Seminario de Algoritmia

CLAVE: I59556

NRC: 59556

2022B

D14

QTableWidget

Arellano Granados Angel Mariano

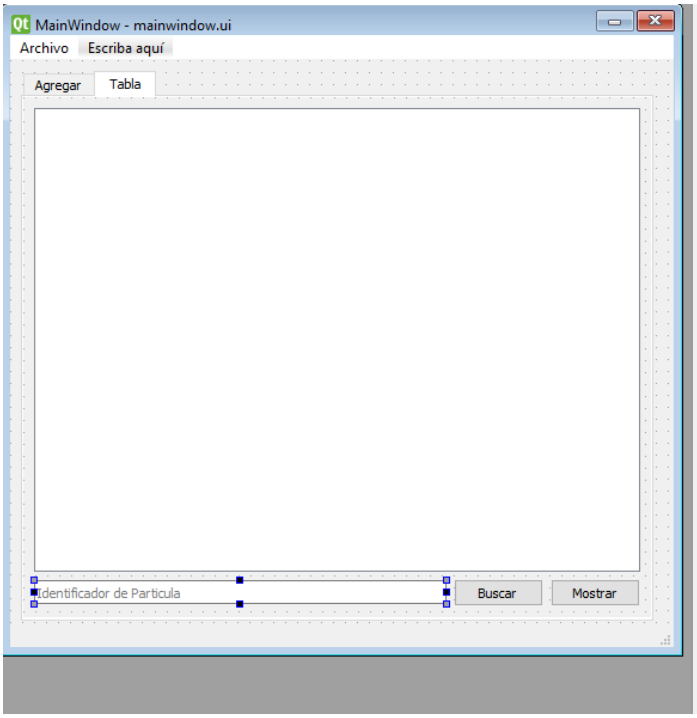
218123444

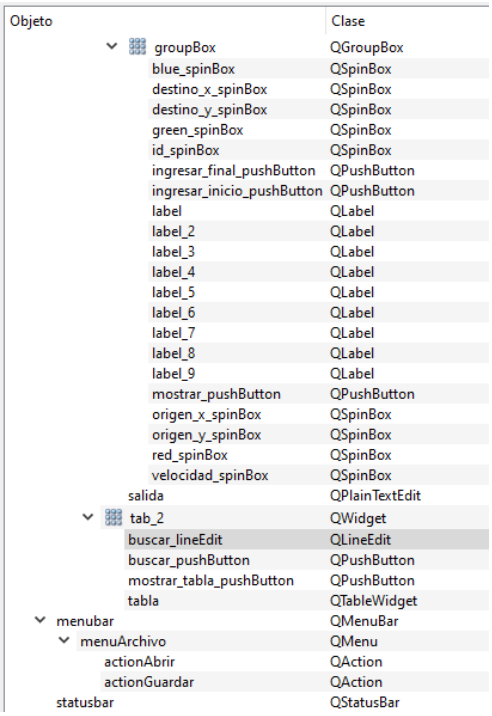
Descripción de la Actividad:

Agregar un widget de pestañas para alternar entre el menú de agregar y el nuevo menú de visualizar los registros en una tabla asi como poder buscar por el ID los registros.

Contenido de la Actividad:

UI usada:





Declaración de botones en mainwindow.py:

```
self.ui.mostrar_tabla_pushButton.clicked.connect(self.mostrar_tabla)
self.ui.buscar_pushButton.clicked.connect(self.buscar_id)
```

Función mostrar tabla en mainwindow.py:

```python
@Slot( )
def mostrar_tabla(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.admin))
    row = 0
    for particula in self.admin:
        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.tabla.setItem(row,0,id_widget)
        self.ui.tabla.setItem(row,1,origen_x_widget)
        self.ui.tabla.setItem(row,2,origen_y_widget)
        self.ui.tabla.setItem(row,3,destino_x_widget)
        self.ui.tabla.setItem(row,4,destino_y_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
```

Función buscar id en mainwindow.py:

```python
def buscar_id(self):
    id = self.ui.buscar_lineEdit.text()

    encontrado = False
    for particula in self.admin:
        if int(id) == particula.id:
            self.ui.tabla.clear()
            self.ui.tabla.setRowCount(1)

            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.tabla.setItem(0,0,id_widget)
            self.ui.tabla.setItem(0,1,origen_x_widget)
            self.ui.tabla.setItem(0,2,origen_y_widget)
            self.ui.tabla.setItem(0,3,destino_x_widget)
            self.ui.tabla.setItem(0,4,destino_y_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'La particula con el identificador {(id)} no fue encontrado'
    )
```

