

Arellano Granados Angel Mariano
Seminario de solución de problemas de bases de
datos

2022B D05

lunes y miércoles de 9-11 beta 09



Entregable 2 U5. Conexión de la base de datos con lenguaje de programación

Videos Seleccionados:

<https://www.youtube.com/watch?v=XnWaZEmIO4k>

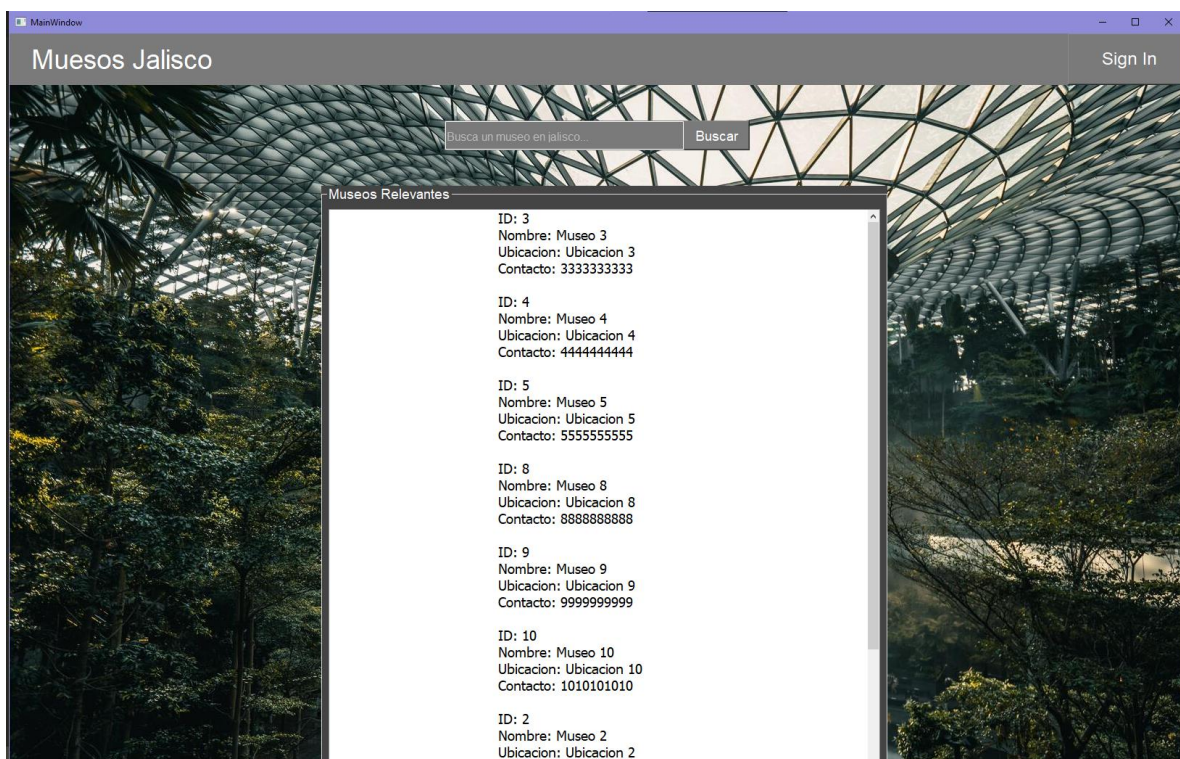
<https://youtu.be/M2NzvnfS-hI>

Conexión:

Para realizar la conexión de mis formularios con mi programa y la base de datos use el lenguaje Python con las librerías PySide2 para conectarlos con los formularios y Psycopg2 para conectarlo con PostgreSQL.

