Actividad 08 - QTableWidget

Roberto Haro González

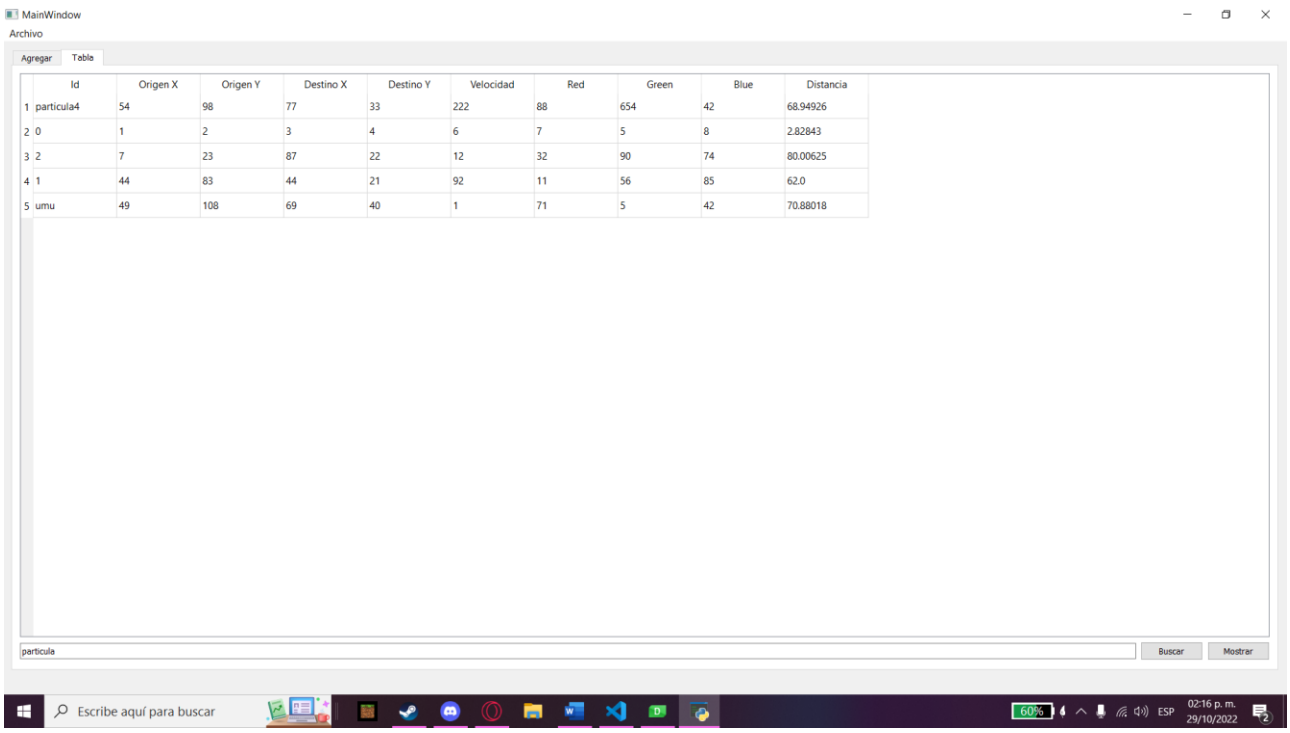
Seminario de solución de problemas de algoritmia

