Universidad de San Carlos de Guatemala
Escuela de Ingeniería en Ciencias y Sistemas, Facultad de Ingeniería
Introducción a la programación y computación 2, 2do. Semestre 2025

# PROYECTO 3 LABORATORIO – SISTEMA TECNOLOGIAS CHAPINAS, S.A

**201906795 – Javier Ricardo Yllescas Barrios**

**Resumen**

El programa presentado propone el desarrollo un sistema de facturacion para servicioes en la nube, compuesto por un frontend en Django y un backend en Flask. El sistema procesa datos mediante archivos XML para gestionar recursos, clientes y configuraciones, calculando consumos y generando facturas y reportes.

El objetivo de la aplicacion es usar paradigmas de programacion orientada a objetos para la construccion del software, la utlilizacion de base datos XML y la interaccion entre dos aplicaciones con diferenes tecnologias utilizando el protocolo HTTP para interactuar con el backend.

**Palabras clave**

Proyecto2, Sistema Frontend (Django), Sistema Backend (Flask), USAC, IPC2.

## Introducción

El presente proyecto tiene como objetivo desarrollar una solución tecnológica integral para la empresa Tecnologías Chapinas, S.A., que le permita administrar su portafolio de servicios de nube y realizar los procesos de facturación correspondientes al consumo de recursos por parte de sus clientes.

La solución propuesta implementa una arquitectura moderna que combina un frontend web desarrollado con Django, que servirá como interfaz de gestión, y un backend construido con Flask que proveerá servicios API para el procesamiento de datos. El sistema utilizará archivos XML para la comunicación y persistencia de datos, aplicando los principios de la programación orientada a objetos y expresiones regulares para el procesamiento de información.

## Desarrollo del tema

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

### Estructura del Proyecto [Archivos]

– DOCUMENTACION_IPC2_Proyecto3_201 90695.pdf

—>ArchivosdePrueba

—>Backed

- AppFlask.py
- ArchivoConfiguraciones.xml
- ArchivoConsumos.xml
- entrada.xml

- entradaconsumos.xml

—>Sistemas

-SistemaCentral.py

-SistemaLecturaXML.py

-SistemaLecturaXMLconsumos.py

-SistemaSalidaXML.py

-SistemaSalidaXMLconsumos.py

-SistemaValidaciones.py

—>Clases

-ArchivoConfiguracion.py

-ArchivoConsumos.py

—>Frontend

—>app

-__init__.py

-admin.py

-apps.py

-models.py

-tests.py

-urls.py

-views.py

—>migrations

—>static

—>templates

—>Frontend

(Generados por Django)

## Funcionamiento Frontend

Donde:

**—>app**

Guarda todos los relacionado a la aplicacion en Django.

