



Universidad de Guadalajara
Centro Universitario de Ciencias Exactas e Ingenierías

Estatus

Alumno: Bryan De Anda Reyes

Código de alumno: 216195537

Materia: Computación Tolerante a Fallas (D06)

Horario: lunes y miércoles (11:00 – 13:00)

Carrera: INCO

Fecha de entrega: 28 de febrero de 2022

Instrucciones

El objetivo de esta practica es hacer un servicio para que levante nuestro programa principal y haciendo también el manejo de hilos para recuperar el estado en el que se cerró el programa.

Desarrollo

Lo primero que hice fue crear la interfaz gráfica de la aplicación principal que se debe levantar si su ejecución se interrumpe, también implemente las diferentes funciones para la clase de las películas, estas funciones son, agregar una película, buscar una película y mostrar película; Al ser una evolución del programa pasado considero no es necesario explicar la implementación de estas funciones ya que estas son las mismas del anterior, solo se adaptaron para la interfaz gráfica.

```
def agregar_pelicula(self):
    nombre = self.lineEdit.text()
    genero = self.lineEdit_2.text()
    director = self.lineEdit_3.text()

    peli = Peliculas(nombre, genero, director)
    milista.AgregarPeliculas(peli)
    contador = 0
    lista = []
    for p in milista.peliculas:
        lista1 = ()
        lista1 = (p.nombre, p.genero, p.director)
        lista.append(lista1)
    self.tableWidget.setRowCount(len(lista))
    for p in lista:
        self.tableWidget.setItem(contador, 0, QTableWidgetItem(p[0]))
        self.tableWidget.setItem(contador, 1, QTableWidgetItem(str(p[1])))
        self.tableWidget.setItem(contador, 2, QTableWidgetItem(str(p[2])))
        contador += 1

    _translate = QtCore.QCoreApplication.translate
    self.lineEdit.setText(_translate("Dialog", ""))
    self.lineEdit_2.setText(_translate("Dialog", ""))
    self.lineEdit_3.setText(_translate("Dialog", ""))
```

Ilustración 1. Agregar Película

```
def mostrar_pelicula(self):
    contador = 0
    lista = []
    for p in milista.peliculas:
        lista1 = ()
        lista1 = (p.nombre, p.genero, p.director)
        lista.append(lista1)
    self.tableWidget.setRowCount(len(lista))
    for p in lista:
        self.tableWidget.setItem(contador, 0, QTableWidgetItem(p[0]))
        self.tableWidget.setItem(contador, 1, QTableWidgetItem(str(p[1])))
        self.tableWidget.setItem(contador, 2, QTableWidgetItem(str(p[2])))
        contador += 1
```

Ilustración 2. Mostrar Películas

```
def Buscar_Pelicula(self):
    nom = self.lineEdit_4.text()
    _translate = QtCore.QCoreApplication.translate
    self.lineEdit_4.setText(_translate("Dialog", ""))
    contador = 0
    lista = []
    for p in milista.peliculas:
        lista1 = ()
        lista1 = (p.nombre, p.genero, p.director)
        lista.append(lista1)
    self.tableWidget.setRowCount(1)
    for p in lista:
        if p[0] == nom:
            self.tableWidget.setItem(contador, 0, QTableWidgetItem(p[0]))
            self.tableWidget.setItem(contador, 1, QTableWidgetItem(str(p[1])))
            self.tableWidget.setItem(contador, 2, QTableWidgetItem(str(p[2])))
            break
```

Ilustración 3. Buscar Película

Implementación del hilo

En Python un objeto “Thread” representa una determinada operación que se ejecuta como un subprocesso independiente, es decir, es la representación de un hilo. El hilo de mi programa se encarga de guardar la información cada 3 segundos para preservar el estado actual en caso de que el programa sea cerrado.

Para esto, fue necesario utilizar la librería “threading”. Primero creé una función llamada “Guardar”, dentro de esta implemente un ciclo While el cual siempre se repetirá hasta que el programa termine, este ciclo espera 3 segundos para ejecutar cada vuelta con ayuda de la función “time.sleep(3)”, después utilice un “try catch” para hacer el guardado de la información que se va escribiendo por el usuario, lo que hice fue guardar en archivos diferentes cada dato (nombre de la película, género y director), para el momento de volver a levantar el programa sea posible recuperar la información de una forma más sencilla.

```
def Guardar(self):  
    '''Contar hasta cien'''  
    while True:  
        time.sleep(3)  
        try:  
            nombre = self.lineEdit.text()  
            genero = self.lineEdit_2.text()  
            director = self.lineEdit_3.text()  
            archivo = open('nombre.txt', 'w')  
            archivo2 = open('genero.txt', 'w')  
            archivo3 = open('director.txt', 'w')  
            archivo.write(nombre)  
            archivo2.write(genero)  
            archivo3.write(director)  
        except Exception as e:  
            print(e)  
        finally:  
            archivo.close()  
            archivo2.close()  
            archivo3.close()
```

