Actividad 10

(Sort)

*Alcaraz Valdivia Marcos Fernando*

*Seminario de Solución de Problemas de Algoritmia*

**Lineamientos de evaluación**

* El reporte está en formato Google Docs o PDF. *(REALIZADO)*
* El reporte sigue las pautas del Formato de Actividades. *(REALIZADO)*
* El reporte tiene desarrollada todas las pautas del Formato de Actividades. *(REALIZADO)*
* Se muestra captura de pantalla de las partículas del antes y después de ser ordenadas por id de manera ascendente tanto en el QPlainTextEdit como en el QTableWidget. *(REALIZADO)*
* Se muestra captura de pantalla de las partículas del antes y después de ser ordenadas por distancia de manera descendente tanto en el QPlainTextEdit como en el QTableWidget. *(REALIZADO)*
* Se muestra captura de pantalla de las partículas del antes y después de ser ordenadas por velocidad de manera ascendente tanto en el QPlainTextEdit como en el QTableWidget. *(REALIZADO)*

***AAAAAAAAAA  
Satisfacción 10/10***

***Desarrollo***

***Toma capturas de pantalla de las partículas del antes y el después de ser ordenadas por el ID de manera ascendente en el QPlainTextEdit y el QTableWidget:***

***QPlainText ANTES:***

Captura de pantalla de computadora

Descripción generada automáticamente

***QTableWidget ANTES:***

Captura de pantalla de computadora

Descripción generada automáticamente

***QPlainText DESPUÉS:***

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

***QTableWidget DESPUÉS:***

Captura de pantalla de computadora

Descripción generada automáticamente

***Toma capturas de pantalla de las partículas del antes y el después de ser ordenadas por la Distancia de manera descendente en el QPlainTextEdit y el QTableWidget:***

***QPlainText ANTES:***

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

***QTableWidget ANTES:***

Captura de pantalla de computadora

Descripción generada automáticamente

***QPlainText DESPUÉS:***

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

***QTableWidget DESPUÉS:***

Captura de pantalla de computadora

Descripción generada automáticamente

***Toma capturas de pantalla de las partículas del antes y el después de ser ordenadas por la Velocidad de manera ascendente en el QPlainTextEdit y el QTableWidget:***

***QPlainText ANTES:***

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

***QTableWidget ANTES:***

Captura de pantalla de computadora

Descripción generada automáticamente

***QPlainText DESPUÉS:***

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

***QTableWidget DESPUÉS:***

Una captura de pantalla de una red social

Descripción generada automáticamente

***Conclusiones***

***Cada interacción con este programa es más divertido, porque me en esta ocasión me quedaron las ganas de ordenar de forma ascendente y descendente cada uno de los datos, pero bueno la tarea solo condicionaba el ID, Velocidad y Distancia.***

***(Detrás de PDFs estoy haciendo los Sort para cada dato)***

***Referencias***

**Python – sort()**

<https://youtu.be/0NZajLIy5qQ>

***Código***

***mainwindow.py***

from PySide2.QtWidgets import QMainWindow, QFileDialog, QMessageBox, QTableWidgetItem, QGraphicsScene

from PySide2.QtCore import Slot

from PySide2.QtGui import QPen, QColor, QTransform

from ui\_mainwindow import Ui\_MainWindow

from manager import Manager

from particula import Particula

class MainWindow(QMainWindow):

    def \_\_init\_\_(self):

        super(MainWindow, self).\_\_init\_\_()

        self.manager = Manager()

        self.ui = Ui\_MainWindow()

        self.ui.setupUi(self)

        self.id = 0

        # Cuando el botón pushbutton es presionado, ejecuta la función click\_agregar

        # self.ui.mostrar.clicked.connect(self.click\_mostrar)

        self.ui.insertar\_inicio.clicked.connect(self.click\_insertar\_inicio)

        self.ui.insertar\_final.clicked.connect(self.click\_insertar\_final)

        self.ui.OrdenarAscendenteID\_pushbutton.clicked.connect(self.accionMostrarTablaSBID)

        self.ui.OrdenarAscendenteVelocidad\_pushbutton.clicked.connect(self.accionMostrarTablaSBS)

