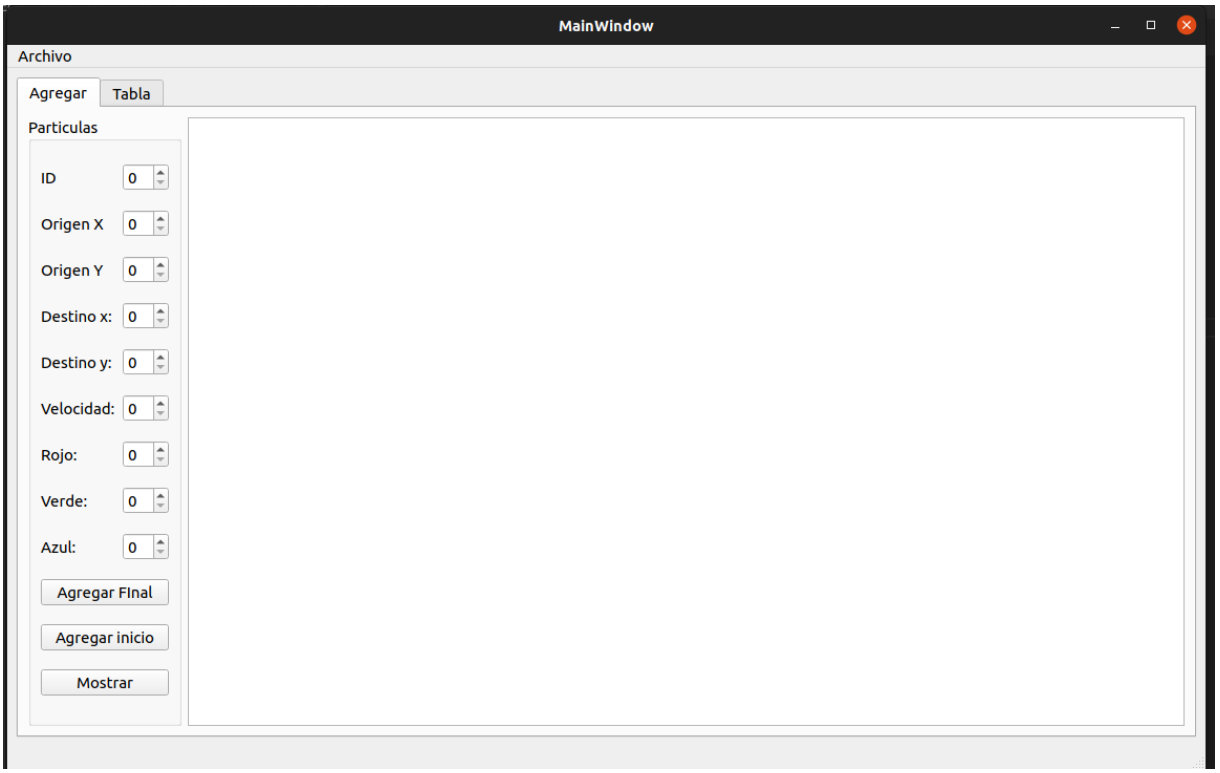
Actividad 07 QTableWidget

Ramirez Orduna Geovanni

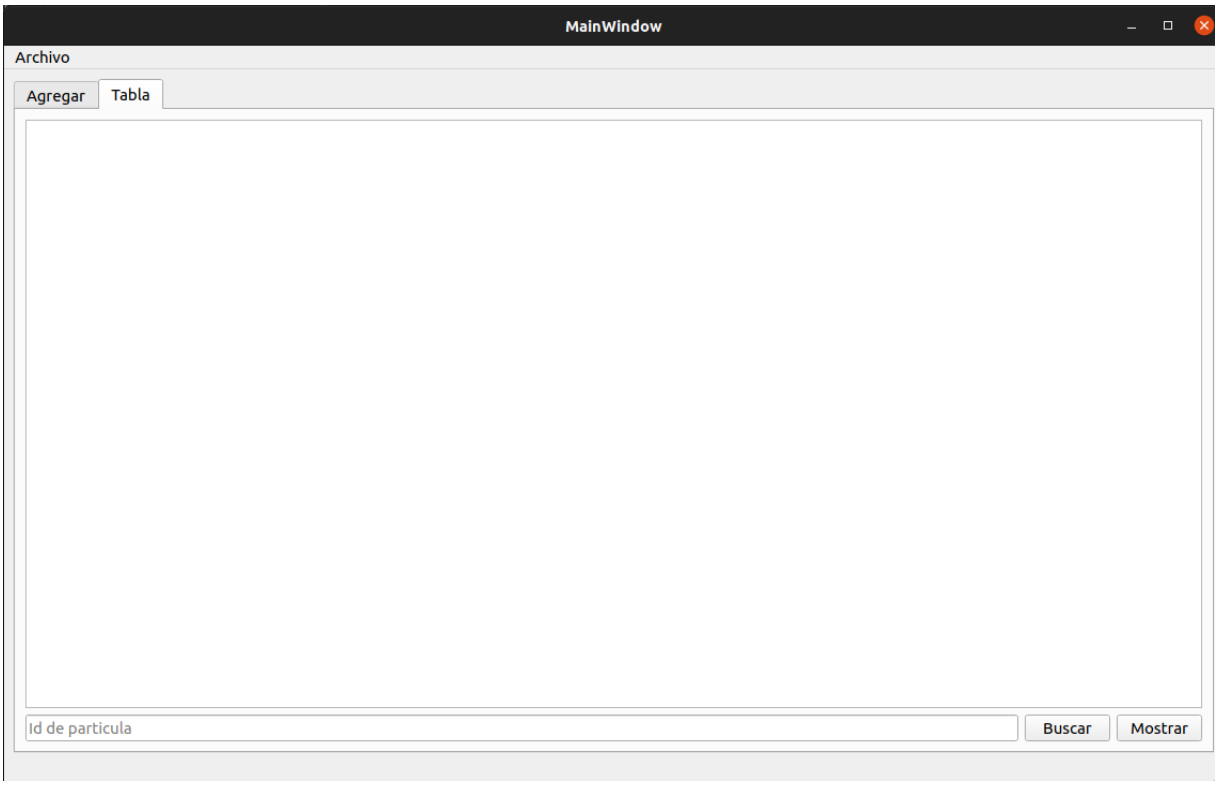
Lineamientos de evaluación

- [ ] 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 captura de pantalla de lo que se pide en el punto 2. sub punto a.