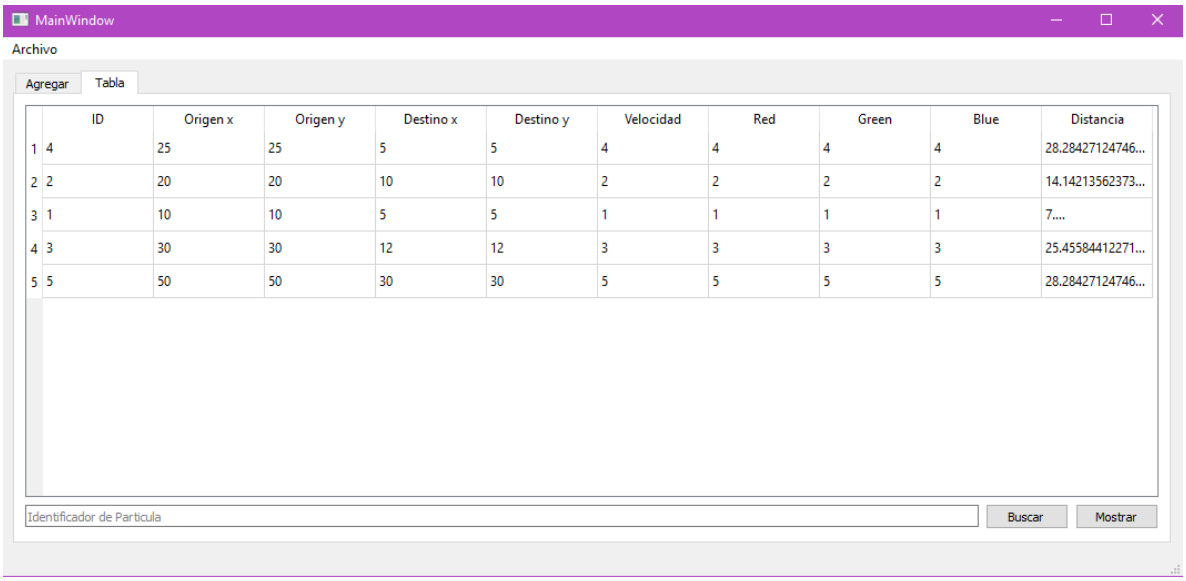
Funciones nuevas para iterar los objetos de la clase administradora:

```
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
```

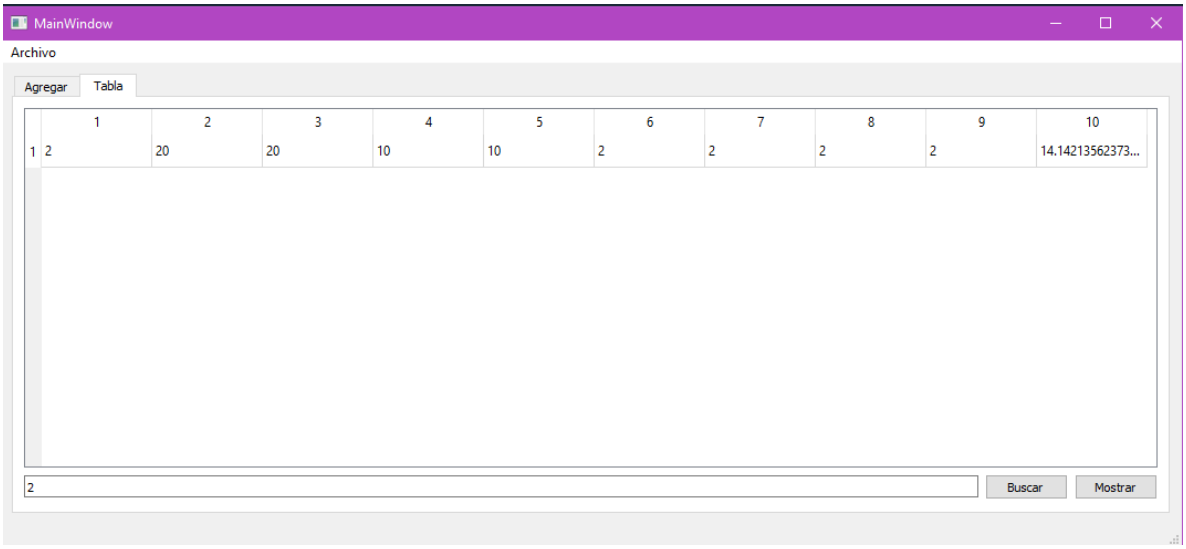
Funciones para acceder a atributos privados de la clase partícula:

```
@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 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
```

Mostrar registros guardados en JSON anteriormente:



Buscar y encontrar registro de id 2:



Buscar id no existente y no encontrarlo: