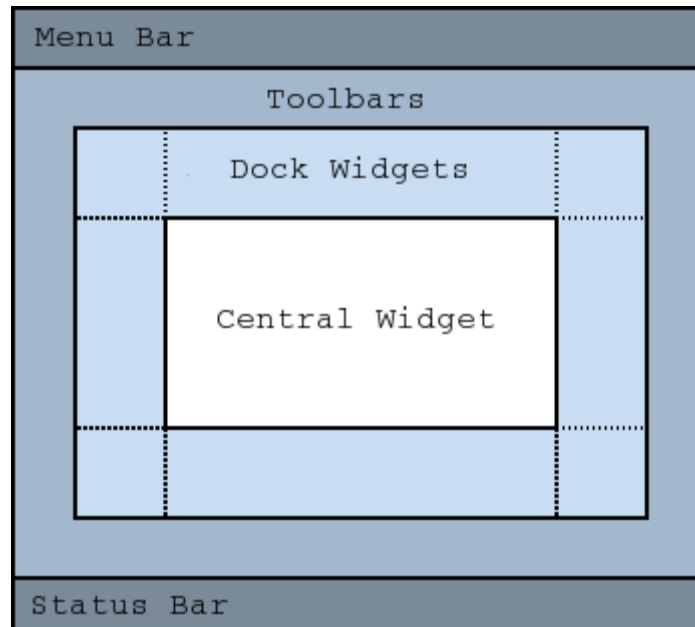


Profundizando en el diseño

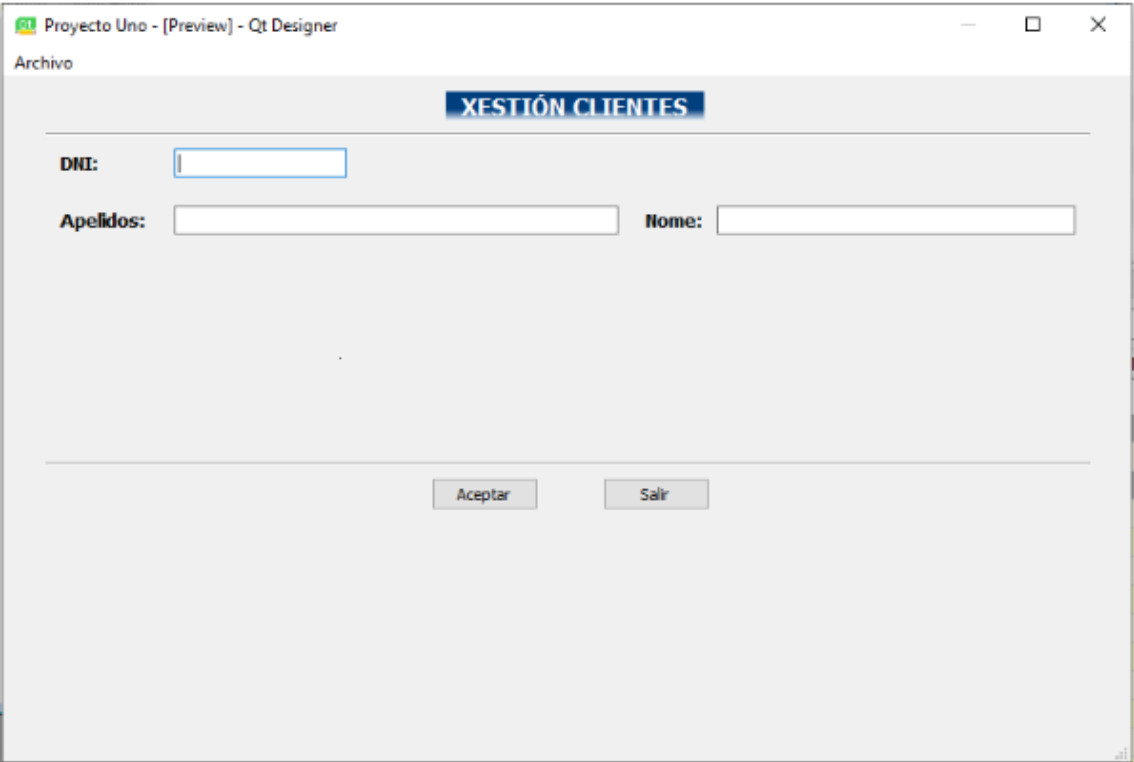
Observemos la siguiente imagen, donde se muestra la disposición típica de una ventana de una aplicación:



- **Menubar:** contiene los menús Archivo, Ver, Acerca de...
- **Toolbars:** contiene la barra de iconos para acceso rápido a las principales funciones.
- **Dock Widgets:** no siempre aparece y apenas lo utilizaremos. En este [enlace](#) se puede ver su objetivo.
- **CentralWidgets:** donde se sitúan los widgets de la aplicación. Para **activarlo, una vez tenga widgets en su interior**, pulsamos **Ctrl+1**. Sin embargo, es mejor dejarlo al principio sin activar para poder tener libertad en la disposición de los widgets.

