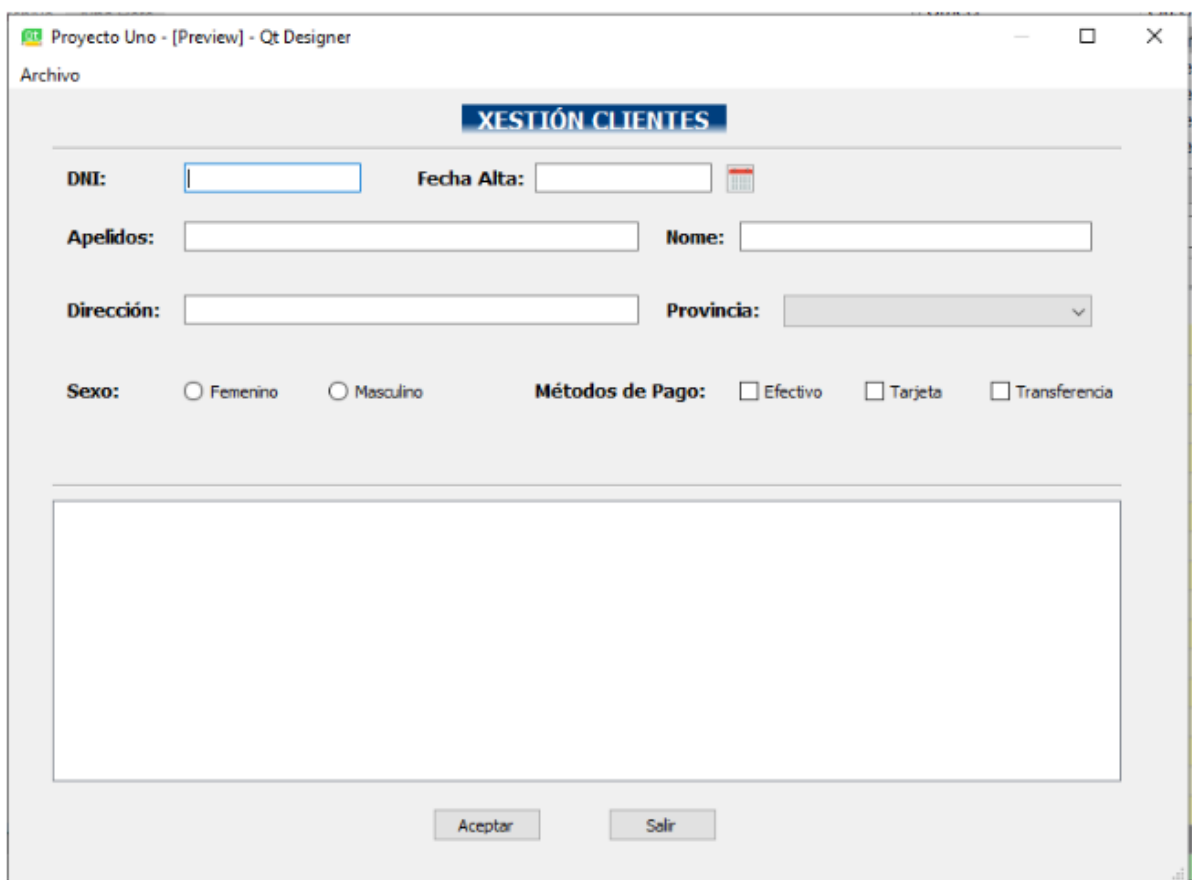


ACTIVIDAD – VISTAS – QtableView

El comportamiento predeterminado de las vistas estándar que se muestran debería ser suficiente para la mayoría de las aplicaciones. Proporcionan servicios de edición básicos y se pueden personalizar para satisfacer las necesidades de interfaces de usuario más especializadas y complejas.