**app/urls.py**

Estan todas las Rutas y asignacion de sus funciones

**app/views.py**

Funciones para captaurar datos y renderizar html

## Funcionamiento Backend

**—>Backed**

Almacena todo lo realcionado a la aplicacion con flask

**Backed/AppFlask.py**

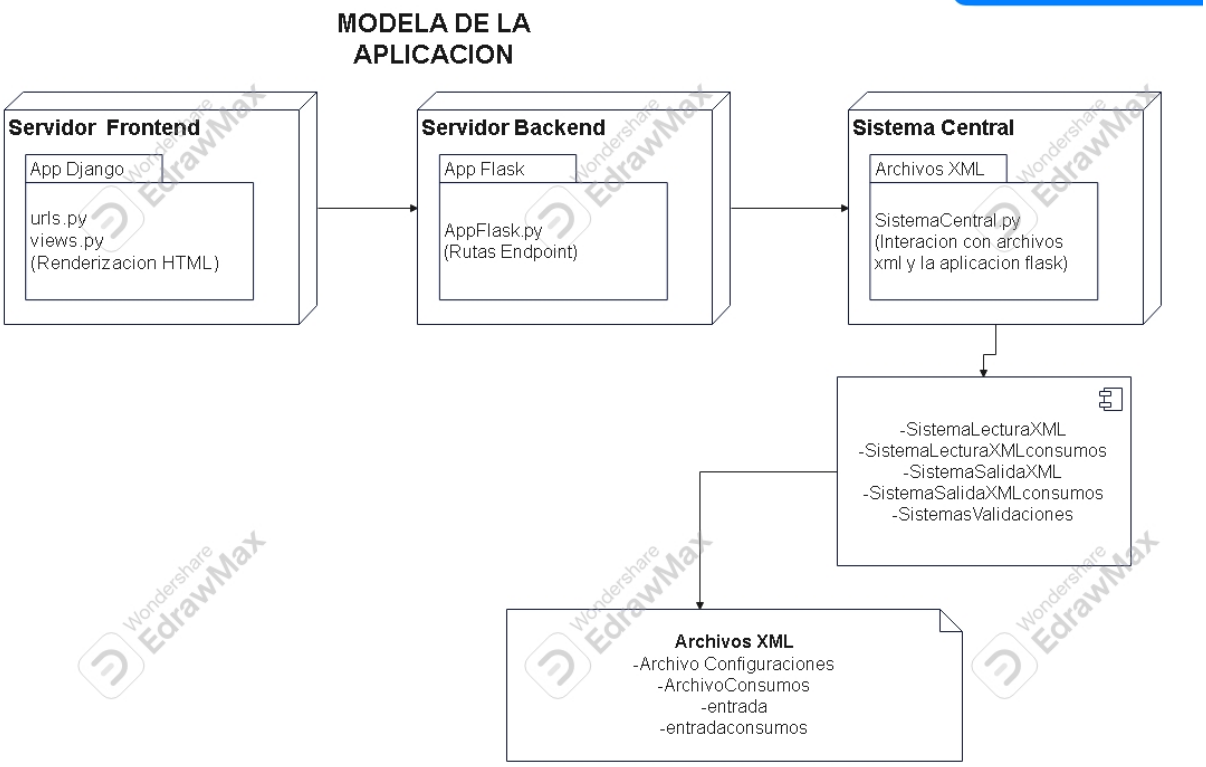
Contiene todas las rutas para los endpint para la interaccion con la aplicaion

**—>Backed/Sistemas**

Estan todos los sistemas para ejecutar la aplicacon con la base de datos XML.

**Backed/Sistemas/SistemaCentral.py**

Es el encargado de recibir de flask las instrucciones para poder interactura con los archivos XML y obtener las clases y creacion, eliminacion y moficacion de los archivos XML


MODELA DE LA APLICACION
Servidor Frontend
App Django
urls.py
views.py
(Renderizacion HTML)
Servidor Backend
App Flask
AppFlask.py
(Rutas Endpoint)
Sistema Central
Archivos XML
SistemaCentral.py
(Interacion con archivos xml y la aplicacion flask)
-SistemaLecturaXML
-SistemaLecturaXMLconsumos
-SistemaSalidaXML
-SistemaSalidaXMLconsumos
-SistemasValidaciones
Archivos XML
-Archivo Configuraciones
-ArchivoConsumos
-entrada
-entradaconsumos