Ilustración 4. Función del hilo

Después cree el hilo que manda a llamar la función explicada anteriormente, con otro parámetro que es un “daemon” y lo que hace es terminar el hilo cuando el programa principal termina su ejecución y con la función “hilo1.start()” que inicia la ejecución del hilo.

```
hilo1 = threading.Thread(target=ui.Guardar, daemon=True)
hilo1.start()
```

Ilustración 5. Hilo

Para el posible caso en que el programa sea interrumpido creé una función llamada “Recuperar”, la cual se ejecuta al inicio del programa, esta va abriendo cada archivo y guardando su información en una variable, para después escribir esa información en los campos pertinentes de la interfaz, en caso de que algún o algunos archivos estén vacíos, en los campos de la interfaz no muestra nada. Todo esto se realizó para que en el caso de que el programa se ejecute nuevamente, vuelva al mismo estado en el momento antes de que se cerrara la ejecución del programa.

```
def Recuperar(self):
    archivo = open("nombre.txt", "r")
    nombre = archivo.read()
    archivo2 = open("genero.txt", "r")
    genero = archivo2.read()
    archivo3 = open("director.txt", "r")
    director = archivo3.read()
    _translate = QtCore.QCoreApplication.translate
    self.lineEdit.setText(_translate("Dialog", nombre))
    self.lineEdit_2.setText(_translate("Dialog", genero))
    self.lineEdit_3.setText(_translate("Dialog", director))
    archivo.close()
    archivo2.close()
    archivo3.close()
```

Ilustración 6. Recuperar estado

Servicio

Para el servicio que mantendrá arriba el programa fue necesario la utilización de 2 librerías (psutil, subprocess). Lo primero que hice fue, crear un while que siempre se estará ejecutando hasta que terminé la ejecución del código, dentro del while, con ayuda de una de las funcionalidades que proporciona la primera librería mencionada, lo que hace es identificar los procesos que se están ejecutando en el sistema operativo. Si el programa principal (estatus.exe) está en ejecución se le asigna “1” a una variable, en caso de que esta variable siga en 0 es porque el programa estatus.exe no se encuentra

en ejecución, para levantarlo utilice la función “subprocess.call(‘programa.exe’)” y lo que hace esta función es volver a abrir el programa que se le indica en sus parámetros, para mi caso el programa principal “estatus.exe” esto hará que el programa siempre este arriba.

```
import psutil
import subprocess

while True:
    band = 0
    for proc in psutil.process_iter():
        if proc.name() == 'estatus.exe':
            band = 1

    if band == 1:
        print("Archivo Activo")
    elif band == 0:
        print("No Encontrado")
        subprocess.call('estatus.exe')
```

Ilustración 7. Servicio

Ejecución

Agregar película

Almacenamiento de películas

Agregar

Nombre: Shrek
Genero: Comedia
Director: Vicky Jenson

Buscar Película

Nombre:

Películas

Nombre	Genero	Director
--------	--------	----------

Mostrar

Buscar Película

Almacenamiento de películas

Agregar

Nombre:

Genero:

Director:

Buscar Película

Nombre: Sing 2

Películas

Nombre	Genero	Director
--------	--------	----------

Mostrar

Mostrar Películas

Form

Almacenamiento de películas

Agregar

Nombre

Genero

Director

Agregar

Buscar Pelicula

Nombre

Buscar

Peliculas

	Nombre	Genero	Director
1	Shrek	Comedia	Vicky Jenson
2	Sing 2	Animada	Garth Jennings
3	Duna	Ciencia Ficción	Denis Villeneuve

Mostrar

Interrumpir ejecución

Antes de la interrupción

Form

Almacenamiento de películas

Agregar

Nombre

Spencer

Genero

Drama

Director

Pablo L

Agregar

Buscar Pelicula

Nombre

Buscar

Peliculas

	Nombre	Genero	Director
1	Shrek	Comedia	Vicky Jenson
2	Sing 2	Animada	Garth Jennings
3	Duna	Ciencia Ficción	Denis Villeneuve

Mostrar

Después de la interrupción

Form

Almacenamiento de películas

Agregar

Nombre

Genero

Director

Buscar Película

Nombre

Películas

	Nombre	Genero	Director
1	Shrek	Comedia	Vicky Jenson
2	Sing 2	Animada	Garth Jenni...
3	Duna	Ciencia Fic...	Denis Ville...

