Actividad 08 - QTableWidget

Gabriel Eduardo Sevilla Chavez

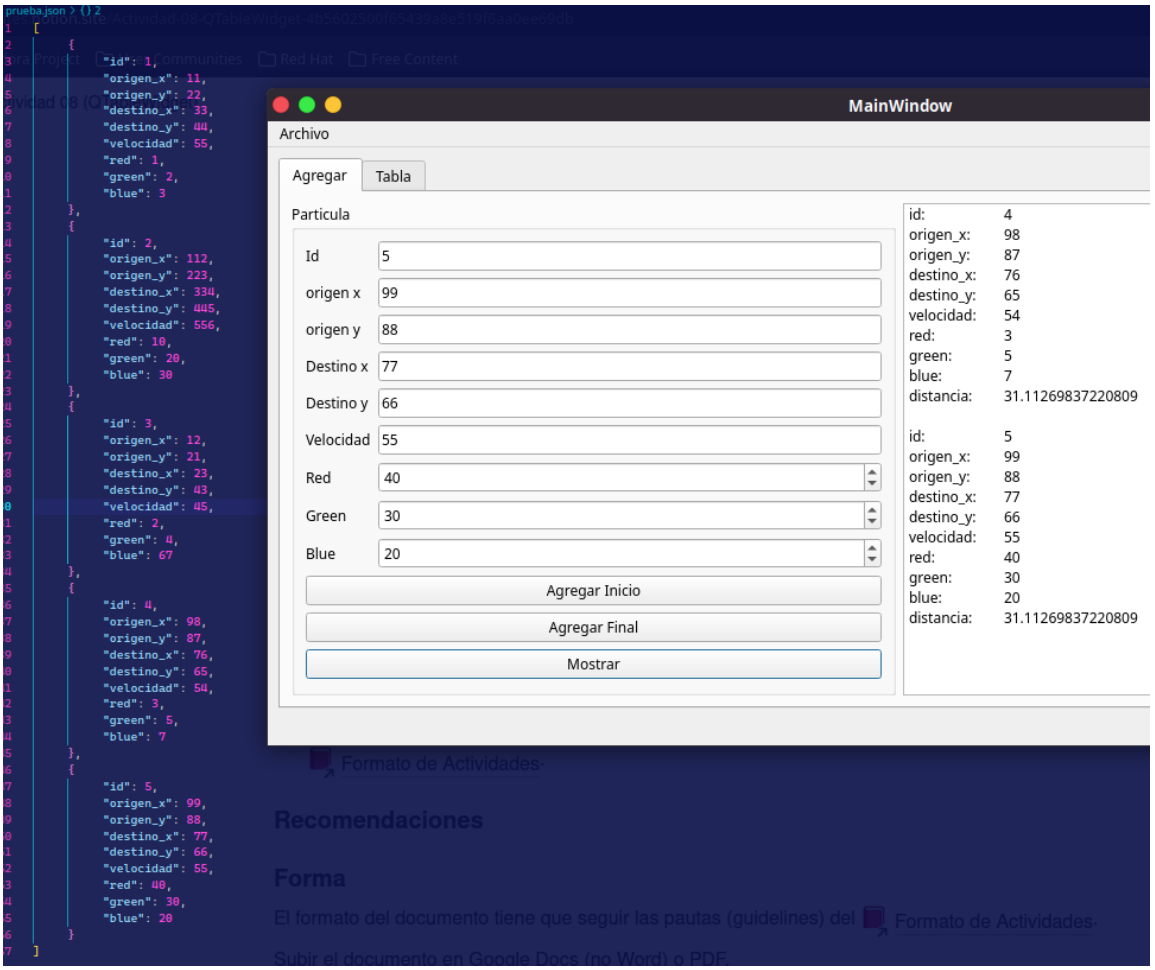
Seminario de algoritmia

Lineamientos de evaluación

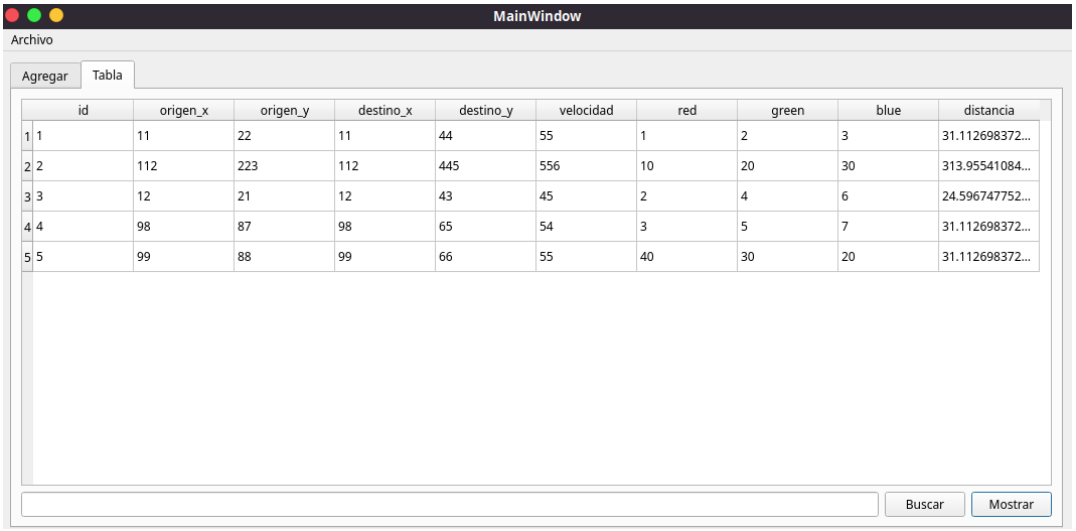
- [ ] El reporte sigue las pautas del Formato de Actividades .
- [ ] El reporte tiene desarrollada todas las pautas del Formato de Actividades.
- [ ] Se muestra captura de pantalla de lo que se pide en el punto 2. sub punto a.
- [ ] Se muestra captura de pantalla de lo que se pide en el punto 2. sub 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.

