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

-1_Grafica.dot

-entrada.xml

- salida.xml

- SistemaCentral.py

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.

1_Grafica.dot

Se genera temporalmente el archivo dot con la inforamcion a graficar.

1_TDAs.png

Muestra imagen genera por el archivo dot y el sistema TDAs.

entrada.xml

Este archivo cambia cada vez que se sube un nuevo archivo xml.

salida.xml

Después de procesar toda la información se almacena un resumen en este archivo.

SistemaCentral.py

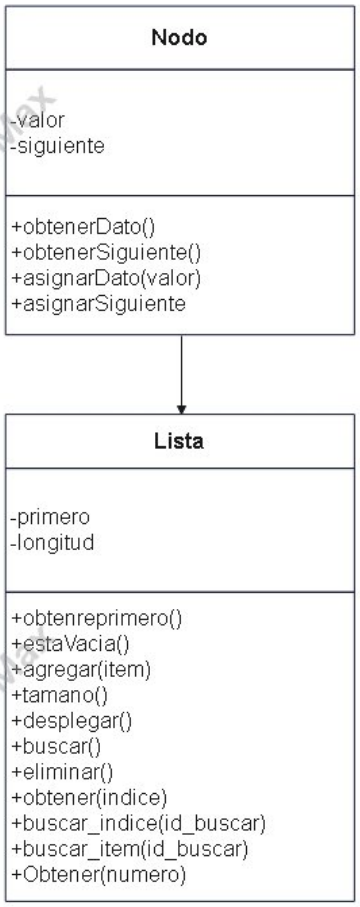
Reune todos los sistemas y define funciones sencillas para ser llamada por Flask

Diagrama de Clases

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, apuntadores, clases en python y herencia.

Se tuvo que implementar la siguiente estructura para las listas



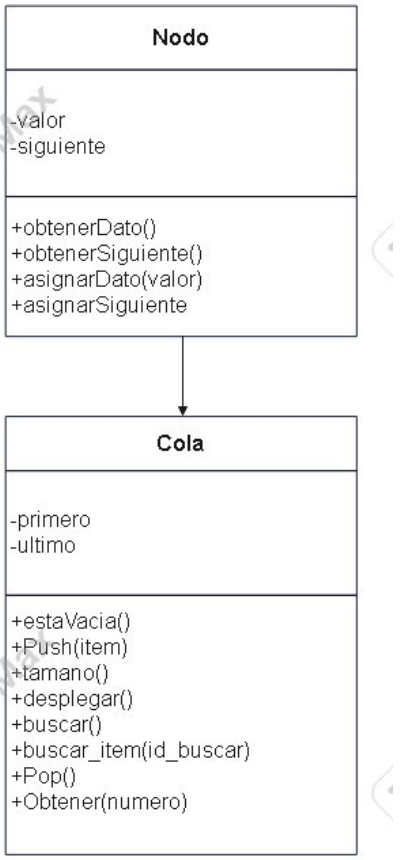
Donde:

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

Lista: 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.

Cola

Son similares a las listas pero su peculiaridad es que el primer objeto en entrar es el primero en salir facilitando la extracción de los elementos

