# Actividad 07 – QFileDialog

Hernández Lomelí Diego Armando

Seminario de algoritmia 2022B D02

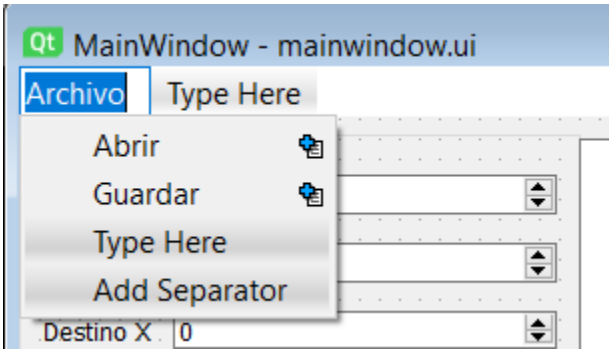
## Lineamientos de evaluación.

- [ ] 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

## Desarrollo

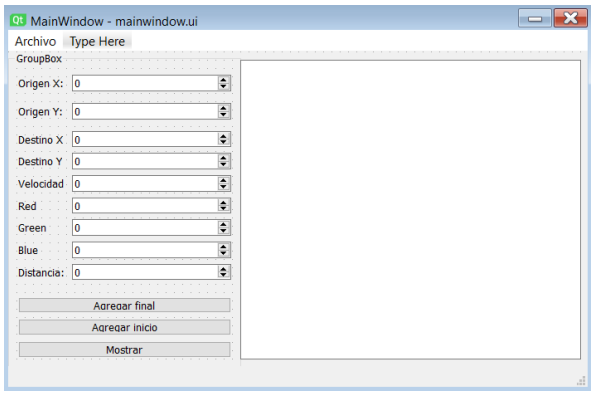
Esta actividad consiste en generar respaldos de partículas en archivos individuales que puedan ser utilizados en cualquier momento, también debemos ser capaces de leer estos archivos respaldados y para ello empezaremos agregando acciones a nuestra interfaz

Para agregar las acciones basta con abrir nuestra interfaz anterior y pulsar el botón **Type Here**, al presionar podemos escribir el texto de la opción de nuestro menú, podemos repetir el proceso si pulsamos el botón que generemos y veremos el resultado de la imagen, para este trabajo agregamos 2 opciones **abrir** y **guardar**.



También es posible agregar atajos de teclado si en el árbol de elementos (ubicado a la derecha de la ventana de QT Designer) damos doble click a la acción, nos despliega una ventana en la que podremos agregar el atajo de teclado.

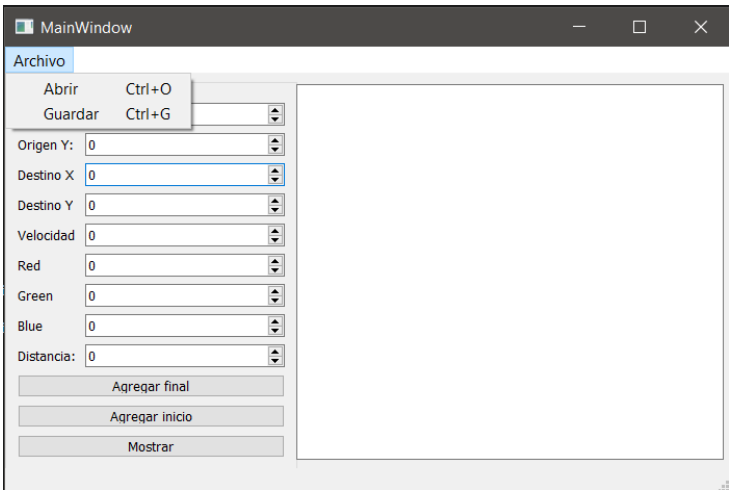
Resultado en la ejecución.




Ahora que tenemos la interfaz creada vamos a la creación de la pantalla a código .py

```
PS C:\Users\Armando\Documents\GitHub\Actividad 7\Actividad-7> C:\Users\Armando\AppData\Local\Programs\Python\Python310\Scripts\pyside2-uic "C:\Users\Armando\Documents\GitHub\Actividad 7\Actividad-7\src\mainwindow.ui" -o src\ui_mainwindow.py
```

Pantalla resultante en ejecución desde Python. Si pulsamos el btn **archivo** podremos ver las opciones agregadas previamente y además estará escrito el atajo de teclado que hayamos configurado.