Formulario principal (home.ui):

Este es el formulario principal donde inicia el programa, aquí solo se pueden hacer tres cosas revisar los museos existentes con información superficial, buscar un museo específico y loggarse.



```

main.py > ...
1 from PySide2.QtWidgets import QApplication
2 from mainwindow import MainWindow
3 import sys
4
5 app = QApplication(sys.argv)
6 window = MainWindow()
7 window.show()
8
9 sys.exit(app.exec_())

```

Código para inicial la aplicación.

```

mainwindow.py > MainWindow > __init__
1 from PySide2.QtWidgets import QMainWindow, QGraphicsScene, QMessageBox
2 from PySide2.QtCore import Slot
3 from PySide2.QtGui import QFont
4 from ui_home import Ui_MainWindow
5 from loginwindow import LoginWindow
6 import posgres
7 import detalleswindow
8
9 class MainWindow(QMainWindow):
10     def __init__(self):
11         super(MainWindow,self).__init__()
12         self.ui = Ui_MainWindow()
13         self.ui.setupUi(self)
14         self.scene = QGraphicsScene()
15         self.ui.graphicsView.setScene(self.scene)
16         self.museos_relevantes()
17         self.ui.buscar_pushButton.clicked.connect(self.buscar_museo)
18         self.ui.sign_in_pushButton.clicked.connect(self.signIn)
19
20     @Slot()
21     def buscar_museo(self):
22         museo_name = self.ui.buscar_lineEdit.text()
23         museo_id = posgres.buscarMuseo(museo_name)
24         if museo_id == None:
25             QMessageBox.warning(self,"Error","No existe ese museo")
26         else:
27             global detalles
28             detalles = detalleswindow.DetallesWindow(museo_id)
29             detalles.show()
30             self.hide()
31
32     @Slot()
33     def signIn(self):
34         global login
35         login = LoginWindow()
36         login.show()
37         self.hide()
38
39     def museos_relevantes(self):
40         font = QFont("Times", 14)
41         cad = posgres.musRelevantes()
42         self.scene.addSimpleText(cad,font)

```

Código de la ventana

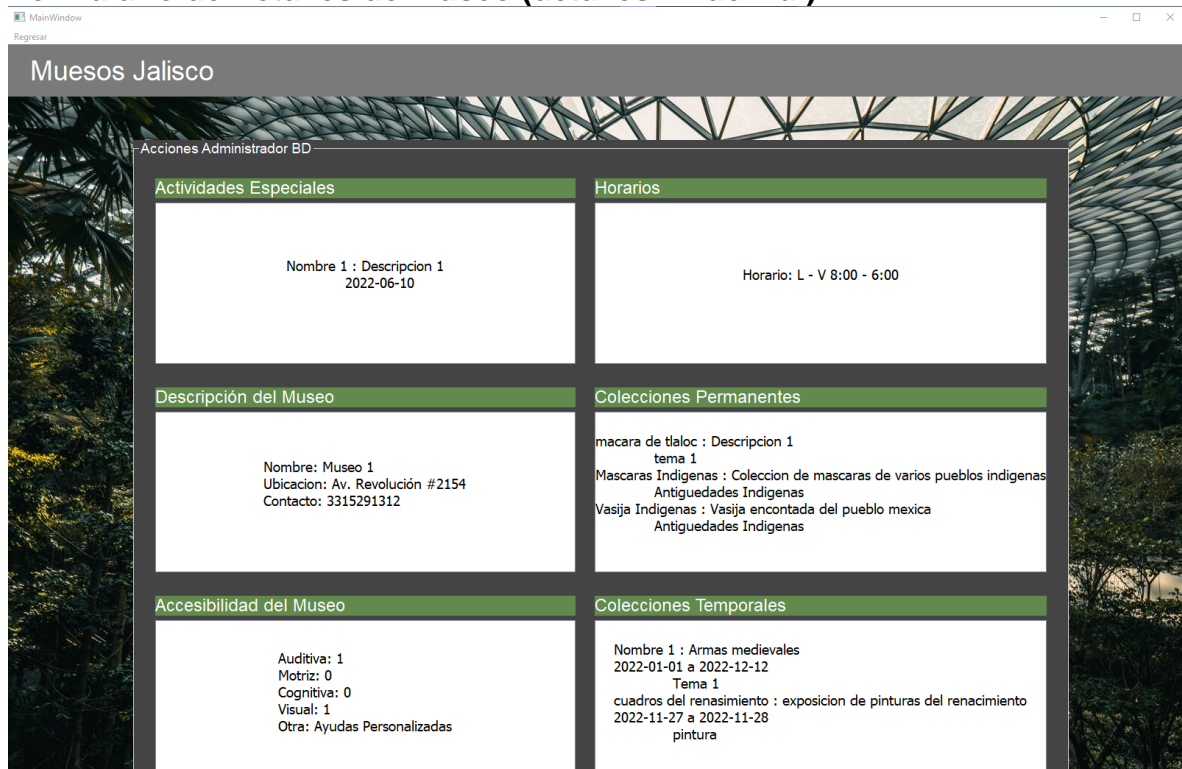
```

def musRelevantes()->str:
    conn = psycopg2.connect(host = hostname,dbname = database,user = username,password = pwd,port = post_id)
    cur = conn.cursor()
    cadena = ""
    cur.execute('SELECT * FROM museo')
    for mus in cur.fetchall():
        cadena += "ID: " + str(mus[0]) + "\n" + \
            "Nombre: " + mus[1] + "\n" + \
            "Ubicacion: " + mus[3] + "\n" + \
            "Contacto: " + mus [4] + "\n\n"
    cur.close()
    conn.close()
    return cadena

