Actividad 07 – (QFileDialog)

Rubio Valenzuela Miguel Angel - 216567795

Seminario de Algoritmia – D02.

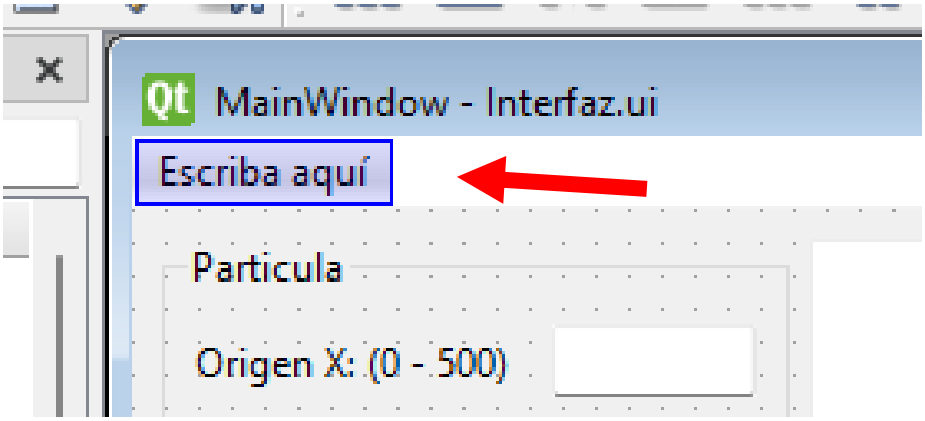
Lineamientos de evaluación

- [ ] 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 el respaldo.

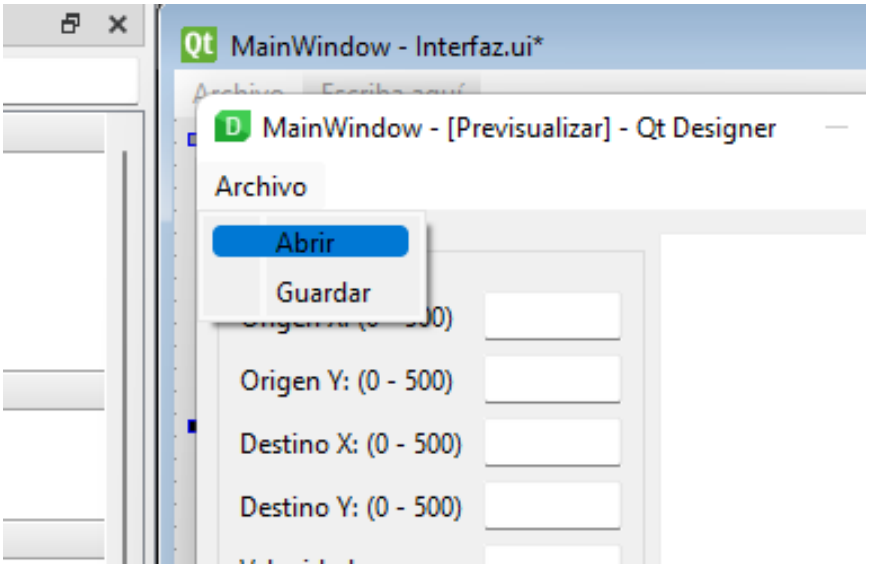
Desarrollo.

Lo primero que realicé fue agregarle a la interfaz las nuevas cajas de información para origen x y origen y, para así poder agregarle estos apartados a el backup.

Lo primero que realice para el desarrollo de esta actividad es la caja de texto, en esta caja, lo que tenemos que realizar es ingresar un nombre en la sección que nos dice escriba aquí en la interfaz de usuario:

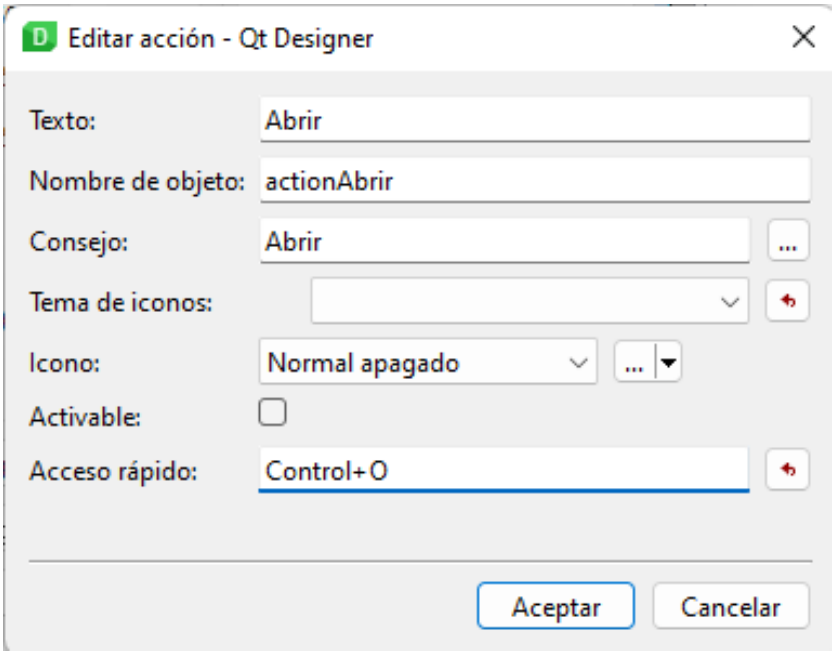


Una vez que damos doble click ahí, se agrega el nombre que deseemos darle, y pues en mi caso, al igual que el video del profe lo tiene, lo llamé con el nombre de Archivo, donde vamos a desplegar dos funciones, guardar y abrir.



Podemos ver aquí, como es que quedaría este apartado, poniendo dos acciones, la de abrir y la de guardar y ahora lo que sigue es la configuración de dichas acciones.