El siguiente paso es agregar eventos a los componentes hechos anteriormente (en la imagen son las ultimas 2 lineas) y para este elemento corresponde el elemento **triggered** que responde al teclado también.

```
def __init__(self):
    super(MainWindow, self).__init__()
    self.__lista = listaParticula()
    self.ui = Ui_MainWindow()
    self.ui.setupUi(self)
    self.ui.btnAgregarInicio.clicked.connect(self.click_agregar_inicio)
    self.ui.btnAgregarFinal.clicked.connect(self.click_agregar_final)
    self.ui.btnMostrar.clicked.connect(self.mostrar)
    """ Metodos para el menu de la ventana """
    self.ui.actionAbrir.triggered.connect(self.action_abrir_archivo)
    self.ui.actionGuardar.triggered.connect(self.action_guardar_archivo)
```

El siguiente paso es generar las funciones que corresponden a los eventos agregados. Para empezar a depurar vamos a mostrar un **QFileDialog** (debe importarse)

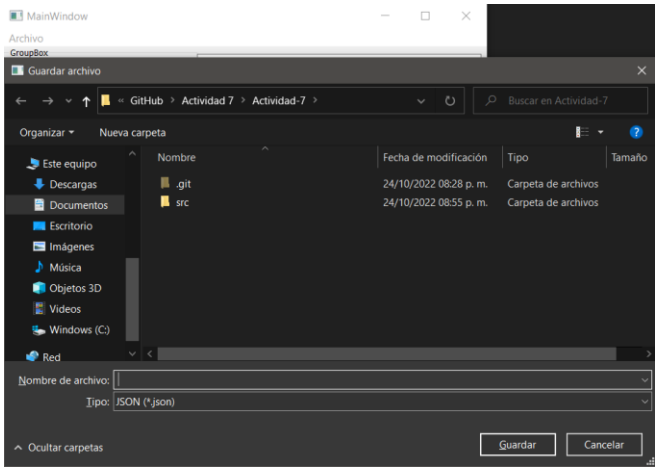
```python
""" Generación de eventos para acciones del menu """
@Slot()
def action_guardar_archivo(self):
    print("guardar")

@Slot()
def action_abrir_archivo(self):
    ubicacion = QFileDialog.getSaveFileName(
        self,
        "Guardar archivo",
        ".",
        "JSON (*.json)"
    )
    print(ubicacion)
```

La función **getSaveFileName** nos abre un dialogo de selección de archivo, lo configuramos para que tenga de titulo **guardar archivo** y filtre todos los archivos para que solo muestre los de extensión **.json.**

Los pasos anteriores los hacemos desde el archivo **mainwindow.py** que gestiona los eventos de la ventana generada.

Visualización de la ventana para seleccionar un archivo al usar la acción **abrir**.




Después de la prueba debemos definir el método guardar dentro de la clase el archivo **listaParticula** nos permitirá guardar los archivos dentro de un archivo **.json.**

```python
def guardar(self, ubicacion):
    with open(ubicacion, 'w') as archivo:
        lista = [particula.to_dict() for particula in self.__particulas]
        print(lista)
```

A su vez, dentro de la clase particula debemos ingresar el método **to_dict**, para que el método anterior funcione. Todos los atributos estarán entre llaves y cada atributo debe estar definido como una cadena de caracteres, después lleva una separación con **":"** y después la variable que le corresponde.

```
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.__veloicidad,
        "Roj": self.__red,
        "Verde": self.__green,
        "Azul": self.__blue,
    }
```

Convierte a la instancia de la clase en un diccionario, lo podemos usar para generar cada archivoe n formato **json**, para eso debemos importar la clase **json** en el archivo de **listaParticulas**.

El siguiente paso es guardar el archivo **.json**, ahora vamos a agregar la función **json.dump()**.

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

Si probamos a guardar el archivo tendremos el siguiente resultado

```
[
    {
        "Id": 0,
        "Origen X": 0,
        "Origen Y": 0,
        "Destino X": 0,
        "Destino Y": 0,
        "Velocidad": "0",
        "Roj": 0,
        "Verde": 0,
        "Azul": 0
    }
]
```

Agregamos un bloque para excepciones y saber si la operación se logró realizar o no.

Si retorna 1, se ha guardado el archivo, si retorna 0, no se ha guardado el archivo, seguramente por un error.

```
def guardar(self, ubicacion):
    try:
        with open(ubicacion, 'w') as archivo:
            lista = [particula.to_dict()
                     for particula in self.__particulas]

            json.dump(lista, archivo, indent=5)
        return 1
    except:
        return 0
```

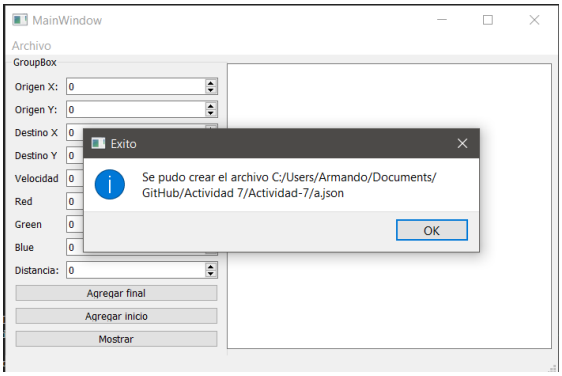
Ahora nos devolvemos a nuestro archivo **mainwindow.py** donde están todos los eventos y agregamos la importación a la clase **QMessageBox** para indicar al usuario si se ha guardado el archivo o no

```
from ui_mainwindow import Ui_MainWindow, QFileDialog, QMessageBox
```

Agregamos una condicional en el evento de guardado para hacer que se muestre el dialogo.

