

Actividad 07 (QFileDialog)

Jose Eduardo Silva Canizales

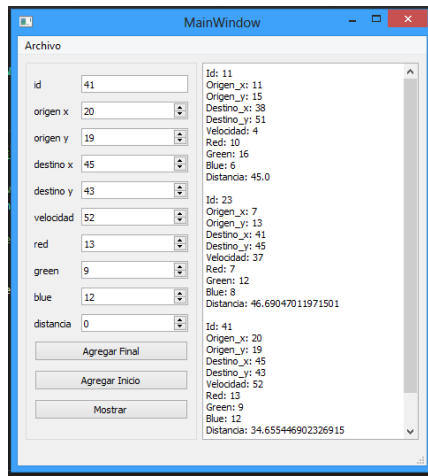
Seminario de solución de problemas de algoritmia

Lineamientos de evaluación

- El reporte está en formato Google Docs o PDF.
- El reporte sigue las pautas del Formato de Actividades .
- El reporte tiene desarrollada todas las pautas del Formato de Actividades.
- Se muestra la captura de pantalla de las partículas con el método mostrar() previo a generar el respaldo.
- Se muestran capturas de pantallas de los pasos que se realizan en la interfaz para generar el respaldo.
- Se muestra el contenido del archivo .json.
- Se muestran capturas de pantallas de los pasos que se realizan en la interfaz para abrir el archivo de respaldo .json.
- Se muestra la captura de pantalla de las partículas con el método mostrar() después de abrir el respaldo.

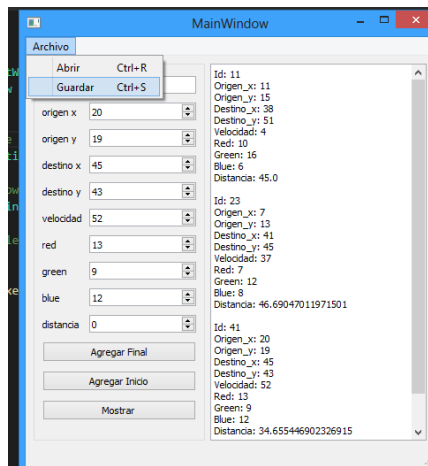
Desarrollo

Captura de pantalla de las partículas con el método mostrar() previo a generar el respaldo.

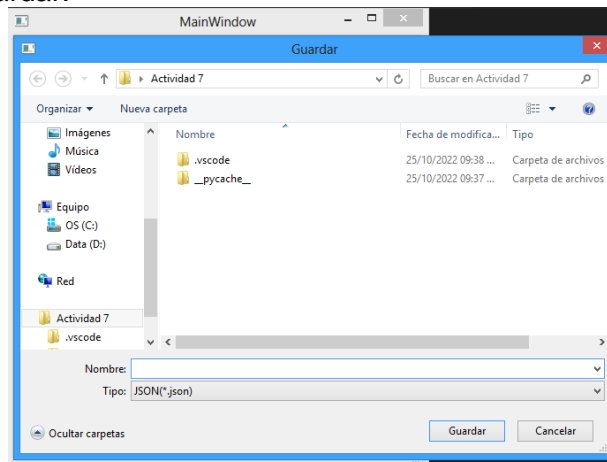


Capturas de pantallas de los pasos que se realizan en la interfaz para generar el respaldo.

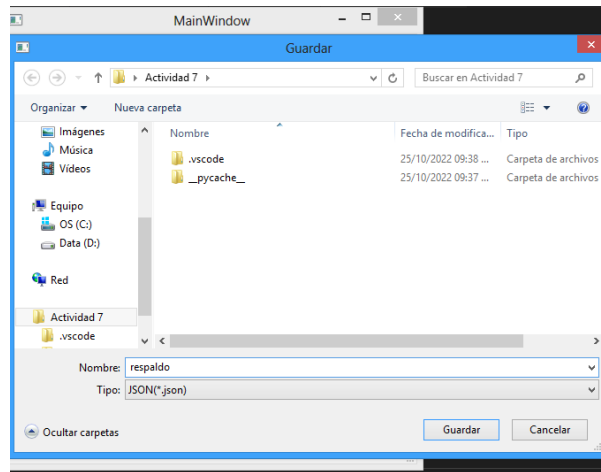
1. Seleccionar guardar.