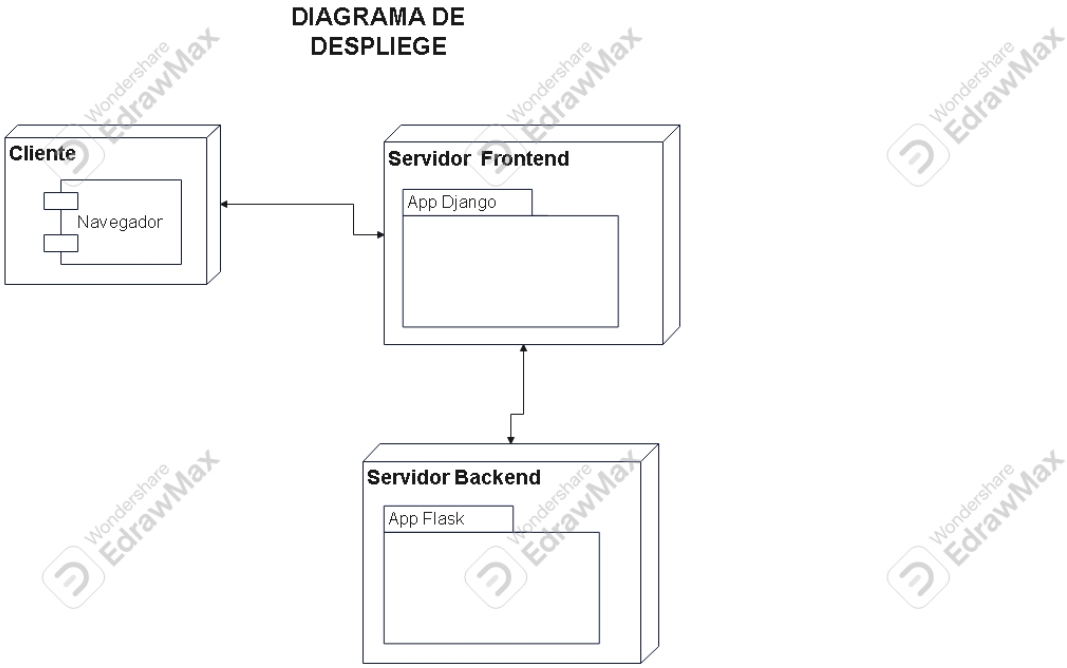
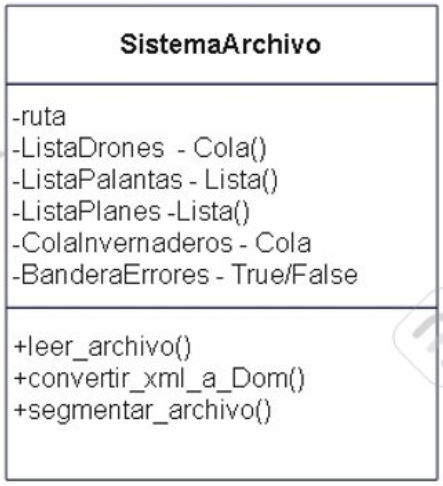
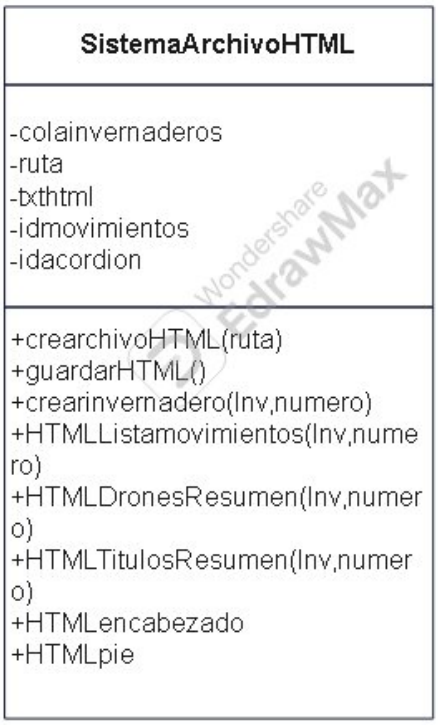


DIAGRAMA DE DESPLIEGE
Cliente
Navegador
Servidor Frontend
App Django
Servidor Backend
App Flask