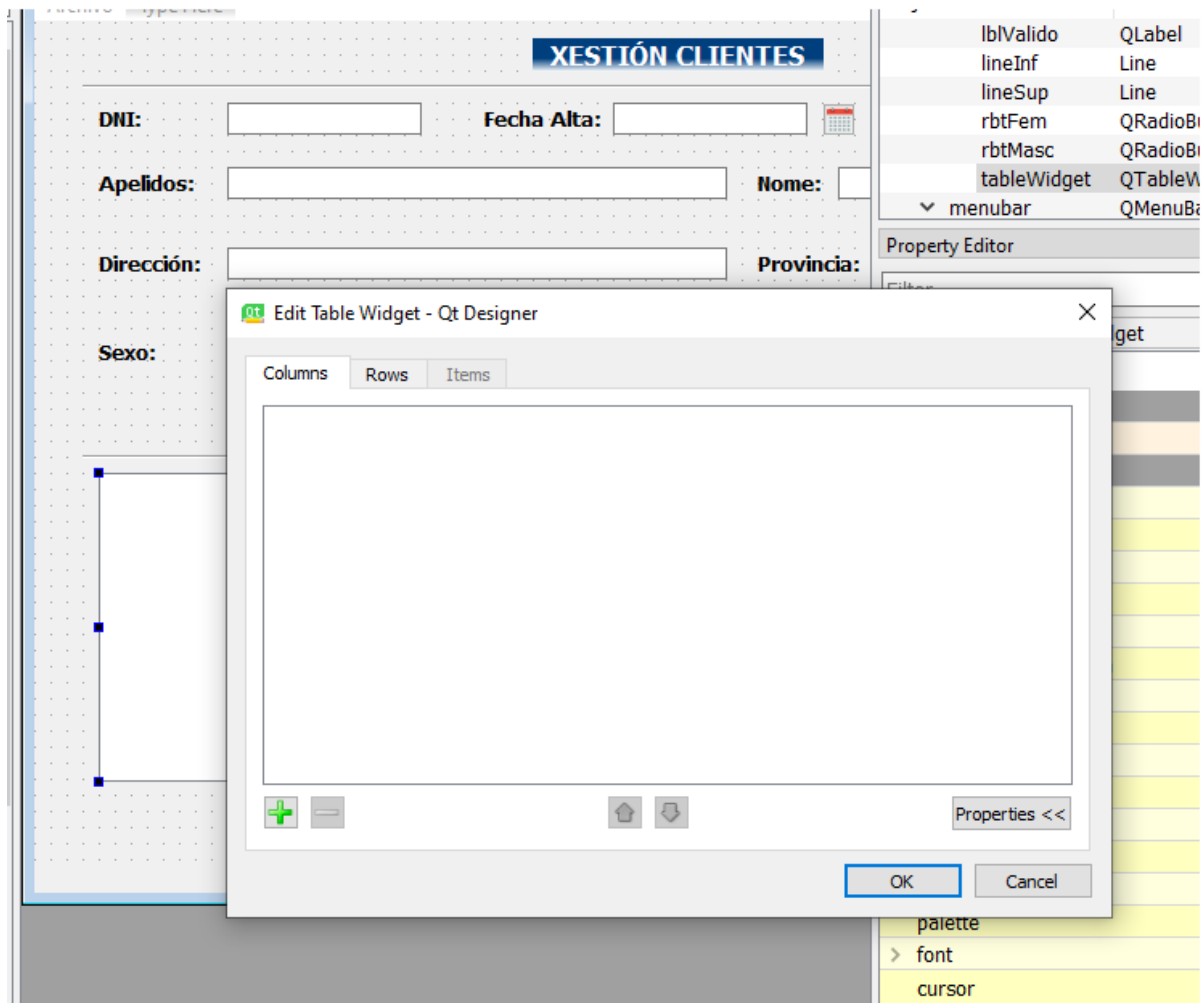
A continuación, vamos a revisar los códigos necesarios para que cuando demos de alta un cliente, usuario, producto, se muestre en un **QtableModel** los valores que hemos cargado.

En primer lugar, añadimos dicho widget a nuestro formulario ejemplo, quedando algo parecido a esto:

