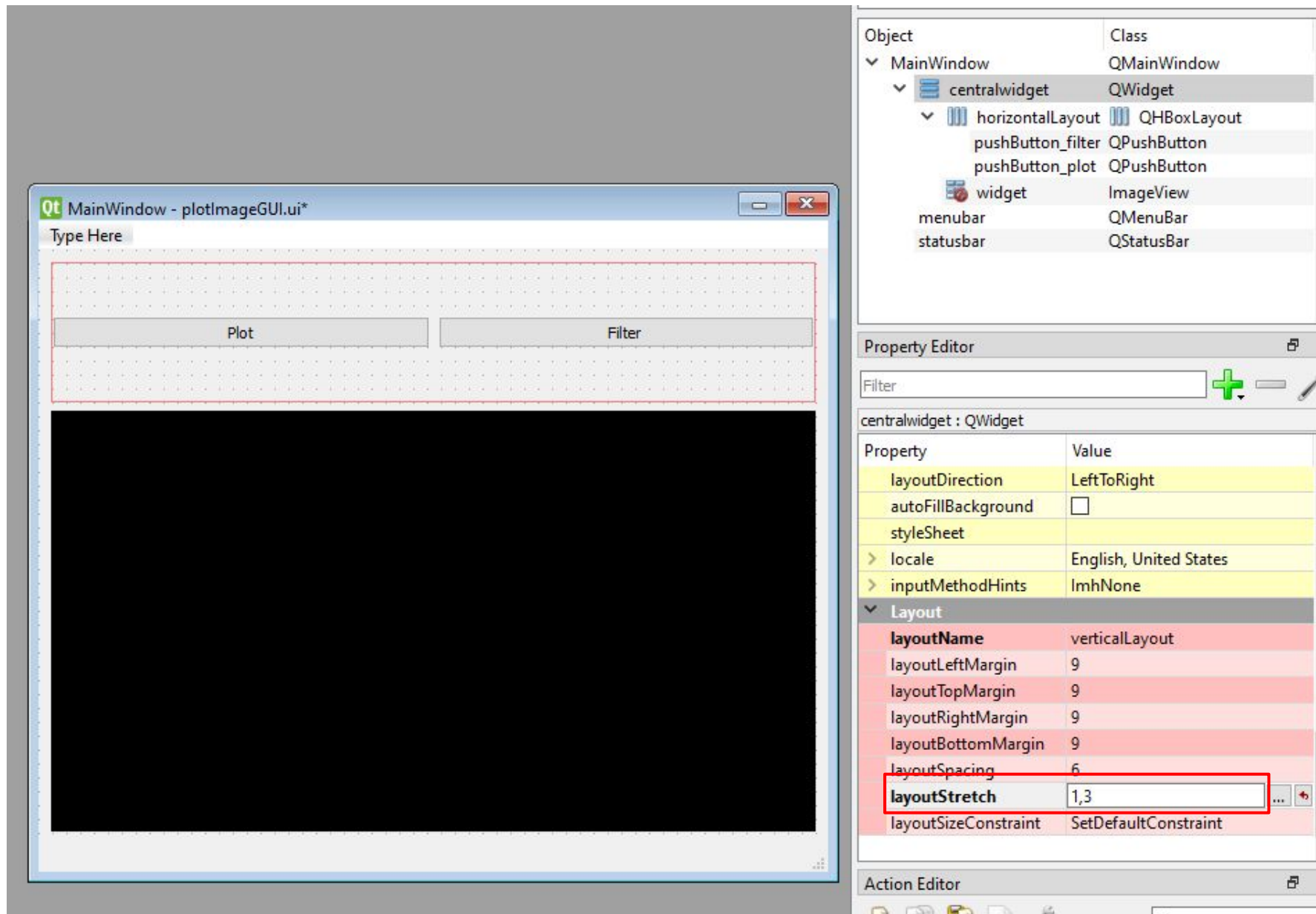
pyqtgraph

Global include

☐

Promote