El programa se abrió por si solo gracias al servicio que lo mantiene arriba y como podemos observar la información se volvió a mostrar como estaba antes de terminar con su ejecución.

Otra Prueba:

Antes de cerrarlo

Form

Almacenamiento de películas

Agregar

Nombre

Genero

Director

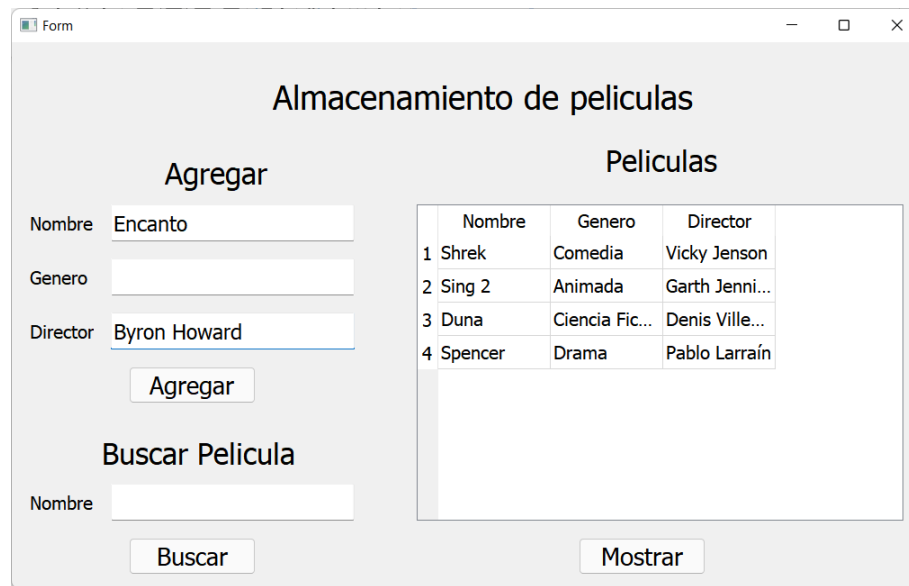
Buscar Película

Nombre

Películas

	Nombre	Genero	Director
1	Shrek	Comedia	Vicky Jenson
2	Sing 2	Animada	Garth Jenni...
3	Duna	Ciencia Fic...	Denis Ville...
4	Spencer	Drama	Pablo Larraín

Después de que se ejecute nuevamente por sí solo



The screenshot shows a Java Swing window titled "Form" with a light gray background. The window is divided into two main sections. The left section is titled "Agregar" (Add) and contains three text input fields labeled "Nombre" (Name), "Genero" (Genre), and "Director" (Director). The "Nombre" field contains the text "Encanto", and the "Director" field contains "Byron Howard". Below these fields is a button labeled "Agregar". The right section is titled "Películas" (Movies) and contains a table with three columns: "Nombre", "Genero", and "Director". The table has four rows of data. Below the table is a button labeled "Mostrar". At the bottom left of the window, there is a section titled "Buscar Película" (Search Movie) with a text input field labeled "Nombre" and a button labeled "Buscar".

	Nombre	Genero	Director
1	Shrek	Comedia	Vicky Jenson
2	Sing 2	Animada	Garth Jenni...
3	Duna	Ciencia Fic...	Denis Ville...
4	Spencer	Drama	Pablo Larraín

Link al código en el repositorio: https://github.com/BryanDeAnda/Estatus_De-Anda-Reyes-Bryan.git

Conclusión

Para finalizar quiero decir que con anterioridad a la realización de este programa yo no había trabajado con servicios ni con hilos, por esta razón en lo personal se me complico bastante su implementación, por otro lado, me pareció bastante interesante como se manejan estos y también pude comprender su gran importancia. Anterior a trabajar con hilos no se me ocurría nada de como recuperar el estado en el que fue cerrado un programa, también me parece muy interesante el hecho de estar ejecutando varios procesos a la vez. En cuanto al servicio es bastante interesante como hacer que un programa se mantenga arriba funcionando a pesar de que sea interrumpido.

Para finalizar quiero decir que a pesar de todos los inconvenientes presentados el resultado fue bastante satisfactorio.

Bibliografía

<https://tecnobillo.com/sections/python-en-windows/servicios-windows-python/servicios-windows-python.html>

<https://python-para-impacientes.blogspot.com/2016/12/threading-programacion-con-hilos-i.html>