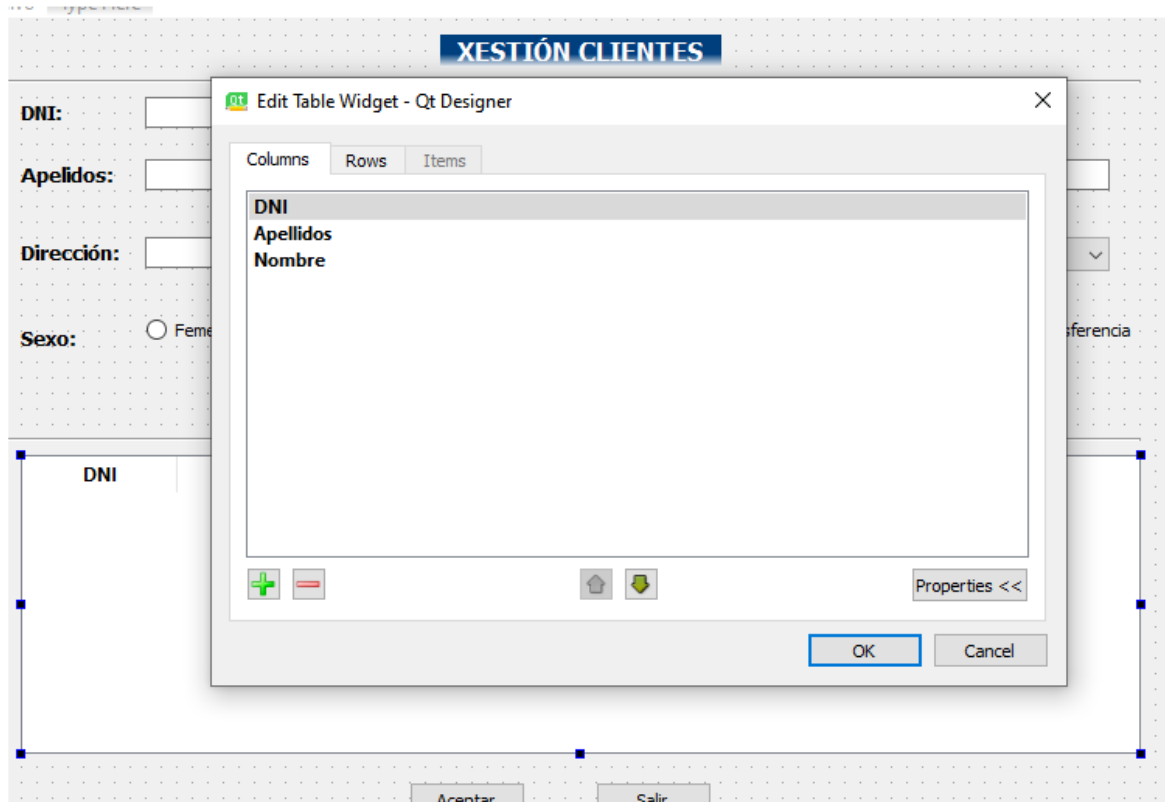


The image shows a Qt Designer window titled 'Proyecto Uno - [Preview] - Qt Designer'. The window contains a form titled 'GESTIÓN CLIENTES'. The form has several input fields and buttons. At the top, there is a menu bar with 'Archivo'. The form fields include: 'DNI:' with a text box, 'Fecha Alta:' with a date picker, 'Apellidos:' with a text box, 'Nombre:' with a text box, 'Dirección:' with a text box, 'Provincia:' with a dropdown menu, 'Sexo:' with radio buttons for 'Femenino' and 'Masculino', and 'Métodos de Pago:' with checkboxes for 'Efectivo', 'Tarjeta', and 'Transferencia'. Below these fields is a large empty rectangular area. At the bottom of the form are two buttons: 'Aceptar' and 'Salir'.

Sobre la misma botón derecho y *Edit Items*:



Y construimos el modelo:



Veamos ahora el código.

En primer lugar, preparamos la lista que contendrá todos los datos. No se va a exponer todo el código, solo aquel que hemos modificado. Tendremos dos funciones *showClients*, *selProov* y *cleanClients*, ambos dentro del fichero ***Cientes.py***. También vamos a realizar modificaciones en *selPago* y *selSex*. Empecemos por estas últimas.

```
def selSexo(self):
    try:
        global sex
        if var.ui.rbtFem.isChecked():
            sex = 'Mujer'
        if var.ui.rbtMasc.isChecked():
            sex = 'Hombre'

    except Exception as error:
        print('Error: %s ' % str(error))
```

En este módulo lo que hemos hecho es declarar una variable *sex* de carácter *global* para poder acceder a ella desde otros módulos de la misma clase. Lo veremos en el módulo *showClients*.

```
def selPago():
    try:
        if var.ui.chkEfectivo.isChecked():
            #print('Pagas con efectivo')
            var.pay.append('Efectivo')
        if var.ui.chkTarjeta.isChecked():
            #print('Pagas con tarjeta')
            var.pay.append('Tarjeta')
        if var.ui.chkTransf.isChecked():
            #print('Pagas con transferencia')
            var.pay.append('Transferencia')
    except Exception as error:
        print('Error: %s ' % str(error))
```

El manejo de los checkbox es algo más complicado. Cada vez que se checkea un valor se llama a este módulo con lo que se duplican o incluso triplican en este caso los valores elegidos.