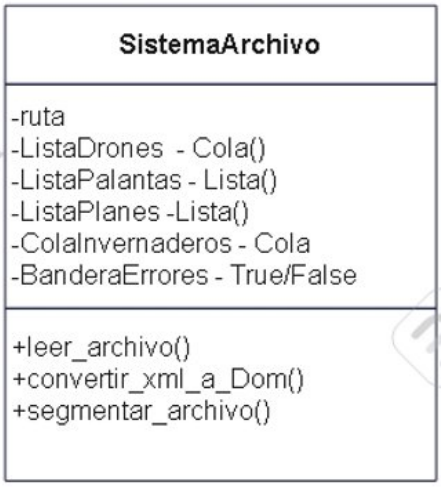


SistemaCentral

Tiene funciones para facilitar la obtencion de datos para Flask

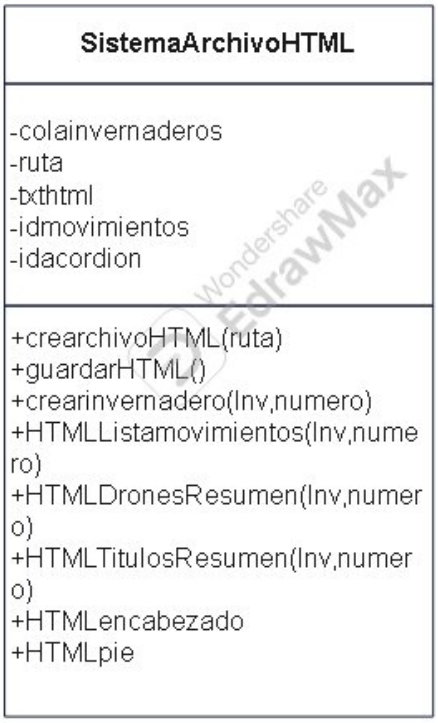
SistemaArchivoXML

extraer informacion del archivo entrada XML



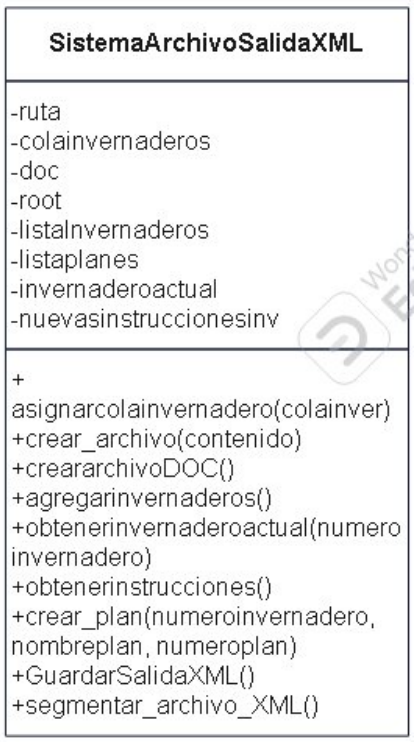
SistemaArchivoHTML

Sistema para generar reportes HTML



Sistema Salida archivos XML

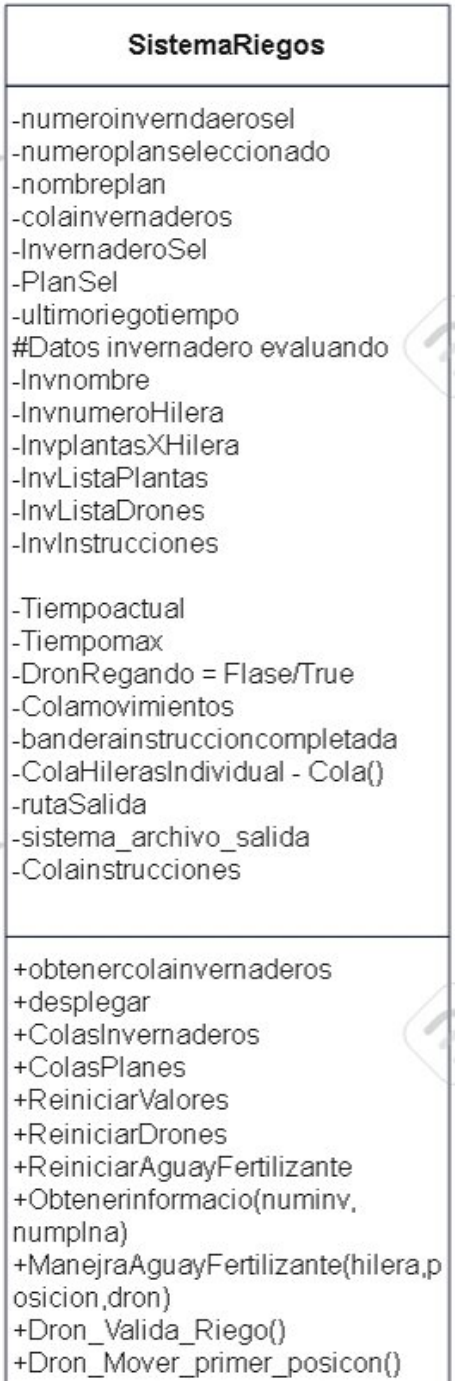
Sistema genera todo lo relacionado con el archivo salida.xml