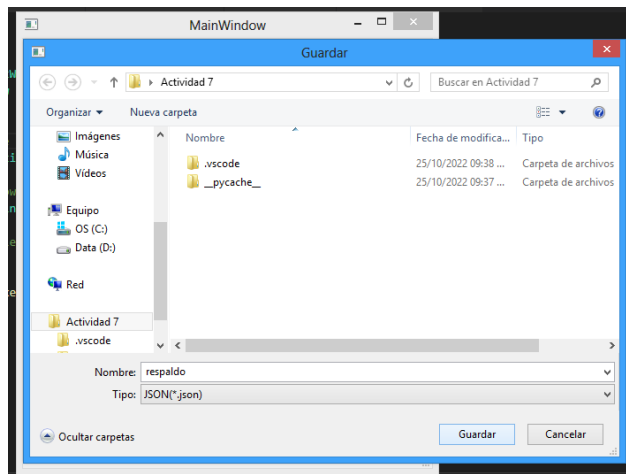
2. Se abre la ventana guardar.



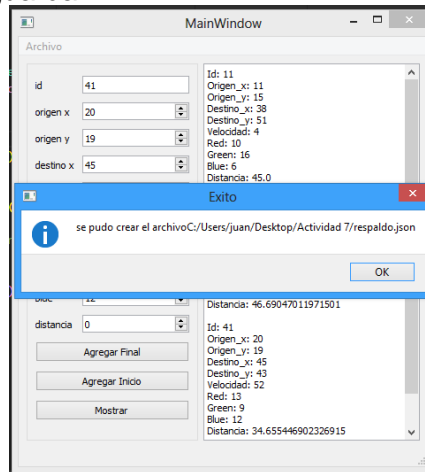
3. Se introduce el nombre del archivo.



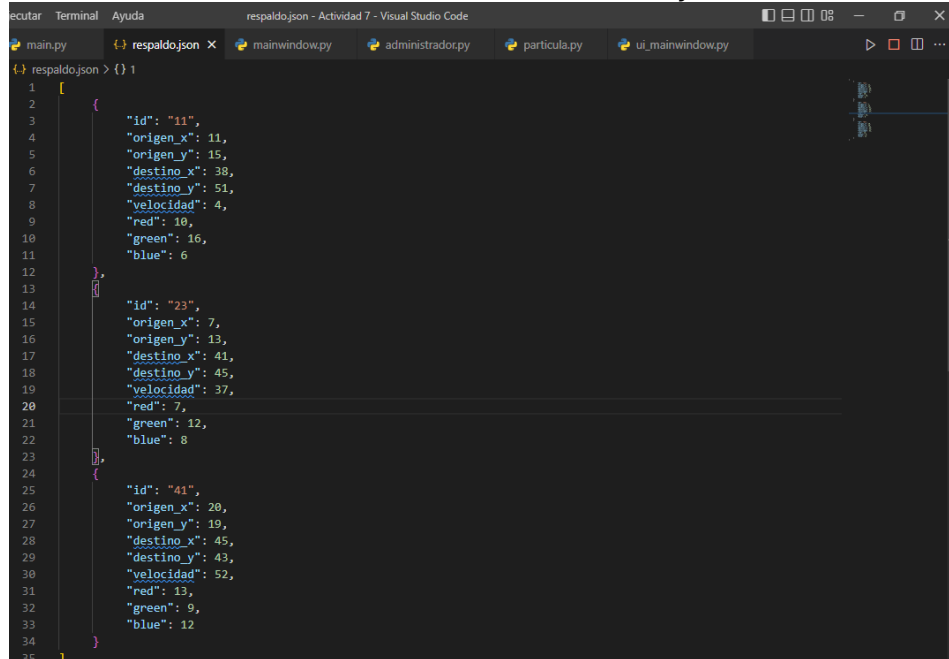
4. Dar clic a guardar.



5. saldrá la ventana de éxito al guardar.



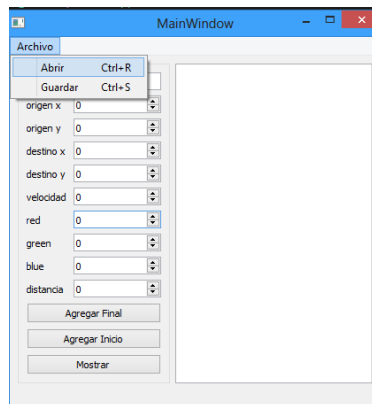
Muestra del contenido del archivo *.json*.