EJERCICIO 1

Crear la siguiente ventana:

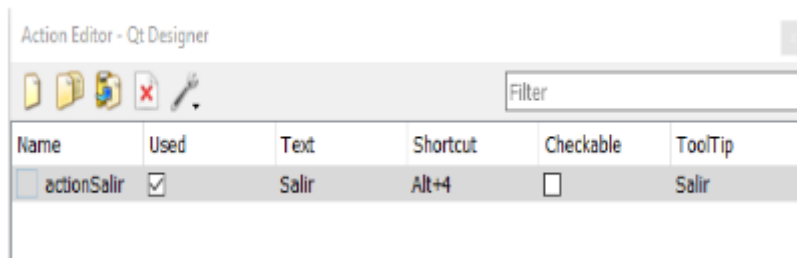


Object	Class
▼ MainWindow	QMainWindow
▼ centralwidget	QWidget
btnAceptar	QPushButton
btnSalir	QPushButton
cmbProv	QComboBox
editApel	QLineEdit
editDir	QLineEdit
editDni	QLineEdit
editNome	QLineEdit
▼ horlaySex	QHBoxLayout
rbtFem	QRadioButton
rbtMasc	QRadioButton
▼ horlayPago	QHBoxLayout
chkEfectivo	QCheckBox
chkTarjeta	QCheckBox
chkTransf	QCheckBox
lblApel	QLabel
lblDir	QLabel
lblDni	QLabel
lblMetpago	QLabel
lblNome	QLabel
lblProv	QLabel
lblSexo	QLabel
lblTitCli	QLabel
lblValido	QLabel
lineInf	Line
lineSup	Line
▼ menubar	QMenuBar
▼ menuArchivo	QMenu
actionSalir	QAction
statusBar	QStatusBar

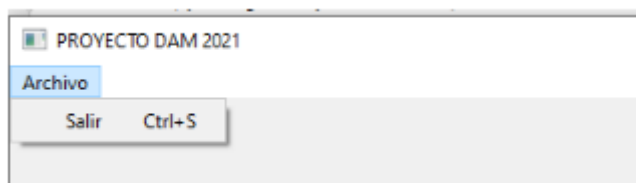
Los label o etiquetas son un elemento que nos permite incluir información.

El cuadro de texto que usaremos es el **LineEdit**. Es el *widget input* más básico y el que se utiliza en la mayoría de los casos.

Tocamos el menubar tal como se indica en la imagen. En dicho menú vamos a incluir la opción Salir con una combinación de teclas (por ejemplo, **Ctrl+S** o **Alt+4**). Para ello vamos a la opción **View -> Action Editor**, y hacemos las modificaciones de la imagen.



Quedando algo así:



Quedaría probar la salida de la aplicación desde la Barra de Menús y desde el símbolo **X** de la ventana añadiendo los códigos siguientes:

```
main.py x events.py x var.py x ventana.py x
1 from ventana import *
2 import sys, var, events
3
4
5 class Main(QtWidgets.QMainWindow):
6
7     def __init__(self):
8         super(Main, self).__init__()
9         var.ui = Ui_MainWindow()
10        var.ui.setupUi(self)
11        var.ui.actionSalir.triggered.connect(events.Eventos.Salir)
12
13
14 if __name__ == '__main__':
15     app = QtWidgets.QApplication([])
16     window = Main()
17     window.show()
18     sys.exit(app.exec())
19
```

```
main.py x events.py x var.py x ventana.py x
1 import sys
2
3
4 class Eventos:
5
6     def Salir():
7         try:
8             sys.exit()
9         except Exception as error:
10            print("Error %s: " % str(error))
11
```

EJERCICIO 2

La aplicación nos pide el DNI y el valor que nos pasen debe ser controlado por la aplicación para ver si es correcto.

