Actividad 07 - QFileDialog

Gabriel Eduardo Sevilla Chavez

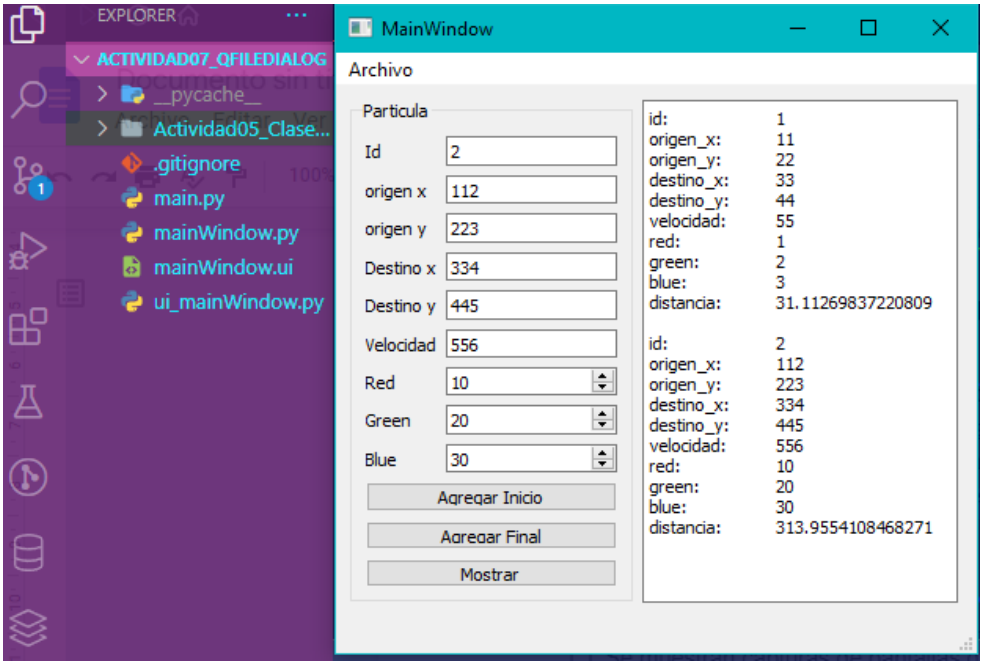
Seminario de algoritmia

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.

