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

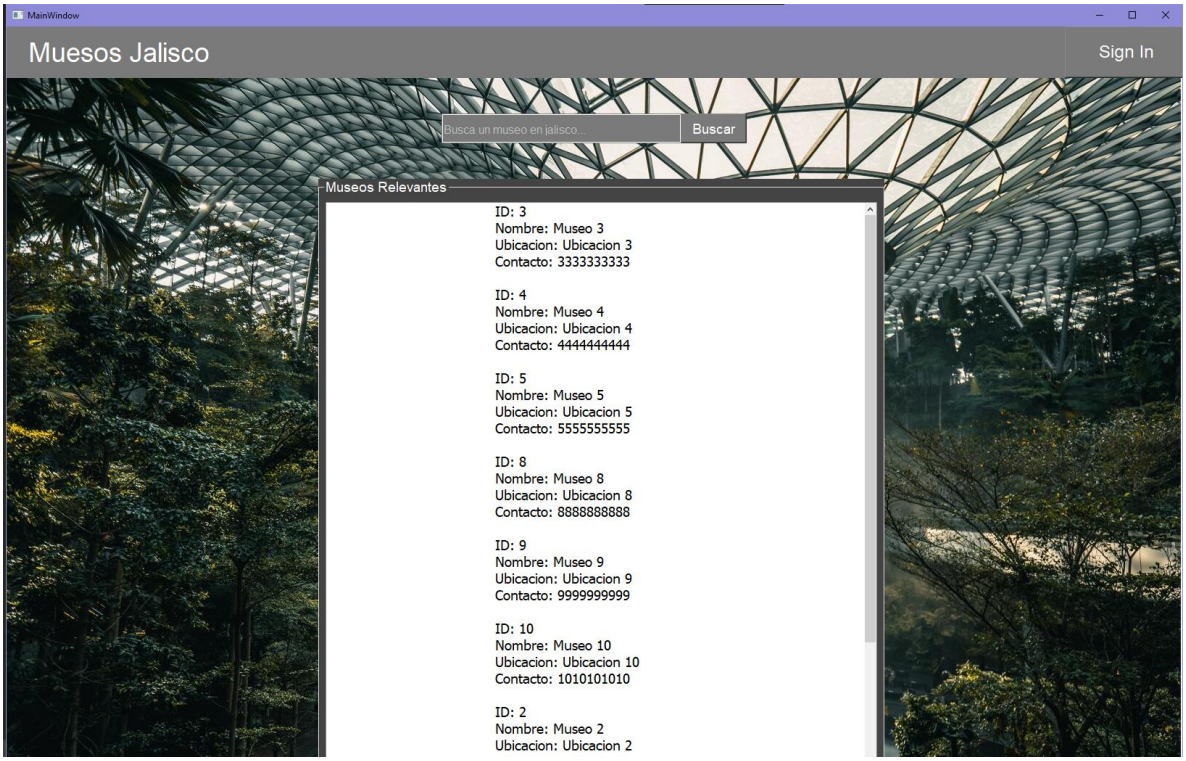
https://youtu.be/M2NzvnfS-hI

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 especifico y loggearse.



main.py > ...

```python
from PySide2.QtWidgets import QApplication
from mainwindow import MainWindow
import sys

app = QApplication(sys.argv)
window = MainWindow()
window.show()

sys.exit(app.exec_())
```

Código para inicial la aplicación.