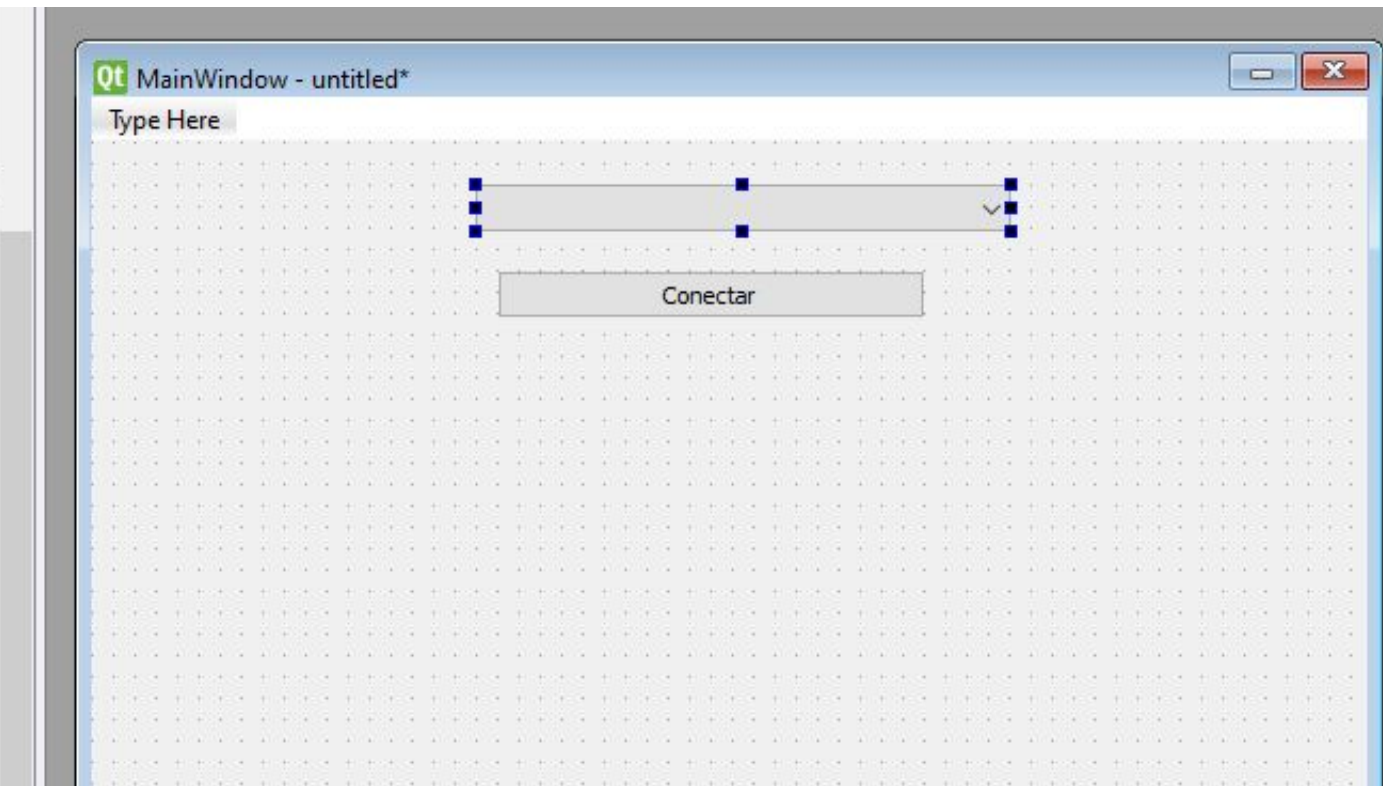
Close



Y después alinear todo como hicimos anteriormente.