        self.ui.OrdenarDescendenteDistancia\_pushbutton.clicked.connect(self.accionMostrarTablaSBD)

        self.ui.OrdenarAscendenteID\_pushbutton\_2.clicked.connect(self.accionMostrarTablaSBID)

        self.ui.OrdenarAscendenteVelocidad\_pushbutton\_2.clicked.connect(self.accionMostrarTablaSBS)

        self.ui.OrdenarDescendenteDistancia\_pushbutton\_2.clicked.connect(self.accionMostrarTablaSBD)

        self.ui.accionAbrir.triggered.connect(self.accionAbrirArchivo)

        self.ui.accionGuardar.triggered.connect(self.accionGuardarArchivo)

        self.ui.Buscar\_pushbutton.clicked.connect(self.accionBuscar)

        self.scene = QGraphicsScene()

        self.ui.graphicsView.setScene(self.scene)

    @Slot()

    def dibujar(self):

        pen = QPen()

        for particula in self.manager:

            pen.setWidth(2)

            color = QColor(particula.red, particula.green, particula.blue)

            pen.setColor(color)

            self.scene.addEllipse(particula.origenX, particula.origenY, 5, 5, pen)

            self.scene.addEllipse(particula.destinoX, particula.destinoY, 5, 5, pen)

            self.scene.addLine(particula.origenX, particula.origenY, particula.destinoX, particula.destinoY, pen)

    @Slot()

    def accionMostrarTabla(self):

        self.ui.tabla.setColumnCount(10)

        headers = ["ID", "Origen X", "Origen Y", "Destino X",

                   "Destino Y", "Velocidad", "Red", "Green", "Blue", "Distancia"]

        self.ui.tabla.setHorizontalHeaderLabels(headers)

        self.ui.tabla.setRowCount(len(self.manager))

        fila = 0

        for particula in self.manager:

            id\_widget = QTableWidgetItem(str(particula.id))

            origenX\_widget = QTableWidgetItem(str(particula.origenX))

            origenY\_widget = QTableWidgetItem(str(particula.origenY))

            destinoX\_widget = QTableWidgetItem(str(particula.destinoX))

            destinoY\_widget = QTableWidgetItem(str(particula.destinoY))

            velocidad\_widget = QTableWidgetItem(str(particula.velocidad))

            red\_widget = QTableWidgetItem(str(particula.red))

            green\_widget = QTableWidgetItem(str(particula.green))

            blue\_widget = QTableWidgetItem(str(particula.blue))

            distancia\_widget = QTableWidgetItem(str(particula.distancia))

            self.ui.tabla.setItem(fila, 0, id\_widget)

            self.ui.tabla.setItem(fila, 1, origenX\_widget)

            self.ui.tabla.setItem(fila, 2, origenY\_widget)

            self.ui.tabla.setItem(fila, 3, destinoX\_widget)

            self.ui.tabla.setItem(fila, 4, destinoY\_widget)

            self.ui.tabla.setItem(fila, 5, velocidad\_widget)

            self.ui.tabla.setItem(fila, 6, red\_widget)

            self.ui.tabla.setItem(fila, 7, green\_widget)

            self.ui.tabla.setItem(fila, 8, blue\_widget)

            self.ui.tabla.setItem(fila, 9, distancia\_widget)

            fila += 1

    @Slot()

    def accionMostrarTablaSBID(self):

        self.ui.tabla.setColumnCount(10)

        headers = ["ID", "Origen X", "Origen Y", "Destino X",

                   "Destino Y", "Velocidad", "Red", "Green", "Blue", "Distancia"]

        self.ui.tabla.setHorizontalHeaderLabels(headers)

        self.ui.tabla.setRowCount(len(self.manager))

        fila = 0

        array = []

        for particula in self.manager:

            array.append(particula)

        array.sort(key=lambda particula: particula.id, reverse=False)

        self.ui.lista\_particulas.clear()

        for particula in array:

            self.ui.lista\_particulas.insertPlainText(str(particula))

        for particula in array:

            id\_widget = QTableWidgetItem(str(particula.id))

            origenX\_widget = QTableWidgetItem(str(particula.origenX))