Al igual que el profe hacemos que la tecla de acceso a la acción abrir tenga Ctrl+O y para guardar voy a poner la tecla de acceso directo Crtl+s y de esta manera vamos a realizar un acceso mucho más rápido a las acciones establecidas.



Y con estas modificaciones solamente tenemos que cargarlas en el programa y ya estarían implementadas en nuestra interfaz.

Pero, aunque estos cambios estén realizados, el problema es que estos cambios aun no tienen funcionalidad, por lo que tenemos que darles funcionalidad.

Para esto tenemos que crear los slots en el programa y darles una funcionalidad en la consola.

Para esto primero debemos darle funcionalidad imprimiendo datos que nosotros queramos que se ejecuten, por lo que quedaría de la siguiente forma:

```
@Slot()
def action_abrir_archivo(self):
    print("Abriendo archivo")

@Slot()
def action_guardar_archivo(self):
    print("Guardando archivo")
```

Y este es el resultado una ves que ingresamos las teclas Crtl+S y Ctrl+O en la interfaz.

```
PS C:\Users\cober\Desktop\Sem de Algoritmia\A7>  c:; cd 'c:\Users\cc
ritmia\A7'; & 'C:\Users\cober\AppData\Local\Programs\Python\Python31
\cober\.vscode\extensions\ms-python.python-2022.16.1\pythonFiles\lib
/../..\debugpy\launcher' '52658' '--' 'c:\Users\cober\Desktop\Sem de
rface.py'
Guardando archivo
Abriendo archivo
```

Por lo que nosotros tenemos como resultado que nuestra funcionalidad si esta implementada en el programa.

Ahora lo que tenemos que realizar son los algoritmos de guardado y de respaldo de los datos.

Para esto tenemos que usar QFileDialog.getSaveFileName()

Este método pide 4 parámetros, primeramente, el self se mantiene de igual forma en la que esta establecida, el segundo parámetro es el titulo que llevara este, el tercero es la dirección donde se va a realizar el guardado y el cuarto es la extensión o el formato del archivo.

Pero en sí lo que realiza este método es la dirección y el nombre en donde nosotros realizaremos el guardado en otro sitio, por lo que de momento su estructura quedaría así.

```python
@Slot()
def action_guardar_archivo(self):
    #print("Guardando archivo")
    QFileDialog.getSaveFileName(
        self,
        "Guardar como:",
        ".",
        "JSON (*.json)"
    )
```

Y esto, como se dijo antes solo nos da la dirección donde se guardará con la extensión.

Una vez que imprimimos la dirección de nuestro archivo de guardado, podemos apreciar que también le agrega el formato:

```
('C:/Users/cober/Desktop/Sem de Algoritmia/A7/Guardado.json', 'JSON (*.json)')
```

Ahora lo que tenemos que hacer es la creación del archivo y ahí vamos a ingresarle todos los datos.

Usando las siguientes líneas de código, podemos hacer que se escriban los datos en forma de texto.

Esta es la función.

```
def guardar(self, direccion):
    with open(direccion, 'w') as archivo:
        archivo.write(str(self))
```

E ingresando los datos este es el resultado:

Creamos un archivo llamado, Guardado.json

Guardado.json

```
ID: 2
Origen X: 2.0
Origen Y: 2.0
Destino X: 2.0
Destino Y: 2.0
Velocidad: 2.0 M/S
Rojo: 2.0
Verde: 2.0
Azul: 2.0
Distancia: 0.0 M

ID: 1
Origen X: 1.0
Origen Y: 1.0
Destino X: 1.0
Destino Y: 1.0
Velocidad: 1.0 M/S
Rojo: 1.0
Verde: 1.0
Azul: 1.0
Distancia: 0.0 M

```

Ahora lo que tenemos que hacer, es convertir estos libros en un diccionario para poder ingresar los datos en el formato JSON.

Antes que nada debemos de añadir la función to_dict(self).

```
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
    }
```

Para esto tenemos que usar este código el cual nos permite hacer una lista con el diccionario para ingresarlo en formato JSON, y usamos json.dump para ingresar estos datos.

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

Luego de tener esto, tenemos que validar si la información que nosotros se manda de forma correcta, y para esto vamos a mandar un mensaje cuando se pudo realizar el guardado con la función QMessageBox.Information().

```
if self.lista.guardar(dir):
    QMessageBox.information(
        self,
        "Éxito",
        "Se pudo crear y guardar datos del archivo " + dir
    )
else:
    QMessageBox.critical(
        self,
        "Error",
        "No se pudo crear y/o guardar datos en el archivo " + dir
    )
```

Ahora lo que nos falta por hacer es la implementación de la recuperación de nuestro archivo de guardado.

Al igual que con la función guardar tenemos que poner condiciones de como es que va a funcionar el programa.

```
@Slot()
def action_abrir_archivo(self):
    dir = QFileDialog.getOpenFileName(
        self,
        "Abrir Archivo:",
        ".",
        "JSON (*.json)"
    )[0]
    if self.lista.abrir(dir):
        QMessageBox.information(
            self,
            "Exito",
            "Se abrió el archivo " + dir
        )
    else:
        QMessageBox.critical(
            self,
            "Error",
            "Error al abrir el archivo " + dir
        )
```

Esa es la manera en que funciona, se obtiene la dirección donde se va a leer el archivo y únicamente leerá archivos tipo JSON, y nos dará una ventana si si funciona y otra si no lo hace.

Luego, si funciona es porque manda a llamar a la función abrir mandándonos la dirección a donde mandaremos el archivo.

