|  |
| --- |
| **PROYECTO 2 LABORATORIO – SISTEMA DE GUATERIEGOS 2.0** |
| **201906795 – Javier Ricardo Yllescas Barrios** |

**Resumen**

El programa presentado propon el desarrollo un sistema de procesado de archivos XML, utilizando la estructura a traves de Tipos de Datos Abstractos (TDA), no esta permitido el uso de estructuras de Python como list, dic, tuple.

Utilizando el framework Flask se creo una interfaz del usuario donde puede subir archivos XML con las instrucciones de riego, agua y fertilizante a usar por planta, una lista con plantas luego sera procesado en el Back-end (archivos python) con clases para generar segmentacion archivos XML, un resumen en un archivo HTML el cual se puede visualizar en el navegador

La segmentacion toma un listado de instrucciones indicando el movimiento de los drones, moviendolos atras, adelante o esperar si llego a su posicion, deben seguir el orden de instrucciones y regar cuando se inidique si los drones cada segundo reciven una nueva insruccion.

Los reporte HTML generados son estaticos y se muestran utilizando la interfaz generada en Flask.

La salida.xml muestra los movimientos del reporte HTML y los segmenta en el orden asignado mostrando cada segundo los movimientos del dron y el resumen del agua y fertilizante en el invernadero

**Palabras clave**

Proyecto2, listas con apuntadores, cola con apuntadores, USAC.

**Introducción**

Este programa se centra en la gestión y análisis de datos provenientes de un archivo XML con las instrucciones para cada dron, a través de la lectura y el procesamiento de información registrada.

Muestra el total de agua y fertilizante utilizado tablas y archivos html esta herramienta permiten automatizar procesos complejos y apoyar la toma de decisiones de manera más eficiente y confiable.

A diferencia de las estructuras tradicionales de Python, como listas o arrays, este programa utiliza clases propias de listas y colas. Estas estructuras personalizadas facilitan un manejo más organizado de los datos y permiten realizar operaciones específicas para agrupar información y generar resultados claros.

El sistema comienza con la lectura de un archivo XML que contiene las instrucciones de los drones y fertilizante y agua a utilizar por movimiento. La información se segmenta y se almacena en nodos y listas adaptadas, lo que asegura un manejo ordenado y comprensible de los datos. Posteriormente, se construyen reportes, reflejando de manera estructurada el comportamiento de cada dron.

Uno de los objetivos principales del programa es agrupar es genera un reporte seleccionando un tiempo especifico para ver que posicion llego el dron y que instrucciones le hacen falta para terminar

**Desarrollo del tema**

Para una mejor compresion del proyecto se aplicaron los siguientes archivos para desarrollarlo.

**Estructura del Proyecto 1 [Archivos]**

—>Nodos

- Clases.py

- Cola.py

- Lista.py

- Nodo.py

—>SistemaArchivos

- SistemaArchivoSalidaHTML.py

- SistemaArchivosSalidaXML.py

- SistemaArchivoTDAs.py

-SistemaArchivoXML.py

—>SistemaRiegos

- SistemaRiegos.py

—>templates

-index.html

- salidaH.html

Donde:

**Nodo/Clase.py**

Guarda todos los clases utilizados en el programa, en lugar de usar fila, tuplas o diccionarios.

**Nodo/Cola.py**

la clase cola utilizada para guardar las clases.

**Nodo/Lista.py**

La clase almacena la lista utilizada para guardar ciertas clases.

**Nodo/Nodo.py**

Se creo el nodo que almacena los apuntadores con un valor y un apuntador siguiente utilizado en cola y fila

**SistemaArchivos/SistemaArchivoSalidaHTML.py:** Genera todo los HTML reporte

**SistemaArchivos/SistemaArchivosSalidaXML.py**

Sistema genera todo lo relacionado con el archivo salida.xml

**SistemaArchivos/SistemaArchivoXML.py**

Lee los archivos xml y los almacena en memoria temporal.

**SistemaArchivos/SistemaArchivoTDAs.py**

Genera las graficas en graphviz

**SistemaRiegos/SistemaRiegos.py**

Es el sistema que procesa la instrucciones para cada dron generando las colas y listas para mostrar los resumenes

**templates/index.html**

Muestra la interfaz de usuario para subir y procesar archivos xml

**templates/salidaH.html**

muestra el reporte con los movimientos de todo los invernaderos y todos los planes

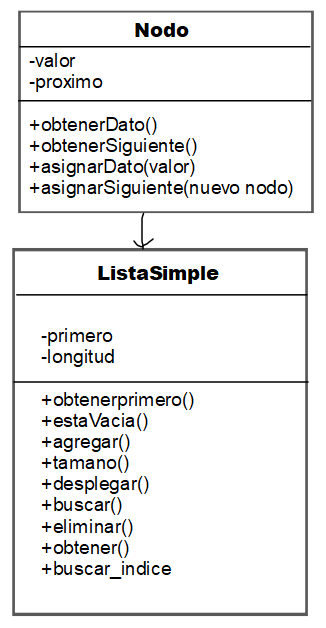
**SistemaArchivos/SistemaArchivoTDAs.py**

Genera las graficas en graphviz

**Listas**

Al desarrollar la practicas se evitaron usar listas y arreglos de python por lo cual se crearon clases especiales para simular listas aplicando los conceptos aprendidos en clase como son los nodos, apuntadore, clases en python y herencia.

Se tuvo que implementar la siguiente estructura para las listas



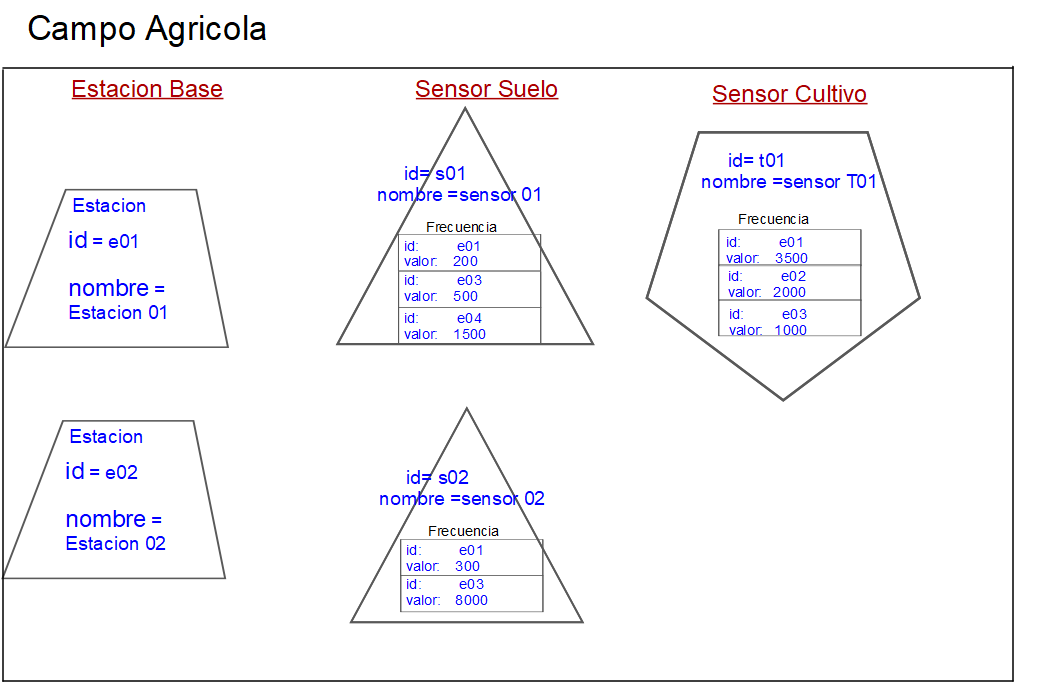
Donde:

**Nodo:** cada elemnto de la lista es un objeto Nodo, que guarda un valor y una referencia al siguiente Nodo.

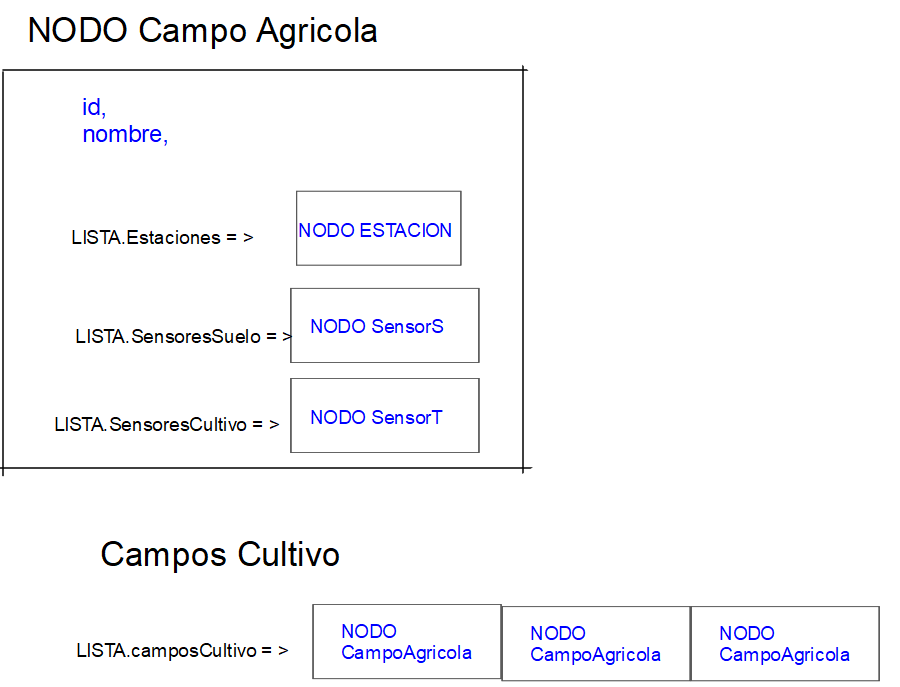
**ListaSimple:** es la estructura principal que administra los Nodos, almacenandolos en primero. Se va guiando atraves de los punteros de Nodo y las funciones de la misma.

**CampoAgricola**

Antes de seguir agregando nodos la siguiente imagen es un diagrama a mano alzada para explicar la estructura del nodo mayor que contiene los datos del campo.

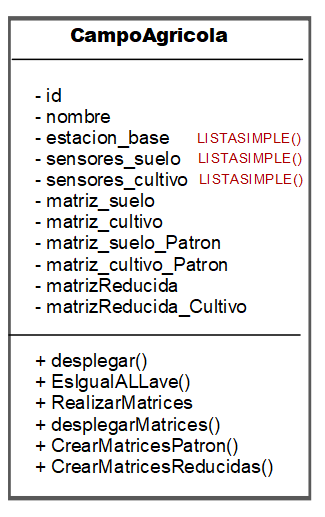


Describiendo seria 3 listas principales serian 1) Estaciones base, 2) Sensores Suelo y 3) sensores cultivo, estas listas serian los pilares para desarrollar el proyecto.



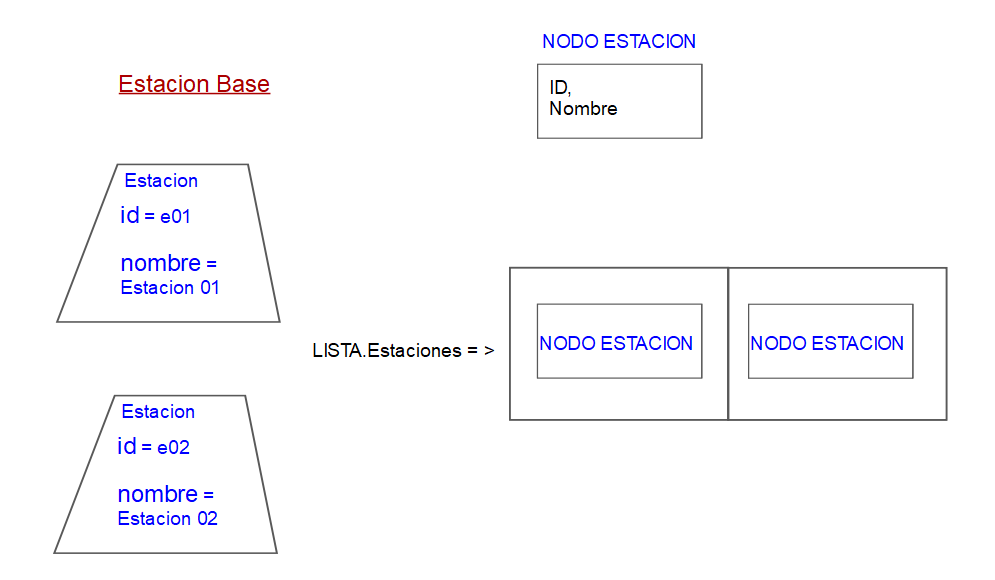
Tomaria el id y el nombre junto con las listas agrupadas en un nodo respectivamente para cada lista.