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

Ahora sabremos si la operación se logró realizar.




Ya que somos capaces de generar los archivos, vamos a leer los archivos que ya estén generados.

Dentro de la clase **listaParticula** agregamos el siguiente código para que sea capaz de recuperar los datos. Usamos la misma estrategias para saber si se logró recuperar o no el archivo.

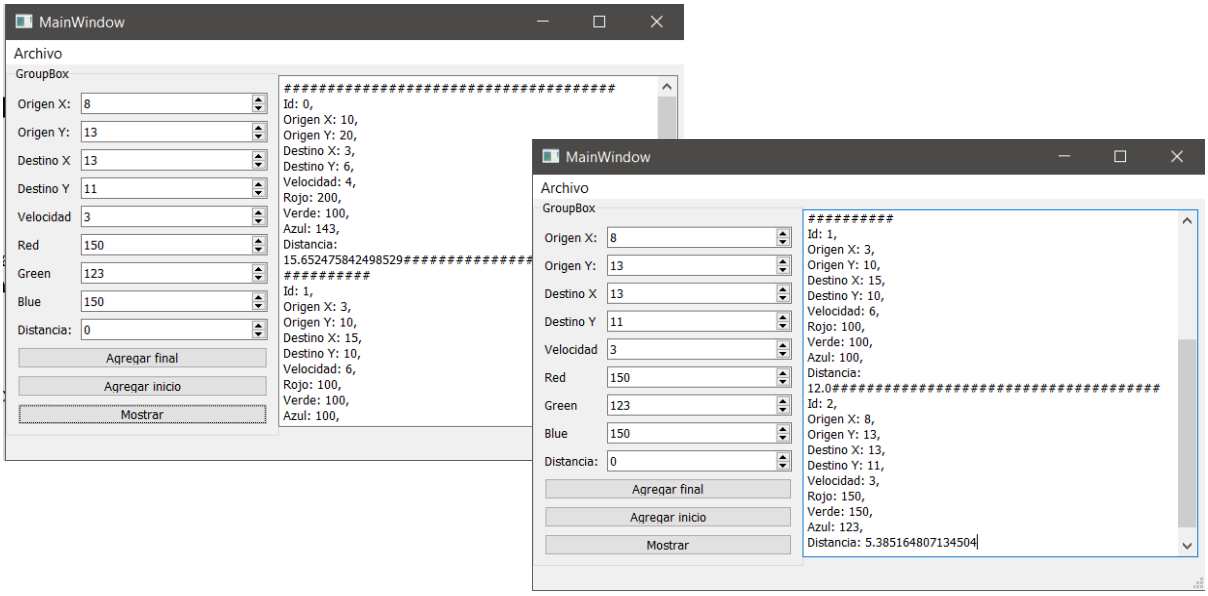
```
def abrir(self, ubicacion):
    try:
        with open(ubicacion, 'r') as archivo:
            lista = json.load(archivo)
            self.__particulas = [Particula(**particula)
                                 for particula in lista]
        return 1
    except:
        return 0
```

Después, en la clase que tiene los eventos de las acciones modificamos el método **action_guardar_archivo** para abrir los archivos, el resultado será muy similar y que además nos mostrará en el **QPlainTextEdit** todas las partículas almacenadas en la lista.

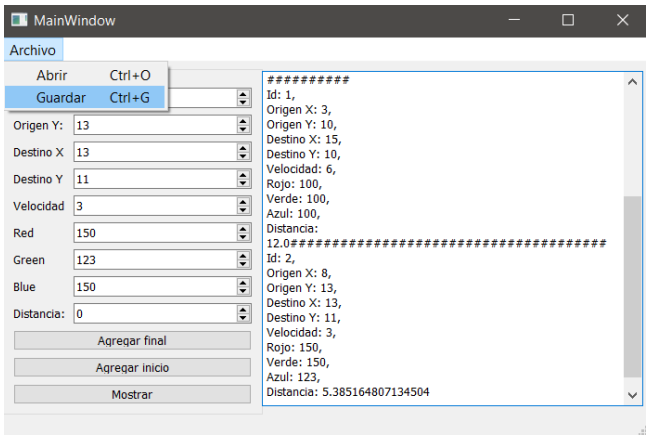
Para terminar, vamos a probar hacer respaldo de 3 particulas y cargarlas después, para demostrar que se están cargando los respaldos, vamos a cerrar completamente la ventana y daremos **abrir** nada más iniciada la vista.

