Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela Ciencias y sistemas

MANUAL TÉCNICO

Lenguajes formales y de programación

Rubén Alejandro Ralda Mejia
202111835

Guatemala 21 de agosto del 2022

Frontend

Librerías

```
from tkinter.messagebox import showerror, showinfo
from PyQt5 import QtWidgets, uic
from tkinter import *
from tkinter import filedialog
```

Cargar ventanas

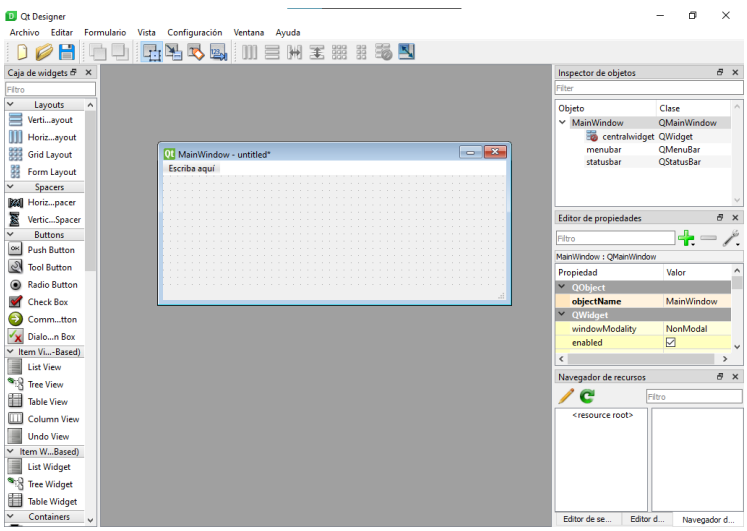
Para cargar las ventanas primero se crea un objeto de la clase QAplication en el constructor con un vector vacío y luego creamos la interfaz de usuario a partir de un archivo .ui con la función loadUi(str).

```
app = QtWidgets.QApplication([])
cursos=[]

# Cargar Ventanas
menuPrincipal = uic.loadUi("ui/menuPrincipal.ui")
cargarArchivo = uic.loadUi("ui/cargarArchivo.ui")
gestionCursos = uic.loadUi("ui/gestionarCursosMenu.ui")
conteoCreditos = uic.loadUi("ui/conteoCreditos.ui")

listaCursos=uic.loadUi("ui/listaCursos.ui")
buscarCurso=uic.loadUi("ui/buscarCurso.ui")
agregarCurso=uic.loadUi("ui/agregarCurso.ui")
editarCurso=uic.loadUi("ui/editarCurso.ui")
eliminarCurso=uic.loadUi("ui/eliminarCurso.ui")
```

Los archivos con extensión ui se crean con una herramienta que PyQt5 proporciona, llamada Designer. Esta aplicación deja utilizar drag and drop y así poder diseñar las ventanas de una manera sencilla, cuando guardamos el archivo se hace con esa extensión que ya solo la función nos genera el objeto para poder utilizarlo. Para eso antes se debe instalar el módulo PyQt5-Designer.



Para iniciar la aplicación se llama al método show() de la ventana principal, en este caso menuPrincipal y luego el método exec() del objeto app.

```
# Iniciar aplicacion
menuPrincipal.show()
app.exec()
```

Backend

Luego de diseñar la interfaz de usuario y crear los objetos de las ventanas lo que sigue es agregar la funcionalidad a los botones. Para agregar el evento se crea la función luego se llama el nombre del botón, la función clicked y después connect como parámetro se le envía la función que contiene la funcionalidad de lo que se desea hacer al pulsar el botón.

```
menuPrincipal.buttonSalir.clicked.connect(salir)
```

Menú principal

En esta ventana se cuenta con 4 botones, el primer botón manda a llamar una ventana:

```
def botonCargarArchivo():
    menuPrincipal.hide()
    cargarArchivo.show()
```