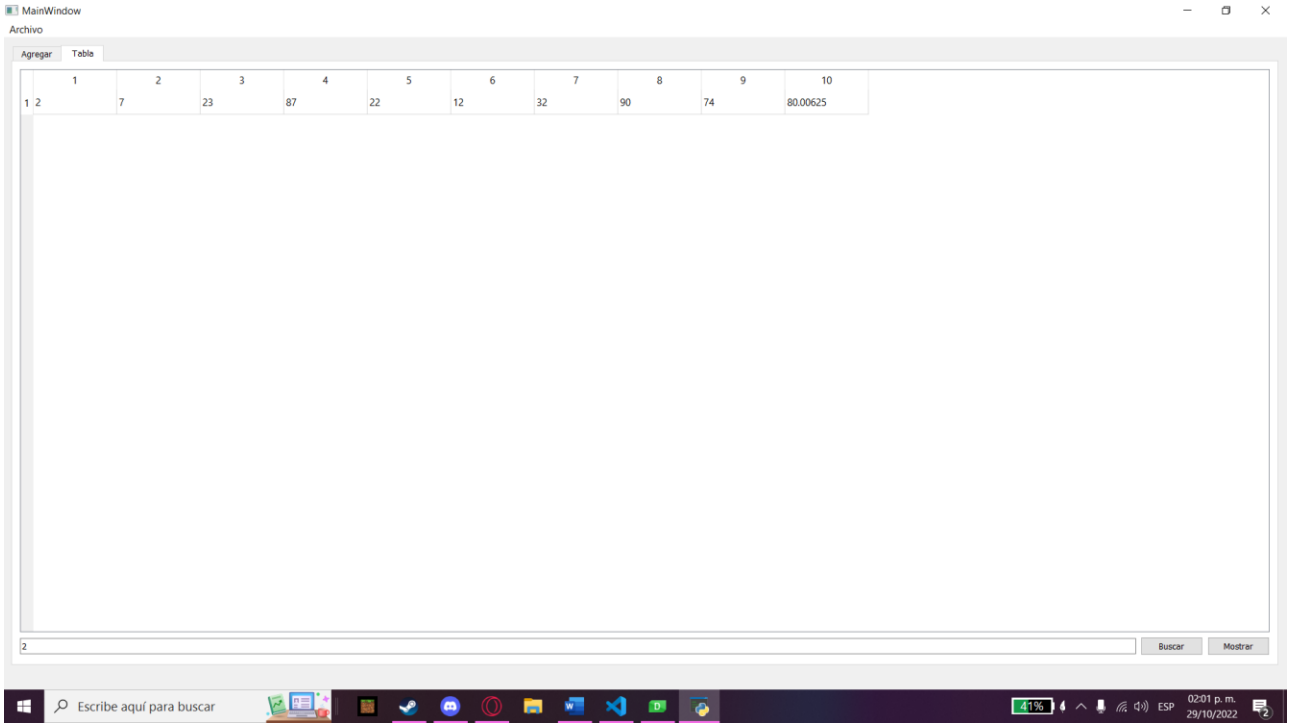
Lineamientos de evaluación

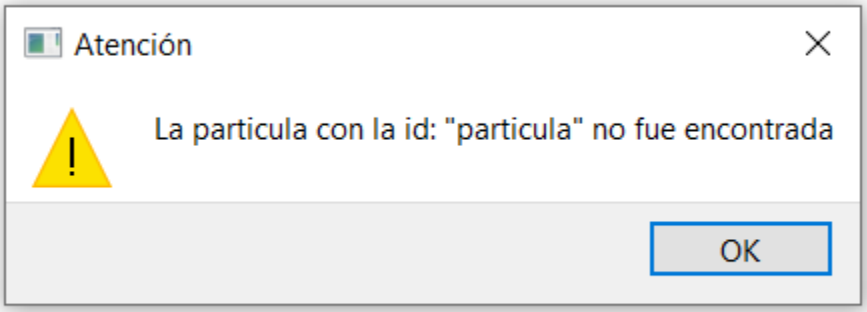
☑ El reporte está en formato Google Docs o PDF.
☑ El reporte sigue las pautas del Formato de Actividades.
☑ Se muestra captura de pantalla de lo que se pide en el punto 2. sub punto a (Agrega o recupera un respaldo de al menos 5 partículas).
☑ Se muestra captura de pantalla de lo que se pide en el punto 2. sub punto b (Muestra las partículas en el QTableWidget).
☑ Se muestra captura de pantalla de lo que se pide en el punto 2. sub punto c (Realiza una búsqueda de una partícula con un id existente).
☑ Se muestra captura de pantalla de lo que se pide en el punto 2. sub punto d (Realiza una búsqueda de una partícula con un id no existente).

Desarrollo

Lo primero en implementarse fue la tabla y seguido de eso el plain text y los botones correspondientes en su propia pestaña, luego de renombrarlos empezaron las conexiones con todos los elementos para probar su funcionamiento, para hacer uso del for con las partículas tuvimos que implementar un iterador y un next en la lista, además de eso creamos unos getters para acceder a los datos de la partícula sin tener que ir directamente a por los atributos privados.