Desarrollo

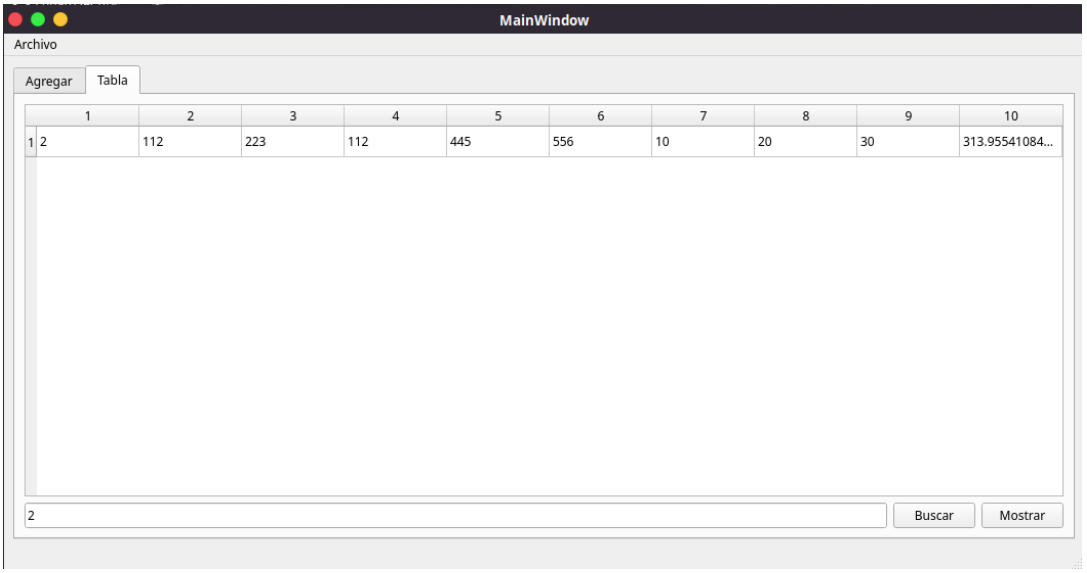
respaldo de al menos 5 partículas



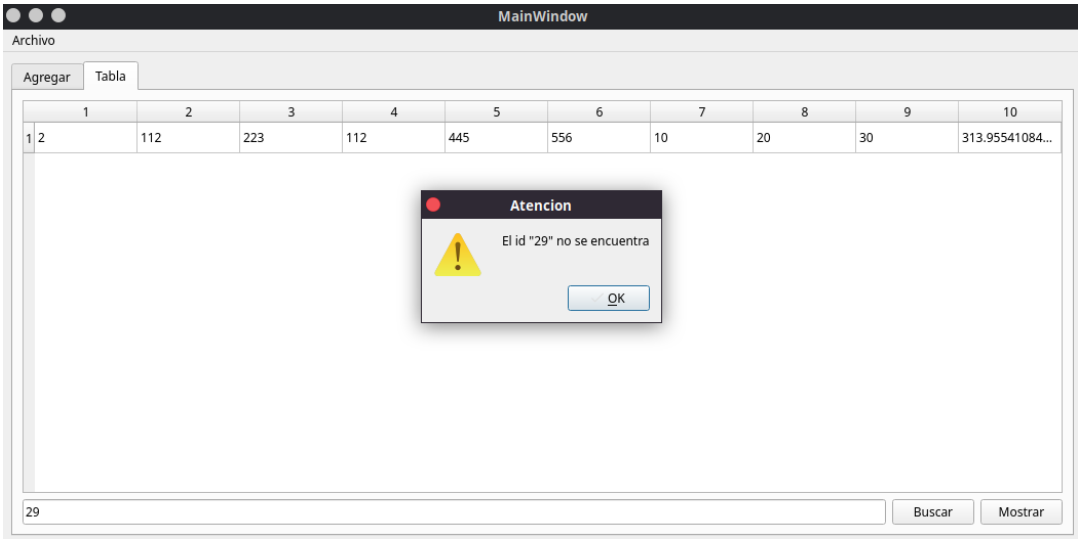
partículas en el `QTableWidget`



búsqueda de una partícula con un `id` existente.



búsqueda de una partícula con un `id` no existente.



Desarrollo

En esta actividad no hubo muchas complicaciones, solamente al ser uno con más requerimientos fue más tedioso que en el video de referencia. Pero hubo aprendizajes logrados como tomar datos de un lineEdit, mostrar datos en una tabla, etc.

Referencias

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

Código

main.py

```
from PySide2.QtWidgets import QApplication

from mainWindow import MainWindow

import sys


app = QApplication()

#Ventana de app

window = MainWindow()

window.show()

sys.exit(app.exec_())
```