```

Código para mostrar museos

Formulario de Detalles de museo (detalleswindow.ui):



Esta ventana aparece una vez se busque y encuentre un museo en la ventana principal, esta solo muestra los datos de cada museo de manera detallada, no hay nada que se pueda hacer en esta ventana más que regresar al inicio.

```

detalleswindow.py > ...
5 import mainwindow
6 import posgres
7 class DetallesWindow(QMainWindow):
8     def __init__(self, id: int):
9         super(DetallesWindow, self).__init__()
10        self.ui = Ui_MainWindow()
11        self.ui.setupUi(self)
12        self.id_museo = id
13        font = QFont("Times", 14)
14        self.act_esp = QGraphicsScene()
15        self.ui.act_esp_graphicsView.setScene(self.act_esp)
16        str1 = posgres.actividades_especiales(self.id_museo)
17        self.act_esp.addSimpleText(str1, font)
18        self.desc_mus = QGraphicsScene()
19        self.ui.desc_mus_graphicsView.setScene(self.desc_mus)
20        str2 = posgres.descripcion(self.id_museo)
21        self.desc_mus.addSimpleText(str2, font)
22        self.acc_mus = QGraphicsScene()
23        self.ui.acc_mus_graphicsView.setScene(self.acc_mus)
24        str3 = posgres.accesibilidad(self.id_museo)
25        self.acc_mus.addSimpleText(str3, font)
26        self.horario = QGraphicsScene()
27        self.ui.horarios_graphicsView.setScene(self.horario)
28        str4 = posgres.horario(self.id_museo)
29        self.horario.addSimpleText(str4, font)
30        self.col_per = QGraphicsScene()
31        self.ui.col_per_graphicsView.setScene(self.col_per)
32        str5 = posgres.colPer(self.id_museo)
33        self.col_per.addSimpleText(str5, font)
34        self.col_tem = QGraphicsScene()
35        self.ui.col_tem_graphicsView.setScene(self.col_tem)
36        str6 = posgres.colTem(self.id_museo)
37        self.col_tem.addSimpleText(str6, font)
38        self.ui.actionVolver_al_inicio.triggered.connect(self.volver_inicio)
39    @Slot()
40    def volver_inicio(self):
41        global main
42        main = mainwindow.MainWindow()
43        main.show()
44        self.hide()