3- GUI Arduino signal plot

- Widget
- MDI Area
- Dock Widget
- Input Widgets
 - Combo Box
 - Font Combo Box
 - Line Edit
 - Text Edit
 - Plain Text Edit
 - Spin Box
 - Double Spin Box
 - Time Edit
 - Date Edit
 - Date/Time Edit
 - Dial

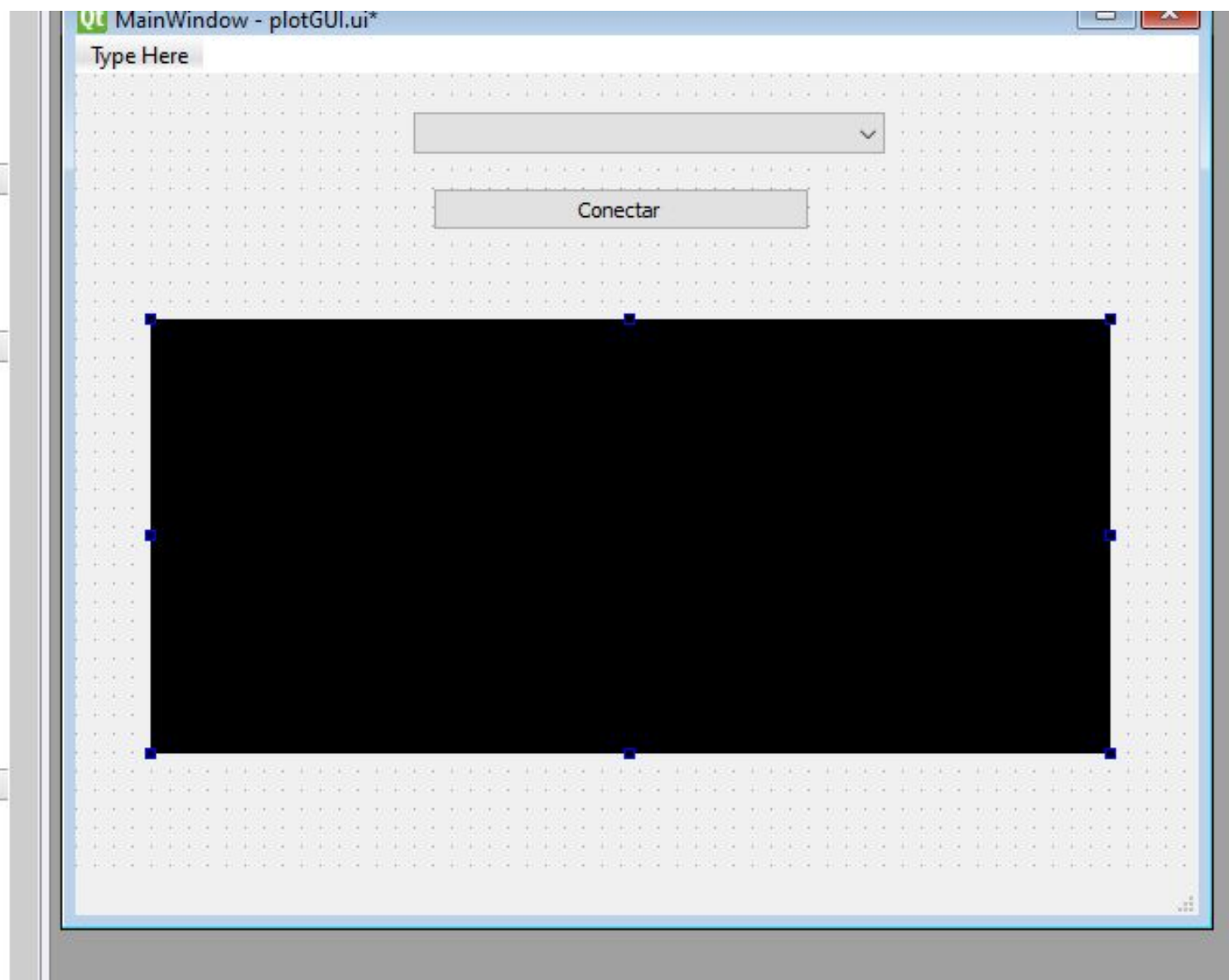
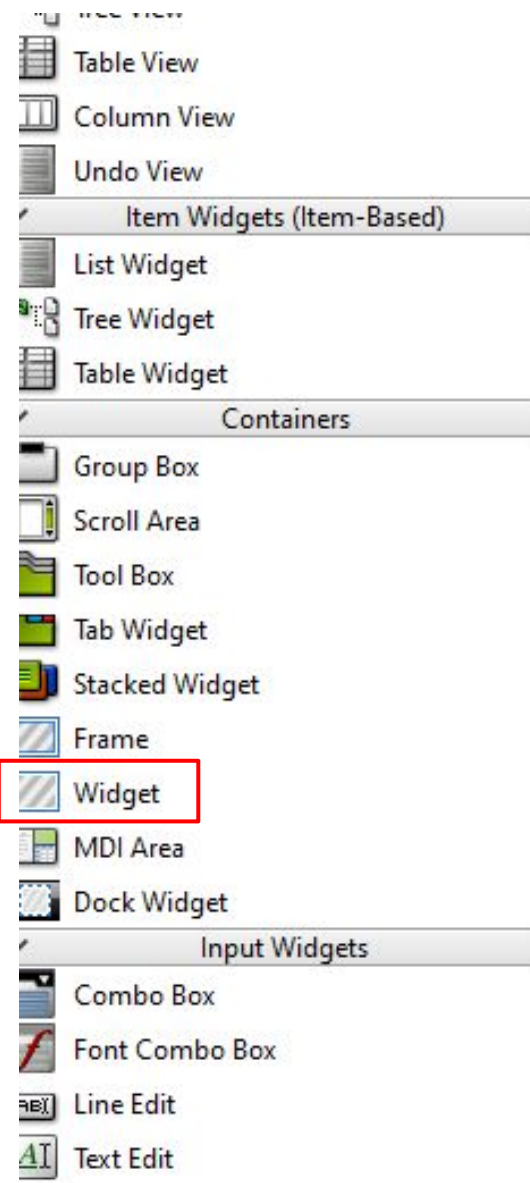