mainWindow.py

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

```
from PySide2.QtCore import Slot

from regex import R

from ui_mainWindow import Ui_MainWindow

#incluir clases particlas

from Actividad05_ClasesyObjetos.particulas import Particulas

from Actividad05_ClasesyObjetos.particula import Particula

class MainWindow(QMainWindow):

   def __init__(self):

        super(MainWindow, self).__init__() #Contructor de QMainWindow

        #Guardar particulas

        self.particulas = Particulas()


        self.ui = Ui_MainWindow()

        #mandar los datos de self.ui a la ventana

        self.ui.setupUi(self)

        # Eventos en botones


self.ui.pbAgregarInicio.clicked.connect(self.click_agregarInicio)


self.ui.pbAgregaFinal.clicked.connect(self.click_agregarFinal)

        self.ui.pbMostrar.clicked.connect(self.click_mostrar)

        #Ad Archivo
```

```
self.ui.actionAbrir_archivo.triggered.connect(self.actionOpenFile)

self.ui.actionGuardar_archivo.triggered.connect(self.actionSaveFile)
        #Table
        self.ui.btnMostrarTabla.clicked.connect(self.mostrarTabla)
        self.ui.btnBuscar.clicked.connect(self.buscarId)


    @Slot()
    def buscarId(self):
        id = self.ui.lineEditTabla.text()
        encontrado = False
        for particula in self.particulas:
            if id == str(particula.id):
                self.ui.tableWidget.clear()
                self.ui.tableWidget.setRowCount(1)
                self.viewData(0,particula)
                encontrado = True
                return
        if not encontrado:
            QMessageBox.warning(
                self,"Atencion",f'El id "{id}" no se encuentra'
            )
```

```python
    @Slot()
    def mostrarTabla(self):
        self.ui.tableWidget.setColumnCount(10)
        headers = ["id","origen_x","origen_y","destino_x",

"destino_y","velocidad","red","green","blue","distancia"]
        self.ui.tableWidget.setHorizontalHeaderLabels(headers)
        self.ui.tableWidget.setRowCount(len(self.particulas))
        row = 0
        for particula in self.particulas:
            self.viewData(row,particula)
            row +=1


    def viewData(self,row,particula):
        id_widget = QTableWidgetItem(str(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))
        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.tableWidget.setItem(row,0,id_widget)
        self.ui.tableWidget.setItem(row,1,origen_x_widget)
        self.ui.tableWidget.setItem(row,2,origen_y_widget)
        self.ui.tableWidget.setItem(row,3,destino_x_widget)
        self.ui.tableWidget.setItem(row,4,destino_y_widget)
        self.ui.tableWidget.setItem(row,5,velocidad_widget)
        self.ui.tableWidget.setItem(row,6,red_widget)
        self.ui.tableWidget.setItem(row,7,green_widget)
        self.ui.tableWidget.setItem(row,8,blue_widget)
        self.ui.tableWidget.setItem(row,9,distancia_widget)

@Slot()
def actionOpenFile(self):
    ubicacion = QFileDialog.getOpenFileName(
        self,
        'Abrir archivo',
```

```
            '.',
            'JSON (*.json)'
        )[0]
        if self.particulas.abrir(ubicacion):
            QMessageBox.information(
                self,'Exito',
                'Se abrio el archivo'+ubicacion
            )
        else:
            QMessageBox.critical(
                self,'Error',
                'Error al abrir el archivo'+ubicacion
            )
    def actionSaveFile(self):
        #print("Guardar archivo")
        ubicacion = QFileDialog.getSaveFileName(
            self,
            'Guardar archivo',
            '.',
            'JSON (*.json)'
        )[0]
        print(ubicacion)
```

```
        if self.particulas.guardar(ubicacion):