**Generando 3 particulas.**

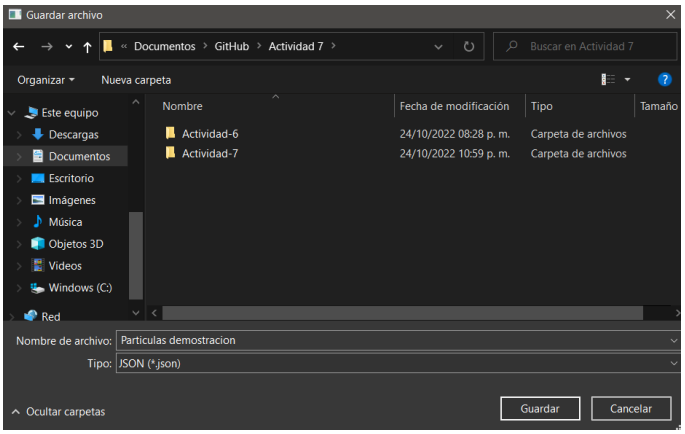
En este paso mostraré las partículas generadas directamente del **QPlainTextEdit**



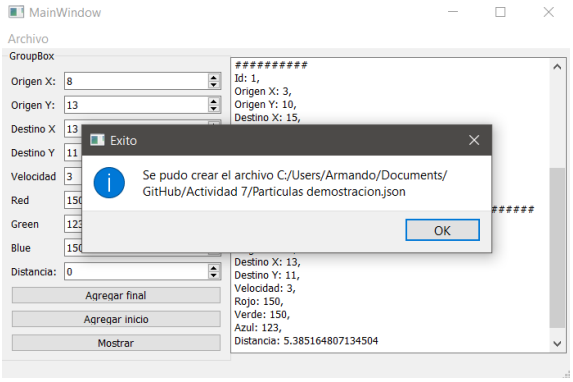

Vamos a guardar las partículas generadas en un nuevo archivo nombrado como **"Partículas demostración.json"**. Activamos la acción **guardar**.



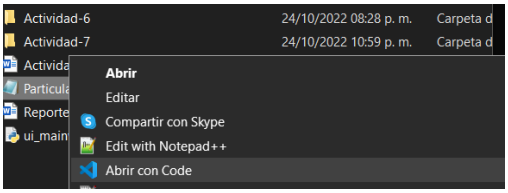

Vemos el cuadro de dialogo y elegimos la ruta.




Pulsamos a guardar y nos mostrará el cuadro de texto indicando que se ha guardado el archivo.




Vamos a abrir el archivo con ayuda de Visual Studio Code

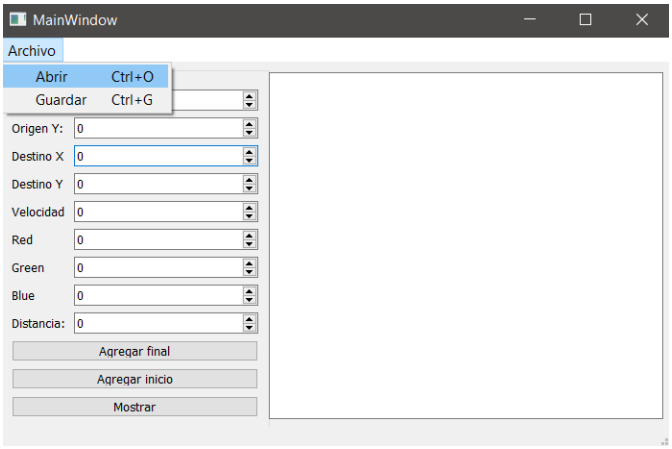


```json
[
    {
        "id": 0,
        "origen_x": 10,
        "origen_y": 20,
        "destino_x": 3,
        "destino_y": 6,
        "veloicidad": "4",
        "red": 200,
        "green": 100,
        "blue": 143,
        "distancia": 15.652475842498529
    },
    {
        "id": 1,
        "origen_x": 3,
        "origen_y": 10,
        "destino_x": 15,
        "destino_y": 10,
        "veloicidad": "6",
        "red": 100,
        "green": 100,
        "blue": 100,
        "distancia": 12.0
    },
    {
        "id": 2,
        "origen_x": 8,
        "origen_y": 13,
        "destino_x": 13,
        "destino_y": 11,
        "veloicidad": "3",
        "red": 150,
        "green": 150,
        "blue": 123,
        "distancia": 5.385164807134504
    }
]
```

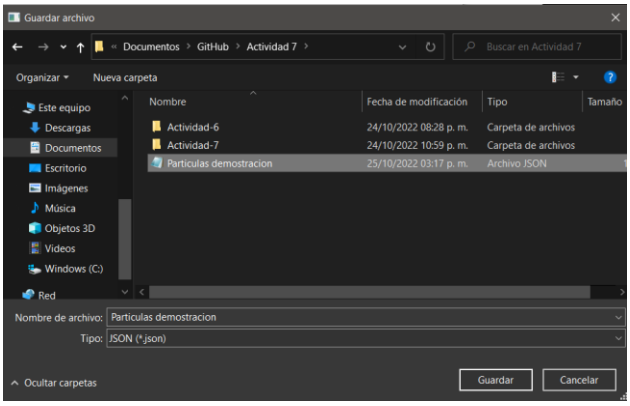

Este es el resultado almacenado dentro del archivo.

El siguiente paso es la apertura del archivo y para eso cerramos terminamos la ejecución completamente y volvemos a abrir.

Es turno de abrir el archivo.