```

Código de la ventana

Funciones para obtener datos

```
def actividades_especiales(id_museo:int)->str:
    conn = psycopg2.connect(host = hostname,dbname = database,user = username,password = pwd,port = post_id)
    cur = conn.cursor()
    cadena = ""
    cur.execute('SELECT * FROM actividades')
    for actividad in cur.fetchall():
        if id_museo == actividad[0]:
            cadena += actividad[1] + " : " + actividad[2] + "\n\t" + str(actividad[3]) + "\n"
    cur.close()
    conn.close()
    return cadena

def accesibilidad(id:int)->str:
    conn = psycopg2.connect(host = hostname,dbname = database,user = username,password = pwd,port = post_id)
    cur = conn.cursor()
    cadena = ""
    cur.execute('SELECT * FROM accesibilidad')
    for acc in cur.fetchall():
        if id == acc[0]:
            if acc[1] == True:
                cadena = "No hay accesibilidad disponibles"
            else:
                cadena = "Auditiva: " + str(int(acc[2])) + "\n" + \
                    "Motriz: " + str(int(acc[3])) + "\n" + \
                    "Cognitiva: " + str(int(acc[4])) + "\n" + \
                    "Visual: " + str(int(acc[5])) + "\n" + \
                    "Otra: " + acc[7] + "\n"
    cur.close()
    conn.close()
    return cadena

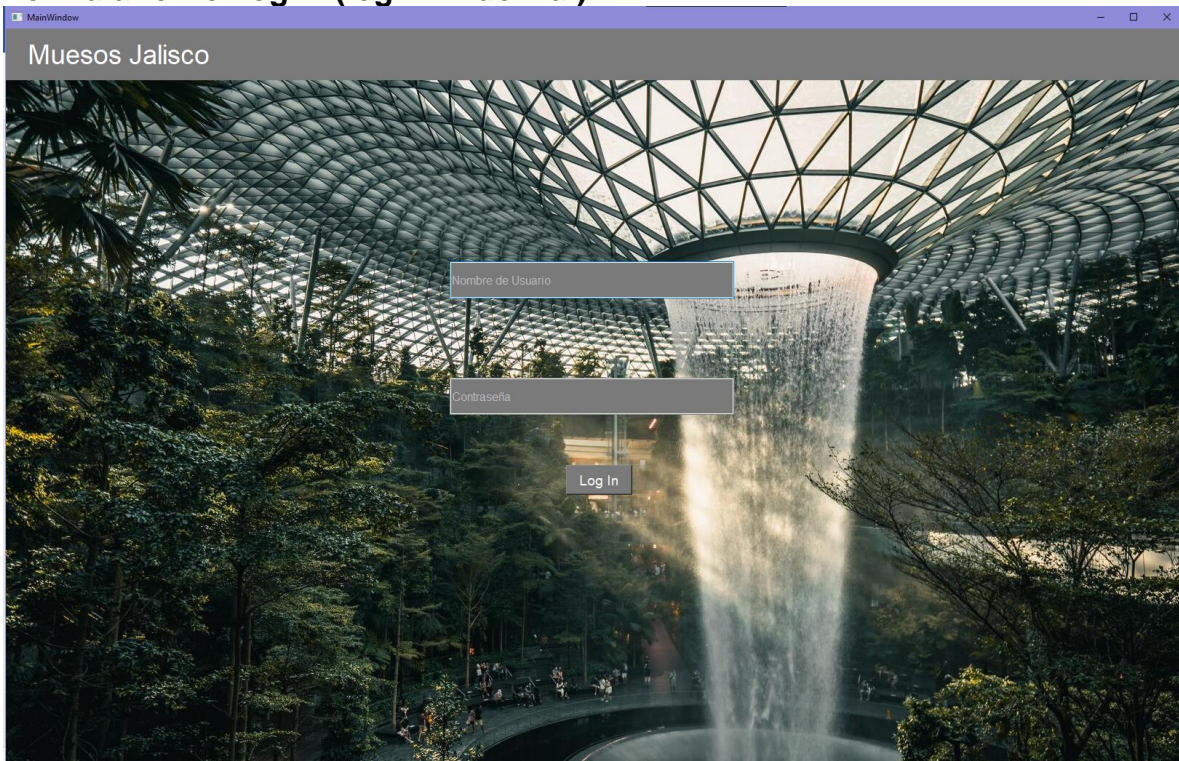
def descripcion(id:int)->str:
    conn = psycopg2.connect(host = hostname,dbname = database,user = username,password = pwd,port = post_id)
    cur = conn.cursor()
    cadena = ""
    cur.execute('SELECT * FROM museo')
    for mus in cur.fetchall():
        if id == mus[0]:
            cadena = "Nombre: " + mus[1] + "\n" + \
                "Ubicacion: " + mus[3] + "\n" + \
                "Contacto: " + mus[4] + "\n"
    cur.close()
    conn.close()
    return cadena

def horario(id:int)->str:
    conn = psycopg2.connect(host = hostname,dbname = database,user = username,password = pwd,port = post_id)
    cur = conn.cursor()
    cadena = ""
    cur.execute('SELECT * FROM museo')
    for mus in cur.fetchall():
        if id == mus[0]:
            cadena = "Horario: " + mus[2] + "\n"
    cur.close()
    conn.close()
    return cadena

def colPer(id:int)->str:
    conn = psycopg2.connect(host = hostname,dbname = database,user = username,password = pwd,port = post_id)
    cur = conn.cursor()
    cadena = ""
    cur.execute('SELECT * FROM colecciones_permanentes')
    for col in cur.fetchall():
        if id == col[0]:
            cadena += col[1] + " : " + col[2] + "\n\t" + col[3] + "\n"
    cur.close()
    conn.close()
    return cadena

def colTem(id:int)->str:
    conn = psycopg2.connect(host = hostname,dbname = database,user = username,password = pwd,port = post_id)
    cur = conn.cursor()
    cadena = ""
    cur.execute('SELECT * FROM colecciones_temporales')
    for col in cur.fetchall():
        if id == col[0]:
            cadena += col[1] + " : " + col[2] + "\n" + str(col[3]) + " a " + str(col[4]) + "\n\t" + col[5] + "\n"
    cur.close()
    conn.close()
    return cadena
```