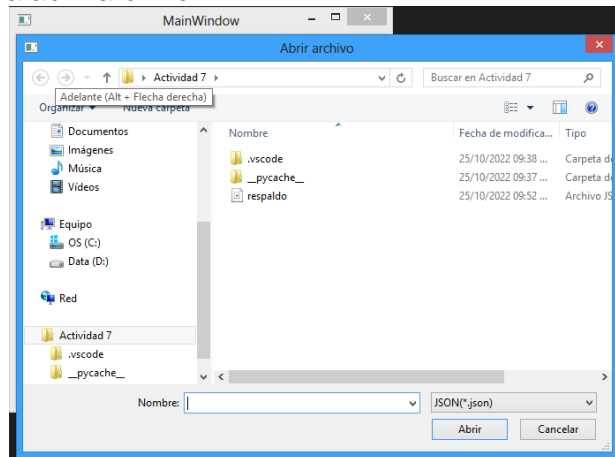
```
1  [
2    {
3      "id": "11",
4      "origen_x": 11,
5      "origen_y": 15,
6      "destino_x": 38,
7      "destino_y": 51,
8      "velocidad": 4,
9      "red": 10,
10     "green": 16,
11     "blue": 6
12   },
13   {
14     "id": "23",
15     "origen_x": 7,
16     "origen_y": 13,
17     "destino_x": 41,
18     "destino_y": 45,
19     "velocidad": 37,
20     "red": 7,
21     "green": 12,
22     "blue": 8
23   },
24   {
25     "id": "41",
26     "origen_x": 20,
27     "origen_y": 19,
28     "destino_x": 45,
29     "destino_y": 43,
30     "velocidad": 52,
31     "red": 13,
32     "green": 9,
33     "blue": 12
34   }
35 ]
```

Capturas de pantallas de los pasos que se realizan en la interfaz para abrir el archivo de respaldo *.json*.

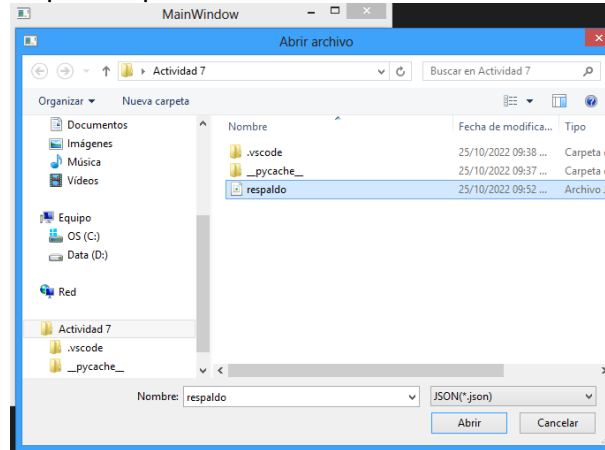
1. Seleccionar abrir.



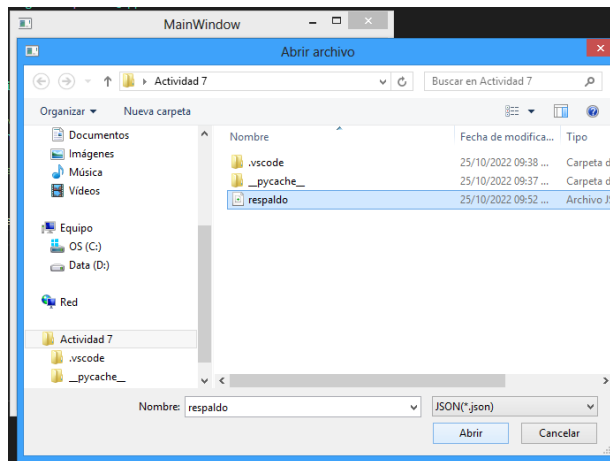
2. Se abre la ventana para abrir archivo.



