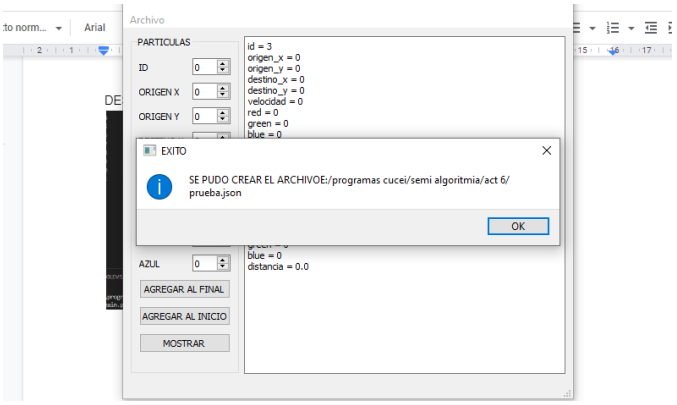
Actividad 07 (QFileDialog)

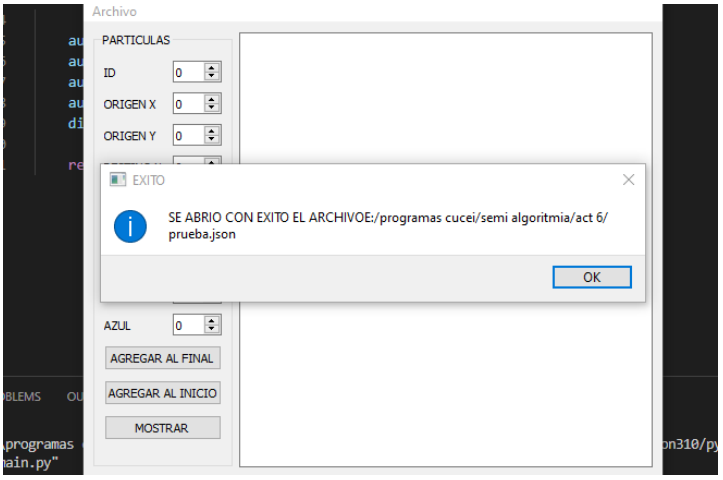
Beracoechea Rosales Jose Francisco
seminario de algoritmia