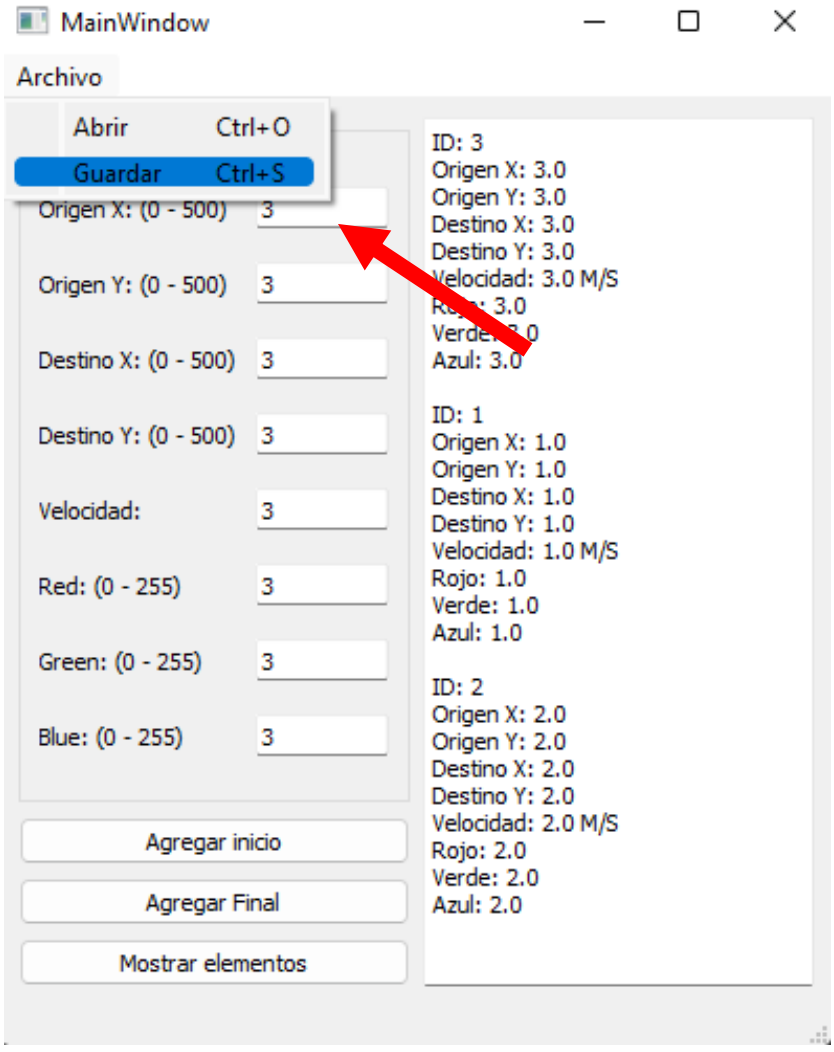
```
def abrir(self, direccion):
    try:
        with open(direccion, 'r') as archivo:
            lista = json.load(archivo)
            self.__particulas = [Particula(**particula) for particula in lista]
        return 1
    except:
        return 0
```

Lo que hace la función anterior es esto: abre el archivo en la dirección donde se le paso con anterioridad en el modo de lectura ‘r’ Read, y se le asigna el nombre de archivo, luego, se cargan los datos del archivo.json a una lista y estos están ingresados por objetos de tipo lista. Luego a la lista de partículas que tiene la clase Lista, le ingresaremos lo siguiente: de toda la información con tamaño de objeto partícula dentro de toda la lista, le ingresaremos la información que tiene dicho objeto referenciado por ** de un objeto tipo partícula.

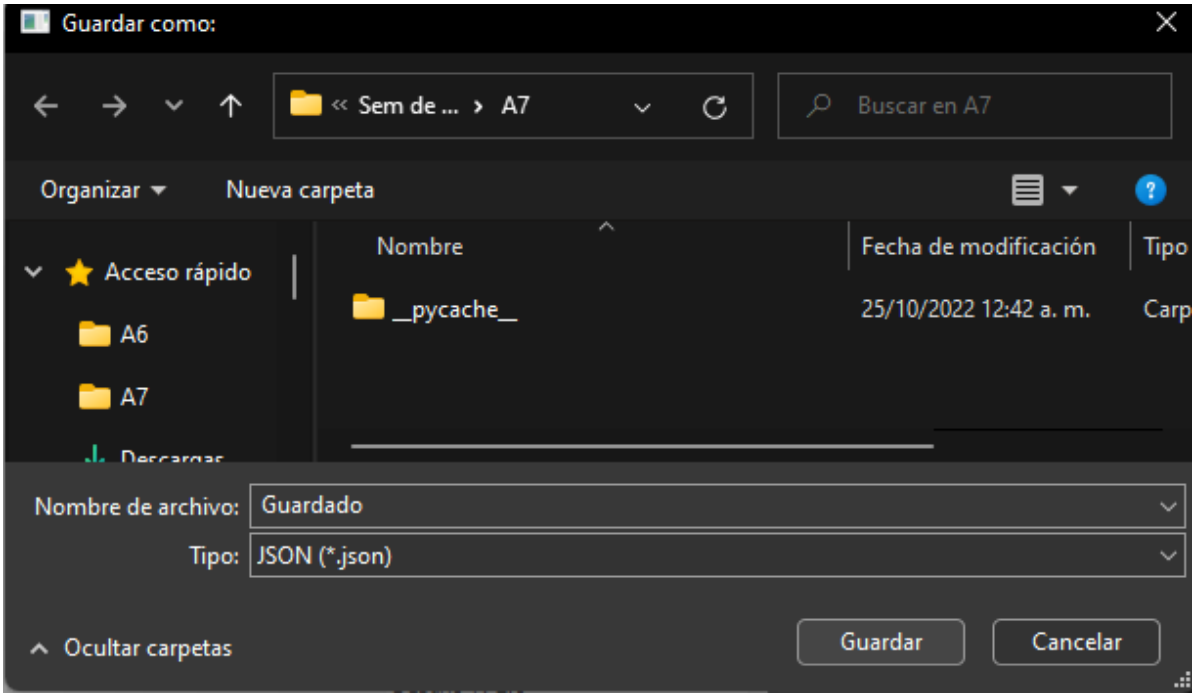
Gracias a esto, nosotros podemos realizar un respaldo entre muchas comillas, porque lo que hace en realidad es abrir un archivo con esa información.