            origenY\_widget = QTableWidgetItem(str(particula.origenY))

            destinoX\_widget = QTableWidgetItem(str(particula.destinoX))

            destinoY\_widget = QTableWidgetItem(str(particula.destinoY))

            velocidad\_widget = QTableWidgetItem(str(particula.velocidad))

            red\_widget = QTableWidgetItem(str(particula.red))

            green\_widget = QTableWidgetItem(str(particula.green))

            blue\_widget = QTableWidgetItem(str(particula.blue))

            distancia\_widget = QTableWidgetItem(str(particula.distancia))

            self.ui.tabla.setItem(fila, 0, id\_widget)

            self.ui.tabla.setItem(fila, 1, origenX\_widget)

            self.ui.tabla.setItem(fila, 2, origenY\_widget)

            self.ui.tabla.setItem(fila, 3, destinoX\_widget)

            self.ui.tabla.setItem(fila, 4, destinoY\_widget)

            self.ui.tabla.setItem(fila, 5, velocidad\_widget)

            self.ui.tabla.setItem(fila, 6, red\_widget)

            self.ui.tabla.setItem(fila, 7, green\_widget)

            self.ui.tabla.setItem(fila, 8, blue\_widget)

            self.ui.tabla.setItem(fila, 9, distancia\_widget)

            fila += 1

    @Slot()

    def accionMostrarTablaSBS(self):

        self.ui.tabla.setColumnCount(10)

        headers = ["ID", "Origen X", "Origen Y", "Destino X",

                   "Destino Y", "Velocidad", "Red", "Green", "Blue", "Distancia"]

        self.ui.tabla.setHorizontalHeaderLabels(headers)

        self.ui.tabla.setRowCount(len(self.manager))

        fila = 0

        array = []

        for particula in self.manager:

            array.append(particula)

        array.sort(key=lambda particula: particula.velocidad, reverse=False)

        self.ui.lista\_particulas.clear()

        for particula in array:

            self.ui.lista\_particulas.insertPlainText(str(particula))

        for particula in array:

            id\_widget = QTableWidgetItem(str(particula.id))

            origenX\_widget = QTableWidgetItem(str(particula.origenX))

            origenY\_widget = QTableWidgetItem(str(particula.origenY))

            destinoX\_widget = QTableWidgetItem(str(particula.destinoX))

            destinoY\_widget = QTableWidgetItem(str(particula.destinoY))

            velocidad\_widget = QTableWidgetItem(str(particula.velocidad))

            red\_widget = QTableWidgetItem(str(particula.red))

            green\_widget = QTableWidgetItem(str(particula.green))

            blue\_widget = QTableWidgetItem(str(particula.blue))

            distancia\_widget = QTableWidgetItem(str(particula.distancia))

            self.ui.tabla.setItem(fila, 0, id\_widget)

            self.ui.tabla.setItem(fila, 1, origenX\_widget)

            self.ui.tabla.setItem(fila, 2, origenY\_widget)

            self.ui.tabla.setItem(fila, 3, destinoX\_widget)

            self.ui.tabla.setItem(fila, 4, destinoY\_widget)

            self.ui.tabla.setItem(fila, 5, velocidad\_widget)

            self.ui.tabla.setItem(fila, 6, red\_widget)

            self.ui.tabla.setItem(fila, 7, green\_widget)

            self.ui.tabla.setItem(fila, 8, blue\_widget)

            self.ui.tabla.setItem(fila, 9, distancia\_widget)

            fila += 1

    @Slot()

    def accionMostrarTablaSBD(self):

        self.ui.tabla.setColumnCount(10)

        headers = ["ID", "Origen X", "Origen Y", "Destino X",

                   "Destino Y", "Velocidad", "Red", "Green", "Blue", "Distancia"]

        self.ui.tabla.setHorizontalHeaderLabels(headers)

        self.ui.tabla.setRowCount(len(self.manager))

        fila = 0

        array = []

        for particula in self.manager:

            array.append(particula)

        array.sort(key=lambda particula: particula.distancia, reverse=True)

        self.ui.lista\_particulas.clear()

        for particula in array:

            self.ui.lista\_particulas.insertPlainText(str(particula))

        for particula in array:

            id\_widget = QTableWidgetItem(str(particula.id))

            origenX\_widget = QTableWidgetItem(str(particula.origenX))

            origenY\_widget = QTableWidgetItem(str(particula.origenY))

            destinoX\_widget = QTableWidgetItem(str(particula.destinoX))

            destinoY\_widget = QTableWidgetItem(str(particula.destinoY))

            velocidad\_widget = QTableWidgetItem(str(particula.velocidad))

            red\_widget = QTableWidgetItem(str(particula.red))

            green\_widget = QTableWidgetItem(str(particula.green))

            blue\_widget = QTableWidgetItem(str(particula.blue))

            distancia\_widget = QTableWidgetItem(str(particula.distancia))

            self.ui.tabla.setItem(fila, 0, id\_widget)

            self.ui.tabla.setItem(fila, 1, origenX\_widget)

            self.ui.tabla.setItem(fila, 2, origenY\_widget)

            self.ui.tabla.setItem(fila, 3, destinoX\_widget)

            self.ui.tabla.setItem(fila, 4, destinoY\_widget)

            self.ui.tabla.setItem(fila, 5, velocidad\_widget)

            self.ui.tabla.setItem(fila, 6, red\_widget)

            self.ui.tabla.setItem(fila, 7, green\_widget)

            self.ui.tabla.setItem(fila, 8, blue\_widget)

            self.ui.tabla.setItem(fila, 9, distancia\_widget)

            fila += 1

    @Slot()

    def accionBuscar(self):

        id = self.ui.buscar\_lineedit.text()

        bandera = False

        for particula in self.manager:

            if str(id) == str(particula.id):

                self.ui.tabla.clear()

                self.ui.tabla.setRowCount(1)

                id\_widget = QTableWidgetItem(str(particula.id))

                origenX\_widget = QTableWidgetItem(str(particula.origenX))

                origenY\_widget = QTableWidgetItem(str(particula.origenY))

                destinoX\_widget = QTableWidgetItem(str(particula.destinoX))

                destinoY\_widget = QTableWidgetItem(str(particula.destinoY))

                velocidad\_widget = QTableWidgetItem(str(particula.velocidad))

                red\_widget = QTableWidgetItem(str(particula.red))

                green\_widget = QTableWidgetItem(str(particula.green))

                blue\_widget = QTableWidgetItem(str(particula.blue))

                distancia\_widget = QTableWidgetItem(str(particula.distancia))

                self.ui.tabla.setItem(0, 0, id\_widget)

                self.ui.tabla.setItem(0, 1, origenX\_widget)

                self.ui.tabla.setItem(0, 2, origenY\_widget)

                self.ui.tabla.setItem(0, 3, destinoX\_widget)

                self.ui.tabla.setItem(0, 4, destinoY\_widget)

                self.ui.tabla.setItem(0, 5, velocidad\_widget)

                self.ui.tabla.setItem(0, 6, red\_widget)

                self.ui.tabla.setItem(0, 7, green\_widget)

                self.ui.tabla.setItem(0, 8, blue\_widget)

                self.ui.tabla.setItem(0, 9, distancia\_widget)

                bandera = True

                return

        if not bandera:

            QMessageBox.warning(self, "Particula no encontrada", f'La particula con el ID " {id} " no fue encontrada')

    @Slot()

    def accionAbrirArchivo(self):

        ubicacion = QFileDialog.getOpenFileName(

            self,

            "Abrir archivo",

            ".",

            "JSON (\*.json)"

        )[0]

        if self.manager.abrir(ubicacion):

            self.click\_mostrar()

            self.accionMostrarTabla()

            self.dibujar()

            QMessageBox.information(

                self, "Abrir archivo", "Archivo abierto Exitosamente : " + ubicacion)

        else:

            QMessageBox.critical(

                self, "Error", "No se puede abrir el archivo : " + ubicacion)

    @Slot()

    def accionGuardarArchivo(self):

        ubicacion = QFileDialog.getSaveFileName(

            self,

            "Guardar Archivo",

            ".",

            "JSON (\*.json)"

        )[0]

        if self.manager.guardar(ubicacion):

            QMessageBox.information(

                self, "Archivo Guardado", "Guardado Exitoso : " + ubicacion)

        else:

            QMessageBox.critical(

                self, "Error", "Archivo no Guardado : " + ubicacion)

    @Slot()

    def click\_insertar\_inicio(self):

        self.id += 1

        aux = Particula(self.id, self.ui.ox.value(), self.ui.oy.value(), self.ui.dx.value(), self.ui.dy.value(

        ), self.ui.velocidad.value(), self.ui.red.value(), self.ui.green.value(), self.ui.blue.value())

        self.manager.agregarInicio(aux)

        self.click\_mostrar()

        self.accionMostrarTabla()

        self.dibujar()

    @Slot()

    def click\_insertar\_final(self):

        self.id += 1

        aux = Particula(self.id, self.ui.ox.value(), self.ui.oy.value(), self.ui.dx.value(), self.ui.dy.value(

        ), self.ui.velocidad.value(), self.ui.red.value(), self.ui.green.value(), self.ui.blue.value())

        self.manager.agregarFinal(aux)

        self.click\_mostrar()

    @Slot()

    def click\_mostrar(self):

        self.ui.lista\_particulas.clear()

        self.ui.lista\_particulas.insertPlainText(str(self.manager))

***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(1093, 797)

        self.accionGuardar = QAction(MainWindow)

        self.accionGuardar.setObjectName(u"accionGuardar")

        self.accionAbrir = QAction(MainWindow)

        self.accionAbrir.setObjectName(u"accionAbrir")

        self.centralwidget = QWidget(MainWindow)

        self.centralwidget.setObjectName(u"centralwidget")

        self.gridLayout\_4 = QGridLayout(self.centralwidget)

        self.gridLayout\_4.setObjectName(u"gridLayout\_4")

        self.tabWidget = QTabWidget(self.centralwidget)

        self.tabWidget.setObjectName(u"tabWidget")

        self.tab = QWidget()

        self.tab.setObjectName(u"tab")

        self.gridLayout\_2 = QGridLayout(self.tab)

        self.gridLayout\_2.setObjectName(u"gridLayout\_2")

        self.lista\_particulas = QPlainTextEdit(self.tab)

        self.lista\_particulas.setObjectName(u"lista\_particulas")

        self.gridLayout\_2.addWidget(self.lista\_particulas, 0, 3, 1, 1)

        self.groupBox = QGroupBox(self.tab)

        self.groupBox.setObjectName(u"groupBox")

        self.gridLayout = QGridLayout(self.groupBox)

        self.gridLayout.setObjectName(u"gridLayout")

        self.label\_3 = QLabel(self.groupBox)

        self.label\_3.setObjectName(u"label\_3")

        self.gridLayout.addWidget(self.label\_3, 4, 0, 1, 1)

        self.dx = QSpinBox(self.groupBox)

        self.dx.setObjectName(u"dx")

        self.dx.setMaximum(500)

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

        self.oy = QSpinBox(self.groupBox)

        self.oy.setObjectName(u"oy")

        self.oy.setMaximum(500)

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

        self.label\_5 = QLabel(self.groupBox)

        self.label\_5.setObjectName(u"label\_5")

        self.gridLayout.addWidget(self.label\_5, 6, 0, 1, 1)

        self.blue = QSpinBox(self.groupBox)

        self.blue.setObjectName(u"blue")

        self.blue.setMaximum(255)

        self.gridLayout.addWidget(self.blue, 7, 1, 1, 1)

        self.insertar\_inicio = QPushButton(self.groupBox)

        self.insertar\_inicio.setObjectName(u"insertar\_inicio")

        self.gridLayout.addWidget(self.insertar\_inicio, 8, 0, 1, 1)

        self.label = QLabel(self.groupBox)

        self.label.setObjectName(u"label")

        self.gridLayout.addWidget(self.label, 2, 0, 1, 1)

        self.label\_7 = QLabel(self.groupBox)

        self.label\_7.setObjectName(u"label\_7")

        self.gridLayout.addWidget(self.label\_7, 1, 0, 1, 1)

        self.velocidad = QSpinBox(self.groupBox)

        self.velocidad.setObjectName(u"velocidad")

        self.velocidad.setMaximum(999)

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

        self.OrdenarAscendenteID\_pushbutton\_2 = QPushButton(self.groupBox)

        self.OrdenarAscendenteID\_pushbutton\_2.setObjectName(u"OrdenarAscendenteID\_pushbutton\_2")

        self.gridLayout.addWidget(self.OrdenarAscendenteID\_pushbutton\_2, 9, 0, 1, 1)

        self.green = QSpinBox(self.groupBox)

        self.green.setObjectName(u"green")

        self.green.setMaximum(255)

        self.gridLayout.addWidget(self.green, 6, 1, 1, 1)

        self.red = QSpinBox(self.groupBox)

        self.red.setObjectName(u"red")

        self.red.setMaximum(255)

        self.gridLayout.addWidget(self.red, 5, 1, 1, 1)

        self.ox = QSpinBox(self.groupBox)

        self.ox.setObjectName(u"ox")

        self.ox.setMaximum(500)

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

        self.OrdenarAscendenteVelocidad\_pushbutton\_2 = QPushButton(self.groupBox)

        self.OrdenarAscendenteVelocidad\_pushbutton\_2.setObjectName(u"OrdenarAscendenteVelocidad\_pushbutton\_2")

        self.gridLayout.addWidget(self.OrdenarAscendenteVelocidad\_pushbutton\_2, 9, 1, 1, 1)

        self.label\_8 = QLabel(self.groupBox)

        self.label\_8.setObjectName(u"label\_8")

        self.gridLayout.addWidget(self.label\_8, 0, 0, 1, 1)

        self.label\_2 = QLabel(self.groupBox)

        self.label\_2.setObjectName(u"label\_2")

        self.gridLayout.addWidget(self.label\_2, 3, 0, 1, 1)

        self.insertar\_final = QPushButton(self.groupBox)

        self.insertar\_final.setObjectName(u"insertar\_final")

        self.gridLayout.addWidget(self.insertar\_final, 8, 1, 1, 1)

        self.dy = QSpinBox(self.groupBox)

        self.dy.setObjectName(u"dy")

        self.dy.setMaximum(500)

        self.gridLayout.addWidget(self.dy, 3, 1, 1, 1)

        self.label\_6 = QLabel(self.groupBox)

        self.label\_6.setObjectName(u"label\_6")

        self.gridLayout.addWidget(self.label\_6, 7, 0, 1, 1)

        self.label\_4 = QLabel(self.groupBox)

        self.label\_4.setObjectName(u"label\_4")

        self.gridLayout.addWidget(self.label\_4, 5, 0, 1, 1)

        self.OrdenarDescendenteDistancia\_pushbutton\_2 = QPushButton(self.groupBox)

        self.OrdenarDescendenteDistancia\_pushbutton\_2.setObjectName(u"OrdenarDescendenteDistancia\_pushbutton\_2")

        self.gridLayout.addWidget(self.OrdenarDescendenteDistancia\_pushbutton\_2, 10, 0, 1, 2)

        self.gridLayout\_2.addWidget(self.groupBox, 0, 0, 1, 1)

        self.tabWidget.addTab(self.tab, "")

        self.tab\_2 = QWidget()

        self.tab\_2.setObjectName(u"tab\_2")

        self.gridLayout\_3 = QGridLayout(self.tab\_2)

        self.gridLayout\_3.setObjectName(u"gridLayout\_3")

        self.buscar\_lineedit = QLineEdit(self.tab\_2)

        self.buscar\_lineedit.setObjectName(u"buscar\_lineedit")

        self.gridLayout\_3.addWidget(self.buscar\_lineedit, 2, 0, 1, 2)

        self.OrdenarAscendenteVelocidad\_pushbutton = QPushButton(self.tab\_2)

        self.OrdenarAscendenteVelocidad\_pushbutton.setObjectName(u"OrdenarAscendenteVelocidad\_pushbutton")

        self.gridLayout\_3.addWidget(self.OrdenarAscendenteVelocidad\_pushbutton, 4, 1, 1, 1)

        self.OrdenarAscendenteID\_pushbutton = QPushButton(self.tab\_2)