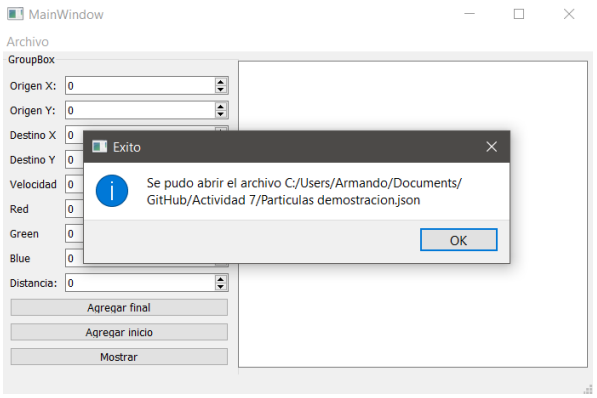



Nos vuelve a mostrar la ventana de selección de archivo y seleccionamos el archivo previamente guardado.

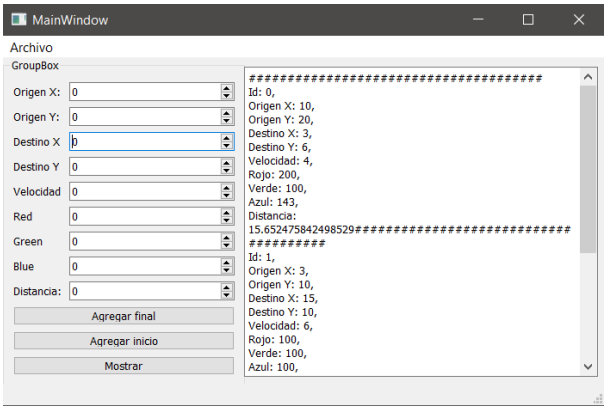


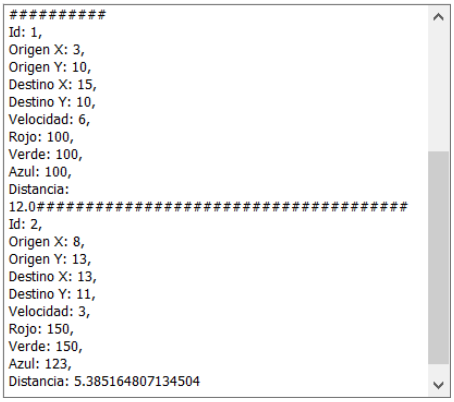

Quizá nos indique que el archivo ya existe, pero podemos continuar con el procedimiento sin problema.

Se muestra el mensaje de éxito al cargar el archivo y podremos cerrar la ventana del mensaje.




Después usamos el método mostrar para que sean visibles las partículas del respaldo.




El respaldo fue cargado con éxito y cargado en el componente para su visualización.

## Conclusiones

Actividad bastante sencilla y útil para poder serializar los datos de una colección a un archivo externos para poder exportar datos que pueden ser incluso como archivos de configuración o para programas más especiales como horarios, controles de gastos etc.

## Referencia

BOITES, M. D. (20 de Octubre de 2022). Obtenido de Youtube:
https://www.youtube.com/watch?v=HRY8QvXmcDM

## Código

Código de archivo **main.py.**

```
from PySide2.QtWidgets import QApplication
from mainwindow import MainWindow
import sys
app = QApplication()
window = MainWindow()
window.show()
sys.exit(app.exec_())
```

Código de archivo **mainwindow.py.**

```
from ui_mainwindow import Ui_MainWindow, QFileDialog, QMessageBox
from PySide2.QtWidgets import QMainWindow
from PySide2.QtCore import Slot
from listaParticulas import listaParticula
from Particula import Particula


class MainWindow(QMainWindow):
    __contador = 0

    def __init__(self):
        super(MainWindow, self).__init__()
        self.__lista = listaParticula()
        self.ui = Ui_MainWindow()
        self.ui.setupUi(self)
        self.ui.btnAgregarInicio.clicked.connect(self.click_agregar_inicio)
        self.ui.btnAgregarFinal.clicked.connect(self.click_agregar_final)
        self.ui.btnMostrar.clicked.connect(self.mostrar)
        """ Metodos para el menu de la ventana """
        self.ui.actionAbrir.triggered.connect(self.action_abrir_archivo)
        self.ui.actionGuardar.triggered.connect(self.action_guardar_archivo)

    """ Generación de eventos para acciones del menu """
    @Slot()
    def action_guardar_archivo(self):
        ubicacion = QFileDialog.getSaveFileName(
            self,
            "Guardar archivo",
            ".",
            "JSON (*.json)"
```

```
        )[0]
        if self.__lista.guardar(ubicacion):
            QMessageBox.information(
                self,
                "Exito",
                ("Se pudo crear el archivo " + ubicacion)
            )
        else:
            QMessageBox.critical(
                self,
                "Error",
                ("No pudo crear el archivo " + ubicacion)
            )

    @Slot()
    def action_abrir_archivo(self):
        ubicacion = QFileDialog.getSaveFileName(
            self,
            "Guardar archivo",
            ".",
            "JSON (*.json)"
        )[0]
        if self.__lista.abrir(ubicacion):
            QMessageBox.information(
                self,
                "Exito",
                ("Se pudo abrir el archivo " + ubicacion)
            )
            self.ui.plainTextEdit.clear()
            self.ui.plainTextEdit.insertPlainText(str(self.__lista))
        else:
            QMessageBox.critical(
                self,
                "Error",
                ("No pudo abrir el archivo " + ubicacion)
            )

    @ Slot()
    def click_agregar_inicio(self):
        self.__lista.agregar_inicio(self.procesarParticula())
        self.__contador += 1

    @ Slot()
    def click_agregar_final(self):
        self.__lista.agregar_final(self.procesarParticula())
```

```
        self.__contador += 1

    @ Slot()
    def mostrar(self):
        self.ui.plainTextEdit.clear()
        self.ui.plainTextEdit.insertPlainText(str(self.__lista))

    def procesarParticula(self):
        """ id, origen_x, origen_y, destino_x, destino_y, veloicidad, red, green, blue,
distancia """
        return Particula(self.__contador,
                         self.ui.spnnOrigenX.value(),
                         self.ui.spnnOrigenY.value(),
                         self.ui.spnnDestinoX.value(),
                         self.ui.spnnDestinoY.value(),
                         self.ui.spnnVelocidad.text(),
                         self.ui.spnnRed.value(),
                         self.ui.spnnBlue.value(),
                         self.ui.spnnGreen.value(),
                         self.ui.spnnDistancia.value())
```