Luego serian agrupado en en una lista campos cultivo donde estarian los nodos campo agricola

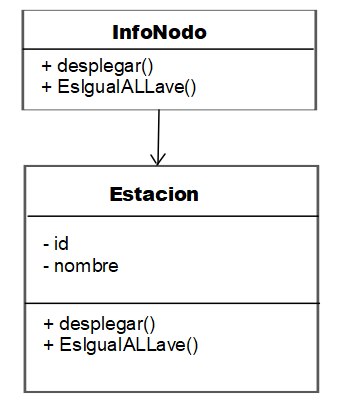


**Lista Estaciones base**

La estructura general para la lista estaciones base seria.

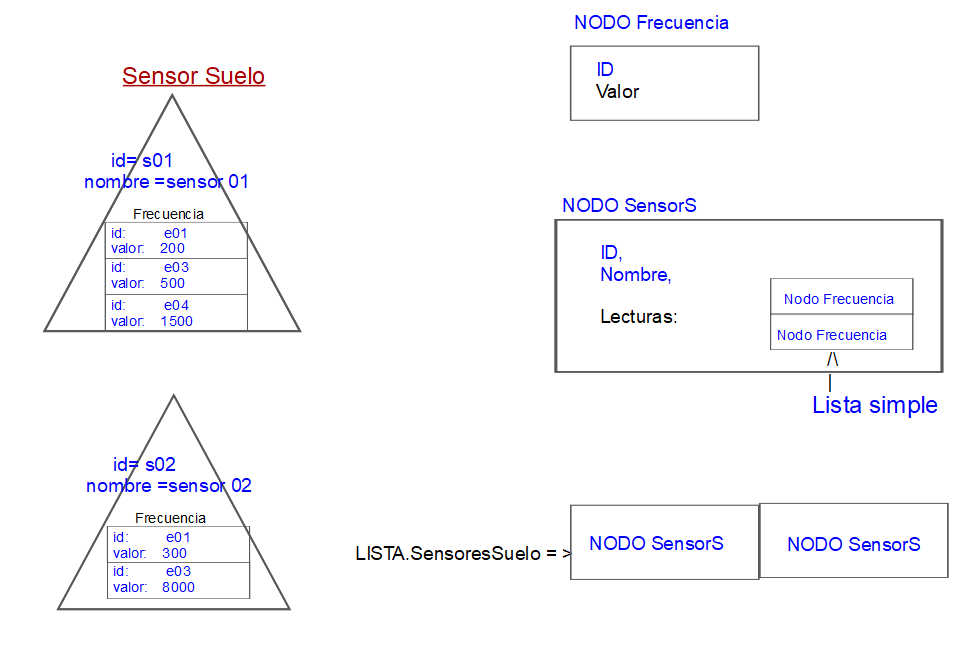


Contaria con nodo estacion con id y nombre de la estacion

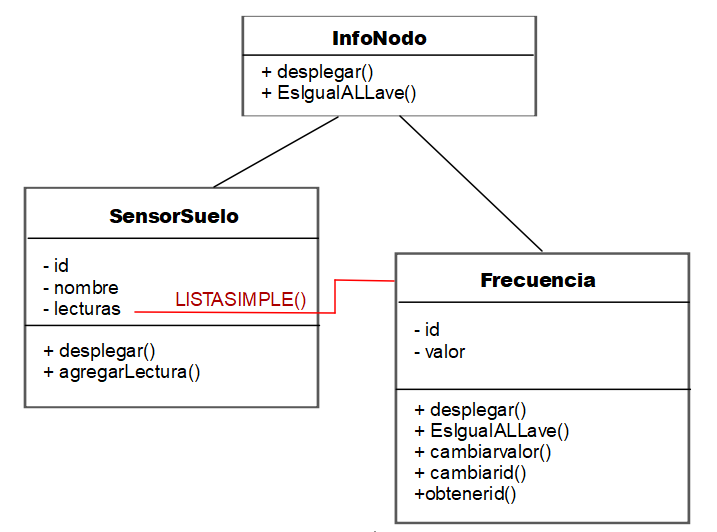