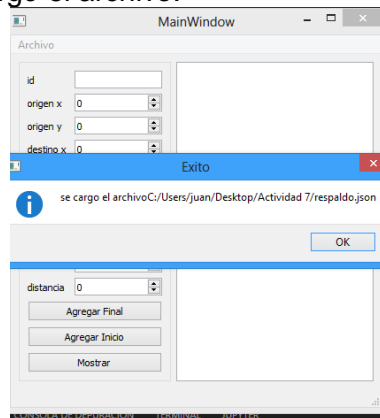
3. Se selecciona el archivo que se quiera abrir.



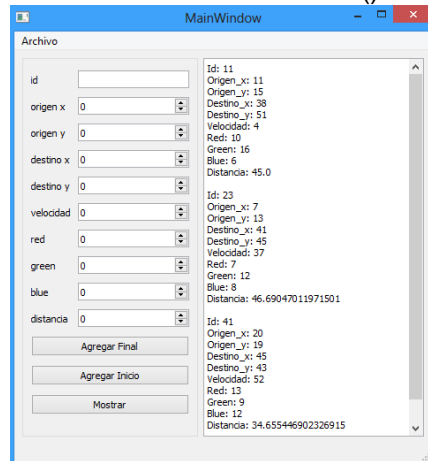
4. Se da clic en abrir



5. Se da el mensaje de que se cargó el archivo.



Captura de pantalla de las partículas con el método mostrar() después de abrir el respaldo.



Conclusiones

Durante esta actividad aprendí a como se puede respaldar la información que se ingresa a la interfaz de usuario, a su vez aprendí a poder abrir ese respaldo de manera correcta, por otra parte aprendí como se a convertir un archivo JSON a una lista de diccionarios.

También tuve en un inicio la problemática de que no podía abrir el archivo previamente respaldado, pero mediante el video de referencia note que el error se debía a que no coincidían los nombres, después de corregir ese problema el código corre sin errores.

Referencias

Primera referencia

Url: <https://www.youtube.com/watch?v=HRY8QvXmcDM>

Título: PySide2 - QFileDialog (Qt for Python)(IV)

Autor: MICHEL DAVALOS BOITES

Código

main.py

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

# Aplicacion de Qt
app = QApplication()

# Se crea window
window = MainWindow()

#se hace visible window
window.show()
# Qt loop
sys.exit(app.exec_())
```

mainwindow.py

```
from PySide2.QtWidgets import QMainWindow, QFileDialog, QMessageBox
from PySide2.QtCore import Slot
from ui_mainwindow import Ui_MainWindow
from particula import Particula
from administrador import Adminisrador

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

        self.administrador=Adminisrador()
        self.ui = Ui_MainWindow()
        self.ui.setupUi(self)
        self.ui.agregar_final_pushButton.clicked.connect(
            self.click_agregar_final)
        self.ui.agregar_inicio_pushButton.clicked.connect(
            self.click_agregar_inicio)
        self.ui.mostrar_pushButton.clicked.connect(self.click_mostrar)
        self.ui.actionAbrir.triggered.connect(self.action_abrir_archivo)
        self.ui.actionGuardar.triggered.connect(self.action_guardar_archivo)

    @Slot()
    def action_abrir_archivo(self):
        ubicacion=QFileDialog.getOpenFileName(
            self,
            'Abrir archivo',
            '.',
            'JSON (*.json) '
        )[0]
        print(ubicacion)
        if self.administrador.abrir(ubicacion):
            QMessageBox.information(
                self,
                "Exito",
                "se cargo el archivo" + ubicacion
```



```

    )
else:
    QMessageBox.critical(
        self,
        "Error",
        "Error al abrir el archivo" + ubicacion
    )

@Slot()
def action_guardar_archivo(self):
    #print('guardar_archivo')
    ubicacion=QFileDialog.getSaveFileName(
        self,
        'Guardar',
        '.',
        'JSON (*.json) '
    )[0]
    print(ubicacion)
    if self.administrador.guardar(ubicacion):
        QMessageBox.information(
            self,
            "Exito",
            "se pudo crear el archivo" + ubicacion
        )
    else:
        QMessageBox.critical(
            self,
            "Error",
            "No se pudo crear el archivo" + ubicacion
        )

@Slot()
def click_mostrar(self):
    self.ui.salida.clear()
    self.ui.salida.insertPlainText(str(self.administrador))