mainwindow.py > MainWindow > __init__

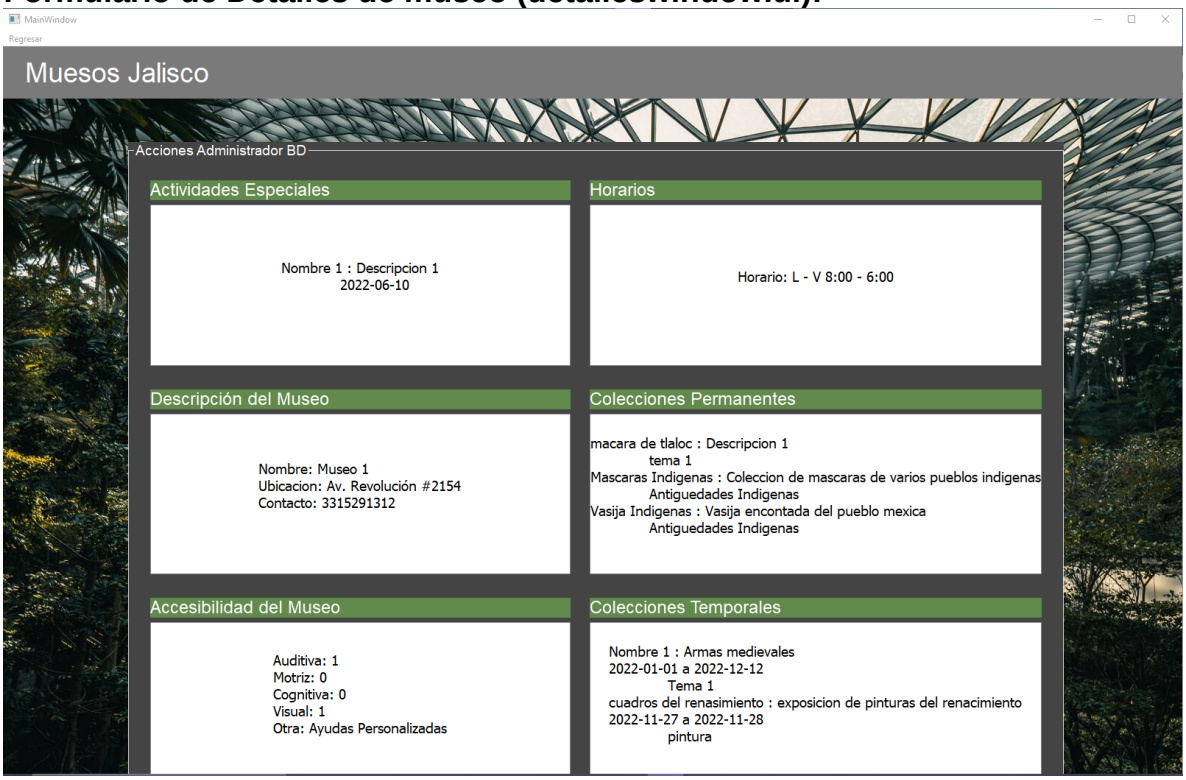
```python
from PySide2.QtWidgets import QMainWindow, QGraphicsScene, QMessageBox
from PySide2.QtCore import Slot
from PySide2.QtGui import QFont
from ui_home import Ui_MainWindow
from loginwindow import LoginWindow
import posgres
import detalleswindow

class MainWindow(QMainWindow):
    def __init__(self):
        super(MainWindow,self).__init__()
        self.ui = Ui_MainWindow()
        self.ui.setupUi(self)
        self.scene = QGraphicsScene()
        self.ui.graphicsView.setScene(self.scene)
        self.museos_relevantes()
        self.ui.buscar_pushButton.clicked.connect(self.buscar_museo)
        self.ui.sign_in_pushButton.clicked.connect(self.signIn)
    @Slot( )
    def buscar_museo(self):
        museo_name = self.ui.buscar_lineEdit.text()
        museo_id = posgres.buscarMuseo(museo_name)
        if museo_id == None:
            QMessageBox.warning(self,"Error","No existe ese museo")
        else:
            global detalles
            detalles = detalleswindow.DetallesWindow(museo_id)
            detalles.show()
            self.hide()
    @Slot( )
    def signIn(self):
        global login
        login = LoginWindow()
        login.show()
        self.hide()
    def museos_relevantes(self):
        font = QFont("Times", 14)
        cad = posgres.musRelevantes()
        self.scene.addSimpleText(cad,font)
```