            QMessageBox.information(
                self,"Exito","Archivo guardado"+ubicacion
            )
        else:
            QMessageBox.critical(
                self,"Error","No se pudo guardar el archivo"+
ubicacion
            )
    @Slot()
    def click_mostrar(self):
        self.ui.salida.clear()
        #self.particulas.mostrar()
        self.ui.salida.insertPlainText(str(self.particulas))


    @Slot() #Guardar los datos obenidos
    def click_agregarInicio(self):
        id = self.ui.leId.text()
        origenx = self.ui.leOrigenx.text()
        origeny = self.ui.leOrigenY.text()
        destinox = self.ui.leDestinoX.text()
        destinoy = self.ui.leDestinoY.text()
```

```
        velocidad = self.ui.leVelocidad.text()
        red = self.ui.sbRed.value()
        green = self.ui.sbGreen.value()
        blue = self.ui.sbBlue.value()
        #Crear particla
        particula =
Particula(int(id),int(origenx),int(origeny),int(destinox),int(destin
oy),int(velocidad),red,green,blue)
        self.particulas.agregar_inicio(particula)

    @Slot() #Guardar los datos obenidos
    def click_agregarFinal(self):
        id = self.ui.leId.text()
        origenx = self.ui.leOrigenx.text()
        origeny = self.ui.leOrigenY.text()
        destinox = self.ui.leDestinoX.text()
        destinoy = self.ui.leDestinoY.text()
        velocidad = self.ui.leVelocidad.text()
        red = self.ui.sbRed.value()
        green = self.ui.sbGreen.value()
        blue = self.ui.sbBlue.value()
        #Crear particla
```

```
        particulafinal =
Particula(int(id),int(origenx),int(origeny),int(destinox),int(destin
oy),int(velocidad),red,green,blue)

        self.particulas.agregar_final(particulafinal)
```

Particula.py

```
from .algoritmos import distancia_euclidiana

#.algoritmos para que la carpeta principal la tome

class Particula():

    def
__init__(self,id=0,origen_x=0,origen_y=0,destino_x=0,destino_y=0,vel
ocidad=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)



    @property
```

```python
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.__origen_x

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

@property
def velocidad(self):
    return self.__velocidad
```

```python
@property
def red(self):
    return self.__red


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


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


@property
def distancia(self):
    return self.__distancia


def __str__(self) -> str:
    return(
        "id: \t"+ str(self.__id)+'\n'+
        "origen_x: \t"+ str(self.__origen_x)+'\n'+
        "origen_y: \t"+ str(self.__origen_y)+'\n'+
```

```
            "destino_x: \t"+ str(self.__destino_x)+'\n'+
            "destino_y: \t"+ str(self.__destino_y)+'\n'+
            "velocidad: \t"+ str(self.__velocidad)+'\n'+
            "red: \t"+ str(self.__red)+'\n'+
            "green: \t"+ str(self.__green)+'\n'+
            "blue: \t"+ str(self.__blue)+'\n'+
            "distancia: \t"+ str(self.__distancia)+'\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
        }
```

particulas.py

```
import json
```

```python
from .particula import Particula

#.partiucla para que la carpeta principal la tome

class Particulas:

    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) -> str:

        return "".join(

            str(particula) + "\n" 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
    #Archivo
    def guardar(self,ubicacion):
        try:
            with open(ubicacion, 'w') as archivo:
                lista = [particula.to_dict() for particula in
self.__particulas]
                print(lista)
                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
```