Conclusiones

Lo que mas puedo rescatar de esta practica es que sigo aprendiendo de objetos en Python más allá de la interfaz, lo del iterador, y los getter para la lista y las particulas. Esta vez no tuve ningún problema ya todo estaba estructurado.

Referencias

MICHEL DAVALOS BOITES. (2020c, octubre 29). *PySide2 - QTableWidget (Qt for Python)(V)*. YouTube. https://www.youtube.com/watch?v=1yEpAHaiMxs

Código

Partícula

```
from .algoritmos import distancia_euclidiana

class Particula:

    def __init__(self, id="", origen_x=0, origen_y=0, destino_x=0, destino_y=0,
velocidad=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 =
round(distancia_euclidiana(origen_x,origen_y,destino_x,destino_y),5)

    def __str__(self):
        return(
            '\nId:            ' + 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'
        )

    @property
    def id(self):
        return self.__id

    @property
    def origen_x(self):
        return self.__origen_x

    @property
    def origen_y(self):
        return self.__origen_y

    @property
    def destino_x(self):
        return self.__destino_x

    @property
    def destino_y(self):
        return self.__destino_y

    @property
    def red(self):
        return self.__velocidad

    @property
    def velocidad(self):
        return self.__red

    @property
    def green(self):
        return self.__green

    @property
    def blue(self):
        return self.__blue
```

```
    @property
    def distancia(self):
        return 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,
            "velocidad": self.__velocidad,
            "red": self.__red,
            "green": self.__green,
            "blue": self.__blue
        }
```

Lista partícula

```
from .particula import Particula
import json

class ListaParticulas:
    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) for particula in self.__particulas
        )

    def __len__(self):
        return len(self.__particulas)
```

```
    def __iter__(self):
        self.cont = 0
        return self

    def __next__(self):
        if self.cont < len(self.__particulas):
            particula = self.__particulas[self.cont]
            self.cont += 1
            return particula
        else:
            raise StopIteration

    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
```