## Estructura Clases



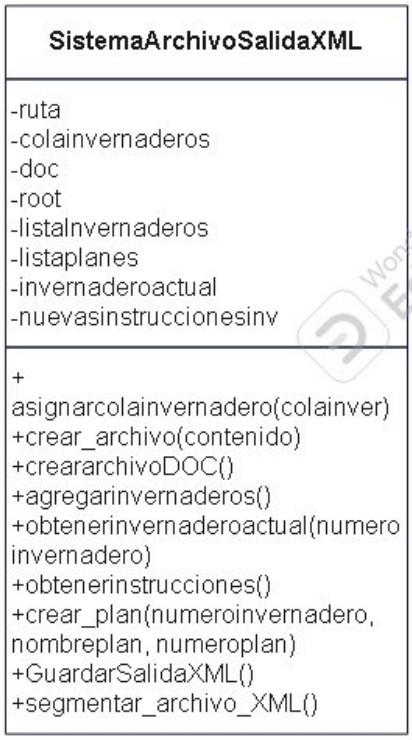




### Sistema Salida archivos XML

Sistema genera todo lo relacionado con el archivo salida.xml

### SistemaArchivoHTML

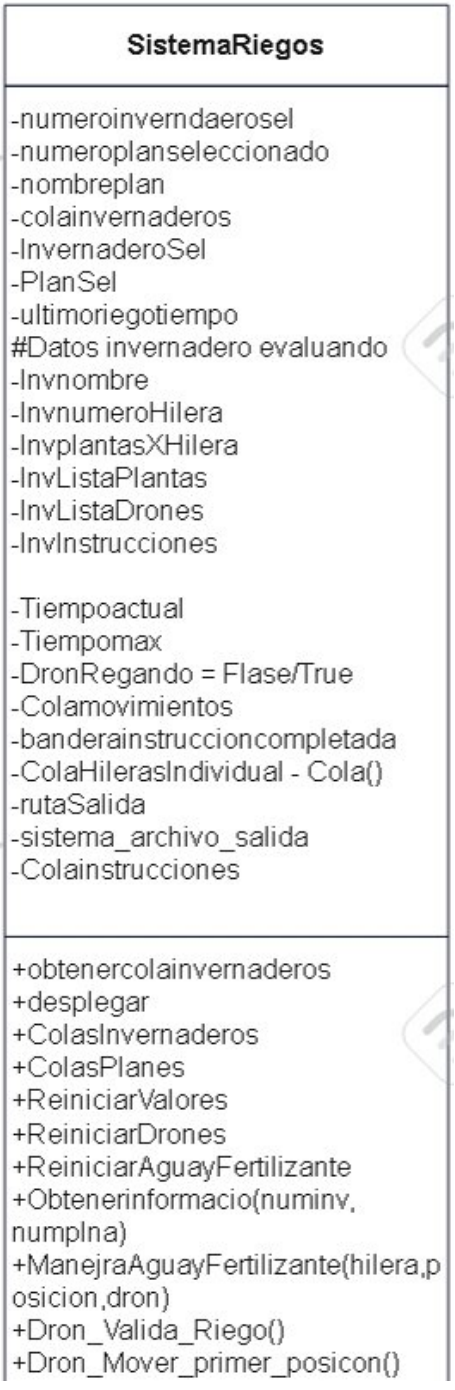
Sistema para generar reportes HTML

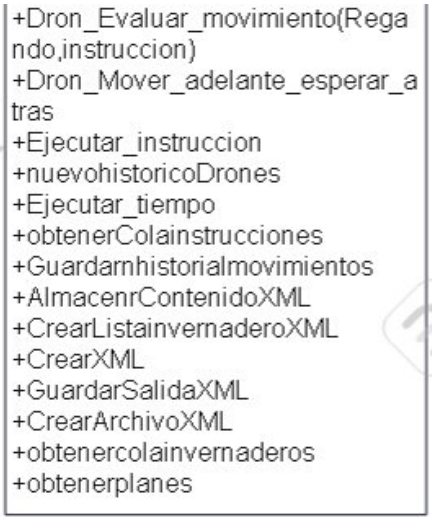



## Sistema Archivo TDA

Genera los graficos en graphviz


## SistemaRiegos



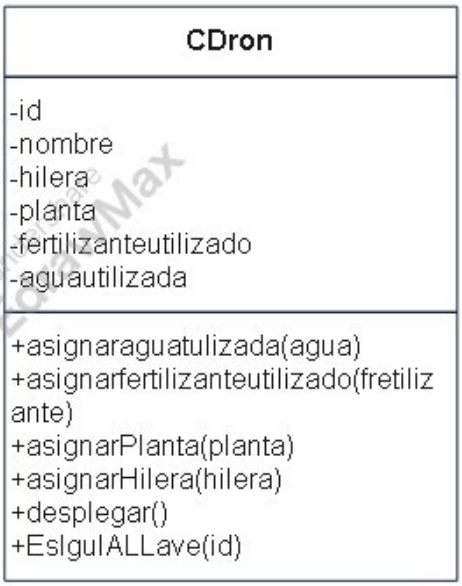



**Clases.py**

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

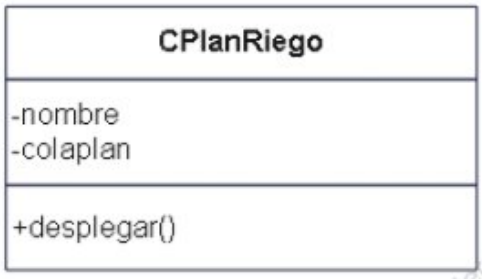



Guarda la informacion de la planta

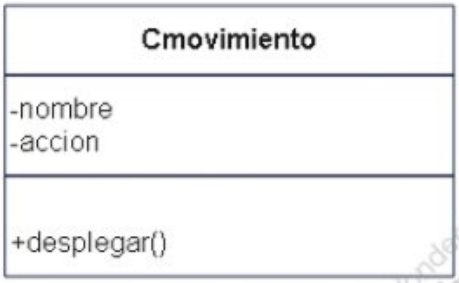

Guardar informacion plan

Guarda toda la informacion del invernadero


Guarda informacion plan riego

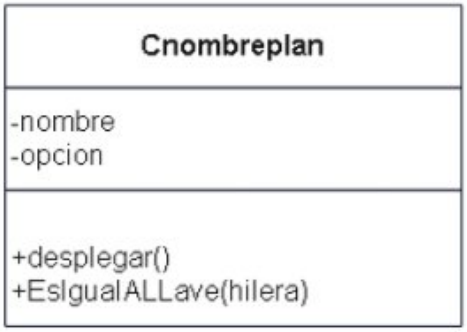