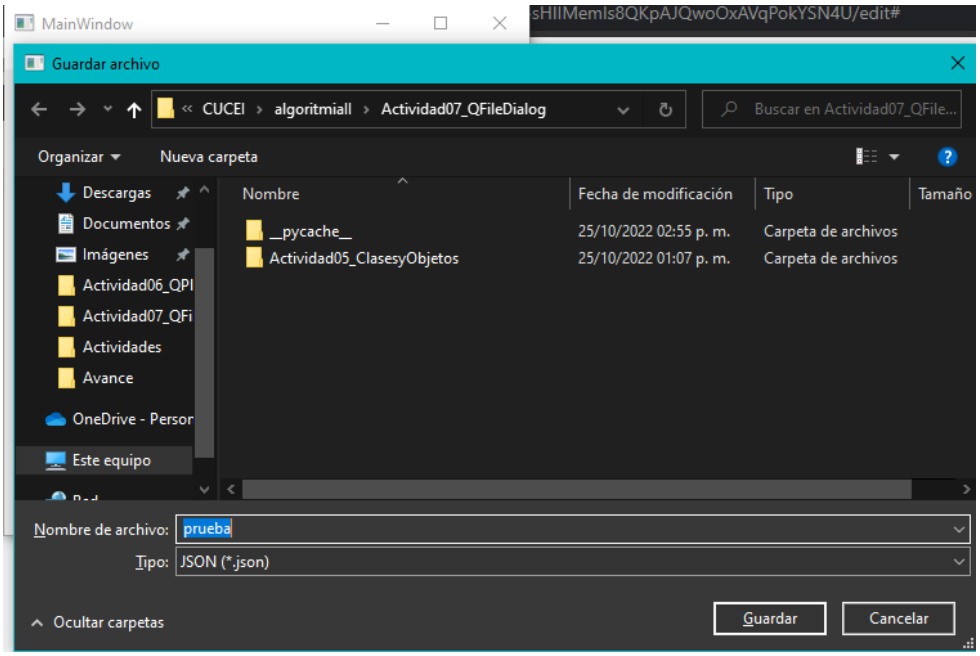
Desarrollo

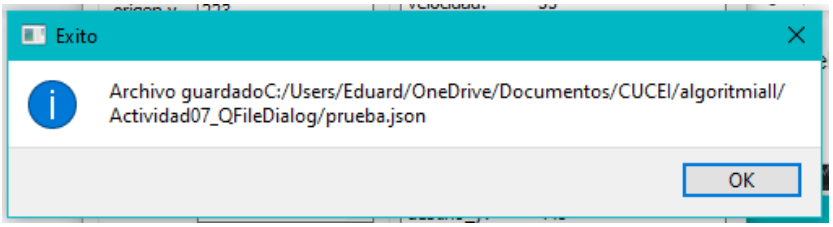
partículas con el método mostrar() previo a generar el respaldo.

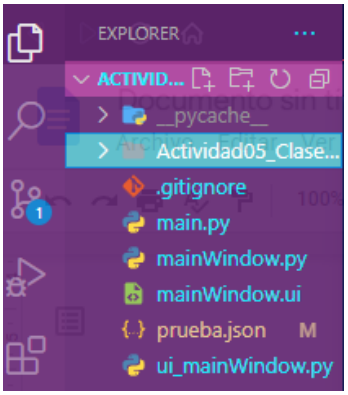


pasos que se realizan en la interfaz para generar el respaldo.