Formulario De Log In (loginwindow.ui):



Clickeando el botón de logeo en la ventana principal no lleva a esta ventana donde podremos ingresar un usuario y contraseña, si estas coinciden con uno de los administradores de la base de datos o con un encargado de un museo lo llevara a la ventana del respectivo tipo de usuario.

```
loginwindow.py > ...
1  from PySide2.QtWidgets import QMainWindow, QMessageBox
2  from PySide2.QtCore import Slot
3  from ui_log_in import Ui_MainWindow
4  import posgres
5  from menuAdminWindow import MenuAdmin
6  from menuEncarWindow import MenuEncar
7
8  class LoginWindow(QMainWindow):
9      def __init__(self):
10         super(LoginWindow, self).__init__()
11
12         self.ui = Ui_MainWindow()
13         self.ui.setupUi(self)
14
15         self.ui.log_in_pushButton.clicked.connect(self.signin)
16
17     @Slot()
18     def signin(self):
19         user = self.ui.nombre_usuario_lineEdit.text()
20         pwd = self.ui.contraseña_lineEdit.text()
21         tipo = posgres.login(user, pwd)
22         if tipo[0] == 'ADMIN':
23             global menuAdmin
24             menuAdmin = MenuAdmin()
25             menuAdmin.show()
26             self.hide()
27         elif tipo[0] == 'ENCAR':
28             global menuEncar
29             menuEncar = MenuEncar(tipo[1])
30             menuEncar.show()
31             self.hide()
32         else:
33             QMessageBox.warning(self, "Error", "Usuario Incorrecto")
34
```

Código de la ventana

```
def login(un:str, pw:str)->list:
    conn = psycopg2.connect(host = hostname,dbname = database,user = username,password = pwd,port = port_id)
    tipo = []
    cur = conn.cursor()
    cur.execute('SELECT * FROM administrador_bd')
    for admin in cur.fetchall():
        if un == admin[2] and pw == admin[3]:
            tipo.append('ADMIN')
            cur.close()
            conn.close()
            return tipo
    cur.execute('SELECT * FROM encargados_museo')
    for encar in cur.fetchall():
        if un == encar[3] and pw == encar[4]:
            tipo.append('ENCAR')
            tipo.append(encar[0])
            cur.close()
            conn.close()
            return tipo
    cur.close()
    conn.close()
    return None
```