Guarda la informacion moviemiento




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

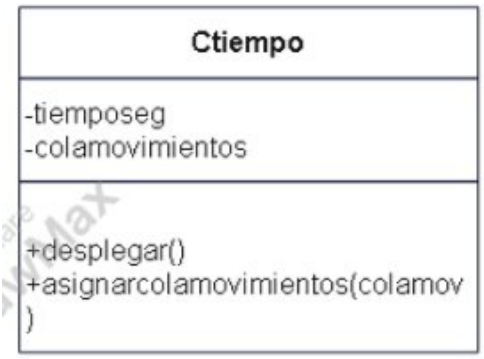
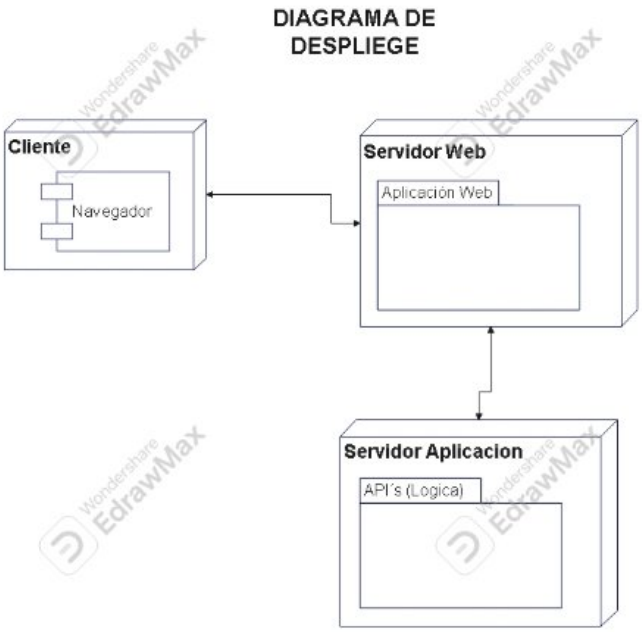



**Diagrama de Despliegue aplicación.**




**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

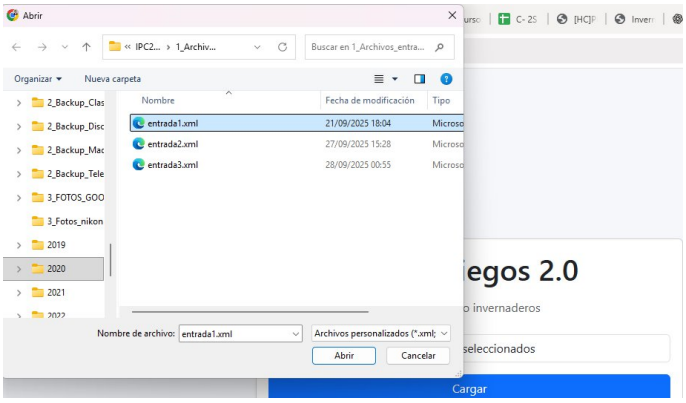
Paso 2: Ejecuta CMD apartir de esa ruta del sistema

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

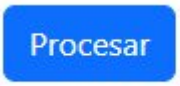

Paso 5: Cargar un archivo XML

Click en selecionar archivo y sube el archivo

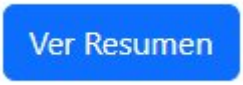



Paso 6: Preciona el boton cargar

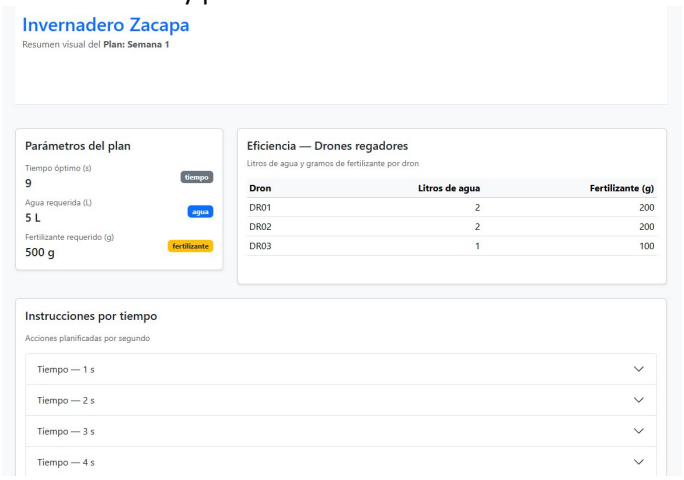

Paso 7:Luego click en procesar




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