Sistema Archivo TDA

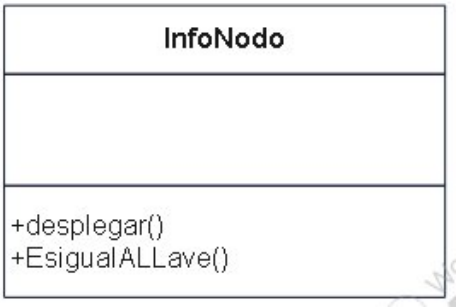
Genera los graficos en graphviz

SistemaRiegos



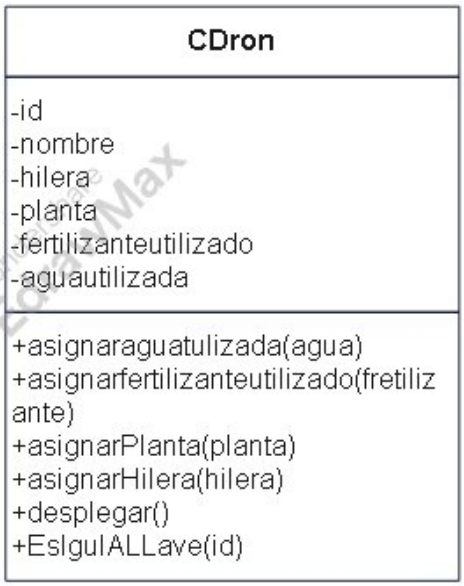
Clases.py

Son las clases para guardar todo el documento en lugar de listas, tuplas u otro forma.

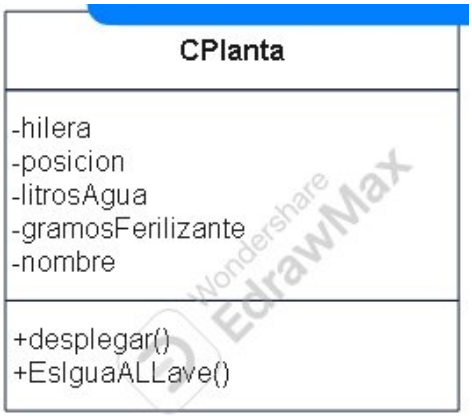


Es el padre que herada las funciones desplegar() y EsigualALlave() a todas las demas clases

Almacena la informacion del dron



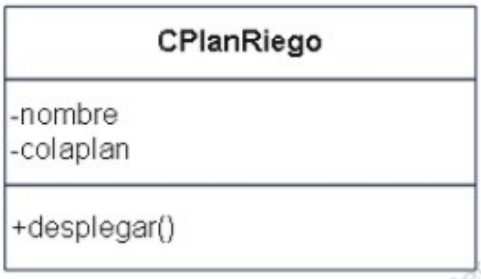
Guarda la informacion de la planta



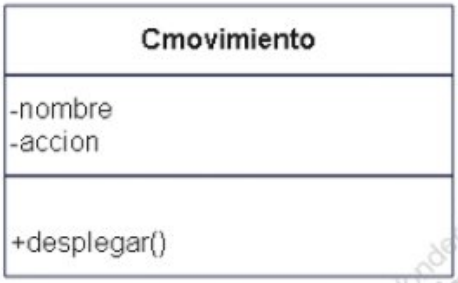
Guardar informacion plan

Guarda toda la informacion del invernadero

Guarda informacion plan riego

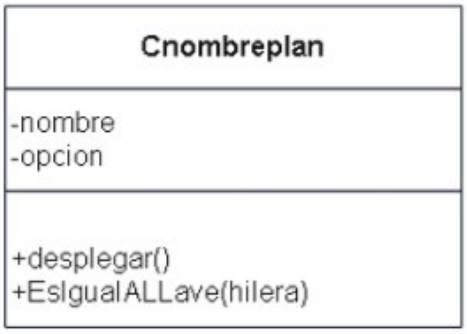


Guarda la informacion moviemiento



Clase temporal para guardar el nombre del invernadero para mostrar solo el nombre