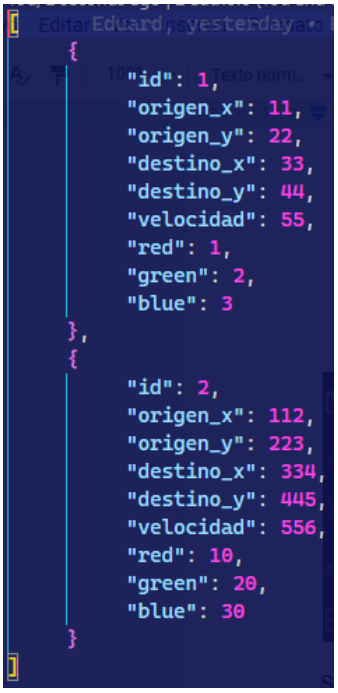
Guardar Archivo



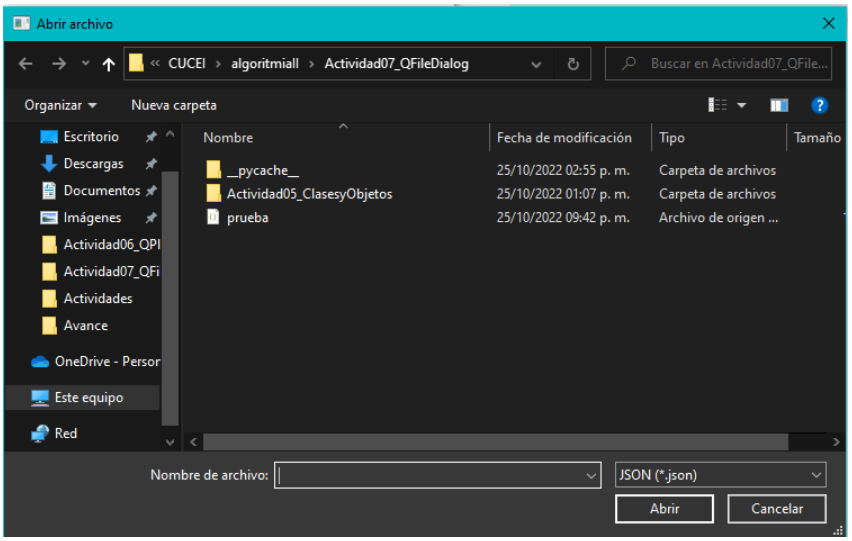


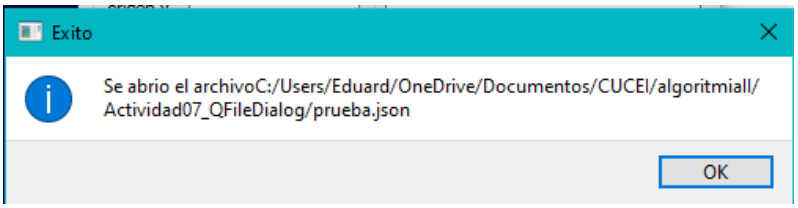


Se muestra el contenido del archivo *.json.*

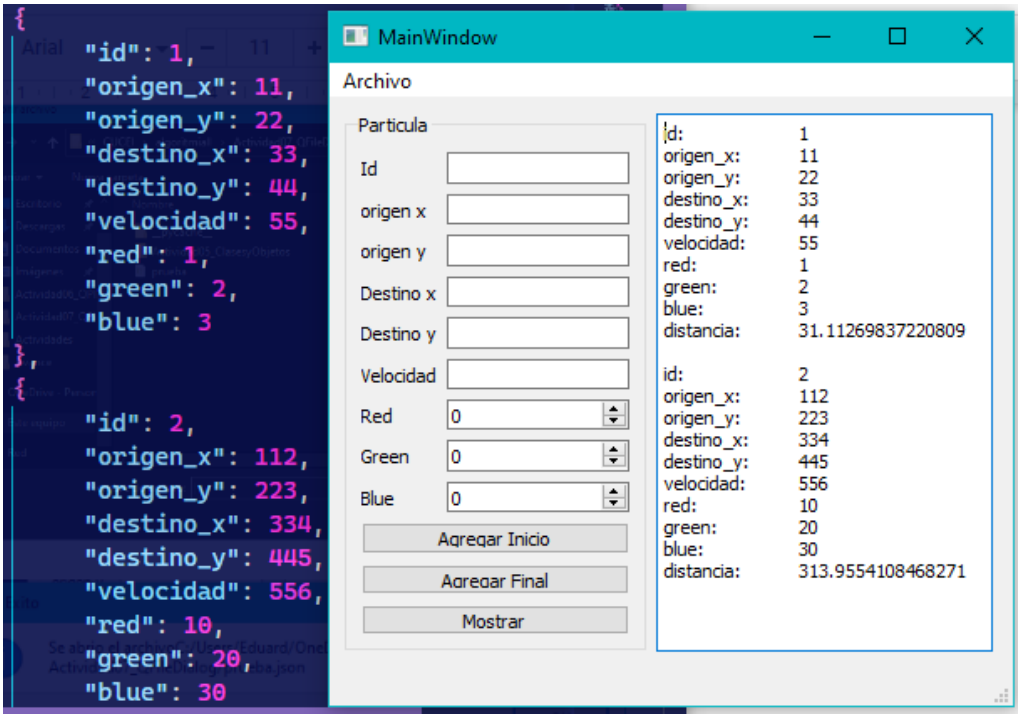


Abrir el archivo de respaldo *.json*.





método mostrar() después de abrir el respaldo.



Conclusiones

Hubo pequeñas dificultades en la parte en la que se requería la ubicación del archivo, pero después de un tiempo pudo resolverse sin problema.

Fue una actividad interesante ya que antes no había usado tanto el uso de archivos .json para guardar datos y el uso de estos.

Referencias

MICHEL DAVALOS BOITES. (2020c, octubre 22). ***PySide2 - QFileDialog (Qt for Python)(IV)*****. YouTube.**

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

Código

`main.py`

```
from PySide2.QtWidgets import QApplication
```

```
from mainWindow import MainWindow
import sys


app = QApplication()
#Ventana de app
window = MainWindow()
window.show()
sys.exit(app.exec_())
```