Eso lo resolveremos en el módulo *showClients*. Por otro lado, hemos creado una variable *var.pay* para almacenar los métodos de pago elegidos por los clientes.

```
def selProv(prov):
    try:
        global vpro
        vpro = prov
    except Exception as error:
        print('Error: %s ' % str(error))
```

Las modificaciones realizadas aquí son mínimas pero necesarias para acceder a la provincia seleccionada.

Ya hemos modificado los módulos necesarios para cargar los datos de cada cliente. Ahora vamos con el módulo *showClients* que veremos en dos partes. La primera nos prepara la lista de los datos del cliente para que luego la segunda parte del código nos lo cargue en el *view*.

```
def showClients():
    try:
        #Preparamos el registro
        newcli = []
        client = [var.ui.editDni, var.ui.editApel, var.ui.editNome, var.ui.editData, var.ui.editDir]
        for i in client:
            newcli.append(i.text())
        newcli.append(vpro)
        #elimina duplicados
        var.pay = set(var.pay)
        for j in var.pay:
            newcli.append(j)
        newcli.append(sex)
        print(newcli)
    except Exception as error:
        print('Error: %s ' % str(error))
```

La lista *newcli* será la encargada de almacenar los valores. El primer **for** es **autoexplicativo**, carga todo el contenido de todos los *lineEdit*. Después añadimos el valor de la provincia que, al ser una variable global accedemos sin problema a ella desde cualquier función de la clase.

En cuanto al código para la selección de los pagos que puede elegir el cliente, indicar que cada vez que clicamos un **checkbox** este carga todos los marcados, el nuevo, más los

anteriores con lo que nos aparecerían duplicados debido al *for* del *main.py*. De ahí la función python: ***var.pay = set(var.pay)*** que elimina los duplicados, y finalmente, a través de otra variable global cargamos el sexo del cliente. Esta primera parte lo que hace es prepararnos la lista con todos los valores que ha introducido el usuario de la aplicación.



A continuación volvemos a QtDesigner para incluir una ***QTableWidget***, donde solo mostraremos el *DNI*, *Apellidos* y *Nombre*. En una tabla de gestión de este tipo no conviene cargar muchos datos ya que se convierte en ilegible. Basta con los datos básicos para localizar la persona o producto o servicio.

PyQt permite las ***View Model-Based***, o basadas en un modelo, pero su manejo es a nivel de código lo que la hace más complicado frente a las *View Item-Based*, basadas en el registro que van a mostrar y que será el que utilicemos. En entornos gráficos como PyGtk habituales en Gnome-Linux, las View se basan en el modelo pero este se construye en el diseñador gráfico cuando se diseña el prototipo, a diferencia de PyQt que obliga a realizar el diseño a nivel de código.

Añadimos entonces en la ventana principal mediante el QtDesigner un ***QtableWidget*** estableciendo el modelo tal como se muestra en las imágenes.

```
def showClients():
    try:
        #Preparamos el registro
        newcli = []
        clitab = [] # serán los datos que carguemos en la tabla
        client = [var.ui.editDni, var.ui.editApel, var.ui.editNome, var.ui.editData, var.ui.editDir]
        k = 0
        for i in client:
            newcli.append(i.text())
            #carguemos los valores para la tabla que solo tiene tres DNI, apellidos y nome
            if k < 3:
                clitab.append(i.text())
                k += 1
        newcli.append(vpro)
        #elimina duplicados
        var.pay = set(var.pay)
        for j in var.pay:
            newcli.append(j)
        newcli.append(sex)
        print(newcli)
        print(clitab)
        row = 0 #posición de la fila, problema: coloca al último como primero en cada click
        column = 0 #posición de la columna
        var.ui.cliTable.insertRow(row) #insertamos una fila nueva con cada click de botón
        for registro in clitab:
            #la celda tiene una posición fila, columna y cargamos en ella el dato
            cell = QtWidgets.QTableWidgetItem(registro) #carga en cell cada dato de la lista
            var.ui.cliTable.setItem(row, column, cell) #lo escribe
            column += 1
        except Exception as error:
            print('Error: %s ' % str(error))
```

GESTIÓN CLIENTES

DNI:  **Fecha Alta:** 

Apellidos: **Nome:**

Dirección: **Provincia:**

Sexo: ☒ Femenino ☐ Masculino **Métodos de Pago:** ☒ Efectivo ☒ Tarjeta ☐ Transferencia

	DNI	Apellidos	Nombre
1	00000000T	Castro Vilela	María
2	11111111H	García Sánchez	Luis

Aceptar

Salir