Empecemos por el IU (ver las imágenes más abajo). Colocaremos una etiqueta a la derecha de la caja de texto del DNI, usando el tipo de letra **Forte**, tamaño 14 y sin texto alguno.

En el código de la aplicación en primer lugar, vamos a crear un nuevo fichero llamado **clients.py** que almacenará aquellas funciones que estén relacionadas con la gestión de los clientes. En el mismo añadimos el siguiente código.

```
class Clientes():
    def validarDNI():
        try:
            dni = var.ui.txtDNI.text()
            var.ui.txtDNI.setText(dni.upper())
            tabla = 'TRWAGMYFPDXBNJZSQVHLCKE' #letras dni
            dig_ext = 'XYZ' #digito
            reemp_dig_ext = {'X': '0', 'Y': '1', 'Z': '2'}
            numeros = '1234567890'
            dni = dni.upper() #conver la letra mayúsculas
            if len(dni) == 9:
                dig_control = dni[8]
                dni = dni[:8]
                if dni[0] in dig_ext:
                    dni = dni.replace(dni[0], reemp_dig_ext[dni[0]])
                if len(dni) == len([n for n in dni if n in numeros]) and tabla[int(dni) % 23] == dig_control:
                    var.ui.lblValidoDNI.setStyleSheet('QLabel {color: green;}')
                    var.ui.lblValidoDNI.setText('V')
                else:
                    var.ui.lblValidoDNI.setStyleSheet('QLabel {color: red;}')
                    var.ui.lblValidoDNI.setText('X')
            else:
                var.ui.lblValidoDNI.setStyleSheet('QLabel {color: red;}')
                var.ui.lblValidoDNI.setText('X')
        except Exception as error:
            print('Error en módulo validar DNI', error)
```

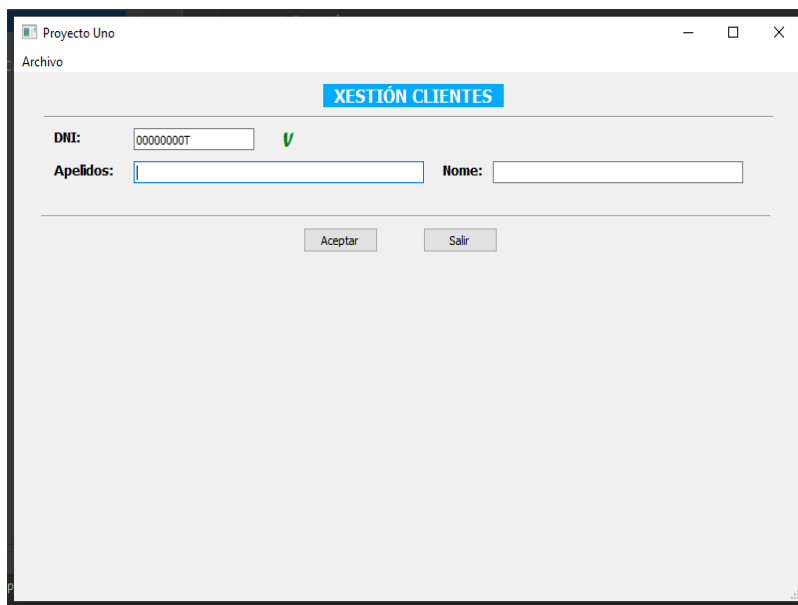
El paso siguiente es asociar el evento que queremos utilizar con esta función. La idea es que cuando el cursor salga de la caja de texto del dni, se ejecute el código expuesto. Para ello añadimos unas líneas al módulo **main.py** y **events.py** (o **clients.py**).

```
...
Eventos cada de texto
...
var.ui.txtDNI.editingFinished.connect(clients.Clientes.validarDNI)

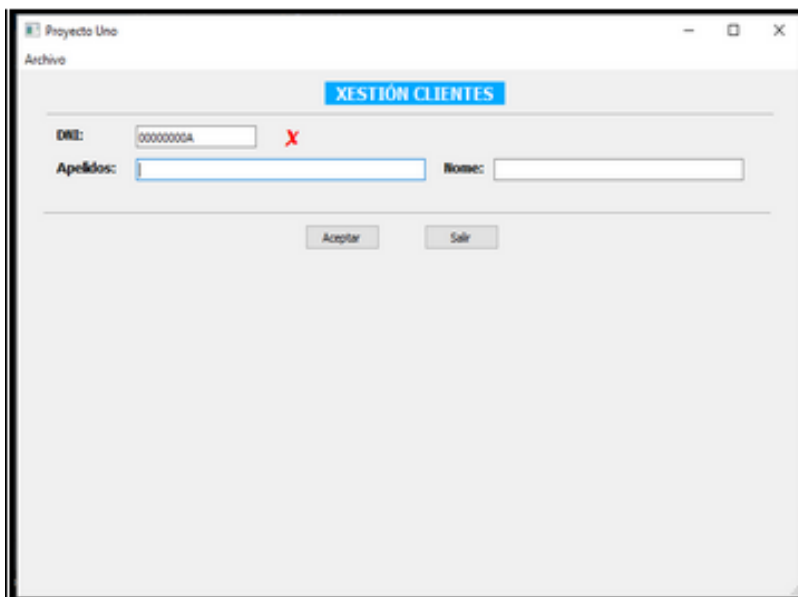
if __name__ == '__main__':
    app = QtWidgets.QApplication([])
    window = Main()
```