Main window

```
from operator import truediv
from PySide2.QtWidgets import QMainWindow, QFileDialog, QMessageBox,
QTableWidgetItem
from PySide2.QtCore import Slot
from ui_mainwindow import Ui_MainWindow
from particulas.listaparticula import ListaParticulas
from particulas.particula import Particula

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

        self.myListaParticulas = ListaParticulas()
```

```
        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)

        self.ui.mostrar_tabla_pushButton.clicked.connect(self.mostrar_tabla)

        self.ui.buscar_pushButton.clicked.connect(self.buscar_particula)

    @Slot()
    def buscar_particula(self):
        id = self.ui.buscar_lineEdit.text()

        encontrado = False
        for particula in self.myListaParticulas:
            if id == particula.id:
                self.ui.tabla_tableWidget.clear()
                self.ui.tabla_tableWidget.setRowCount(1)

                id_widget = QTableWidgetItem(particula.id)
                origen_x_widget = QTableWidgetItem(str(particula.origen_x))
                origen_y_widget = QTableWidgetItem(str(particula.origen_y))
                destino_x_widget = QTableWidgetItem(str(particula.destino_x))
                destino_y_widget = QTableWidgetItem(str(particula.destino_y))
                velocidad_widget = QTableWidgetItem(str(particula.velocidad))
                green_widget = QTableWidgetItem(str(particula.green))
                red_widget = QTableWidgetItem(str(particula.red))
                blue_widget = QTableWidgetItem(str(particula.blue))
                distancia_widget = QTableWidgetItem(str(particula.distancia))

                self.ui.tabla_tableWidget.setItem(0, 0, id_widget)
                self.ui.tabla_tableWidget.setItem(0, 1, origen_x_widget)
                self.ui.tabla_tableWidget.setItem(0, 2, origen_y_widget)
                self.ui.tabla_tableWidget.setItem(0, 3, destino_x_widget)
                self.ui.tabla_tableWidget.setItem(0, 4, destino_y_widget)
                self.ui.tabla_tableWidget.setItem(0, 5, velocidad_widget)
                self.ui.tabla_tableWidget.setItem(0, 6, green_widget)
                self.ui.tabla_tableWidget.setItem(0, 7, red_widget)
                self.ui.tabla_tableWidget.setItem(0, 8, blue_widget)
                self.ui.tabla_tableWidget.setItem(0, 9, distancia_widget)
```

```
                encontrado = True
                return
        if not encontrado:
            QMessageBox.warning(
                self,
                "Atención",
                f'La particula con la id: "{id}" no fue encontrada'
            )


    @Slot()
    def mostrar_tabla(self):
        self.ui.tabla_tableWidget.setColumnCount(10)
        headers =["Id", "Origen X", "Origen Y ", "Destino X", "Destino Y",
"Velocidad", "Red", "Green", "Blue", "Distancia"]
        self.ui.tabla_tableWidget.setHorizontalHeaderLabels(headers)

        self.ui.tabla_tableWidget.setRowCount(len(self.myListaParticulas))

        row = 0
        for particula in self.myListaParticulas:
            id_widget = QTableWidgetItem(particula.id)
            origen_x_widget = QTableWidgetItem(str(particula.origen_x))
            origen_y_widget = QTableWidgetItem(str(particula.origen_y))
            destino_x_widget = QTableWidgetItem(str(particula.destino_x))
            destino_y_widget = QTableWidgetItem(str(particula.destino_y))
            velocidad_widget = QTableWidgetItem(str(particula.velocidad))
            green_widget = QTableWidgetItem(str(particula.green))
            red_widget = QTableWidgetItem(str(particula.red))
            blue_widget = QTableWidgetItem(str(particula.blue))
            distancia_widget = QTableWidgetItem(str(particula.distancia))

            self.ui.tabla_tableWidget.setItem(row, 0, id_widget)
            self.ui.tabla_tableWidget.setItem(row, 1, origen_x_widget)
            self.ui.tabla_tableWidget.setItem(row, 2, origen_y_widget)
            self.ui.tabla_tableWidget.setItem(row, 3, destino_x_widget)
            self.ui.tabla_tableWidget.setItem(row, 4, destino_y_widget)
            self.ui.tabla_tableWidget.setItem(row, 5, velocidad_widget)
            self.ui.tabla_tableWidget.setItem(row, 6, green_widget)
            self.ui.tabla_tableWidget.setItem(row, 7, red_widget)
            self.ui.tabla_tableWidget.setItem(row, 8, blue_widget)
            self.ui.tabla_tableWidget.setItem(row, 9, distancia_widget)

            row += 1
```

```
    @Slot()
    def action_abrir_archivo(self):
        ubicacion = QFileDialog.getOpenFileName(
            self,
            'Abrir Archivo',
            '.',
            'JSON (*.json)'
        )[0]
        if self.myListaParticulas.abrir(ubicacion):
            QMessageBox.information(
                self,
                "Exito",
                "Se ha abierto el archivo " + ubicacion
            )
        else:
            QMessageBox.critical(
                self,
                "Error",
                "No se ha abierto el archivo " + ubicacion
            )


    @Slot()
    def action_guardar_archivo(self):
        #print('Guardar archivo')
        ubicacion = QFileDialog.getSaveFileName(
            self,
            'Guardar Archivo',
            '.',
            'JSON (*.json)'
        )[0]
        print(ubicacion)
        if self.myListaParticulas.guardar(ubicacion):
            QMessageBox.information(
                self,
                "Exito",
                "Se ha guardado el archivo " + ubicacion
            )
        else:
            QMessageBox.critical(
                self,
                "Error",
                "No se ha creado el archivo " + ubicacion
            )

    @Slot()
```

```
    def click_mostrar(self):
        self.ui.out_plainTextEdit.clear()
        self.ui.out_plainTextEdit.insertPlainText(str(self.myListaParticulas))

    @Slot()
    def click_agregar_inicio(self):
        myId = self.ui.id_lineEdit.text()
        myOrigenX = self.ui.origenX_spinBox.value()
        myOrigenY = self.ui.origenY_spinBox.value()
        myDestinoX = self.ui.destinoX_spinBox.value()
        myDestinoY = self.ui.destinoY_spinBox.value()
        myVelocidad = self.ui.velocidad_spinBox.value()
        myRed = self.ui.red_spinBox.value()
        myGreen = self.ui.green_spinBox.value()
        myBlue = self.ui.blue__spinBox.value()

        myParticula = Particula(myId, myOrigenX, myOrigenY,
myDestinoX,myDestinoY,myVelocidad,myRed,myGreen,myBlue)
        self.myListaParticulas.agregar_inicio(myParticula)

    @Slot()
    def click_agregar_final(self):
        myId = self.ui.id_lineEdit.text()
        myOrigenX = self.ui.origenX_spinBox.value()
        myOrigenY = self.ui.origenY_spinBox.value()
        myDestinoX = self.ui.destinoX_spinBox.value()
        myDestinoY = self.ui.destinoY_spinBox.value()
        myVelocidad = self.ui.velocidad_spinBox.value()
        myRed = self.ui.red_spinBox.value()
        myGreen = self.ui.green_spinBox.value()
        myBlue = self.ui.blue__spinBox.value()

        myParticula = Particula(myId, myOrigenX, myOrigenY,
myDestinoX,myDestinoY,myVelocidad,myRed,myGreen,myBlue)
        self.myListaParticulas.agregar_final(myParticula)
```