@Slot()
def click_agregar_final(self):
    id = self.ui.id_lineEdit.text()
    origen_x = self.ui.origen_x_spinBox.value()
    origen_y = self.ui.origen_y_spinBox.value()
    destino_x = self.ui.destino_x_spinBox.value()
    destino_y = self.ui.destino_y_spinBox.value()
    velocidad = self.ui.velocidad_spinBox.value()
    red = self.ui.red_spinBox.value()
    green = self.ui.green_spinBox.value()
    blue = self.ui.blue_spinBox.value()
    distancia = self.ui.distancia_spinBox.value()

```

```

        particula=Particula(id,origen_x,origen_y,destino_x,destino_y,velocidad,
ad, red, green, blue, distancia)
        self.administrador.agregar_final(particula)

@Slot()
def click_agregar_inicio(self):
    id = self.ui.id_lineEdit.text()
    origen_x = self.ui.origen_x_spinBox.value()
    origen_y = self.ui.origen_y_spinBox.value()
    destino_x = self.ui.destino_x_spinBox.value()
    destino_y = self.ui.destino_y_spinBox.value()
    velocidad = self.ui.velocidad_spinBox.value()
    red = self.ui.red_spinBox.value()
    green = self.ui.green_spinBox.value()
    blue = self.ui.blue_spinBox.value()
    distancia = self.ui.distancia_spinBox.value()

    particula=Particula(id,origen_x,origen_y,destino_x,destino_y,velocidad,
ad, red, green, blue, distancia)
    self.administrador.agregar_inicio(particula)

```

administrador.py

```

from algoritmos import distancia_euclidiana
from particula import Particula
import json

class Adminisrador:
    def __init__(self):
        self.__particulas = []

    def agregar_final(self, particula: Particula):
        self.__particulas.append(particula)

    def agregar_inicio(self, particula: Particula):
        self.__particulas.insert(0, particula)

    def mostrar(self):
        for particula in self.__particulas:
            print(particula)

    def __str__(self):
        return "".join(
            str(particula)+'\n' for particula in self.__particulas
        )

    def guardar(self, ubicacion):
        try:
            with open(ubicacion, 'w') as archivo:
                lista = [particula.to_dict() for particula in
self.__particulas]
                print(lista)
                json.dump(lista, archivo, indent=5)
            return 1
        except :

```

```

        return 0

    def abrir(self, ubicacion):
        try:
            with open(ubicacion, 'r') as archivo:
                lista=json.load(archivo)
                self.__particulas = [Particula(**particula) for particula in
lista]

            return 1
        except:
            return 0

```

particula.py

```

from algoritmos import distancia_euclidiana

class Particula:
    def __init__(self,
                    id=0, origen_x=0, origen_y=0, destino_x=0,
destino_y=0, velocidad=0, red=0, green=0, blue=0, distancia=0.0):

        self.__id = id
        self.__origen_x = origen_x
        self.__origen_y = origen_y
        self.__destino_x = destino_x
        self.__destino_y = destino_y
        self.__velocidad = velocidad
        self.__red = red
        self.__green = green
        self.__blue = blue
        self.__distancia = distancia_euclidiana(origen_x, origen_y,
destino_x, destino_y)

    def __str__(self):
        return (
            'Id: ' + str(self.__id) + '\n' +
            'Origen_x: ' + str(self.__origen_x) + '\n' +
            'Origen_y: ' + str(self.__origen_y) + '\n' +
            'Destino_x: ' + str(self.__destino_x) + '\n' +
            'Destino_y: ' + str(self.__destino_y) + '\n' +
            'Velocidad: ' + str(self.__velocidad) + '\n' +
            'Red: ' + str(self.__red) + '\n' +
            'Green: ' + str(self.__green) + '\n' +
            'Blue: ' + str(self.__blue) + '\n'+
            'Distancia: ' + str(self.__distancia) + '\n'
        )

    def to_dict(self):
        return {
            "id": self.__id,
            "origen_x": self.__origen_x,
            "origen_y": self.__origen_y,
            "destino_x": self.__destino_x,
            "destino_y": self.__destino_y,
            "velocidad": self.__velocidad,

```

```
"red": self.__red,  
"green": self.__green,  
"blue": self.__blue  
}
```