Ahora que se encuentra completo el programa, vamos a mostrar cómo es su funcionamiento.

Primero, vamos a ingresar 3 partículas y crear un respaldo:

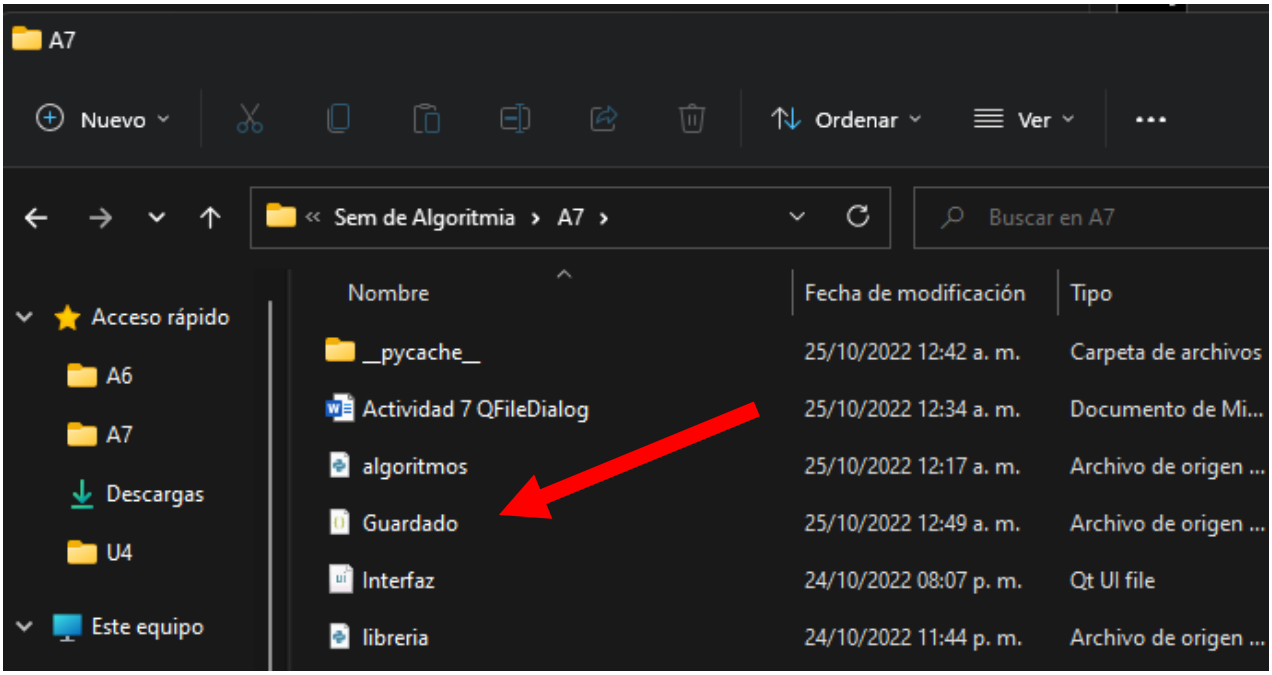


Ahí podemos ver los 3 registros de partículas, únicamente si, tuve que quitar el otro dato de la distancia para que funcionase el programa y además ingrese en la opción de guardar.