Código de la ventana

```python
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 ves se busque y encuentre un mueso 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 > ...

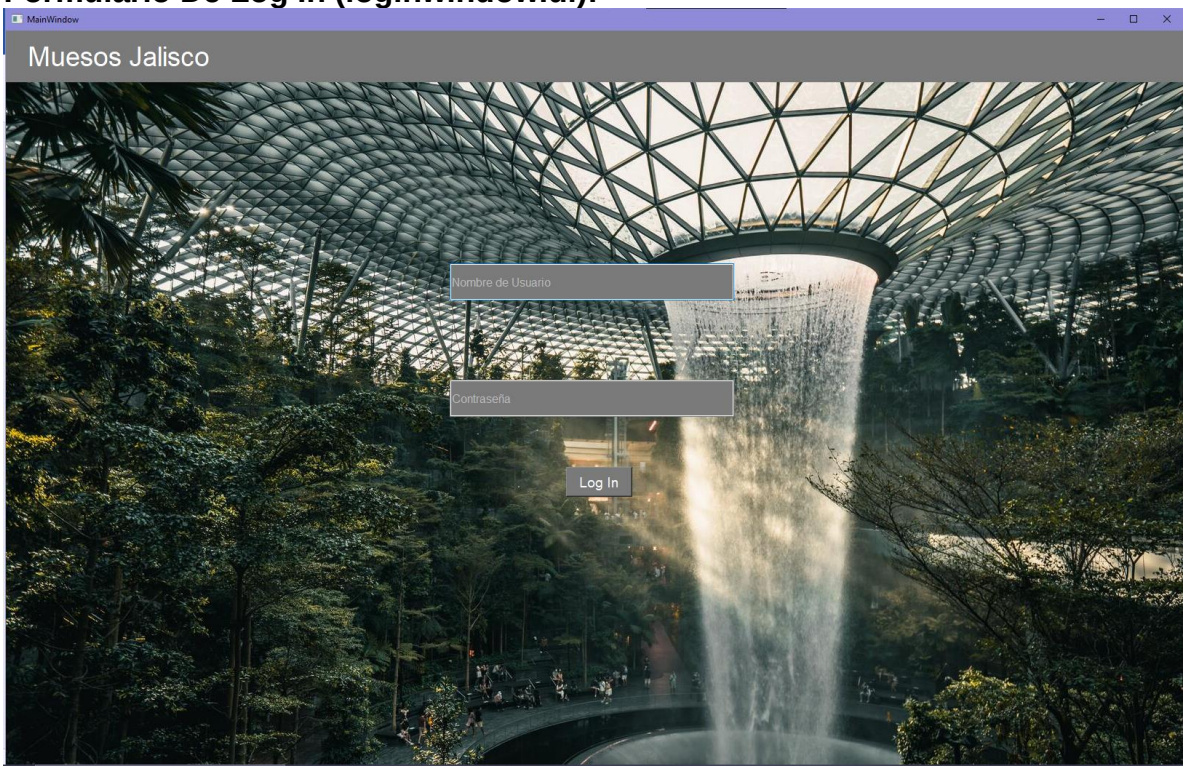
```python
import mainwindow
import posgres
class DetallesWindow(QMainWindow):
    def __init__(self,id:int):
        super(DetallesWindow,self).__init__()
        self.ui = Ui_MainWindow()
        self.ui.setupUi(self)
        self.id_museo = id
        font = QFont("Times", 14)
        self.act_esp = QGraphicsScene()
        self.ui.act_esp_graphicsView.setScene(self.act_esp)
        str1 = posgres.actividades_especiales(self.id_museo)
        self.act_esp.addSimpleText(str1,font)
        self.desc_mus = QGraphicsScene()
        self.ui.desc_mus_graphicsView.setScene(self.desc_mus)
        str2 = posgres.descripcion(self.id_museo)
        self.desc_mus.addSimpleText(str2,font)
        self.acc_mus = QGraphicsScene()
        self.ui.acc_mus_graphicsView.setScene(self.acc_mus)
        str3 = posgres.accesivilidad(self.id_museo)
        self.acc_mus.addSimpleText(str3,font)
        self.horario = QGraphicsScene()
        self.ui.horarios_graphicsView.setScene(self.horario)
        str4 = posgres.horario(self.id_museo)
        self.horario.addSimpleText(str4,font)
        self.col_per = QGraphicsScene()
        self.ui.col_per_graphicsView.setScene(self.col_per)
        str5 = posgres.colPer(self.id_museo)
        self.col_per.addSimpleText(str5,font)
        self.col_tem = QGraphicsScene()
        self.ui.col_tem_graphicsView.setScene(self.col_tem)
        str6 = posgres.colTem(self.id_museo)
        self.col_tem.addSimpleText(str6,font)
        self.ui.actionVolver_al_inicio.triggered.connect(self.volver_inicio)
    @Slot()
    def volver_inicio(self):
        global main
        main = mainwindow.MainWindow()
        main.show()
        self.hide()
```

Código de la ventana

Funciones para obtener datos

```python
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 accesivilidad(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 accesivilidad 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 > ...