Código para logeo

Formulario Menú de Encargado de Muso (menuencarwindow.ui):

The screenshot shows a window titled 'MainWindow Encargado Museo' with a header 'Museos Jalisco'. The main content area has a background image of a modern glass and steel structure. Overlaid on this is a panel titled 'Acciones Encargados de Museos' with three sections:

- Añadir Colección Temporal:** Contains input fields for 'Nombre museo', 'Descripción Museo', and 'Tema Museo'. There are radio buttons for 'Permanente' and 'Temporal'. There are also input fields for 'Fecha Inicio' and 'Fecha Final'. An 'Añadir' button is at the bottom.
- Eliminar Colección Temporal:** Displays a list of collections with details like 'Nombre 1 : Armas medievales', '2022-01-01 a 2022-12-12', 'Tema 1', 'cuadros del renacimiento : exposicion de p', '2022-11-27 a 2022-11-28', and 'pintura'. There is an 'Eliminar' button.
- Modificar Información de Museo:** Contains input fields for 'Nombre Museo', 'Ubicación Museo', 'Horario Museo', and 'Contacto Museo'. A 'Modificar' button is at the bottom.

Llegamos a esta ventana si el logeo coincidió con una cuenta de un encargado, aquí se mostrará las 3 acciones disponibles que pude hacer el encargado añadir una colección permanente, eliminarlas y modificar la información de su museo. Cada una alterara las tablas dentro de la Base de Datos en Postgres.

```

menuEncarWindow.py > MenuEncar > _init_
import posgres

7 class MenuEncar(QMainWindow):
8     def __init__(self, id:int):
9         super(MenuEncar, self).__init__()
10        self.ui = Ui_MainWindow()
11        self.ui.setupUi(self)
12        self.id_museo = id
13        self.scene = QGraphicsScene()
14        self.ui.eleminar_graphicsView.setScene(self.scene)
15        self.act_col_tem()
16        self.ui.pushButton_3.clicked.connect(self.addCol)
17        self.ui.eli_col_pushButton.clicked.connect(self.eliminarCol)
18        self.ui.mod_pushButton.clicked.connect(self.modMuseo)
19        self.ui.actionSing_Out.triggered.connect(self.volver_inicio)
20    Slot()
21    def addCol(self):
22        nom = self.ui.add_nom_lineEdit.text()
23        desc = self.ui.add_des_lineEdit.text()
24        tem = self.ui.add_tem_lineEdit.text()
25
26        if self.ui.per_radioButton.isChecked() == True:
27            posgres.addColPer(self.id_museo, nom, desc, tem)
28
29        elif self.ui.tem_radioButton.isChecked() == True:
30            ini = self.ui.add_fec_ini_lineEdit.text()
31            fin = self.ui.add_fec_fin_lineEdit.text()
32            posgres.addColTem(self.id_museo, nom, desc, tem, ini, fin)
33            self.act_col_tem()
34    @Slot()
35    def eliminarCol(self):
36        nom = self.ui.eli_col_lineEdit.text()
37        posgres.eliminarColTem(nom)
38        self.act_col_tem()
39    @Slot()
40    def modMuseo(self):
41        nom = self.ui.mod_nom_lineEdit.text()
42        hor = self.ui.mod_hor_lineEdit.text()
43        ubi = self.ui.mod_ubi_lineEdit.text()
44        con = self.ui.mod_con_lineEdit.text()
45        posgres.modificar_datos_museo(self.id_museo, nom, hor, ubi, con)
46    @Slot()
47    def volver_inicio(self):
48        global main
49        main = mainwindow.MainWindow()
50        main.show()
51        self.hide()
52    def act_col_tem(self):
53        self.scene.clear()
54        font = QFont("Times", 14)