- [ ] Se muestra captura de pantalla de lo que se pide en el punto 2. sub punto b.
- [ ] Se muestra captura de pantalla de lo que se pide en el punto 2. sub punto c.
- [ ] Se muestra captura de pantalla de lo que se pide en el punto 2. sub punto d

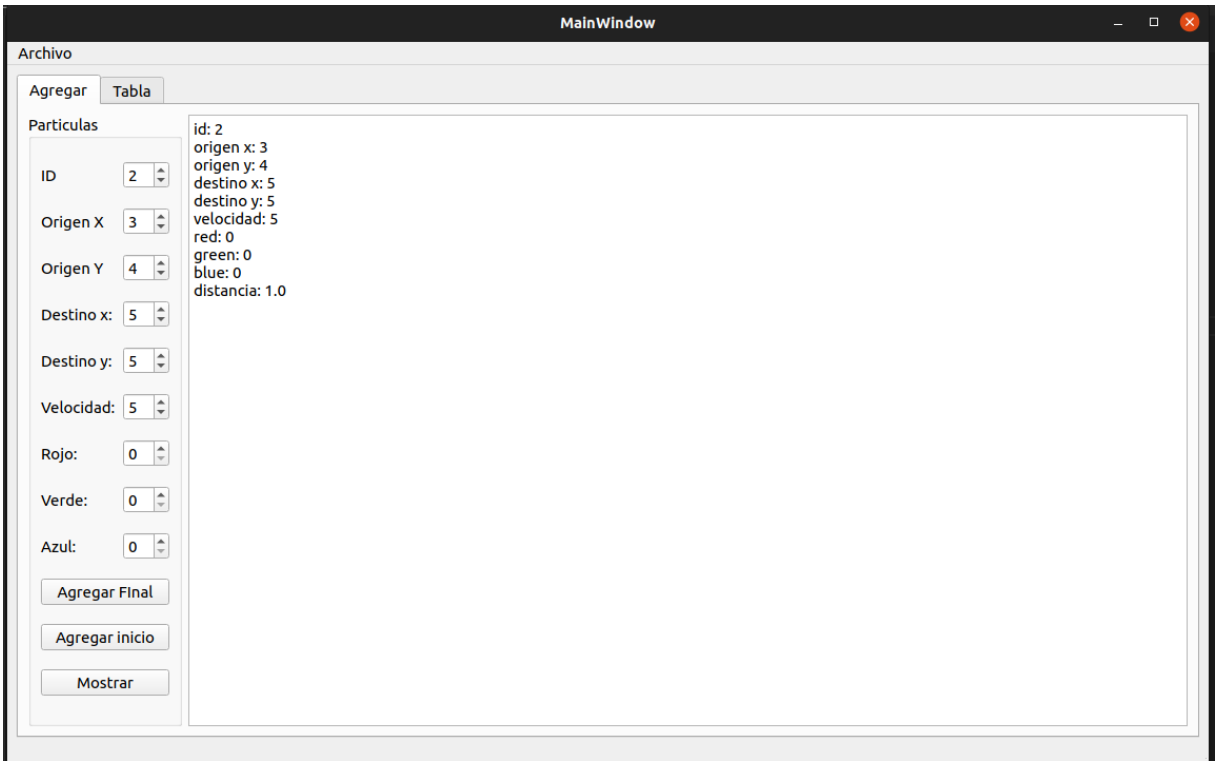
tenemos el main principal



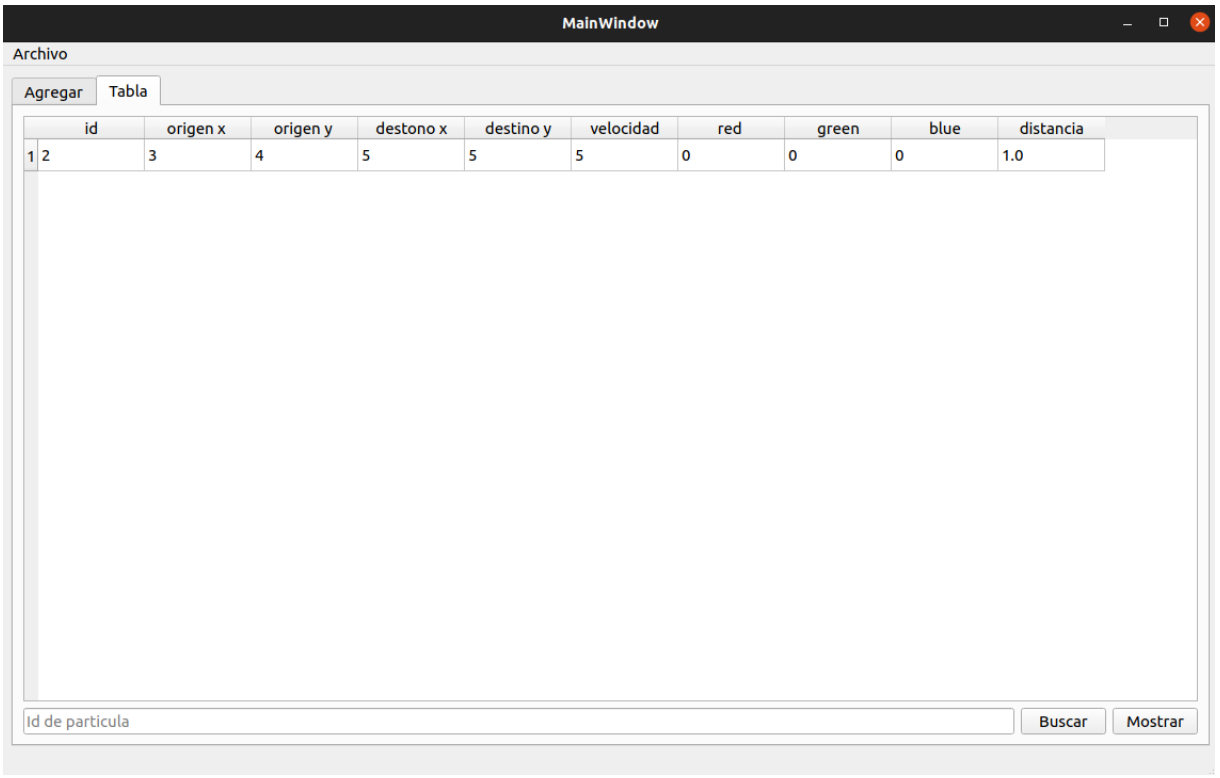
Tenemos la interfaz actualizada ya con la tabla



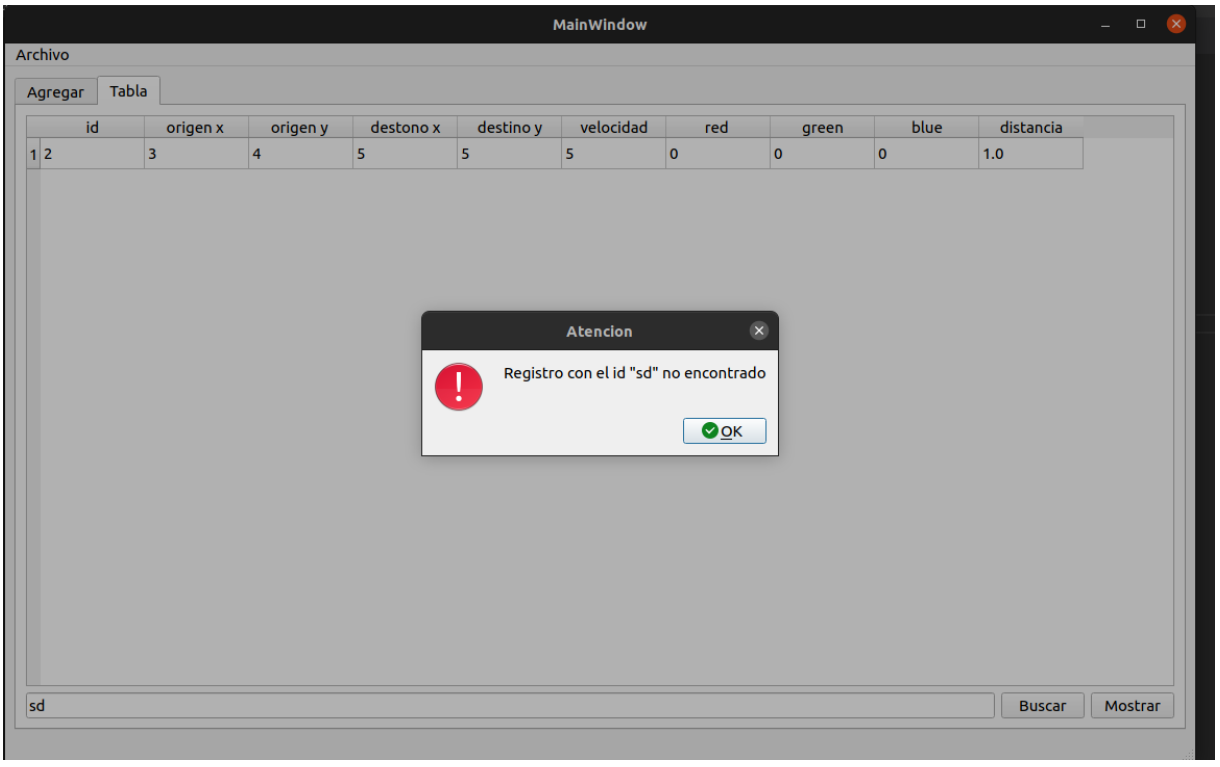
mostramos los datos pra poder comprobar que son los mismos que en la tabla



tenemos los datos agregados en la tabla



Buscamos un registro



Conclusión
En esta actividad puede realizar una tabla donde vaya mostrando nuestros registros, así como implementar una búsqueda.