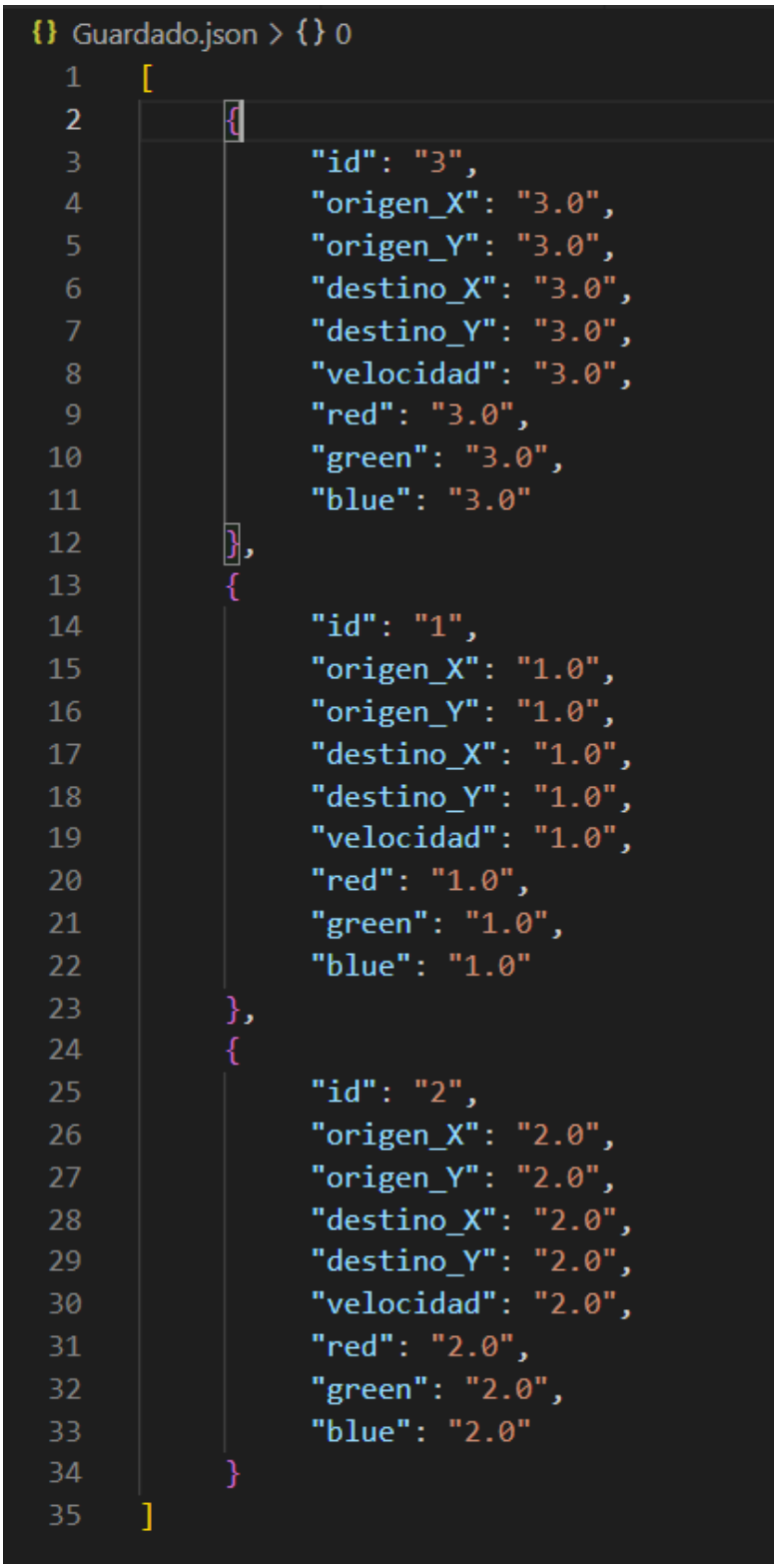


En la carpeta de C:\Users\cober\Desktop\Sem de Algoritmia\A7, cree un archivo llamado guardado.json, donde guardaré los datos de las 3 partículas.

Aquí podemos ver el archivo generado.

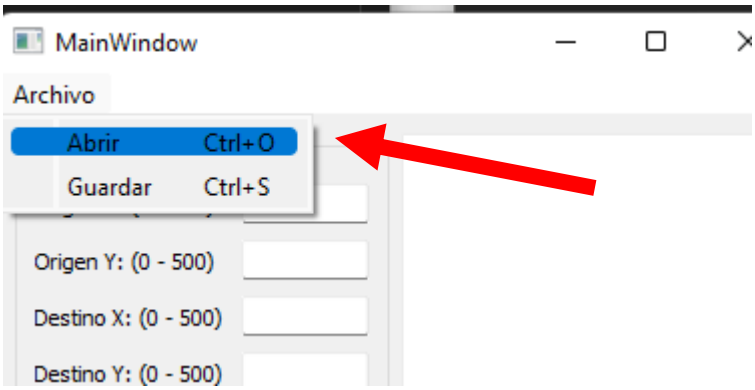


Y una ves abrimos el archivo, esta es la información que nosotros tenemos.

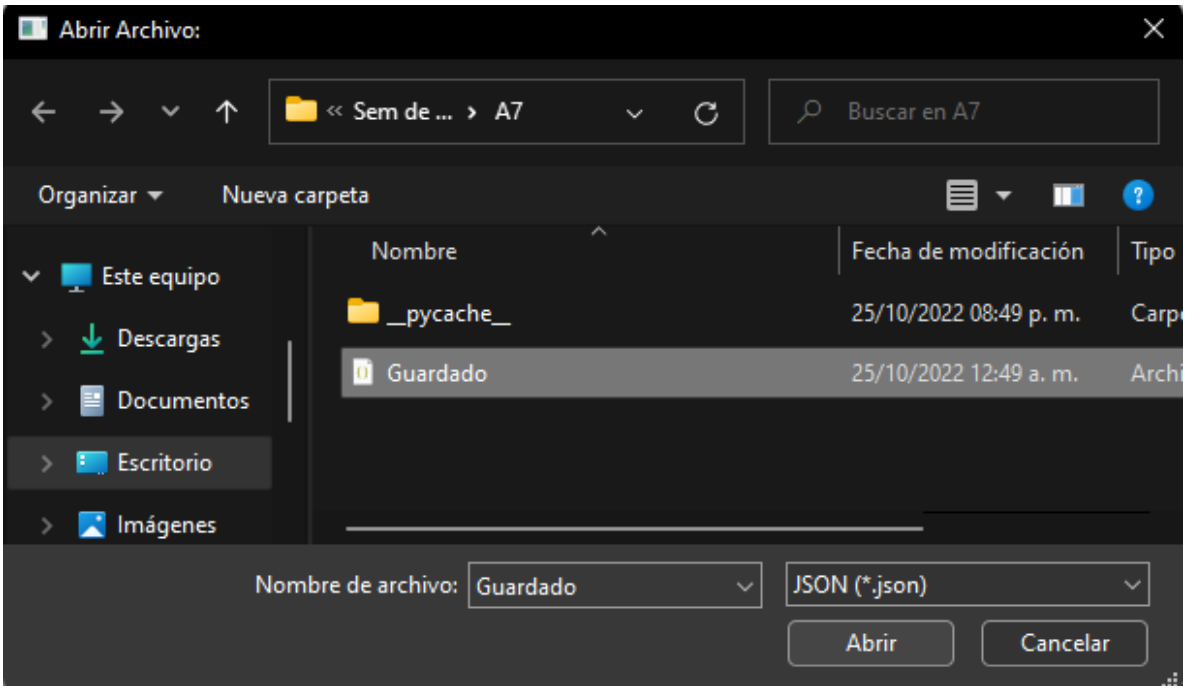


Aquí podemos ver las 3 partículas que ingresamos con anterioridad.

Ahora lo que tenemos que hacer es abrir el archivo con la información

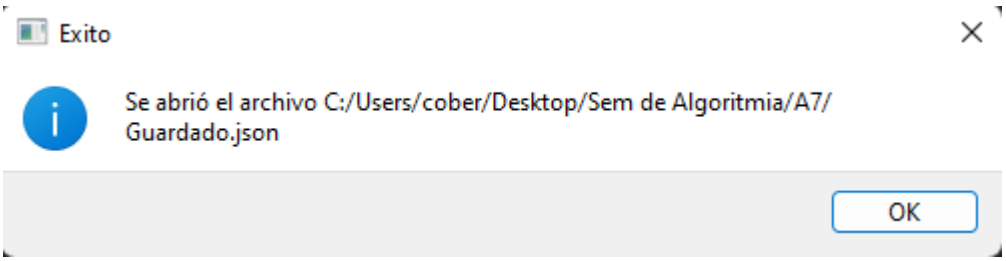


Seguido de esto, tenemos que ingresar a la carpeta en donde nosotros guardamos la información con la opción de guardado.