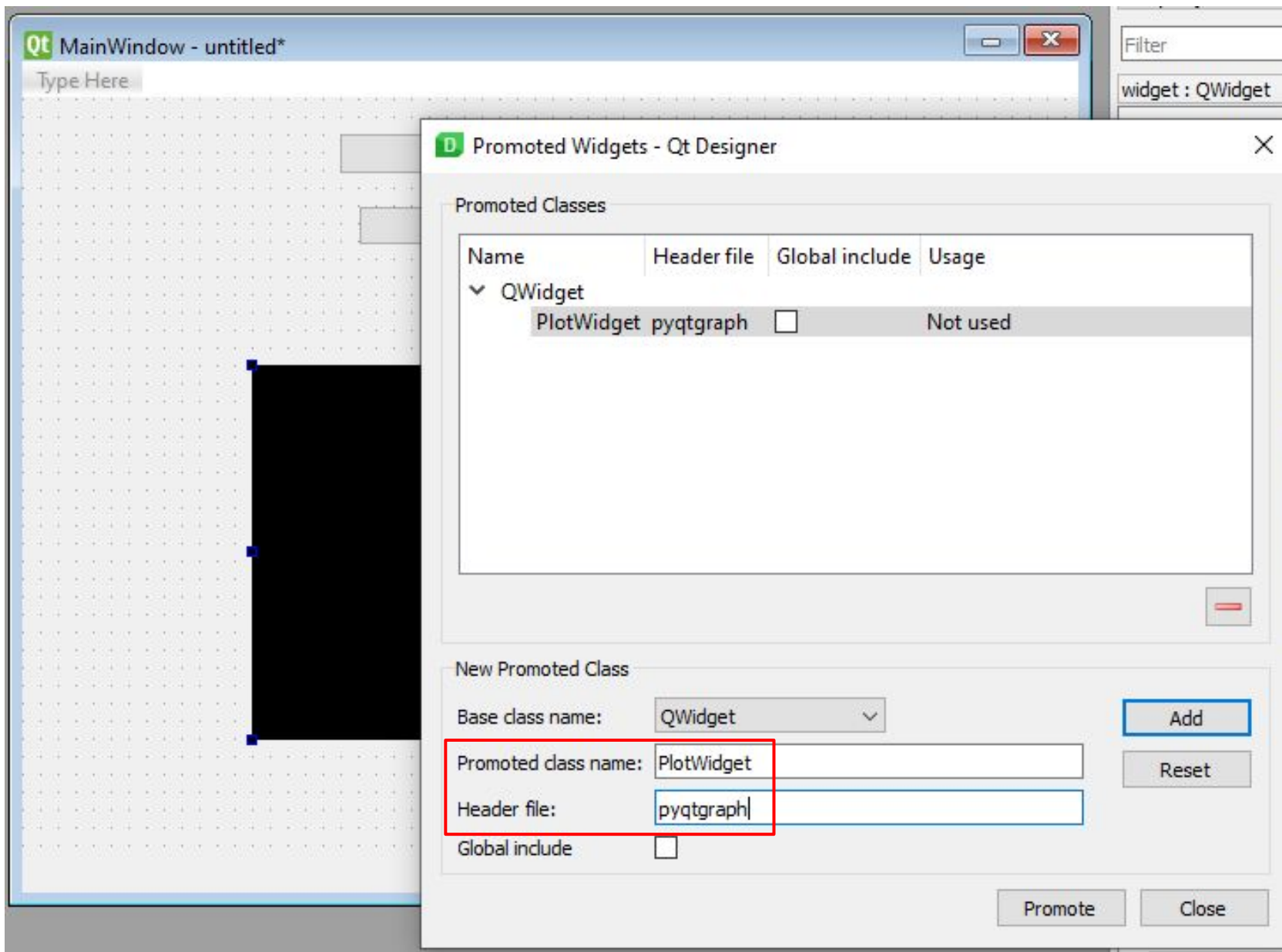
Property Editor

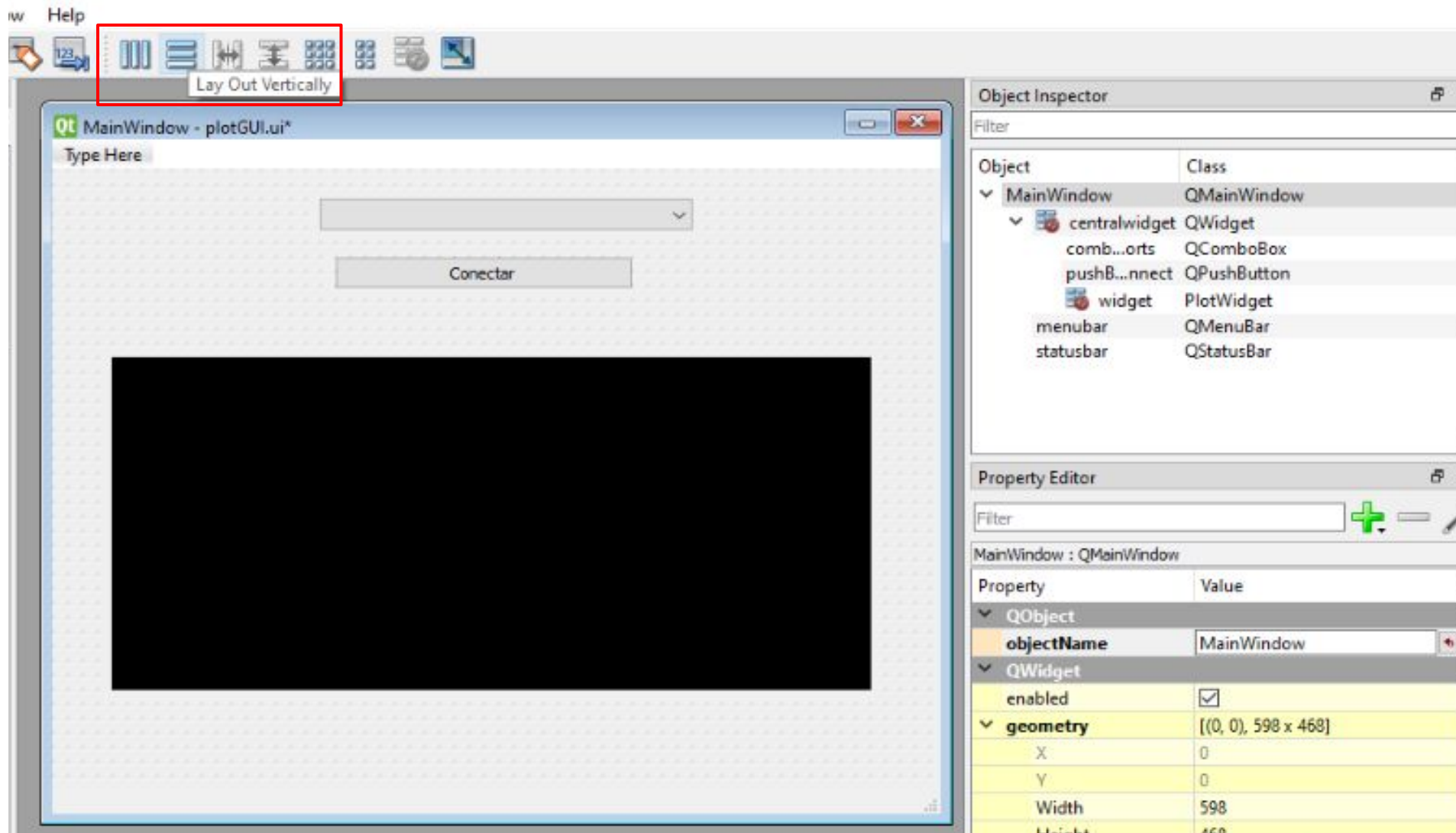
Filter

comboBox_ports : QComboBox

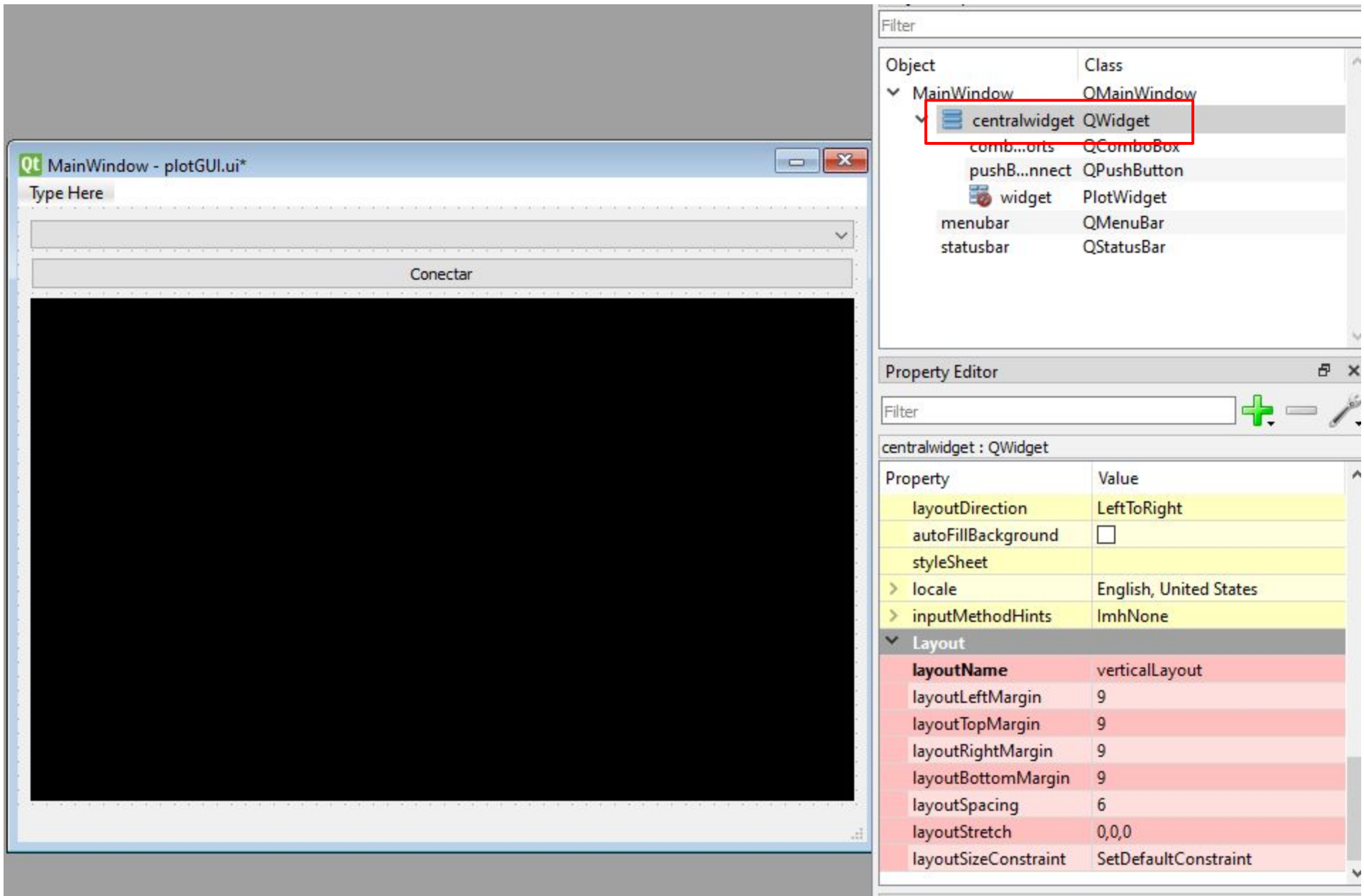
Property	Value
QObject	
objectName	comboBox_ports
QWidget	
enabled	<input checked="" type="checkbox"/>
geometry	[(180, 20), 251 x 22]
X	180
Y	20
Width	251
Height	22
sizePolicy	[Preferred, Fixed, 0, 0]
minimumSize	0 x 0