Referencias :
https://www.youtube.com/watch?v=HRY8QvXmcDM&t=2668s

codigo

```
from asyncio.format_helpers import _format_callback_source
from wsgiref import headers
import PySide2
from PySide2.QtWidgets import QMainWindow, QFileDialog, QMessageBox,
QTableWidgetItem
from PySide2.QtCore import Slot
from ui_mainwindow import Ui_MainWindow
from admin import Admin
from particula import Particula
from algoritmos import distacia_euclidiana

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

        self.admin=Admin()

        self.ui=Ui_MainWindow()
        self.ui.setupUi(self) ##mete la intrerfaz
        self.ui.Agre_Final_buttom.clicked.connect(self.click_agregar)
##conectar clase a boton

self.ui.Agre_Inicio_Buttom.clicked.connect(self.click_agregar_inicio)
        self.ui.Mostrar_buttom.clicked.connect(self.click_mostrar)



        self.ui.actionAbrir.triggered.connect(self.action_abrir_archivo)
        self.ui.actionGuardar.triggered.connect(self.action_guardar)

        self.ui.Mostrar_Dates.clicked.connect(self.mostrar_tabla)
        self.ui.Buscar_Butonn.clicked.connect(self.buscar_id)

    @Slot()
    def buscar_id(self):
        id=self.ui.txt_line.text()
        encontrado=False
        for Particula in self.admin:
            if id == Particula.id:
                self.ui.tabla.clear()
                self.ui.tabla.setRowCount(1)
                id_widget= QTableWidgetItem(Particula.id)
```

```
                origenX_widget=
QTableWidgetItem(str(Particula.origen_x))
                origenY_widget=
QTableWidgetItem(str(Particula.origen_y))
                destinox_widget=
QTableWidgetItem(str(Particula.destino_x))
                destinoY_widget=
QTableWidgetItem(str(Particula.destino_y))
                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)

                encontrado=True
                return
        if not encontrado:
                QMessageBox.warning(
                    self,
                    "Atencion",
                    f'Registro con el id "{str(id)}" no encontrado'
                )



    @Slot()
    def mostrar_tabla(self):
        self.ui.tabla.setColumnCount(10)
        headers=["id",  "origen x ","origen y ","destono x" ,"destino y"
,"velocidad" ,"red ","green","blue","distancia"]
```

```
        self.ui.tabla.setHorizontalHeaderLabels(headers)
        self.ui.tabla.setRowCount(len(self.admin))


        row=0

        for Particula in self.admin:
          id_widget= QTableWidgetItem(str(Particula.id))
          origenX_widget= QTableWidgetItem(str(Particula.origen_x))
          origenY_widget= QTableWidgetItem(str(Particula.origen_y))
          destinox_widget= QTableWidgetItem(str(Particula.destino_x))
          destinoY_widget= QTableWidgetItem(str(Particula.destino_y))
          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(row, 0, id_widget)
          self.ui.tabla.setItem(row, 1, origenX_widget)
          self.ui.tabla.setItem(row, 2, origenY_widget)
          self.ui.tabla.setItem(row, 3, destinox_widget)
          self.ui.tabla.setItem(row, 4, destinoY_widget)
          self.ui.tabla.setItem(row, 5, velocidad_widget)
          self.ui.tabla.setItem(row, 6, red_widget)
          self.ui.tabla.setItem(row, 7, green_widget)
          self.ui.tabla.setItem(row, 8, blue_widget)
          self.ui.tabla.setItem(row, 9, distancia_widget)

          row+=1


    @Slot()
    def action_abrir_archivo(self):
        #print("abrir archivo")
            ubicacion=QFileDialog.getOpenFileName(
                self,
                "abrir archivo",
                ".",
                "JSON (*.json)"
            )[0]
            if self.admin.open(ubicacion):
                QMessageBox.information(
```

```
                self,
                "exito",
                "Se abrio el archivo"+ubicacion
            )
        else:
            QMessageBox.critical(
                self,
                "Error",
                "Error abrir archivo "
            )



@Slot()
def action_guardar(self):