**Lista Sensores Suelo**

La estructura general para la lista sensorse suelo.

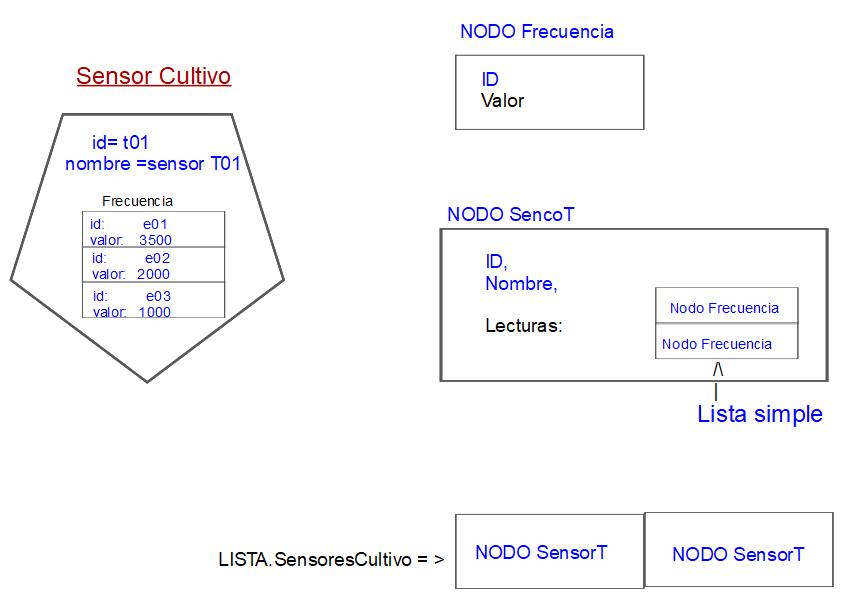


Cuenta con unnodo de frecuencias que se almacena dentro de una lista de lectura para luego agregarla al nodo SensorS y pasarlo a la Lista sensores suelo.

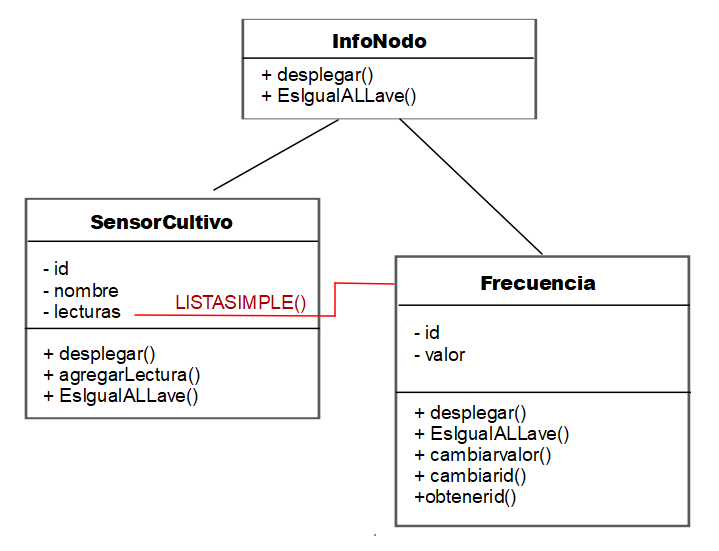


**Lista Cultivo**

Estructura general para la lista sensor cultivo.