El segundo botón hace lo mismo que el primero, pero el tercero hace el conteo de cuando cargue la ventana, la información pueda ser visualizada de inmediato. Con un ciclo for de la línea 237 recorremos el vector cursos que contiene todos los datos, se compara en la siguiente línea si la posición 6 que dice si esta pendiente, aprobado o perdido con 0, indicando que ese curso ha sido aprobado para luego, autoincrementar la variable conteoaprobados que suma los créditos del curso. La otra comparación hace lo mismo y la tercera agrega otra condición de si en la posición 3 que indica si es obligatorio u opcional es igual a 1 hará su respectiva suma, para finalizar modifica el texto se los labels en la ventana con los conteos hechos.

```
231    def buttonAbrirConteo():
232        conteoCreditos.show()
233        menuPrincipal.hide()
234        conteoaprobados=0
235        conteocursando=0
236        conteopendientes=0
237        for curso in cursos:
238            if int(curso[6])==0:#aprobado
239                conteoaprobados+=int(curso[5])
240            if int(curso[6])==1:#cursando
241                conteocursando+=int(curso[5])
242            if int(curso[6])==-1:#pendientes
243                if int(curso[3])==1:#obligatorios
244                    conteopendientes+=int(curso[5])
245        conteoCreditos.aprobados.setText("Créditos Aprobados: "+str(conteoaprobado
246        conteoCreditos.cursados.setText("Créditos Cursando: "+str(conteocursando))
247        conteoCreditos.pendientes.setText("Créditos Pendientes: "+str(conteopendie
```

El ultimo botón solo invoca la función exit que termina con el programa.

Cargar archivo

En esta ventana se carga el archivo con extensión lfp donde está la información de los cursos. El botón buscar archivo invoca la función buscarArchivo(), esta abre una ventana para guardar la ruta con el modulo que nos ofrece Tkinter, la variable fileTypes configura que solo los archivos con esa extensión se pueda escoger, luego en la línea 40 guarda la ruta en el text de la ventana.

```
33  def buscarArchivo():
34      filetypes = (
35          ('text files', '*.lfp'),
36          ('All files', '*.*')
37      )
38      f = filedialog.askopenfilename(
39          initialdir="/", title="Escoge el Archivo", filetypes=filetypes)
40      cargarArchivo.txtRuta.setText(f)
```

El segundo botón “cargar” invoca la función cargarFile() esta primero valida si el txt esta vacío de ser verdadero muestra una ventana del Tkinter showerror(), si es falso abre el archivo con la ruta del txt y recorre cada línea del archivo guardándola en un vector que separa los datos en otro vector con el separador “,” y después invoca validar() esta función busca si hay repetidos.

```
42  def cargarFile():
43      if cargarArchivo.txtRuta.text()!="":
44          try:
45              file = open(cargarArchivo.txtRuta.text(),"r+",encoding="utf-8")
46              cursos.clear()
47              for linea in file:
48                  cursos.append(linea.replace("\n","").split(","))
49              file.close()
50          except:
51              showerror("Error", "Error al cargar el archivo")
52          showinfo("Información","El archivo fue cargado con exito")
53          validar()
54          cargarArchivo.hide()
55          menuPrincipal.show()
56      else:
57          showerror("Error", "Ingrese una ruta")
```

```
59    def validar():
60        for codigo in cursos:
61            repitencia,i=0,0
62            while i<len(cursos):
63                if cursos[i][0]==codigo[0]:
64                    repitencia+=1
65                    if repitencia==1:
66                        borrar=i
67                if repitencia>1:
68                    cursos.pop(borrar)
69                    borrar=i-1
70                    repitencia-=1
71                i+=1
```

Gestionar Cursos

Es menú que muestra las ventanas, el primero botón es diferente a los demás pues agrega los datos a la tabla antes de mostrar la ventana

```
def buttonLista():
    listaCursos.show()
    gestionCursos.hide()
    if cursos==[]:
        showerror("Error", "Cargue un archivo")
    else:
        m=len(cursos)
        listaCursos.tableWidget.setRowCount(m)
        tableRow=0
        for curso in cursos:
            listaCursos.tableWidget.setItem(tableRow,0,QtWidgets.QTableWidgetI
            listaCursos.tableWidget.setItem(tableRow,1,QtWidgets.QTableWidgetI
            listaCursos.tableWidget.setItem(tableRow,2,QtWidgets.QTableWidgetI
            listaCursos.tableWidget.setItem(tableRow,3,QtWidgets.QTableWidgetI
            listaCursos.tableWidget.setItem(tableRow,4,QtWidgets.QTableWidgetI
            listaCursos.tableWidget.setItem(tableRow,5,QtWidgets.QTableWidgetI
            listaCursos.tableWidget.setItem(tableRow,6,QtWidgets.QTableWidgetI
            tableRow+=1
```