    #print("guardar")
    ubicacion= QFileDialog.getSaveFileName(
     self,
     'Guardar Archivo',
     '.',
     'JSON (*.json)'
    )[0]
    print( ubicacion)
    self.admin.save(ubicacion)
    if self.admin.save(ubicacion):
         QMessageBox.information(
             self,
             "exito",
             "se pudo crear archivo"+ubicacion
         )
    else:
         QMessageBox.critical(
             self,
             "Error",
             "no se pudo crear archivo"
         )


@Slot()
def click_agregar(self):
        id=self.ui.Id_spinBox.value()
        OrigenX=self.ui.OrigenX_spinBox.value()
        OrigenY=self.ui.OrigenY_spinBox.value()
```

```
            DestinoX=self.ui.DestinoX_spinBox.value()
            DestinoY=self.ui.DestinoY_spinBox.value()
            Velocidad=self.ui.Velocidad_spinBox.value()
            Red=self.ui.Red_spinBox.value()
            Green=self.ui.Green_spinBox.value()
            Blue=self.ui.Blue_spinBox.value()
            distancia=distacia_euclidiana


particula=Particula(id,OrigenX,OrigenY,DestinoX,DestinoY,Velocidad,
Red,Green,Blue,distancia)
            self.admin.agregar_final(particula)

            #print(OrigenX,OrigenY,
DestinoX,DestinoY,Velocidad,Red,Green,Blue)
            #self.ui.salida.insertPlainText(str(OrigenX) + str( OrigenY)
+ str(DestinoX) + str (DestinoY) + str (Velocidad) +str( Red)+
str(Green) + str(Blue))
    @Slot()
    def click_agregar_inicio(self):
            id=self.ui.Id_spinBox.value()
            OrigenX=self.ui.OrigenX_spinBox.value()
            OrigenY=self.ui.OrigenY_spinBox.value()
            DestinoX=self.ui.DestinoX_spinBox.value()
            DestinoY=self.ui.DestinoY_spinBox.value()
            Velocidad=self.ui.Velocidad_spinBox.value()
            Red=self.ui.Red_spinBox.value()
            Green=self.ui.Green_spinBox.value()
            Blue=self.ui.Blue_spinBox.value()
            distancia=distacia_euclidiana


particula=Particula(id,OrigenX,OrigenY,DestinoX,DestinoY,Velocidad,
Red,Green,Blue,distancia)
            self.admin.agregar_inicio(particula)

    @Slot()
    def click_mostrar(self):
           # self.admin.mostrar()
            self.ui.salida.clear()
            self.ui.salida.inser
from asyncio.format_helpers import _format_callback_source
from wsgiref import headers
```

```
import PySide2
from PySide2.QtWidgets import QMainWindow, QFileDialog, QMessageBox,
QTableWidgetItem
from PySide2.QtCore import Slot
from ui_mainwindow import Ui_MainWindow
from admin import Admin
from particula import Particula
from algoritmos import distacia_euclidiana

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

       self.admin=Admin()

       self.ui=Ui_MainWindow()
       self.ui.setupUi(self) ##mete la intrerfaz
       self.ui.Agre_Final_buttom.clicked.connect(self.click_agregar)
##conectar clase a boton

self.ui.Agre_Inicio_Buttom.clicked.connect(self.click_agregar_inicio)
       self.ui.Mostrar_buttom.clicked.connect(self.click_mostrar)


       self.ui.actionAbrir.triggered.connect(self.action_abrir_archivo)
       self.ui.actionGuardar.triggered.connect(self.action_guardar)

       self.ui.Mostrar_Dates.clicked.connect(self.mostrar_tabla)
       self.ui.Buscar_Butonn.clicked.connect(self.buscar_id)

   @Slot()
   def buscar_id(self):
       id=self.ui.txt_line.text()
       encontrado=False
       for Particula in self.admin:
           if id == Particula.id:
               self.ui.tabla.clear()
               self.ui.tabla.setRowCount(1)
               id_widget= QTableWidgetItem(Particula.id)
               origenX_widget=
QTableWidgetItem(str(Particula.origen_x))
               origenY_widget=
QTableWidgetItem(str(Particula.origen_y))
```

```
                destinox_widget=
QTableWidgetItem(str(Particula.destino_x))
                destinoY_widget=
QTableWidgetItem(str(Particula.destino_y))
                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)

                encontrado=True
                return
        if not encontrado:
                QMessageBox.warning(
                    self,
                    "Atencion",
                    f'Registro con el id "{str(id)}" no encontrado'
                )



    @Slot()
    def mostrar_tabla(self):
        self.ui.tabla.setColumnCount(10)
        headers=["id",  "origen x ","origen y ","destono x" ,"destino y"
,"velocidad" ,"red ","green","blue","distancia"]
        self.ui.tabla.setHorizontalHeaderLabels(headers)
        self.ui.tabla.setRowCount(len(self.admin))
```

```
        row=0

        for Particula in self.admin:
          id_widget= QTableWidgetItem(str(Particula.id))
          origenX_widget= QTableWidgetItem(str(Particula.origen_x))
          origenY_widget= QTableWidgetItem(str(Particula.origen_y))
          destinox_widget= QTableWidgetItem(str(Particula.destino_x))
          destinoY_widget= QTableWidgetItem(str(Particula.destino_y))
          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(row, 0, id_widget)
          self.ui.tabla.setItem(row, 1, origenX_widget)
          self.ui.tabla.setItem(row, 2, origenY_widget)
          self.ui.tabla.setItem(row, 3, destinox_widget)
          self.ui.tabla.setItem(row, 4, destinoY_widget)
          self.ui.tabla.setItem(row, 5, velocidad_widget)
          self.ui.tabla.setItem(row, 6, red_widget)
          self.ui.tabla.setItem(row, 7, green_widget)
          self.ui.tabla.setItem(row, 8, blue_widget)
          self.ui.tabla.setItem(row, 9, distancia_widget)

          row+=1


    @Slot()
    def action_abrir_archivo(self):
        #print("abrir archivo")
            ubicacion=QFileDialog.getOpenFileName(
                self,
                "abrir archivo",
                ".",
                "JSON (*.json)"
            )[0]
            if self.admin.open(ubicacion):
                QMessageBox.information(
                    self,
                    "exito",
                    "Se abrio el archivo"+ubicacion
                )
```

```
        else:
            QMessageBox.critical(
                self,
                "Error",
                "Error abrir archivo "
            )



@Slot()
def action_guardar(self):
    #print("guardar")
    ubicacion= QFileDialog.getSaveFileName(
     self,
     'Guardar Archivo',
     '.',
     'JSON (*.json)'
    )[0]
    print( ubicacion)
    self.admin.save(ubicacion)
    if self.admin.save(ubicacion):
         QMessageBox.information(
             self,
             "exito",
             "se pudo crear archivo"+ubicacion
         )
    else:
         QMessageBox.critical(
             self,
             "Error",
             "no se pudo crear archivo"
         )


@Slot()
def click_agregar(self):
        id=self.ui.Id_spinBox.value()
        OrigenX=self.ui.OrigenX_spinBox.value()
        OrigenY=self.ui.OrigenY_spinBox.value()
        DestinoX=self.ui.DestinoX_spinBox.value()
        DestinoY=self.ui.DestinoY_spinBox.value()
        Velocidad=self.ui.Velocidad_spinBox.value()
        Red=self.ui.Red_spinBox.value()
```

```
            Green=self.ui.Green_spinBox.value()
            Blue=self.ui.Blue_spinBox.value()
            distancia=distacia_euclidiana


particula=Particula(id,OrigenX,OrigenY,DestinoX,DestinoY,Velocidad,
Red,Green,Blue,distancia)
            self.admin.agregar_final(particula)

            #print(OrigenX,OrigenY,
DestinoX,DestinoY,Velocidad,Red,Green,Blue)
            #self.ui.salida.insertPlainText(str(OrigenX) + str( OrigenY)
+ str(DestinoX) + str (DestinoY) + str (Velocidad) +str( Red)+
str(Green) + str(Blue))
    @Slot()
    def click_agregar_inicio(self):
            id=self.ui.Id_spinBox.value()
            OrigenX=self.ui.OrigenX_spinBox.value()
            OrigenY=self.ui.OrigenY_spinBox.value()
            DestinoX=self.ui.DestinoX_spinBox.value()
            DestinoY=self.ui.DestinoY_spinBox.value()
            Velocidad=self.ui.Velocidad_spinBox.value()
            Red=self.ui.Red_spinBox.value()
            Green=self.ui.Green_spinBox.value()