```

Código de la ventana

Funciones para conectarse con la Base de datos

```
def addColTem(id:int,nombre:str,desc:str,tema:str,ini:str,fin:str):
    conn = psycopg2.connect(host = hostname,dbname = database,user = username,password = pwd,port = post_id)
    cur = conn.cursor()
    script = 'INSERT INTO public.colecciones_temporales(id_museo,nombre,descripcion,fecha_inicio,fecha_final,tema)VALUES (%s, %s, %s, %s, %s, %s)'
    values = (id,nombre,desc,dt.date(int(ini[0:4]),int(ini[5:7]),int(ini[8:11])),dt.date(int(fin[0:4]),int(fin[5:7]),int(fin[8:11])),tema)
    cur.execute(script,values)
    conn.commit()
    cur.close()
    conn.close()

def eliminarColTem(nombre:str):
    conn = psycopg2.connect(host = hostname,dbname = database,user = username,password = pwd,port = post_id)
    cur = conn.cursor()
    script = 'DELETE FROM colecciones_temporales WHERE nombre = %s'
    values = (nombre,)
    cur.execute(script,values)
    conn.commit()
    cur.close()
    conn.close()

def modificar_datos_museo(id:int,nombre:str,horario:str,ubicacio:str,contacto:str):
    conn = psycopg2.connect(host = hostname,dbname = database,user = username,password = pwd,port = post_id)
    cur = conn.cursor()
    if nombre != '':
        script = 'UPDATE museo SET nombre = %s WHERE id_museo = %s'
        values = (nombre,id)
        cur.execute(script,values)
    if horario != '':
        script = 'UPDATE museo SET horario = %s WHERE id_museo = %s'
        values = (horario,id)
        cur.execute(script,values)
    if ubicacio != '':
        script = 'UPDATE museo SET ubicacion = %s WHERE id_museo = %s'
        values = (ubicacio,id)
        cur.execute(script,values)
    if contacto != '':
        script = 'UPDATE museo SET contacto = %s WHERE id_museo = %s'
        values = (contacto,id)
        cur.execute(script,values)
    conn.commit()
    cur.close()
    conn.close()
```

Formulario Menú Administrador Base de Datos (menuadminwindow.ui):

The screenshot shows a window titled "Muesos Jalisco" with a background image of a modern building with a glass and steel structure. Overlaid on this is a dark-themed menu titled "Acciones Administrador BD".

The menu contains three main sections:

- Añadir Museo:** A form with five input fields: "Identificador Museo", "Nombre Museo", "Ubicación Museo", "Horario Museo", and "Contacto Museo". Below these fields is an "Añadir" button.
- Eliminar Museo:** A section with a list of museums. The list shows two entries:
 - ID: 3, Nombre: Museo 3, Ubicacion: Ubicacion 3, Contacto: 3333333333
 - ID: 4, Nombre: Museo 4, Ubicacion: Ubicacion 4, Contacto: 4444444444To the right of the list is an "ID Museo" input field and an "Eliminar" button.
- Definir Encargado de Museo:** A form with five input fields: "Identificador Museo", "Nombre Encargado", "Nombre de usuario", "Contacto Encargado", and "Contraseña". Below these fields is an "Añadir" button.

Por último esta la ventana del menú para los administradores de base de datos a la cual se entra si el logeo coincide con la cuenta de una administrados, aquí se muestran las 3 acciones que puede hacer este usuario añadir museos, eliminarlos y definir el encargado de cada museo.