ui mainwindow

```
from operator import truediv
from PySide2.QtWidgets import QMainWindow, QFileDialog, QMessageBox,
QTableWidgetItem
from PySide2.QtCore import Slot
from ui_mainwindow import Ui_MainWindow
from particulas.listaparticula import ListaParticulas
from particulas.particula import Particula
```

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

        self.myListaParticulas = ListaParticulas()

        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)

        self.ui.mostrar_tabla_pushButton.clicked.connect(self.mostrar_tabla)

        self.ui.buscar_pushButton.clicked.connect(self.buscar_particula)

    @Slot()
    def buscar_particula(self):
        id = self.ui.buscar_lineEdit.text()

        encontrado = False
        for particula in self.myListaParticulas:
            if id == particula.id:
                self.ui.tabla_tableWidget.clear()
                self.ui.tabla_tableWidget.setRowCount(1)

                id_widget = QTableWidgetItem(particula.id)
                origen_x_widget = QTableWidgetItem(str(particula.origen_x))
                origen_y_widget = QTableWidgetItem(str(particula.origen_y))
                destino_x_widget = QTableWidgetItem(str(particula.destino_x))
                destino_y_widget = QTableWidgetItem(str(particula.destino_y))
                velocidad_widget = QTableWidgetItem(str(particula.velocidad))
                green_widget = QTableWidgetItem(str(particula.green))
                red_widget = QTableWidgetItem(str(particula.red))
                blue_widget = QTableWidgetItem(str(particula.blue))
                distancia_widget = QTableWidgetItem(str(particula.distancia))

                self.ui.tabla_tableWidget.setItem(0, 0, id_widget)
                self.ui.tabla_tableWidget.setItem(0, 1, origen_x_widget)
                self.ui.tabla_tableWidget.setItem(0, 2, origen_y_widget)
```

```
                self.ui.tabla_tableWidget.setItem(0, 3, destino_x_widget)
                self.ui.tabla_tableWidget.setItem(0, 4, destino_y_widget)
                self.ui.tabla_tableWidget.setItem(0, 5, velocidad_widget)
                self.ui.tabla_tableWidget.setItem(0, 6, green_widget)
                self.ui.tabla_tableWidget.setItem(0, 7, red_widget)
                self.ui.tabla_tableWidget.setItem(0, 8, blue_widget)
                self.ui.tabla_tableWidget.setItem(0, 9, distancia_widget)

                encontrado = True
                return
        if not encontrado:
            QMessageBox.warning(
                self,
                "Atención",
                f'La particula con la id: "{id}" no fue encontrada'
            )


    @Slot()
    def mostrar_tabla(self):
        self.ui.tabla_tableWidget.setColumnCount(10)
        headers =["Id", "Origen X", "Origen Y ", "Destino X", "Destino Y",
"Velocidad", "Red", "Green", "Blue", "Distancia"]
        self.ui.tabla_tableWidget.setHorizontalHeaderLabels(headers)

        self.ui.tabla_tableWidget.setRowCount(len(self.myListaParticulas))