El segundo botón muestra la ventana buscarCurso, esta invoca el método buscar que como el nombre indica, busca el curso por código recorriendo el vector cursos comparando si la posición 0 es igual al código del txt.

```
112  def buscar():
113      encontrado=False
114      if buscarCurso.txtCodigo.text()!="":
115          for curso in cursos:
116              if buscarCurso.txtCodigo.text()==curso[0]:
117                  encontrado=True
118                  obligatorio= 'obligatorio' if int(curso[3]) == 0 else 'opciona
119                  if int(curso[6])==0:
120                      estado="Aprobado"
121                  elif int(curso[6])==1:
122                      estado="Cursando"
123                  elif int(curso[6])==-1:
124                      estado="Pendiente"
125                  buscarCurso.curso.setText(f"Nombre: {curso[1]}, es {obligatori
126                  break
127          if encontrado==False:
128              showerror("Error", "El código no existe")
129      else:
130          showerror("Error", "Ingrese el código")
```

El tercer botón muestra la ventana agregar curso, este invoca la función buttonAgregar, primero hace una búsqueda si el código del curso ya existe y si no existe inserta al vector curso los datos extraídos de los textbox

```
151  def buttonAgregar():
152      if agregarCurso.txtCodigo.text()=="":
153          showerror("Error", "Ingrese un codigo")
154      else:
155          encontrado=False
156          for curso in cursos:
157              if agregarCurso.txtCodigo.text()==curso[0]:
158                  showerror("Error", "El codigo del curso ya existe")
159                  agregarCurso.txtCodigo.setText("")
160                  encontrado=True
161                  break
162          if encontrado==False:
163              if agregarCurso.txtNombre.text()!="" and agregarCurso.txtSemestre.
164                  cursos.append([agregarCurso.txtCodigo.text(),agregarCurso.txtN
165                  showinfo("Información","El curso se agregó exitosamente")
166              else:
167                  showerror("Error", "Algunos campos son obligatorios")
```

El cuarto botón llama la ventana editarCurso y este invoca dos métodos importantes, el primero es buttonBuscar2 que hace un búsqueda con el código del curso y extrae los datos del vector a los textbox correspondientes para editarlos, luego habilita el botón actualizar información que este hace lo mismo que el método del tercer botón, agregar el curso después de haber eliminado el anterior.

```
170  def buttonBuscar2():
171      if editarCurso.txtCodigo.text()=="":
172          showerror("Error", "Ingrese un codigo")
173      else:
174          i=0
175          existe=False
176          for curso in cursos:
177              if editarCurso.txtCodigo.text()==curso[0]:
178                  editarCurso.txtNombre.setText(curso[1])
179                  editarCurso.txtSemestre: Any  xt(curso[2])
180                  editarCurso.txtSemestre.setText(curso[4])
181                  editarCurso.txtOpcion.setText(curso[3])
182                  editarCurso.txtCreditos.setText(curso[5])
183                  editarCurso.txtEstado.setText(curso[6])
184                  editarCurso.buttonEditar.setEnabled(True)
185                  editarCurso.buttonRegresar.setEnabled(False)
186                  editarCurso.buttonBuscar.setEnabled(False)
187                  cursos.pop(i)
188                  existe=True
189                  break
190              i+=1
191          if existe==True:
192              showinfo("Información","Se ha cargado la informacion del curso")
193          else:
194              showinfo("Información","El curso no existe")
```

```
197  def buttonEditar():
198      if editarCurso.txtNombre.text()!="" and editarCurso.txtSemestre.text()!="
199          cursos.append([editarCurso.txtCodigo.text(),editarCurso.txtNombre.text
200          showinfo("Información","El curso se ha editado exitosamente")
201          editarCurso.buttonEditar.setEnabled(False)
202          editarCurso.buttonRegresar.setEnabled(True)
203          editarCurso.buttonBuscar.setEnabled(True)
204      else:
205          showerror("Error", "Algunos campos son obligatorios")
```