Código de la ventana

```
menuAdminWindow.py > MenuAdmin > volver_inicio
4 from PySide2.QtGui import QFont
5 import mainwindow
6 import posgres
7 class MenuAdmin(QMainWindow):
8     def __init__(self):
9         super(MenuAdmin, self).__init__()
10
11         self.ui = Ui_MainWindow()
12         self.ui.setupUi(self)
13         self.scene = QGraphicsScene()
14         self.ui.elemimar_graphicsView.setScene(self.scene)
15         self.act_mus()
16         self.ui.add_mus_pushButton.clicked.connect(self.addMus)
17         self.ui.eli_mus_pushButton.clicked.connect(self.eliminarMus)
18         self.ui.eli_mus_lineEdit.setText("ID Museo")
19         self.ui.add_encar_pushButton.clicked.connect(self.addEmcar)
20         self.ui.actionSing_Out.triggered.connect(self.volver_inicio)
21
22     @Slot()
23     def addMus(self):
24         id = int(self.ui.add_id_lineEdit.text())
25         nom = self.ui.add_nom_lineEdit.text()
26         hor = self.ui.add_hor_lineEdit.text()
27         ubi = self.ui.add_ubi_lineEdit.text()
28         con = self.ui.add_con_lineEdit.text()
29         posgres.addMuseo(id,nom,hor,ubi,con)
30
31     @Slot()
32     def eliminarMus(self):
33         id = int(self.ui.eli_mus_lineEdit.text())
34         posgres.eliMuseo(id)
35
36     @Slot()
37     def addEmcar(self):
38         id = int(self.ui.add_id_encar_lineEdit.text())
39         nom = self.ui.add_nom_encar_lineEdit.text()
40         con = self.ui.add_con_encar_lineEdit.text()
41         usu = self.ui.add_usu_encar_lineEdit_3.text()
42         pwd = self.ui.add_pwd_encar_lineEdit.text()
43         posgres.addEncar(id,nom,con,usu,pwd)
44
45     @Slot()
46     def volver_inicio(self):
47         global main
48         main = mainwindow.MainWindow()
49         main.show()
50         self.hide()
51
52     def act_mus(self):
53         self.scene.clear()
54         font = QFont("Times", 14)
55         cad = posgres.musRelevantes()
56         self.scene.addSimpleText(cad,font)
```

Funciones para conectarse con la Base de datos

```
def eliMuseo(id:int):
    conn = psycopg2.connect(host = hostname,dbname = database,user = username,password = pwd,port = post_id)
    cur = conn.cursor()
    value = (id,)
    script1 = 'DELETE FROM encargados_museo WHERE id_museo = %s'
    cur.execute(script1,value)
    script2 = 'DELETE FROM colecciones_temporales WHERE id_museo = %s'
    cur.execute(script2,value)
    script3 = 'DELETE FROM colecciones_permanentes WHERE id_museo = %s'
    cur.execute(script3,value)
    script4 = 'DELETE FROM accesibilidad WHERE id_museo = %s'
    cur.execute(script4,value)
    script5 = 'DELETE FROM actividades WHERE id_museo = %s'
    cur.execute(script5,value)
    script6 = 'DELETE FROM museo WHERE id_museo = %s'
    cur.execute(script6,value)
    conn.commit()
    cur.close()
    conn.close()

def addEncar(id:int,nombre:str,contacto:str,usuario:str,contr:str):
    conn = psycopg2.connect(host = hostname,dbname = database,user = username,password = pwd,port = post_id)
    cur = conn.cursor()
    script = 'INSERT INTO encargados_museo(id_museo, nombre, contacto, nombre_usuario, "contaseña")VALUES (%s, %s, %s, %s, %s)'
    values = (id,nombre,contacto,usuario,contr)
    cur.execute(script,values)
    conn.commit()
    cur.close()
    conn.close()

def addMuseo(id:int,nombre:str,horario:str,ubicacio:str,contacto:str):
    conn = psycopg2.connect(host = hostname,dbname = database,user = username,password = pwd,port = post_id)
    cur = conn.cursor()
    script = 'INSERT INTO museo(id_museo, nombre, horario, ubicacion, contacto)VALUES (%s, %s, %s, %s, %s)'
    values = (id,nombre,horario,ubicacio,contacto)
    cur.execute(script,values)
    conn.commit()
    cur.close()
    conn.close()
```

Conclusión:

Estoy muy contento con el resultado todo funciona perfectamente y se que se puede mejorar muchísimo el código y la interfaz para que sea más estética, pero por cuestión de tiempo lo tuve que dejar en lo funcional en vez de lo optimo.