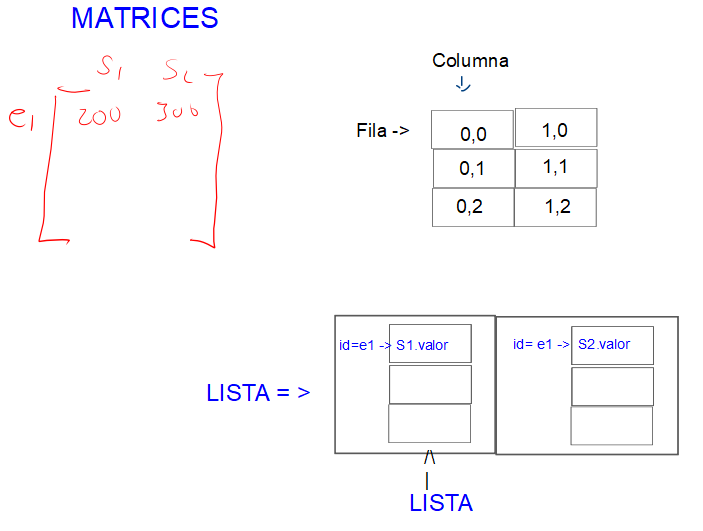
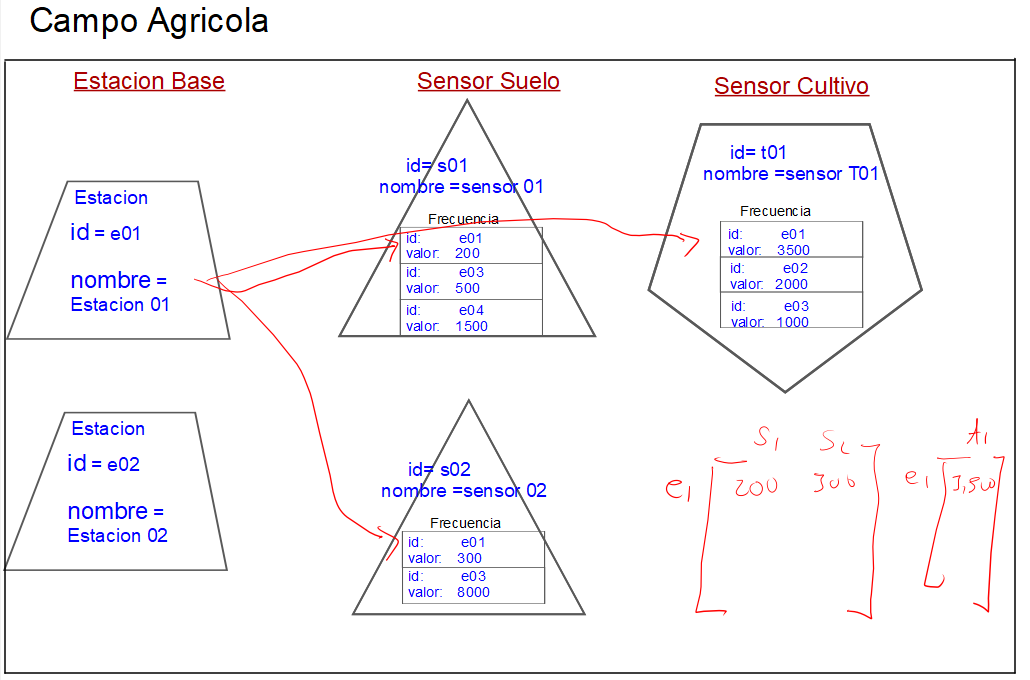


Su nodo primario es el de frecuencias, tambien se almacenan en lecturas y luego en un nodo sensor cultivo para guardarlas en una lista sensores cultivos.