Clase temporal para guardar solo el nombre del plan para poderlo listar mas facilmente



Clase que almacena los movimientos

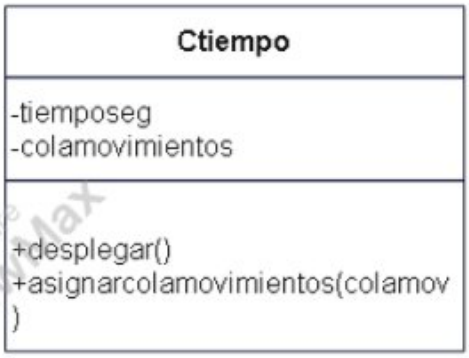
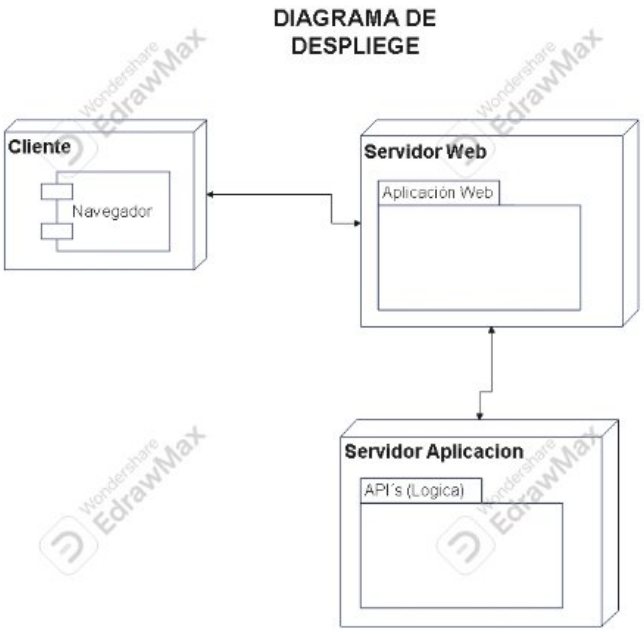


Diagrama de Despliegue aplicación.



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 app.py

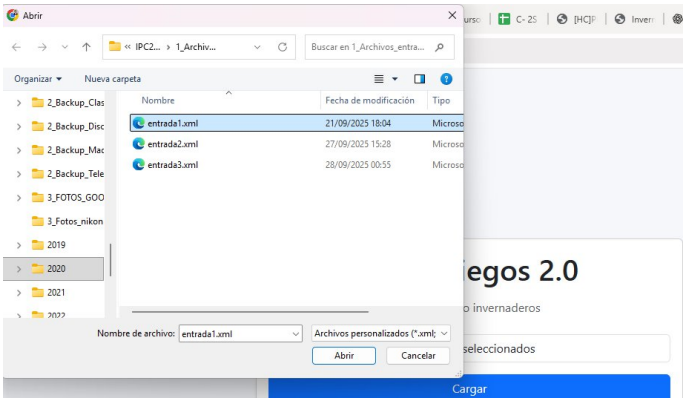
Paso 2: Ejecuta CMD apartir de esa ruta del sistema

Paso 3: Colocar en las lineas de comando el siguiente ocmando "**python app.py**" luego presiona Enter.

Paso 4: Accede a tu navegador preferido e ingresa al siguiente link http://127.0.0.1:4000/
Aparecera la siguiente ventan