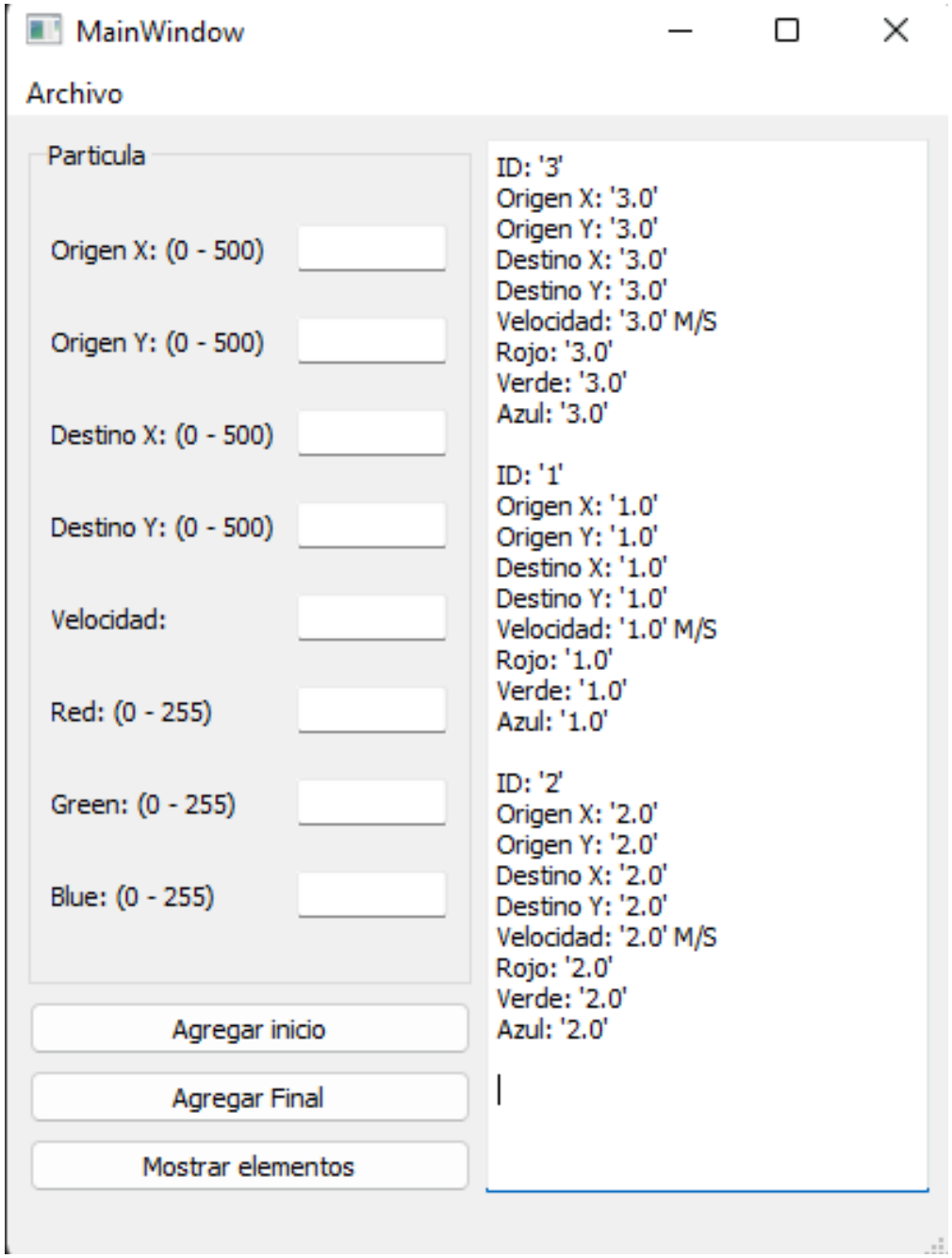


Aquí podemos ver que nosotros vamos a abrir un archivo que se encuentra en la carpeta A7 y esto accediendo a un archivo .json.

Al darle a abrir el programa nos manda este siguiente mensaje:



Después de esto, nosotros le damos Ok, y nos dirigimos en el apartado de mostrar elementos, en donde podemos encontrar que los datos que se habían ingresado con anterioridad se abrieron de la misma forma, que como fueron introducidos.



Conclusiones

Al momento de desarrollar esta actividad no tuve muchos problemas en realidad, esto porque siguiendo todos los pasos que el profesor nos presenta en el video uno puede desarrollar el trabajo sin muchas complicaciones, además de aprender mucho más con bastante facilidad.

El único problema que se me presento fue que cuando se realizaba la lectura del archivo, este no podía realizarse a menos que la variable de distancia no se ejecutara en tiempo real, supongo que es porque la manera de ingresar a la información al momento de abrir el archivo json, este necesita que el valor de distancia se encuentre, sino no podrá abrirlo, traté de buscar de muchas formas que la variable distancia se calculara en tiempo real pero el programa no me permitía abrir el archivo json si es que se tenia que hacer el calculo al momento de ingresar dichos datos.

Además de ese problema, se me presentó el problema de no haber llamado a las variables con el mismo nombre al momento de ingresarlas en el constructo, y esto hizo que tuviese que llamar a las variables del mismo nombre en los parámetros que a los datos de la clase.

De ahí en más, no tuve mayores complicaciones al momento de hacer la actividad.