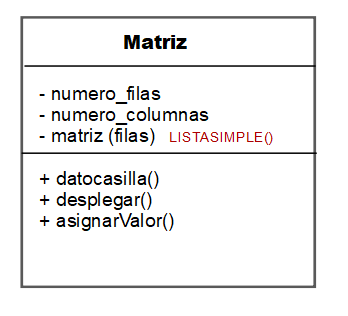
**Matrices**

Para el agrupamiento de los datos en matrices se aplico el siguiente diagrama

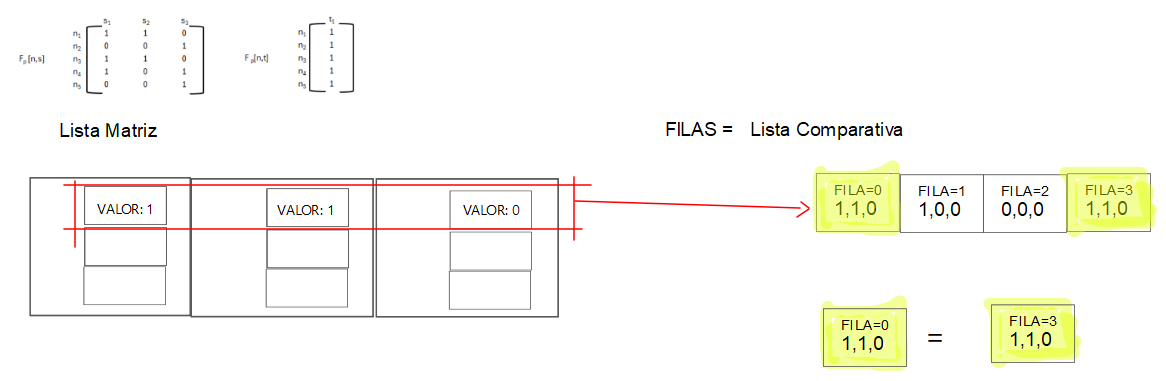


Se crearon las matrices a partir de dos listas la primera almacena las filas y luego se agrega a otra lista para que cuenten como columnas.

Se crearon clases matrices para acceder cada valor como si fuera un excel dando su ubicacion numero de columna y numero de fila y obtener sus valores ubicados en esa casilla especifica.



**Construcción matrices reducidas**



Se utilizaron listas auxiliares para alamcenar temporalmente los datos agrupados.

Primero se obtuvo los primeros valores de cada columna para luego concatenar sus valores en un texto para luego realizar asi con todas la filas.

Luego la lista auxiliar se compara cada nodo si los textos eran iguales entonces las filas eran iguales.

**Como ejecutar el Proyecto 1.**

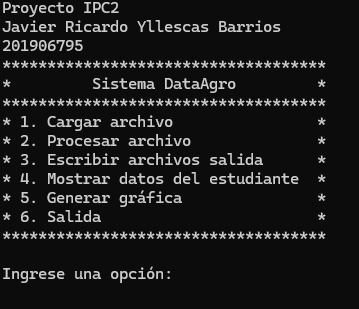
Para ejecutar el proyecto necesitas tener python instalado y saber usar CMD

Paso 1: Busca la carpeta donde este el documento main.py

Paso 2: Ejecuta CMD apartir de esa ruta del sistema

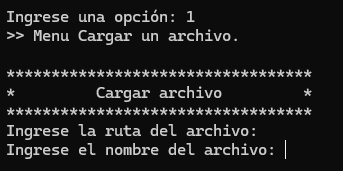
Paso 3: Colocar en las lineas de comando el siguiente ocmando “**python main.py**” luego presiona Enter.

Veras una ventana parecida a esta



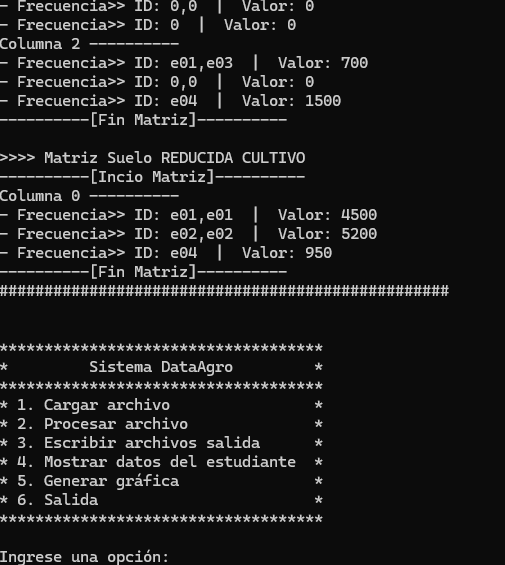
Paso 4:Ingres la opcion deseada del 1 al 6.