mainWindow.py

```
#from cgitb import text
from PySide2.QtWidgets import QMainWindow, QFileDialog, QMessageBox
from PySide2.QtCore import Slot
from ui_mainWindow import Ui_MainWindow
#incluir clases particlas
from Actividad05_ClasesyObjetos.particulas import Particulas
from Actividad05_ClasesyObjetos.particula import Particula
class MainWindow(QMainWindow):
    def __init__(self):
        super(MainWindow, self).__init__() #Contructor de
QMainWindow
        #Guardar particulas
        self.particulas = Particulas()
```

```
        self.ui = Ui_MainWindow()
        #mandar los datos de self.ui a la ventana
        self.ui.setupUi(self)
        # Eventos en botones

self.ui.pbAgregarInicio.clicked.connect(self.click_agregarInicio)

self.ui.pbAgregaFinal.clicked.connect(self.click_agregarFinal)
        self.ui.pbMostrar.clicked.connect(self.click_mostrar)
        #Ad Archivo

self.ui.actionAbrir_archivo.triggered.connect(self.actionOpenFile)

self.ui.actionGuardar_archivo.triggered.connect(self.actionSaveFile)
    @Slot()
    def actionOpenFile(self):
        ubicacion = QFileDialog.getOpenFileName(
            self,
            'Abrir archivo',
            '.',
            'JSON (*.json)'
        )[0]
        if self.particulas.abrir(ubicacion):
```

```
            QMessageBox.information(
                self,'Exito',
                'Se abrio el archivo'+ubicacion
            )
        else:
            QMessageBox.critical(
                self,'Error',
                'Error al abrir el archivo'+ubicacion
            )
    def actionSaveFile(self):
        #print("Guardar archivo")
        ubicacion = QFileDialog.getSaveFileName(
            self,
            'Guardar archivo',
            '.',
            'JSON (*.json)'
        )[0]
        print(ubicacion)
        if self.particulas.guardar(ubicacion):
            QMessageBox.information(
                self,"Exito","Archivo guardado"+ubicacion
            )
```

```
        else:
            QMessageBox.critical(
                self,"Error","No se pudo guardar el archivo"+
ubicacion
            )
    @Slot()
    def click_mostrar(self):
        self.ui.salida.clear()
        self.particulas.mostrar()
        self.ui.salida.insertPlainText(str(self.particulas))

    @Slot() #Guardar los datos obenidos
    def click_agregarInicio(self):
        id = self.ui.leId.text()
        origenx = self.ui.leOrigenx.text()
        origeny = self.ui.leOrigenY.text()
        destinox = self.ui.leDestinoX.text()
        destinoy = self.ui.leDestinoY.text()
        velocidad = self.ui.leVelocidad.text()
        red = self.ui.sbRed.value()
        green = self.ui.sbGreen.value()
        blue = self.ui.sbBlue.value()
```

```
        #Crear particla
        particula =
Particula(int(id),int(origenx),int(origeny),int(destinox),int(destin
oy),int(velocidad),red,green,blue)
        self.particulas.agregar_inicio(particula)


    @Slot() #Guardar los datos obenidos
    def click_agregarFinal(self):
        id = self.ui.leId.text()
        origenx = self.ui.leOrigenx.text()
        origeny = self.ui.leOrigenY.text()
        destinox = self.ui.leDestinoX.text()
        destinoy = self.ui.leDestinoY.text()
        velocidad = self.ui.leVelocidad.text()
        red = self.ui.sbRed.value()
        green = self.ui.sbGreen.value()
        blue = self.ui.sbBlue.value()
        #Crear particla
        particulafinal =
Particula(int(id),int(origenx),int(origeny),int(destinox),int(destin
oy),int(velocidad),red,green,blue)
        self.particulas.agregar_final(particulafinal)
```

particulas.py

```python
import json
from .particula import Particula
#.partiucla para que la carpeta principal la tome
class Particulas:
    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) -> str:
        return "".join(
            str(particula) + "\n" for particula in self.__particulas
        )
    #Archivo
```

```
    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
#.algoritmos para que la carpeta principal la tome
class Particula():
```

```
    def
__init__(self,id=0,origen_x=0,origen_y=0,destino_x=0,destino_y=0,vel
ocidad=0,red=0,green=0,blue=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) -> str:
        return(
            "id: \t"+ str(self.__id)+'\n'+
            "origen_x: \t"+ str(self.__origen_x)+'\n'+
            "origen_y: \t"+ str(self.__origen_y)+'\n'+
            "destino_x: \t"+ str(self.__destino_x)+'\n'+
            "destino_y: \t"+ str(self.__destino_y)+'\n'+
            "velocidad: \t"+ str(self.__velocidad)+'\n'+
```

```
            "red: \t"+ str(self.__red)+'\n'+
            "green: \t"+ str(self.__green)+'\n'+
            "blue: \t"+ str(self.__blue)+'\n'+
            "distancia: \t"+ 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
        }
```