Lo importante de conocer como es el funcionamiento de este tipo de programas, es que no solamente estamos ingresando y respaldando información de un archivo, sino que estamos creando un archivo con una extensión diferente a la txt y conociendo esto, podemos conocer cómo es que accedemos a diferente información de diferentes tipos de archivos, pero, conociendo como es la manera de obtener esa información.

Referencias

https://www.youtube.com/watch?v=HRY8QvXmcDM – Davalos Boites Michel – PySide2 – QfileDialog (Qt for Python) (V)

Código.

Para el código de este programa, tenemos 5 archivos con código, los cuales son los siguientes:

1. **UserInterface.py**

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

#Se crea la aplicacion de QT
app = QApplication()

#Se crea un botton con la palabra hola
window = MainWindow()

#Se hace visible el boton.
window.show()

#QT Loop
sys.exit(app.exec_())
```

2. **Mainwindow.py**

```
from PySide2.QtWidgets import QMainWindow, QFileDialog, QMessageBox
from PySide2.QtCore import Slot
from ui_interfaz import Ui_MainWindow
from libreria import Lista
from particula import Particula

class MainWindow(QMainWindow):

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

        self.lista = Lista()
        self.id = int(0)
        self.ui = Ui_MainWindow()
        self.ui.setupUi(self)
        self.ui.Agregar_inicio_pushButton.clicked.connect(self.click_agregar
_inicio)
        self.ui.Agregar_final_pushButton.clicked.connect(self.click_agregar_
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):
        dir = QFileDialog.getOpenFileName(
            self,
            "Abrir Archivo:",
            ".",
            "JSON (*.json)"
        )[0]
        if self.lista.abrir(dir):
            QMessageBox.information(
                self,
                "Exito",
                "Se abrió el archivo " + dir
            )
        else:
            QMessageBox.critical(
                self,
                "Error",
                "Error al abrir el archivo " + dir
            )

    @Slot()
    def action_guardar_archivo(self):
        #print("Guardando archivo")
        dir = QFileDialog.getSaveFileName(
            self,
            "Guardar como:",
            ".",
            "JSON (*.json)"
        )[0]

        if self.lista.guardar(dir):
            QMessageBox.information(
                self,
                "Éxito",
                "Se pudo crear y guardar datos del archivo " + dir
            )
        else:
            QMessageBox.critical(
                self,
                "Error",
                "No se pudo crear y/o guardar datos en el archivo " + dir
```

```
            )

    @Slot()
    def click_agregar_inicio(self):
        self.id = self.id + int(1)
        origenx = float(self.ui.Origen_X_lineEdit.text())
        origeny = float(self.ui.Origen_Y_lineEdit.text())
        destinox = float(self.ui.DestinoX_lineEdit.text())
        destinoy = float(self.ui.DestinoY_lineEdit.text())
        velocidad = float(self.ui.Velocidad_lineEdit.text())
        red = float(self.ui.Red_lineEdit.text())
        green = float(self.ui.Green_lineEdit.text())
        blue = float(self.ui.Blue_lineEdit.text())

        particula = Particula(self.id, origenx, origeny, destinox, destinoy,
velocidad, red, green, blue)

        self.lista.agregar_inicio(particula)

    @Slot()
    def click_agregar_final(self):
        self.id = self.id + int(1)
        origenx = float(self.ui.Origen_X_lineEdit.text())
        origeny = float(self.ui.Origen_Y_lineEdit.text())
        destinox = float(self.ui.DestinoX_lineEdit.text())
        destinoy = float(self.ui.DestinoY_lineEdit.text())
        velocidad = float(self.ui.Velocidad_lineEdit.text())
        red = float(self.ui.Red_lineEdit.text())
        green = float(self.ui.Green_lineEdit.text())
        blue = float(self.ui.Blue_lineEdit.text())

        particula = Particula(self.id, origenx, origeny, destinox, destinoy,
velocidad, red, green, blue)

        self.lista.agregar_final(particula)

        #self.ui.Salida.insertPlainText()

    @Slot()
    def click_mostrar(self):

        self.ui.Salida.clear()
        self.ui.Salida.insertPlainText(str(self.lista))
```

3. **Particula.py**

```
class Particula:
    def __init__(self, id = 0.0, origen_X= 0.0, origen_Y= 0.0,
                 destino_X= 0.0, destino_Y= 0.0, velocidad= 0.0,
                 red= 0.0, green= 0.0, blue= 0.0):
        self.__id = repr(id)
        self.__origen_x = repr(origen_X)
        self.__origen_y = repr(origen_Y)
        self.__destino_x = repr(destino_X)
        self.__destino_y = repr(destino_Y)
        self.__velocidad = repr(velocidad)
        self.__red = repr(red)
        self.__green = repr(green)
        self.__blue = repr(blue)
        #self.__distancia = repr(distancia_euclidiana(origen_X, origen_Y,
destino_X, destino_Y))