**Cargar archivo**



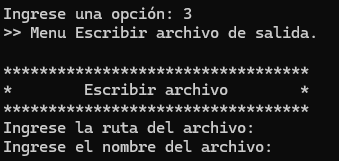
Apareceran dos opciones una ingresando la ruta donde esta el archivo y otra con el nombre de archivo incluyendo la extension xml ejemplo(archivo.xml)

**Procesar archivos**



Muestra en la consola todos los procesos para generar las listas y matrices.

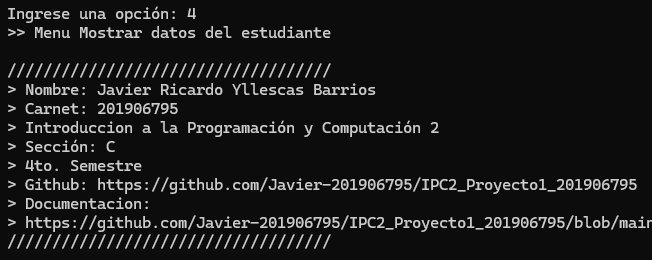
**Escribir archivos salida**



Pedira la ruta y nombre del archivo (coloca la extension xml al archivo) luego generar un archivo con los datos procesados.

**Mostrar datos del estudiante**

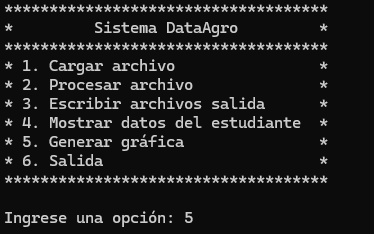
Opcion 4 muestra los datos del estudiante en las consola



**Generar grafica**

Opcion 5 genera un grafica

Para que se genera la grafica es necesario tener instalado **Graphviz**



Y la ultima opcion es la 6 que finaliza el programa.

**Conclusiones**

Las listas enlazadas son una alternativa a los arrays, con mucha mas facilidad y un manejo de la informacion con un seguimiento constante de la ubicacion de los datos.

Al crear clase nodos se pueden almacenar y procesar los datos de una manera mas ordenada eficiente proque se pueden evitar ciclos repetitivos sin llamar a funciones externas.

La aplicacion de los conceptos de listas y nodos resulto muy enriquecedor, pero su nivel de complegidad va aumentando a la hora de almacenar muchos nodos dentro de listas, dentro de otras listas, llevando un acceso mas lento a la informacion y una complejidad para procesar los datos.

**Comentarios:**

Al desarrollar el programa sin el uso de listas y aplicando la tecnica de apuntadores resulto un reto a al hora de mantener la logica en que nodo esta ubicado, ya que despues de almacenar los datos en varias listas con sus nodos respectivos y la opciones de obtener siguiente no daba mucha informacion solo a la hora de depurar el codigo se podia saber en que nodo se encontraba manejando.

**Referencias bibliográficas**

Universidad de San Carlos de Guatemala. (s.f.). Enunciado proyecto 1. [https://uedi.ingenieria.usac.edu.gt/campus/pluginfile.php/258349/mod\_resource/content/1/%5BIPC2%5DProyecto202502\_v2.pdf](https://uedi.ingenieria.usac.edu.gt/campus/pluginfile.php/258349/mod_resource/content/1/%5BIPC2%5DProyecto202502_v2.pdf?utm_source=chatgpt.com)

Argueta, Hesban. (s.f.). *Clases laboratorio IPC2* [Repositorio en GitHub]. GitHub. [https://github.com/Hes-007/IPC2-2S2025/tree/main](https://github.com/Hes-007/IPC2-2S2025/tree/main?utm_source=chatgpt.com).

Ruiz Juarez, J. M. (s.f.). Contenido Unidad 1. <https://uedi.ingenieria.usac.edu.gt/campus/course/view.php?id=2547>

Graphviz. (s.f.). Documentacion para crear graficas. <https://graphviz.org/documentation>