            Blue=self.ui.Blue_spinBox.value()
            distancia=distacia_euclidiana


particula=Particula(id,OrigenX,OrigenY,DestinoX,DestinoY,Velocidad,
Red,Green,Blue,distancia)
            self.admin.agregar_inicio(particula)

    @Slot()
    def click_mostrar(self):
           # self.admin.mostrar()
            self.ui.salida.clear()
            self.ui.salida.inser
```

mainWindow

```
import PySide2
from PySide2.QtWidgets import QMainWindow, QFileDialog, QMessageBox
from PySide2.QtCore import Slot
from ui_mainwindow import Ui_MainWindow
from admin import Admin
from particula import Particula
from algoritmos import distacia_euclidiana

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

        self.admin=Admin()

        self.ui=Ui_MainWindow()
        self.ui.setupUi(self) ##mete la intrerfaz
        self.ui.Agre_Final_buttom.clicked.connect(self.click_agregar)
##conectar clase a boton

self.ui.Agre_Inicio_Buttom.clicked.connect(self.click_agregar_inicio)
        self.ui.Mostrar_buttom.clicked.connect(self.click_mostrar)


        self.ui.actionAbrir.triggered.connect(self.action_abrir_archivo)
        self.ui.actionGuardar.triggered.connect(self.action_guardar)


    @Slot()
    def action_abrir_archivo(self):
        #print("abrir archivo")
        ubicacion=QFileDialog.getOpenFileName(
            self,
            "abrir archivo",
            ".",
            "JSON (*.json)"
        )[0]
        if self.admin.open(ubicacion):
            QMessageBox.information(
                 self,
                "exito",
                "Se abrio el archivo"+ubicacion
            )
```

```
     else:
         QMessageBox.critical(
             self,
             "Error",
             "Error abrir archivo "
         )



@Slot()
def action_guardar(self):
    #print("guardar")
    ubicacion= QFileDialog.getSaveFileName(
     self,
     'Guardar Archivo',
     '.',
     'JSON (*.json)'
    )[0]
    print( ubicacion)
    self.admin.save(ubicacion)
    if self.admin.save(ubicacion):
         QMessageBox.information(
             self,
             "exito",
             "se pudo crear archivo"+ubicacion
         )
    else:
         QMessageBox.critical(
             self,
             "Error",
             "no se pudo crear archivo"
         )



@Slot()
def click_agregar(self):
         id=self.ui.Id_spinBox.value()
         OrigenX=self.ui.OrigenX_spinBox.value()
         OrigenY=self.ui.OrigenY_spinBox.value()
         DestinoX=self.ui.DestinoX_spinBox.value()
         DestinoY=self.ui.DestinoY_spinBox.value()
         Velocidad=self.ui.Velocidad_spinBox.value()
         Red=self.ui.Red_spinBox.value()
```

```
            Green=self.ui.Green_spinBox.value()
            Blue=self.ui.Blue_spinBox.value()
            distancia=distacia_euclidiana


particula=Particula(id,OrigenX,OrigenY,DestinoX,DestinoY,Velocidad,
Red,Green,Blue,distancia)
            self.admin.agregar_final(particula)

            #print(OrigenX,OrigenY,
DestinoX,DestinoY,Velocidad,Red,Green,Blue)
            #self.ui.salida.insertPlainText(str(OrigenX) + str( OrigenY)
+ str(DestinoX) + str (DestinoY) + str (Velocidad) +str( Red)+
str(Green) + str(Blue))
    @Slot()
    def click_agregar_inicio(self):
            id=self.ui.Id_spinBox.value()
            OrigenX=self.ui.OrigenX_spinBox.value()
            OrigenY=self.ui.OrigenY_spinBox.value()
            DestinoX=self.ui.DestinoX_spinBox.value()
            DestinoY=self.ui.DestinoY_spinBox.value()
            Velocidad=self.ui.Velocidad_spinBox.value()
            Red=self.ui.Red_spinBox.value()
            Green=self.ui.Green_spinBox.value()
            Blue=self.ui.Blue_spinBox.value()
            distancia=distacia_euclidiana


particula=Particula(id,OrigenX,OrigenY,DestinoX,DestinoY,Velocidad,
Red,Green,Blue,distancia)
            self.admin.agregar_inicio(particula)

    @Slot()
    def click_mostrar(self):
           # self.admin.mostrar()
            self.ui.salida.clear()
            self.ui.salida.insertPlainText(str(self.admin))
```