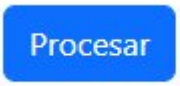
Paso 5: Cargar un archivo XML

Click en selecionar archivo y sube el archivo



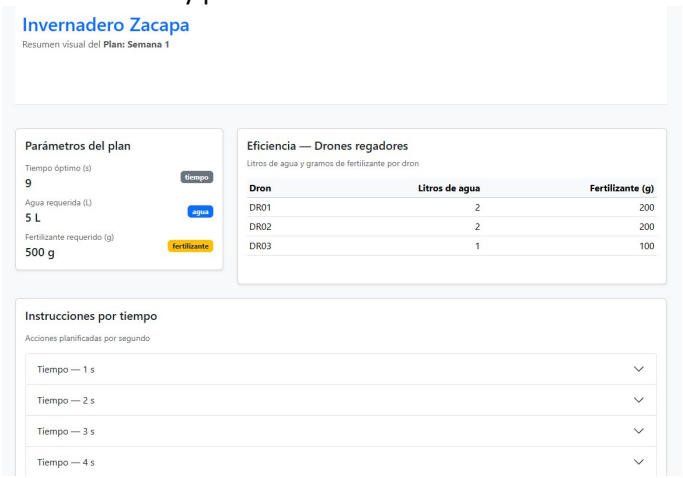
Paso 6: Preciona el boton cargar

Paso 7:Luego click en procesar



Paso 8:Para ver resumen del archivo click en ver resumen

Rediccionara a la siguiente ventana con la inforamacion del invernader y plan de todo el documento XML



Donde podrás ver el tiempo exacto cada movimiento

En la carpeta del proyecto se puede ver salida.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<datoSalida>
 <listaInvernaderos>
  <invernadero nombre="Invernadero Zacapa">
   <listaPlanes>
    <plan nombre="Semana 1">
     <tiempoOptimoSegundos>9</tiempoOptimoSegundos>
     <aguaRequeridaLitros>5</aguaRequeridaLitros>
     <fertilizanteRequeridoGramos>500</fertilizanteRequeridoGramos>
     <eficienciaDronesRegadores>
      <dron nombre="DR01" litrosAgua="2" gramosFertilizante="200"/>
      <dron nombre="DR02" litrosAgua="2" gramosFertilizante="200"/>
      <dron nombre="DR03" litrosAgua="1" gramosFertilizante="100"/>
     </eficienciaDronesRegadores>
     <instrucciones>
      <tiempo segundos="1">
       <dron nombre="DR01" accion="Adelante (H1P1)"/>
       <dron nombre="DR02" accion="Adelante (H2P1)"/>
       <dron nombre="DR03" accion="Adelante (H3P1)"/>
      </tiempo>
      <tiempo segundos="2">
       <dron nombre="DR01" accion="Adelante (H1P2)"/>
       <dron nombre="DR02" accion="Esperar  (H2P1)"/>
       <dron nombre="DR03" accion="Adelante (H3P2)"/>
      </tiempo>
      <tiempo segundos="3">
       <dron nombre="DR01" accion="Regar    (H1P2)"/>
       <dron nombre="DR02" accion="Esperar  (H2P1)"/>
       <dron nombre="DR03" accion="Adelante (H3P3)"/>
      </tiempo>
      <tiempo segundos="4">
       <dron nombre="DR01" accion="Adelante (H1P3)"/>
       <dron nombre="DR02" accion="Regar    (H2P1)"/>
       <dron nombre="DR03" accion="Esperar  (H3P3)"/>
```

Conclusiones

Las listas y colas son una alternativa a los arrays, list, diccionarios o tuplas, con mucha mas flexibilidad y un manejo de la informacion mas personalizado 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 eficiente en memoria ya que el mismo archivo apuntando a un elemento en memoria va optimizando el proyecto si va creciendo

Flask al ser de un framework basado en python es muy sencillo para ejecutar toda nuestra logica de back-end

.

Comentarios:

La aplicacion de los conceptos de listas, colas y nodos resulto muy enriquecedor, pero su nivel de complegidad va aumentando a la hora de almacenar muchos nodos dentro de ellas, el manejo de la informacion parece un poco repetitivo y no se pueden crear funciones para todo porque se termina formando duplicados de las mismas y la modificaciones se generan imposibles si no se usa bien los try exception para identicar donde ocurrieron los errores.

Referencias bibliográficas

Universidad de San Carlos de Guatemala. (s.f.). Enunciado proyecto 1. https://uedi.ingenieria.usac.edu.gt/campus/pluginfile.php/265700/mod_resource/content/1/%5BIPC2%5DProyecto2_202502_v2.pdf

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

Ruiz Juarez, J. M. (s.f.). Contenido Unidad 2 y 3. https://uedi.ingenieria.usac.edu.gt/campus/course/view.php?id=2547

Flask (s.f) Documentacion para uso framework Flask https://flask.palletsprojects.com/en/stable/tutorial/

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