Código de archivo **listaParticula.py.**

```
import json
from Particula import Particula


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

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

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

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

    def __str__(self):
        return "".join(
            str(particula) 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]

                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
```

Código de archivo **Particula.py.**

```
from algoritmos import distancia_euclidiana


class Particula(object):
    __id = 0
    __origen_x = 0
    __origen_y = 0
    __destino_x = 0
    __destino_y = 0
    __veloicidad = 0
    __red = 0
    __green = 0
    __blue = 0
    __distancia = 0.0

    def __init__(self, id, origen_x, origen_y, destino_x, destino_y, veloicidad, red,
green, blue, distancia):
        """ Propiedades de la clase """
        self.__id = id
```

```
        self.__origen_x = origen_x
        self.__origen_y = origen_y
        self.__destino_x = destino_x
        self.__destino_y = destino_y
        self.__veloicidad = veloicidad
        self.__red = red
        self.__green = green
        self.__blue = blue
        self.__distancia = distancia
        """ Calculo de la distancia euclidiana """
        self.__distancia = distancia_euclidiana(
            origen_x, origen_y, destino_x, destino_y)

    def __str__(self):
        return (
            "#####################################\n"
            + "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.__veloicidad) + ",\n"
            + "Rojo: " + str(self.__red) + ",\n"
            + "Verde: " + str(self.__green) + ",\n"
            + "Azul: " + str(self.__blue) + ",\n"
            + "Distancia: " + str(self.__distancia))

    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,
            "veloicidad": self.__veloicidad,
            "red": self.__red,
            "green": self.__green,
            "blue": self.__blue,
            "distancia": + self.__distancia
        }
```