        self.OrdenarAscendenteID\_pushbutton.setObjectName(u"OrdenarAscendenteID\_pushbutton")

        self.gridLayout\_3.addWidget(self.OrdenarAscendenteID\_pushbutton, 4, 0, 1, 1)

        self.tabla = QTableWidget(self.tab\_2)

        self.tabla.setObjectName(u"tabla")

        self.gridLayout\_3.addWidget(self.tabla, 0, 0, 1, 3)

        self.Buscar\_pushbutton = QPushButton(self.tab\_2)

        self.Buscar\_pushbutton.setObjectName(u"Buscar\_pushbutton")

        self.gridLayout\_3.addWidget(self.Buscar\_pushbutton, 2, 2, 1, 1)

        self.OrdenarDescendenteDistancia\_pushbutton = QPushButton(self.tab\_2)

        self.OrdenarDescendenteDistancia\_pushbutton.setObjectName(u"OrdenarDescendenteDistancia\_pushbutton")

        self.gridLayout\_3.addWidget(self.OrdenarDescendenteDistancia\_pushbutton, 4, 2, 1, 1)

        self.tabWidget.addTab(self.tab\_2, "")

        self.tab\_3 = QWidget()

        self.tab\_3.setObjectName(u"tab\_3")

        self.graphicsView = QGraphicsView(self.tab\_3)

        self.graphicsView.setObjectName(u"graphicsView")

        self.graphicsView.setGeometry(QRect(10, 10, 1051, 701))

        self.tabWidget.addTab(self.tab\_3, "")

        self.gridLayout\_4.addWidget(self.tabWidget, 1, 0, 1, 1)

        MainWindow.setCentralWidget(self.centralwidget)

        self.menubar = QMenuBar(MainWindow)

        self.menubar.setObjectName(u"menubar")

        self.menubar.setGeometry(QRect(0, 0, 1093, 21))

        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.accionAbrir)

        self.menuArchivo.addAction(self.accionGuardar)

        self.retranslateUi(MainWindow)

        self.tabWidget.setCurrentIndex(1)

        QMetaObject.connectSlotsByName(MainWindow)

    # setupUi

    def retranslateUi(self, MainWindow):

        MainWindow.setWindowTitle(QCoreApplication.translate("MainWindow", u"MainWindow", None))

        self.accionGuardar.setText(QCoreApplication.translate("MainWindow", u"Guardar", None))

#if QT\_CONFIG(shortcut)

        self.accionGuardar.setShortcut(QCoreApplication.translate("MainWindow", u"Ctrl+S", None))

#endif // QT\_CONFIG(shortcut)

        self.accionAbrir.setText(QCoreApplication.translate("MainWindow", u"Abrir", None))

#if QT\_CONFIG(shortcut)

        self.accionAbrir.setShortcut(QCoreApplication.translate("MainWindow", u"Ctrl+L", None))

#endif // QT\_CONFIG(shortcut)

        self.groupBox.setTitle(QCoreApplication.translate("MainWindow", u"Particulas", None))

        self.label\_3.setText(QCoreApplication.translate("MainWindow", u"Velocidad ( KM/h )", None))

        self.label\_5.setText(QCoreApplication.translate("MainWindow", u"GREEN ( 0-255 )", None))

        self.insertar\_inicio.setText(QCoreApplication.translate("MainWindow", u"Insertar al Inicio", None))

        self.label.setText(QCoreApplication.translate("MainWindow", u"Destino X  ( 0-500 )", None))

        self.label\_7.setText(QCoreApplication.translate("MainWindow", u"Origen Y ( 0-500 )", None))

        self.OrdenarAscendenteID\_pushbutton\_2.setText(QCoreApplication.translate("MainWindow", u"Ordenar de forma Ascendente ID", None))

        self.OrdenarAscendenteVelocidad\_pushbutton\_2.setText(QCoreApplication.translate("MainWindow", u"Ordenar de forma Ascendente Velocidad", None))

        self.label\_8.setText(QCoreApplication.translate("MainWindow", u"Origen X ( 0-500 )", None))

        self.label\_2.setText(QCoreApplication.translate("MainWindow", u"Destino Y  ( 0-500 )", None))