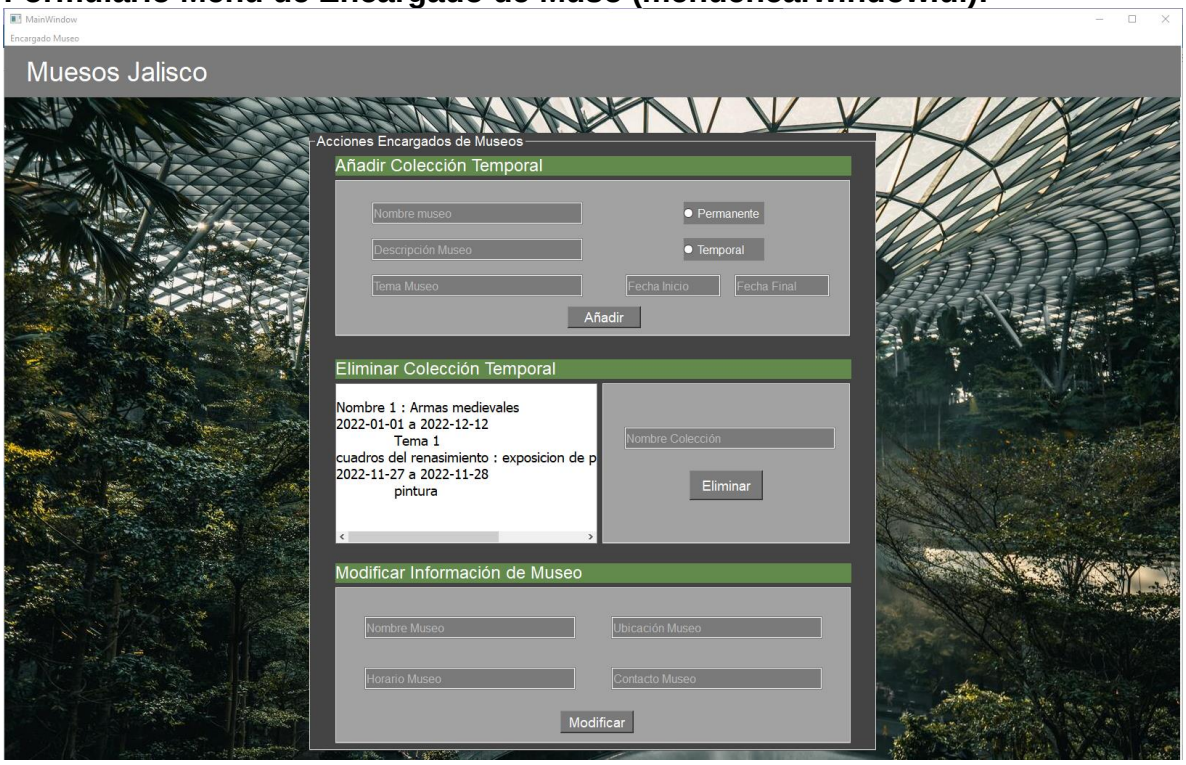
```python
from PySide2.QtWidgets import QMainWindow, QMessageBox
from PySide2.QtCore import Slot
from ui_log_in import Ui_MainWindow
import posgres
from menuAdminWindow import MenuAdmin
from menuEncarWindow import MenuEncar

class LoginWindow(QMainWindow):
    def __init__(self):
        super(LoginWindow,self).__init__()

        self.ui = Ui_MainWindow()
        self.ui.setupUi(self)

        self.ui.log_in_pushButton.clicked.connect(self.signin)

    @Slot( )
    def signin(self):
        user = self.ui.nombre_usuario_lineEdit.text()
        pwd = self.ui.contasea_lineEdid.text()
        tipo = posgres.login(user,pwd)
        if tipo[0] == 'ADMIN':
            global menuAdmin
            menuAdmin = MenuAdmin()
            menuAdmin.show()
            self.hide()
        elif tipo[0] == 'ENCAR':
            global menuEncar
            menuEncar = MenuEncar(tipo[1])
            menuEncar.show()
            self.hide()
        else:
            QMessageBox.warning(self,"Error","Usuario Incorrecto")