Código de archivo **ui_mainwindow.py**

```
# -*- coding: utf-8 -*-
```

```
################################################################################
## Form generated from reading UI file 'mainwindow.ui'
##
## Created by: Qt User Interface Compiler version 5.15.2
##
## WARNING! All changes made in this file will be lost when recompiling UI file!
################################################################################

from PySide2.QtCore import *
from PySide2.QtGui import *
from PySide2.QtWidgets import *


class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        if not MainWindow.objectName():
            MainWindow.setObjectName(u"MainWindow")
        MainWindow.resize(699, 430)
        self.actionAbrir = QAction(MainWindow)
        self.actionAbrir.setObjectName(u"actionAbrir")
        self.actionGuardar = QAction(MainWindow)
        self.actionGuardar.setObjectName(u"actionGuardar")
        self.centralwidget = QWidget(MainWindow)
        self.centralwidget.setObjectName(u"centralwidget")
        self.groupBox = QGroupBox(self.centralwidget)
        self.groupBox.setObjectName(u"groupBox")
        self.groupBox.setGeometry(QRect(0, 0, 281, 381))
        self.formLayout = QFormLayout(self.groupBox)
        self.formLayout.setObjectName(u"formLayout")
        self.label = QLabel(self.groupBox)
        self.label.setObjectName(u"label")

        self.formLayout.setWidget(4, QFormLayout.LabelRole, self.label)

        self.spnnDestinoX = QSpinBox(self.groupBox)
        self.spnnDestinoX.setObjectName(u"spnnDestinoX")
        self.spnnDestinoX.setMaximum(500)

        self.formLayout.setWidget(4, QFormLayout.FieldRole, self.spnnDestinoX)

        self.label_2 = QLabel(self.groupBox)
        self.label_2.setObjectName(u"label_2")

        self.formLayout.setWidget(5, QFormLayout.LabelRole, self.label_2)
```

```
        self.spnnDestinoY = QSpinBox(self.groupBox)
        self.spnnDestinoY.setObjectName(u"spnnDestinoY")
        self.spnnDestinoY.setMaximum(500)

        self.formLayout.setWidget(5, QFormLayout.FieldRole, self.spnnDestinoY)

        self.label_3 = QLabel(self.groupBox)
        self.label_3.setObjectName(u"label_3")

        self.formLayout.setWidget(6, QFormLayout.LabelRole, self.label_3)

        self.label_5 = QLabel(self.groupBox)
        self.label_5.setObjectName(u"label_5")

        self.formLayout.setWidget(7, QFormLayout.LabelRole, self.label_5)

        self.spnnRed = QSpinBox(self.groupBox)
        self.spnnRed.setObjectName(u"spnnRed")
        self.spnnRed.setMaximum(255)

        self.formLayout.setWidget(7, QFormLayout.FieldRole, self.spnnRed)

        self.label_6 = QLabel(self.groupBox)
        self.label_6.setObjectName(u"label_6")

        self.formLayout.setWidget(8, QFormLayout.LabelRole, self.label_6)

        self.spnnGreen = QSpinBox(self.groupBox)
        self.spnnGreen.setObjectName(u"spnnGreen")
        self.spnnGreen.setMaximum(255)

        self.formLayout.setWidget(8, QFormLayout.FieldRole, self.spnnGreen)

        self.label_7 = QLabel(self.groupBox)
        self.label_7.setObjectName(u"label_7")

        self.formLayout.setWidget(9, QFormLayout.LabelRole, self.label_7)

        self.spnnBlue = QSpinBox(self.groupBox)
        self.spnnBlue.setObjectName(u"spnnBlue")
        self.spnnBlue.setMaximum(255)

        self.formLayout.setWidget(9, QFormLayout.FieldRole, self.spnnBlue)
```

```
        self.btnAgregarFinal = QPushButton(self.groupBox)
        self.btnAgregarFinal.setObjectName(u"btnAgregarFinal")
        self.btnAgregarFinal.setCursor(QCursor(Qt.PointingHandCursor))

        self.formLayout.setWidget(13, QFormLayout.SpanningRole, self.btnAgregarFinal)

        self.label_4 = QLabel(self.groupBox)
        self.label_4.setObjectName(u"label_4")

        self.formLayout.setWidget(0, QFormLayout.LabelRole, self.label_4)

        self.spnnOrigenX = QSpinBox(self.groupBox)
        self.spnnOrigenX.setObjectName(u"spnnOrigenX")
        self.spnnOrigenX.setMaximum(500)

        self.formLayout.setWidget(0, QFormLayout.FieldRole, self.spnnOrigenX)

        self.spnnOrigenY = QSpinBox(self.groupBox)
        self.spnnOrigenY.setObjectName(u"spnnOrigenY")
        self.spnnOrigenY.setMaximum(500)

        self.formLayout.setWidget(2, QFormLayout.FieldRole, self.spnnOrigenY)

        self.label_8 = QLabel(self.groupBox)
        self.label_8.setObjectName(u"label_8")

        self.formLayout.setWidget(2, QFormLayout.LabelRole, self.label_8)

        self.spnnVelocidad = QSpinBox(self.groupBox)
        self.spnnVelocidad.setObjectName(u"spnnVelocidad")
        self.spnnVelocidad.setMaximum(500)

        self.formLayout.setWidget(6, QFormLayout.FieldRole, self.spnnVelocidad)

        self.btnAgregarInicio = QPushButton(self.groupBox)
        self.btnAgregarInicio.setObjectName(u"btnAgregarInicio")
        self.btnAgregarInicio.setCursor(QCursor(Qt.PointingHandCursor))

        self.formLayout.setWidget(14, QFormLayout.SpanningRole, self.btnAgregarInicio)

        self.btnMostrar = QPushButton(self.groupBox)
        self.btnMostrar.setObjectName(u"btnMostrar")
        self.btnMostrar.setCursor(QCursor(Qt.PointingHandCursor))

        self.formLayout.setWidget(15, QFormLayout.SpanningRole, self.btnMostrar)
```

```
        self.label_9 = QLabel(self.groupBox)
        self.label_9.setObjectName(u"label_9")

        self.formLayout.setWidget(10, QFormLayout.LabelRole, self.label_9)

        self.spnnDistancia = QSpinBox(self.groupBox)
        self.spnnDistancia.setObjectName(u"spnnDistancia")
        self.spnnDistancia.setMaximum(255)

        self.formLayout.setWidget(10, QFormLayout.FieldRole, self.spnnDistancia)

        self.plainTextEdit = QPlainTextEdit(self.centralwidget)
        self.plainTextEdit.setObjectName(u"plainTextEdit")
        self.plainTextEdit.setGeometry(QRect(280, 10, 411, 361))
        MainWindow.setCentralWidget(self.centralwidget)
        self.menubar = QMenuBar(MainWindow)
        self.menubar.setObjectName(u"menubar")
        self.menubar.setGeometry(QRect(0, 0, 699, 26))
        self.menuArchivo = QMenu(self.menubar)
        self.menuArchivo.setObjectName(u"menuArchivo")
        MainWindow.setMenuBar(self.menubar)
        self.statusbar = QStatusBar(MainWindow)
        self.statusbar.setObjectName(u"statusbar")
        MainWindow.setStatusBar(self.statusbar)

        self.menubar.addAction(self.menuArchivo.menuAction())
        self.menuArchivo.addAction(self.actionAbrir)
        self.menuArchivo.addAction(self.actionGuardar)

        self.retranslateUi(MainWindow)

        QMetaObject.connectSlotsByName(MainWindow)
    # setupUi

    def retranslateUi(self, MainWindow):
        MainWindow.setWindowTitle(QCoreApplication.translate("MainWindow",
u"MainWindow", None))
        self.actionAbrir.setText(QCoreApplication.translate("MainWindow", u"Abrir",
None))
#if QT_CONFIG(shortcut)
        self.actionAbrir.setShortcut(QCoreApplication.translate("MainWindow",
u"Ctrl+O", None))
#endif // QT_CONFIG(shortcut)
```

```
        self.actionGuardar.setText(QCoreApplication.translate("MainWindow", u"Guardar",
None))
#if QT_CONFIG(shortcut)
        self.actionGuardar.setShortcut(QCoreApplication.translate("MainWindow",
u"Ctrl+G", None))
#endif // QT_CONFIG(shortcut)
        self.groupBox.setTitle(QCoreApplication.translate("MainWindow", u"GroupBox",
None))
        self.label.setText(QCoreApplication.translate("MainWindow", u"Destino X",
None))
        self.label_2.setText(QCoreApplication.translate("MainWindow", u"Destino Y",
None))
        self.label_3.setText(QCoreApplication.translate("MainWindow", u"Velocidad",
None))
        self.label_5.setText(QCoreApplication.translate("MainWindow", u"Red", None))
        self.label_6.setText(QCoreApplication.translate("MainWindow", u"Green", None))
        self.label_7.setText(QCoreApplication.translate("MainWindow", u"Blue", None))
        self.btnAgregarFinal.setText(QCoreApplication.translate("MainWindow", u"Agregar
final", None))
        self.label_4.setText(QCoreApplication.translate("MainWindow", u"Origen X:",
None))
        self.label_8.setText(QCoreApplication.translate("MainWindow", u"Origen Y: ",
None))
        self.btnAgregarInicio.setText(QCoreApplication.translate("MainWindow",
u"Agregar inicio", None))
        self.btnMostrar.setText(QCoreApplication.translate("MainWindow", u"Mostrar",
None))
        self.label_9.setText(QCoreApplication.translate("MainWindow", u"Distancia:",
None))
        self.menuArchivo.setTitle(QCoreApplication.translate("MainWindow", u"Archivo",
None))
    # retranslateUi
```