        row = 0
        for particula in self.myListaParticulas:
            id_widget = QTableWidgetItem(particula.id)
            origen_x_widget = QTableWidgetItem(str(particula.origen_x))
            origen_y_widget = QTableWidgetItem(str(particula.origen_y))
            destino_x_widget = QTableWidgetItem(str(particula.destino_x))
            destino_y_widget = QTableWidgetItem(str(particula.destino_y))
            velocidad_widget = QTableWidgetItem(str(particula.velocidad))
            green_widget = QTableWidgetItem(str(particula.green))
            red_widget = QTableWidgetItem(str(particula.red))
            blue_widget = QTableWidgetItem(str(particula.blue))
            distancia_widget = QTableWidgetItem(str(particula.distancia))

            self.ui.tabla_tableWidget.setItem(row, 0, id_widget)
            self.ui.tabla_tableWidget.setItem(row, 1, origen_x_widget)
            self.ui.tabla_tableWidget.setItem(row, 2, origen_y_widget)
            self.ui.tabla_tableWidget.setItem(row, 3, destino_x_widget)
            self.ui.tabla_tableWidget.setItem(row, 4, destino_y_widget)
            self.ui.tabla_tableWidget.setItem(row, 5, velocidad_widget)
```

```
            self.ui.tabla_tableWidget.setItem(row, 6, green_widget)
            self.ui.tabla_tableWidget.setItem(row, 7, red_widget)
            self.ui.tabla_tableWidget.setItem(row, 8, blue_widget)
            self.ui.tabla_tableWidget.setItem(row, 9, distancia_widget)

            row += 1

    @Slot()
    def action_abrir_archivo(self):
        ubicacion = QFileDialog.getOpenFileName(
            self,
            'Abrir Archivo',
            '.',
            'JSON (*.json)'
        )[0]
        if self.myListaParticulas.abrir(ubicacion):
            QMessageBox.information(
                self,
                "Exito",
                "Se ha abierto el archivo " + ubicacion
            )
        else:
            QMessageBox.critical(
                self,
                "Error",
                "No se ha abierto el archivo " + ubicacion
            )


    @Slot()
    def action_guardar_archivo(self):
        #print('Guardar archivo')
        ubicacion = QFileDialog.getSaveFileName(
            self,
            'Guardar Archivo',
            '.',
            'JSON (*.json)'
        )[0]
        print(ubicacion)
        if self.myListaParticulas.guardar(ubicacion):
            QMessageBox.information(
                self,
                "Exito",
                "Se ha guardado el archivo " + ubicacion
            )
        else:
```

```
            QMessageBox.critical(
                self,
                "Error",
                "No se ha creado el archivo " + ubicacion
            )

    @Slot()
    def click_mostrar(self):
        self.ui.out_plainTextEdit.clear()
        self.ui.out_plainTextEdit.insertPlainText(str(self.myListaParticulas))

    @Slot()
    def click_agregar_inicio(self):
        myId = self.ui.id_lineEdit.text()
        myOrigenX = self.ui.origenX_spinBox.value()
        myOrigenY = self.ui.origenY_spinBox.value()
        myDestinoX = self.ui.destinoX_spinBox.value()
        myDestinoY = self.ui.destinoY_spinBox.value()
        myVelocidad = self.ui.velocidad_spinBox.value()
        myRed = self.ui.red_spinBox.value()
        myGreen = self.ui.green_spinBox.value()
        myBlue = self.ui.blue__spinBox.value()

        myParticula = Particula(myId, myOrigenX, myOrigenY,
myDestinoX,myDestinoY,myVelocidad,myRed,myGreen,myBlue)
        self.myListaParticulas.agregar_inicio(myParticula)

    @Slot()
    def click_agregar_final(self):
        myId = self.ui.id_lineEdit.text()
        myOrigenX = self.ui.origenX_spinBox.value()
        myOrigenY = self.ui.origenY_spinBox.value()
        myDestinoX = self.ui.destinoX_spinBox.value()
        myDestinoY = self.ui.destinoY_spinBox.value()
        myVelocidad = self.ui.velocidad_spinBox.value()
        myRed = self.ui.red_spinBox.value()
        myGreen = self.ui.green_spinBox.value()
        myBlue = self.ui.blue__spinBox.value()

        myParticula = Particula(myId, myOrigenX, myOrigenY,
myDestinoX,myDestinoY,myVelocidad,myRed,myGreen,myBlue)
        self.myListaParticulas.agregar_final(myParticula)
```