- **main.py:** conectamos el evento **editingFinished** editDni para que cuando el cursor salga de la caja de texto del dni se ejecute **validoDNI** en **events.py**
- **events.py** (o en un módulo **clients.py**): este evento primero coge el valor **dni** introducido en la caja de texto con la función **.text()** y se lo pasamos a la función **validarDNI** en **clients.py**. Esta función nos devuelve **True** o **False**. Si nos devuelve **True** escribiremos una V en color verde (estilos), si nos devuelve **False** escribiremos una X en color rojo. En ambos casos, además lo pone en mayúscula con la función **setText** y **upper()**.

Vemos el resultado en los dos casos:



The screenshot shows a window titled 'Proyecto Uno' with a menu bar containing 'Archivo'. Below the menu bar is a blue button labeled 'GESTIÓN CLIENTES'. The main area contains a form with two text input fields: 'DNI:' and 'Apellidos:'. The 'DNI:' field contains the text '00000000T' and has a green checkmark 'V' to its right. The 'Apellidos:' field is empty. Below the fields are two buttons: 'Aceptar' and 'Salir'.



The screenshot shows the same window as above, but the 'DNI:' field now contains the text '00000000A' and has a red X to its right. The 'Apellidos:' field remains empty. The 'Aceptar' and 'Salir' buttons are still present at the bottom.

EJERCICIO 2

La aplicación nos pide el DNI y el valor que nos pasen debe ser controlado por la aplicación para ver si es correcto.

Empecemos por el IU (ver las imágenes más abajo). Colocaremos una etiqueta a la derecha de la caja de texto del DNI, usando el tipo de letra **Forte**, tamaño 14 y sin texto alguno.

En el código de la aplicación en primer lugar, vamos a crear un nuevo fichero llamado **clients.py** que almacenará aquellas funciones que estén relacionadas con la gestión de los clientes. En el mismo añadimos el siguiente código.

```
class Clientes():
    def validarDNI():
        try:
            dni = var.ui.txtDNI.text()
            var.ui.txtDNI.setText(dni.upper())
            tabla = 'TRWAGMYFPDXBNJZSQVHLCKE' #letras dni
            dig_ext = 'XYZ' #digito
            reemp_dig_ext = {'X': '0', 'Y': '1', 'Z': '2'}
            numeros = '1234567890'
            dni = dni.upper() #conver la letra mayúsculas
            if len(dni) == 9:
                dig_control = dni[8]
                dni = dni[:8]
                if dni[0] in dig_ext:
                    dni = dni.replace(dni[0], reemp_dig_ext[dni[0]])
                if len(dni) == len([n for n in dni if n in numeros]) and tabla[int(dni) % 23] == dig_control:
                    var.ui.lblValidoDNI.setStyleSheet('QLabel {color: green;}')
                    var.ui.lblValidoDNI.setText('V')
                else:
                    var.ui.lblValidoDNI.setStyleSheet('QLabel {color: red;}')
                    var.ui.lblValidoDNI.setText('X')
            else:
                var.ui.lblValidoDNI.setStyleSheet('QLabel {color: red;}')
                var.ui.lblValidoDNI.setText('X')
        except Exception as error:
            print('Error en módulo validar DNI', error)
```