    def __str__(self):
        return(
            'ID: ' + self.__id + '\n' +
            'Origen X: ' + self.__origen_x + '\n' +
            'Origen Y: ' + self.__origen_y + '\n' +
            'Destino X: ' + self.__destino_x + '\n' +
            'Destino Y: ' + self.__destino_y + '\n' +
            'Velocidad: ' + self.__velocidad + ' M/S \n' +
            'Rojo: ' + self.__red + '\n' +
            'Verde: ' + self.__green + '\n' +
            'Azul: ' + self.__blue + '\n'
            #'Distancia: ' + self.__distancia +  ' M \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
        }
```

4. **Ui_interfaz.py**

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

################################################################################
####
## Form generated from reading UI file 'interfaz.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(398, 405)
        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.gridLayout_2 = QGridLayout(self.centralwidget)
        self.gridLayout_2.setObjectName(u"gridLayout_2")
        self.Mostrar_pushButton = QPushButton(self.centralwidget)
        self.Mostrar_pushButton.setObjectName(u"Mostrar_pushButton")

        self.gridLayout_2.addWidget(self.Mostrar_pushButton, 4, 0, 1, 1)

        self.Agregar_inicio_pushButton = QPushButton(self.centralwidget)
        self.Agregar_inicio_pushButton.setObjectName(u"Agregar_inicio_pushBu
tton")

        self.gridLayout_2.addWidget(self.Agregar_inicio_pushButton, 2, 0, 1,
1)
```

```
        self.groupBox = QGroupBox(self.centralwidget)
        self.groupBox.setObjectName(u"groupBox")
        self.gridLayout = QGridLayout(self.groupBox)
        self.gridLayout.setObjectName(u"gridLayout")
        self.Velocidad = QLabel(self.groupBox)
        self.Velocidad.setObjectName(u"Velocidad")

        self.gridLayout.addWidget(self.Velocidad, 4, 1, 1, 1)

        self.DestinoX_lineEdit = QLineEdit(self.groupBox)
        self.DestinoX_lineEdit.setObjectName(u"DestinoX_lineEdit")

        self.gridLayout.addWidget(self.DestinoX_lineEdit, 2, 3, 1, 2)

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

        self.gridLayout.addWidget(self.DestinoY, 3, 1, 1, 2)

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

        self.gridLayout.addWidget(self.DestinoX, 2, 1, 1, 2)

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

        self.gridLayout.addWidget(self.Origen_Y, 1, 1, 1, 1)

        self.DestinoY_lineEdit = QLineEdit(self.groupBox)
        self.DestinoY_lineEdit.setObjectName(u"DestinoY_lineEdit")

        self.gridLayout.addWidget(self.DestinoY_lineEdit, 3, 3, 1, 2)

        self.Red_lineEdit = QLineEdit(self.groupBox)
        self.Red_lineEdit.setObjectName(u"Red_lineEdit")

        self.gridLayout.addWidget(self.Red_lineEdit, 8, 3, 1, 2)

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

        self.gridLayout.addWidget(self.Green, 12, 1, 1, 1)
```

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

        self.gridLayout.addWidget(self.Red, 8, 1, 1, 1)

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

        self.gridLayout.addWidget(self.Blue, 14, 1, 1, 1)

        self.Green_lineEdit = QLineEdit(self.groupBox)
        self.Green_lineEdit.setObjectName(u"Green_lineEdit")

        self.gridLayout.addWidget(self.Green_lineEdit, 12, 3, 1, 2)

        self.Velocidad_lineEdit = QLineEdit(self.groupBox)
        self.Velocidad_lineEdit.setObjectName(u"Velocidad_lineEdit")

        self.gridLayout.addWidget(self.Velocidad_lineEdit, 4, 3, 1, 2)

        self.Blue_lineEdit = QLineEdit(self.groupBox)
        self.Blue_lineEdit.setObjectName(u"Blue_lineEdit")

        self.gridLayout.addWidget(self.Blue_lineEdit, 14, 3, 1, 2)

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

        self.gridLayout.addWidget(self.Origen_X, 0, 1, 1, 1)

        self.Origen_Y_lineEdit = QLineEdit(self.groupBox)
        self.Origen_Y_lineEdit.setObjectName(u"Origen_Y_lineEdit")

        self.gridLayout.addWidget(self.Origen_Y_lineEdit, 1, 3, 1, 2)

        self.Origen_X_lineEdit = QLineEdit(self.groupBox)
        self.Origen_X_lineEdit.setObjectName(u"Origen_X_lineEdit")

        self.gridLayout.addWidget(self.Origen_X_lineEdit, 0, 3, 1, 2)


        self.gridLayout_2.addWidget(self.groupBox, 1, 0, 1, 1)

        self.Agregar_final_pushButton = QPushButton(self.centralwidget)
```

```
        self.Agregar_final_pushButton.setObjectName(u"Agregar_final_pushButt
on")

        self.gridLayout_2.addWidget(self.Agregar_final_pushButton, 3, 0, 1,
1)

        self.Salida = QPlainTextEdit(self.centralwidget)
        self.Salida.setObjectName(u"Salida")

        self.gridLayout_2.addWidget(self.Salida, 1, 1, 4, 1)

        MainWindow.setCentralWidget(self.centralwidget)
        self.menubar = QMenuBar(MainWindow)
        self.menubar.setObjectName(u"menubar")
        self.menubar.setGeometry(QRect(0, 0, 398, 22))
        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("MainWindo
w", u"Ctrl+S", None))
```

```
#endif // QT_CONFIG(shortcut)
        self.Mostrar_pushButton.setText(QCoreApplication.translate("MainWind
ow", u"Mostrar elementos", None))
        self.Agregar_inicio_pushButton.setText(QCoreApplication.translate("M
ainWindow", u"Agregar inicio", None))
        self.groupBox.setTitle(QCoreApplication.translate("MainWindow",
u"Particula", None))
        self.Velocidad.setText(QCoreApplication.translate("MainWindow",
u"Velocidad:", None))
        self.DestinoY.setText(QCoreApplication.translate("MainWindow",
u"Destino Y: (0 - 500) ", None))
        self.DestinoX.setText(QCoreApplication.translate("MainWindow",
u"Destino X: (0 - 500) ", None))
        self.Origen_Y.setText(QCoreApplication.translate("MainWindow",
u"Origen Y: (0 - 500)", None))
        self.Green.setText(QCoreApplication.translate("MainWindow", u"Green:
(0 - 255)", None))
        self.Red.setText(QCoreApplication.translate("MainWindow", u"Red: (0
- 255)", None))
        self.Blue.setText(QCoreApplication.translate("MainWindow", u"Blue:
(0 - 255)", None))
        self.Origen_X.setText(QCoreApplication.translate("MainWindow",
u"Origen X: (0 - 500)", None))
        self.Agregar_final_pushButton.setText(QCoreApplication.translate("Ma
inWindow", u"Agregar Final", None))
        self.menuArchivo.setTitle(QCoreApplication.translate("MainWindow",
u"Archivo", None))
    # retranslateUi
```

5. **Librería.py**

```
from particula import Particula
import json

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