```

Código de la ventana

```python
def login(un:str, pw:str)->list:
    conn = psycopg2.connect(host = hostname,dbname = database,user = username,password = pwd,port = post_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):



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.

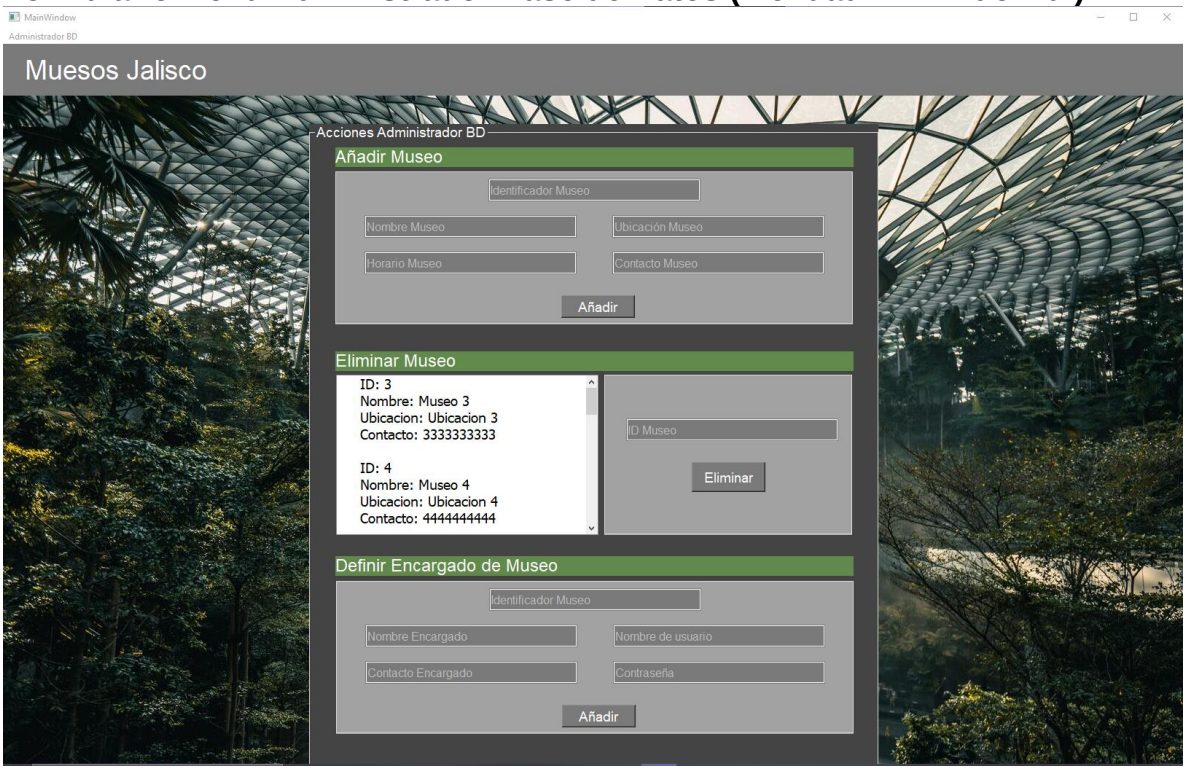