        self.insertar\_final.setText(QCoreApplication.translate("MainWindow", u"Insertar al Final", None))

        self.label\_6.setText(QCoreApplication.translate("MainWindow", u"BLUE ( 0-255 )", None))

        self.label\_4.setText(QCoreApplication.translate("MainWindow", u"RED ( 0-255 )", None))

        self.OrdenarDescendenteDistancia\_pushbutton\_2.setText(QCoreApplication.translate("MainWindow", u"Ordenar de forma Decendente Distancia", None))

        self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab), QCoreApplication.translate("MainWindow", u"Agregar Particulas", None))

        self.buscar\_lineedit.setPlaceholderText(QCoreApplication.translate("MainWindow", u"Escribe un ID", None))

        self.OrdenarAscendenteVelocidad\_pushbutton.setText(QCoreApplication.translate("MainWindow", u"Ordenar de forma Ascendente Velocidad", None))

        self.OrdenarAscendenteID\_pushbutton.setText(QCoreApplication.translate("MainWindow", u"Ordenar de forma Ascendente ID", None))

        self.Buscar\_pushbutton.setText(QCoreApplication.translate("MainWindow", u"Buscar", None))

        self.OrdenarDescendenteDistancia\_pushbutton.setText(QCoreApplication.translate("MainWindow", u"Ordenar de forma Decendente Distancia", None))

        self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab\_2), QCoreApplication.translate("MainWindow", u"Tabla de Particulas", None))

        self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab\_3), QCoreApplication.translate("MainWindow", u"Gr\u00e1fico de Particulas", None))

        self.menuArchivo.setTitle(QCoreApplication.translate("MainWindow", u"Archivo", None))

    # retranslateUi

***particula.py***

# Archivo particula.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):

        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 ('\n\nParticula\n' +

                '\nID       : ' + str(self.\_\_id) +

                '\nOrigen X : ' + str(self.\_\_origen\_x) +

                '\nOrigen Y : ' + str(self.\_\_origen\_y) +

                '\nDestino X: ' + str(self.\_\_destino\_x) +

                '\nDestino Y: ' + str(self.\_\_destino\_y) +

                '\nVelocidad: ' + str(self.\_\_velocidad) +

                '\nRed      : ' + str(self.\_\_blue) +

                '\nGreen    : ' + str(self.\_\_green) +

                '\nBlue     : ' + str(self.\_\_blue) +

                '\nDistancia: ' + str(self.\_\_distancia)

                )

    def \_\_lt\_\_(self, other):

        return self.id < other.id

    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,

        }

    @property

    def id(self):

        return self.\_\_id

    @property

    def origenX(self):

        return self.\_\_origen\_x

    @property

    def origenY(self):

        return self.\_\_origen\_y

    @property

    def destinoX(self):

        return self.\_\_destino\_x

    @property

    def destinoY(self):

        return self.\_\_destino\_y

    @property

    def velocidad(self):

        return self.\_\_velocidad

    @property

    def red(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

***manager.py***

# Archivo manager.py

from particula import Particula

import json

class Manager:

    def \_\_init\_\_(self):

        self.\_\_particulas = []

    def agregarInicio(self, particula: Particula):

        self.\_\_particulas.insert(0, particula)

    def agregarFinal(self, particula: Particula):

        self.\_\_particulas.append(particula)

    def imprimir(self):

        for particula in self.\_\_particulas:

            print(particula)

    def \_\_str\_\_(self):

        return "".join(

            str(particula) for particula in self.\_\_particulas

        )

    def sortById(self):

        array = []

        for particula in self.\_\_particulas:

            array.append(particula)

        array.sort(key=particula.\_\_id, reverse=False)

        return "".join(

            str(particula) for particula in array

        )

    def sortBySpeed(self):

        array = []

        for particula in self.\_\_particulas:

            array.append(particula)

        array.sort(key=particula.velocidad, reverse=False)

        return "".join(

            str(particula) for particula in array

        )

    def sortByDistance(self):

        array = []

        for particula in self.\_\_particulas:

            array.append(particula)

        array.sort(key=particula.distancia, reverse=True)

        return "".join(

            str(particula) for particula in array

        )

    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