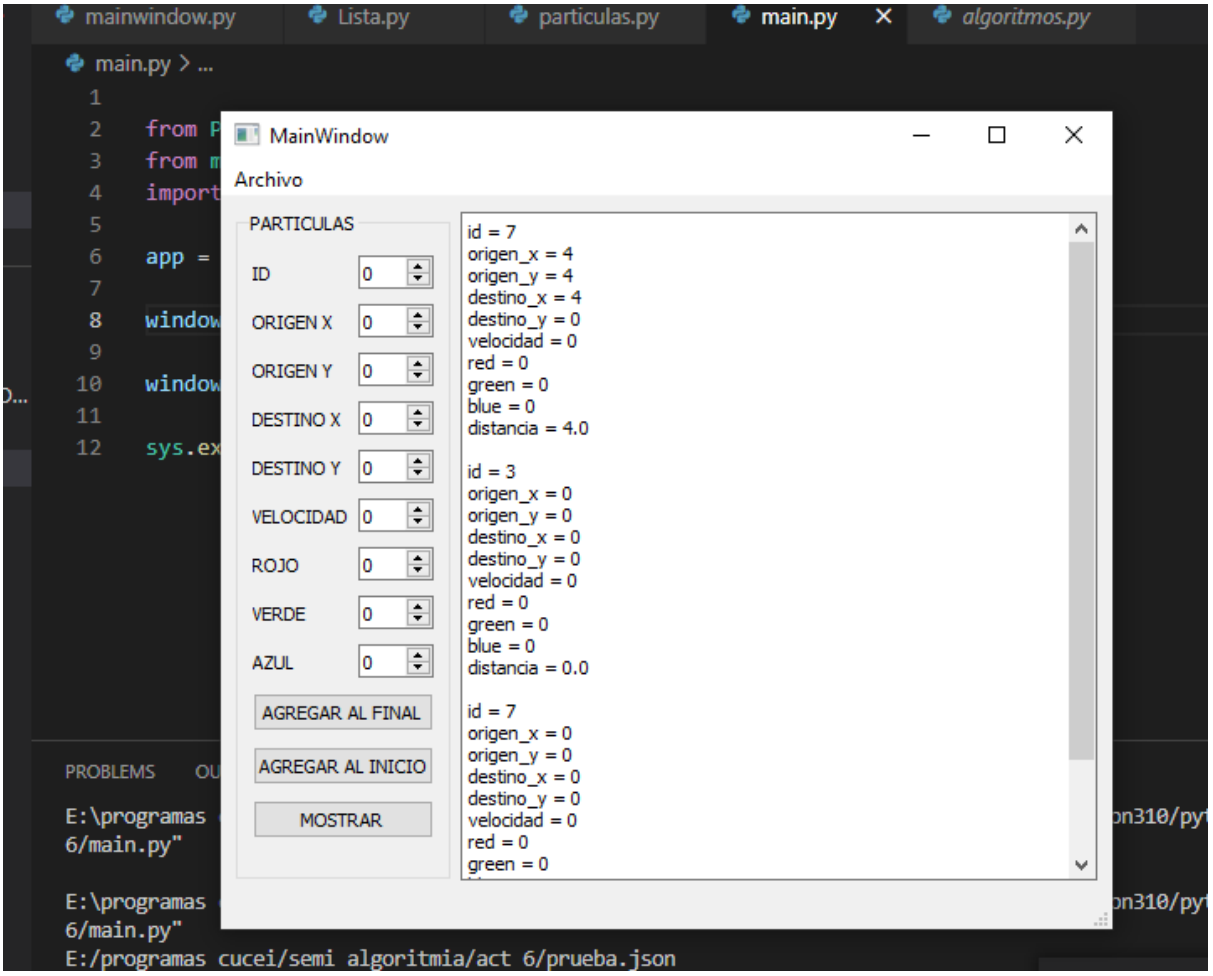
Lineamientos de evaluación

- [ ] 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:







CONCLUSIONES:

En esta actividad aprendí a guardar y abrir los archivos creados para una interfaz gráfica lo que se me complicó un poco fue al momento de abrir los datos guardados ya que los había guardado con un nombre de variable diferente

Referencias:

https://youtu.be/HRY8QvXmcDM

Codigo:

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

app = QApplication()

window = MainWindow()

window.show()

sys.exit(app.exec_())

## particulas.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):

        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,
        }
from particulas import Particula
import json

class Lista:

    def __init__ (self):
            self.__Lista = []

    def agregar_final(self, particulas:Particula ):
            self.__Lista.append(particulas)


    def agregar_inicio(self, particulas:Particula ):
            self.__Lista.insert(0, particulas)


    def mostrar(self):
            for particulas in self.__Lista:
                print(particulas)

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

        )

    def guardar (self, ubicacion):
        try:

                with open(ubicacion, 'w') as archivo:
                    lista = [particulas.to_dict() for particulas in self.__Lista]

                    json.dump(lista, archivo,indent=11)
                return 1
        except:
                return 0
```

```
    def abrir(self, ubicacion):
        try:
            with open(ubicacion,'r') as archivo:
                lista = json.load(archivo)
                self.__Lista = [Particula(**particulas) for particulas in lista]
            return 1
        except:
            return 0


from PySide2.QtWidgets import QMainWindow , QFileDialog, QMessageBox
from PySide2.QtCore import Slot
from ui_mainwindow  import Ui_MainWindow
from particulas import Particula
from Lista import Lista


class MainWindow(QMainWindow):
    def __init__(self):
        super(MainWindow,self).__init__()
        self.lista = Lista()
        self.ui = Ui_MainWindow()
        self.ui.setupUi(self)
        self.ui.inicio_pushButton.clicked.connect(self.click_agregar)
        self.ui.FINAL_pushButton.clicked.connect(self.click_final)
        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]
        if self.lista.abrir(ubicacion):
            QMessageBox.information(
              self,
              "EXITO",
              "SE ABRIO CON EXITO EL ARCHIVO"  + ubicacion
         )
        else:
            QMessageBox.critical(
```

```
            self,
            "ERORR",
            "ERROR AL ABRIR EL ARCHIVO"  + ubicacion
        )



@Slot()
def action_guardar_archivo(self):
    ubicacion = QFileDialog.getSaveFileName(
            self,
            'Guardar',
            '.',
            'JSON (*.json)'
        )[0]
    print(ubicacion)
    if self.lista.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.insertPlainText(str(self.lista))


@Slot()
def click_agregar(self):
    id = self.ui.ID_spinBox.value()
    origen_x = self.ui.ORIGEN_X_spinBox.value()
    origen_y = self.ui.ORIGEN_Y_spinBox.value()

    destino_x = self.ui.x_spinBox.value()
    destino_y = self.ui.y_spinBox.value()
    velocidad = self.ui.velocidad_spinBox.value()
    rojo = self.ui.rojo_spinBox.value()
    verde = self.ui.verde_spinBox.value()
```

```
        azul = self.ui.azul_spinBox.value()

        particula = Particula(id,origen_x,origen_y,destino_x,destino_y,velocidad,rojo,verde,azul)
        self.lista.agregar_inicio(particula)


    @Slot()
    def click_final(self):
        id = self.ui.ID_spinBox.value()
        origen_x = self.ui.ORIGEN_X_spinBox.value()
        origen_y = self.ui.ORIGEN_Y_spinBox.value()

        destino_x = self.ui.x_spinBox.value()
        destino_y = self.ui.y_spinBox.value()
        velocidad = self.ui.velocidad_spinBox.value()
        rojo = self.ui.rojo_spinBox.value()
        verde = self.ui.verde_spinBox.value()
        azul = self.ui.azul_spinBox.value()

        particula = Particula(id,origen_x,origen_y,destino_x,destino_y,velocidad,rojo,verde,azul)
        self.lista.agregar_final(particula)

# algoritmos.py
import math
def distancia_euclidiana(x_1, y_1, x_2, y_2):

    aux1 =  x_2-x_1
    aux1 = aux1*aux1
    aux2 = y_2-y_1
    aux2 = aux2*aux2
    distancia = math.sqrt(aux1+aux2)

    return (distancia)
```