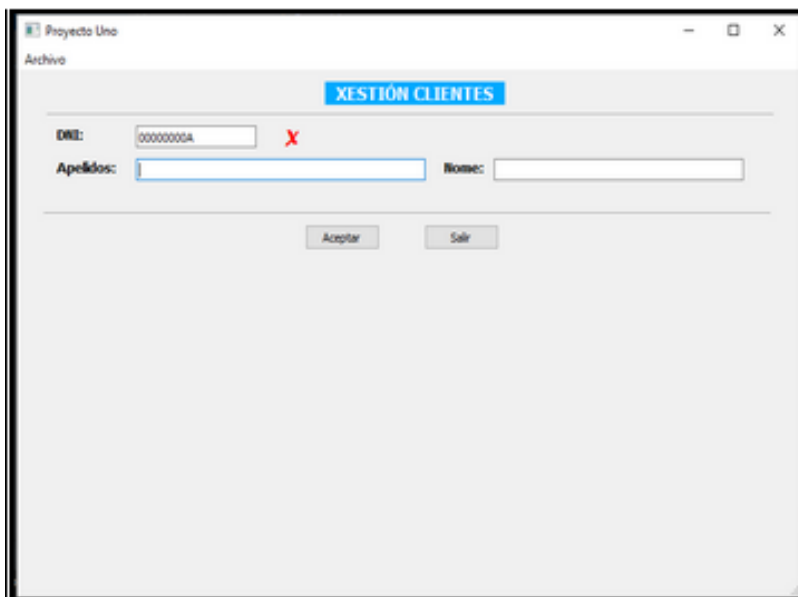
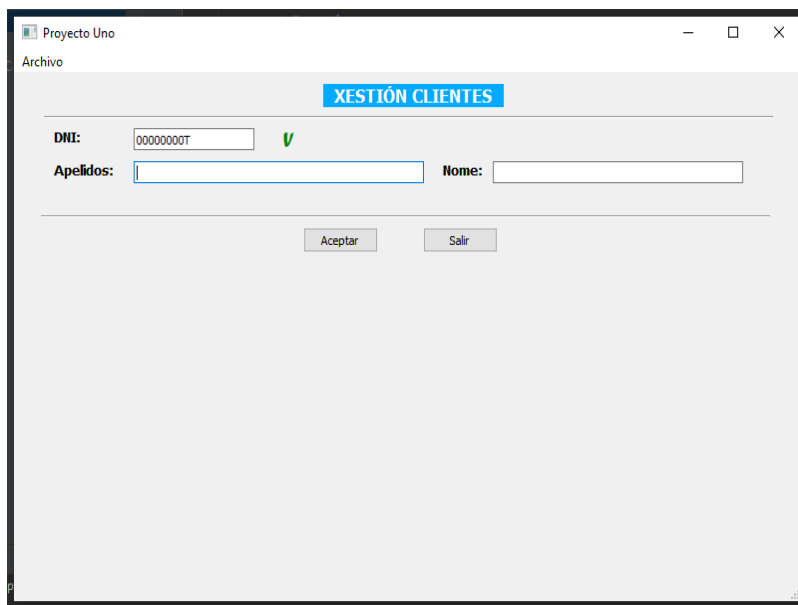
El paso siguiente es asociar el evento que queremos utilizar con esta función. La idea es que cuando el cursor salga de la caja de texto del dni, se ejecute el código expuesto. Para ello añadimos unas líneas al módulo **main.py** y **events.py** (o **clients.py**).

```
...
Eventos cada de texto
...
var.ui.txtDNI.editingFinished.connect(clients.Clientes.validarDNI)

if __name__ == '__main__':
    app = QtWidgets.QApplication([])
    window = Main()
```

- **main.py:** conectamos el evento **editingFinished** editDni para que cuando el cursor salga de la caja de texto del dni se ejecute **validoDNI** en **events.py**
- **events.py** (o en un módulo **clients.py**): este evento primero coge el valor **dni** introducido en la caja de texto con la función **.text()** y se lo pasamos a la función **validarDNI** en **clients.py**. Esta función nos devuelve **True** o **False**. Si nos devuelve **True** escribiremos una V en color verde (estilos), si nos devuelve **False** escribiremos una X en color rojo. En ambos casos, además lo pone en mayúscula con la función **setText** y **upper()**.