```python
import posgres
class MenuEncar(QMainWindow):
    def __init__(self,id:int):
        super(MenuEncar,self).__init__()
        self.ui = Ui_MainWindow()
        self.ui.setupUi(self)
        self.id_museo = id
        self.scene = QGraphicsScene()
        self.ui.eleminar_graphicsView.setScene(self.scene)
        self.act_col_tem()
        self.ui.pushButton_3.clicked.connect(self.addCol)
        self.ui.eli_col_pushButton.clicked.connect(self.eliminarCol)
        self.ui.mod_pushButton.clicked.connect(self.modMuseo)
        self.ui.actionSing_Out.triggered.connect(self.volver_inicio)
    Slot()
    def addCol(self):
        nom = self.ui.add_nom_lineEdit.text()
        desc = self.ui.add_des_lineEdit.text()
        tem = self.ui.add_tem_lineEdit.text()

        if self.ui.per_radioButton.isChecked() == True:
            posgres.addColPer(self.id_museo,nom,desc,tem)

        elif self.ui.tem_radioButton.isChecked() == True:
            ini = self.ui.add_fec_ini_lineEdit.text()
            fin = self.ui.add_fec_fin_lineEdit.text()
            posgres.addColTem(self.id_museo,nom,desc,tem,ini,fin)
            self.act_col_tem()
    @Slot()
    def eliminarCol(self):
        nom = self.ui.eli_col_lineEdit.text()
        posgres.eliminarColTem(nom)
        self.act_col_tem()
    @Slot()
    def modMuseo(self):
        nom = self.ui.mod_nom_lineEdit.text()
        hor = self.ui.mod_hor_lineEdit.text()
        ubi = self.ui.mod_ubi_lineEdit.text()
        con = self.ui.mod_con_lineEdit.text()
        posgres.modificar_datos_museo(self.id_museo,nom,hor,ubi,con)
    @Slot()
    def volver_inicio(self):
        global main
        main = mainwindow.MainWindow()
        main.show()
        self.hide()
    def act_col_tem(self):
        self.scene.clear()
        font = QFont("Times", 14)
```

Código de la ventana

Funciones para conectarse con la Base de datos

```python
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):



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

```python
from PySide2.QtGui import QFont
import mainwindow
import posgres
class MenuAdmin(QMainWindow):
    def __init__(self):
        super(MenuAdmin,self).__init__()

        self.ui = Ui_MainWindow()
        self.ui.setupUi(self)
        self.scene = QGraphicsScene()
        self.ui.eleminar_graphicsView.setScene(self.scene)
        self.act_mus()
        self.ui.add_mus_pushButton.clicked.connect(self.addMus)
        self.ui.eli_mus_pushButton.clicked.connect(self.eliminarMus)
        self.ui.eli_mus_lineEdit.setPlaceholderText("ID Museo")
        self.ui.add_encar_pushButton.clicked.connect(self.addEmcar)
        self.ui.actionSing_Out.triggered.connect(self.volver_inicio)
    @Slot()
    def addMus(self):
        id =  int(self.ui.add_id_lineEdit.text())
        nom = self.ui.add_nom_lineEdit.text()
        hor = self.ui.add_hor_lineEdit.text()
        ubi = self.ui.add_ubi_lineEdit.text()
        con = self.ui.add_con_lineEdit.text()
        posgres.addMuseo(id,nom,hor,ubi,con)
    @Slot()
    def eliminarMus(self):
        id = int(self.ui.eli_mus_lineEdit.text())
        posgres.eliMuseo(id)
    @Slot()
    def addEmcar(self):
        id =  int(self.ui.add_id_encar_lineEdit.text())
        nom = self.ui.add_nom_encar_lineEdit.text()
        con = self.ui.add_con_encar_lineEdit.text()
        usu = self.ui.add_usu_encar_lineEdit_3.text()
        pwd = self.ui.add_pwd_encar_lineEdit.text()
        posgres.addEncar(id,nom,con,usu,pwd)
    @Slot()
    def volver_inicio(self):
        global main
        main = mainwindow.MainWindow()
        main.show()
        self.hide()
    def act_mus(self):
        self.scene.clear()
        font = QFont("Times", 14)
        cad = posgres.musRelevantes()
        self.scene.addSimpleText(cad,font)
```

Funciones para conectarse con la Base de datos

```python
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 funciono 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 ves de lo optimo.