Subida a **git hub.**

```
Armando@Armando04 MINGW64 ~/Documents/GitHub/Actividad 7/Actividad-7 (main)
$ git add .

Armando@Armando04 MINGW64 ~/Documents/GitHub/Actividad 7/Actividad-7 (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   src/Main.py
        new file:   src/Particula.py
        new file:   src/__pycache__/Particula.cpython-310.pyc
        new file:   src/__pycache__/algoritmos.cpython-310.pyc
        new file:   src/__pycache__/listaParticulas.cpython-310.pyc
```

```
        new file:   src/__pycache__/mainwindow.cpython-310.pyc
        new file:   src/__pycache__/ui_mainwindow.cpython-310.pyc
        new file:   src/algoritmos.py
        new file:   src/listaParticulas.py
        new file:   src/mainwindow.py
        new file:   src/mainwindow.ui
        new file:   src/ui_mainwindow.py

Armando@Armando04 MINGW64 ~/Documents/GitHub/Actividad 7/Actividad-7 (main)
$ git commit -m "Subida actividad 7"
[main 34ba856] Subida actividad 7
 12 files changed, 648 insertions(+)
 create mode 100644 src/Main.py
 create mode 100644 src/Particula.py
 create mode 100644 src/__pycache__/Particula.cpython-310.pyc
 create mode 100644 src/__pycache__/algoritmos.cpython-310.pyc
 create mode 100644 src/__pycache__/listaParticulas.cpython-310.pyc
 create mode 100644 src/__pycache__/mainwindow.cpython-310.pyc
 create mode 100644 src/__pycache__/ui_mainwindow.cpython-310.pyc
 create mode 100644 src/algoritmos.py
 create mode 100644 src/listaParticulas.py
 create mode 100644 src/mainwindow.py
 create mode 100644 src/mainwindow.ui
 create mode 100644 src/ui_mainwindow.py

Armando@Armando04 MINGW64 ~/Documents/GitHub/Actividad 7/Actividad-7 (main)
$ git push
Enumerating objects: 17, done.
Counting objects: 100% (17/17), done.
Delta compression using up to 4 threads
Compressing objects: 100% (16/16), done.
Writing objects: 100% (16/16), 10.50 KiB | 2.10 MiB/s, done.
Total 16 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Diego-Armando-H/Actividad-7
   685cadb..34ba856  main -> main
```