maximumSize	16777215 x 16777215
sizeIncrement	0 x 0
baseSize	0 x 0

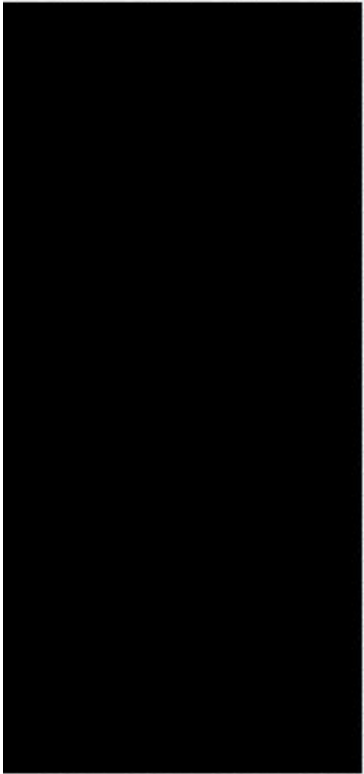






Seleccionada la MainWindow, apretar el botón de “Lay Out Vertically”





Object	Class
▼ MainWindow	QMainWindow
▼ centralwidget	QWidget
comb...orts	QComboBox
pushB...nnect	QPushButton
widget	PlotWidget
menubar	QMenuBar
statusbar	QStatusBar

Property Editor

Filter

centralwidget : QWidget

Property	Value
layoutDirection	LeftToRight
autoFillBackground	<input type="checkbox"/>
styleSheet	
> locale	English, United States
> inputMethodHints	ImhNone
▼ Layout	
layoutName	verticalLayout
layoutLeftMargin	9
layoutTopMargin	9
layoutRightMargin	9
layoutBottomMargin	9
layoutSpacing	6
layoutStretch	1,1,4
layoutSizeConstraint	SetDefaultConstraint

Vista Final