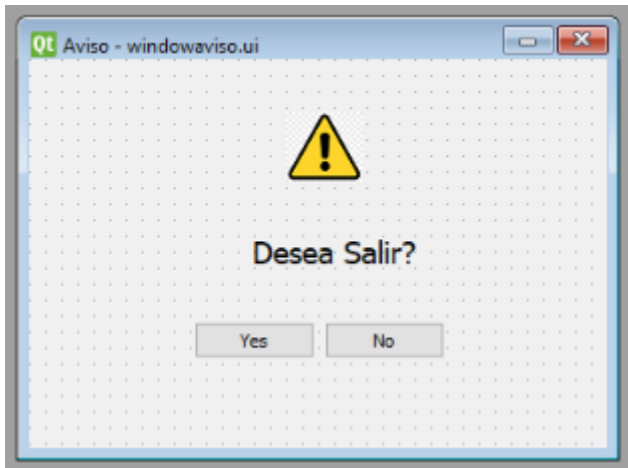
Vemos el resultado en los dos casos:



EJERCICIO 3

Vamos a perfeccionar la salida de la aplicación. En la mayoría de los programas existe un mensaje similar que te pregunta si deseas salir o no de esta. Diseñemos algo parecido utilizando QtDesigner con el Cuadro Dialog de dos botones en horizontal que podemos guardar con el nombre de **windowaviso.ui**.

La ventana la haremos **modal**, es decir, que no se ejecuta nada hasta que la cerremos o contestemos a lo que pregunta.



El icono del **warning** lo descargamos de Internet y ponemos en un **label** a través de la propiedad **pixmap**.

Como en el caso de la ventana principal en el fichero main, creamos la clase para instanciarla posteriormente.

```
# Press Double Shift to search everywhere for classes
class DialogSalir(QtWidgets.QDialog):
    def __init__(self):
        """
        Clase que instanci la ventana de aviso salir
        """
        super(DialogSalir, self).__init__()
        var.dlgSalir = Ui_windowaviso()
        var.dlgSalir.setupUi(self)
```

```

    ...
    Eventos de la barra de menús
    ...
    var.ui.actionSalir.triggered.conne

if __name__ == '__main__':
    app = QtWidgets.QApplication([])
    window = Main()
    var.dlgSalir = DialogSalir()
    window.show()
    sys.exit(app.exec())

```

Ahora modificamos el evento del botón salir de la siguiente manera:

```

import sys, var

class Eventos():
    def Salir(self):
        try:
            var.dlgSalir.show()
            if var.dlgSalir.exec():
                sys.exit()
            else:
                var.dlgSalir.hide()
        except Exception as error:
            print('Error en módulo salir ', error)

```

Si ejecutamos el programa, observamos que para salir antes nos muestra la ventana que nos pregunta si queremos salir. Si pulsamos *Si*, salimos, si pulsamos *No* la ventana diálogo de aviso se oculta.

Nota. - En algún caso se ha detectado que la conversión de QDialog de la ventana aviso *windowaviso.ui* en *windowaviso.py* no incluye dos líneas de código básicas para la ejecución correcta de la función. De ser así, se hace necesario su inclusión manual. Concretamente tenemos que editar *windowaviso.py* e incluir las líneas de código que se indican a continuación:

```
self.lblImageAviso.setText("")
self.lblImageAviso.setPixmap(QtGui.QPixmap("img/warning.png"))
self.lblImageAviso.setScaledContents(True)
self.lblImageAviso.setObjectName("lblImageAviso")
self.retranslateUi(windowaviso)
self.btnBoxAviso.accepted.connect(windowaviso.accept)
self.btnBoxAviso.rejected.connect(windowaviso.reject)
QtCore.QMetaObject.connectSlotsByName(windowaviso)

def retranslateUi(self, windowaviso):
    _translate = QtCore.QCoreApplication.translate
    windowaviso.setWindowTitle(_translate("windowaviso", "Aviso"))
    self.lblAviso.setText(_translate("windowaviso", "Desea Salir?"))
```

Otra opción es modificar la clase Dialog incluyendo lo siguiente:

```
class DialogAviso(QtWidgets.QDialog):
    def __init__(self):
        """
        Clase que instancia la ventana de avisos
        """
        super(DialogAviso, self).__init__()
        var.dlgaviso = Ui_windowaviso()
        var.dlgaviso.setupUi(self)
        var.dlgaviso.btnBoxAviso.accepted.connect(self.accept)
        var.dlgaviso.btnBoxAviso.rejected.connect(self.reject)
```