

---

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

---

**201906795 – Javier Ricardo Yllescas Barrios**

### Resumen

El programa presentado propone el desarrollo un sistema de facturación para servicios 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 aplicación es usar paradigmas de programación orientada a objetos para la construcción del software, la utilización de base de datos XML y la interacción entre dos aplicaciones con diferentes tecnologías 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 comprensión del proyecto se aplicaron los siguientes archivos para desarrollarlo.

### Estructura del Proyecto [Archivos]

- DOCUMENTACION\_IPC2\_Proyecto3\_20190695.pdf

—>ArchivosdePrueba

—>Backed

- AppFlask.py
- ArchivoConfiguraciones.xml
- ArchivoConsumos.xml
- entrada.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 capturar 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 endpoint para la interaccion con la aplicaion

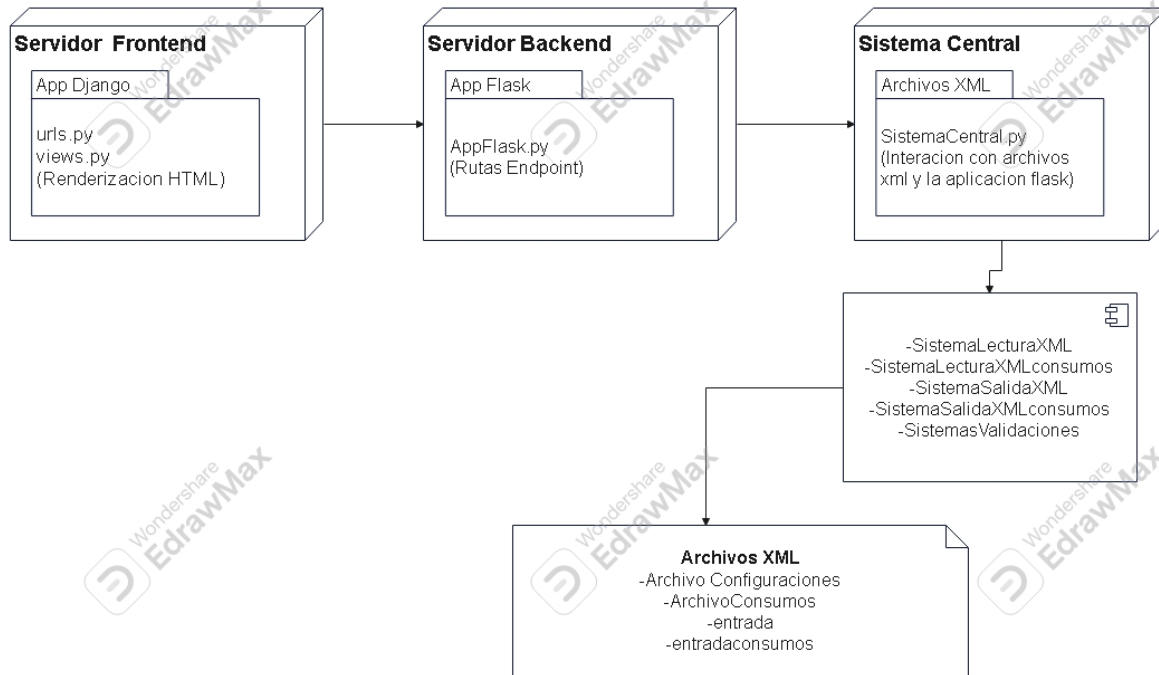
—>**Backed/Sistemas**

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

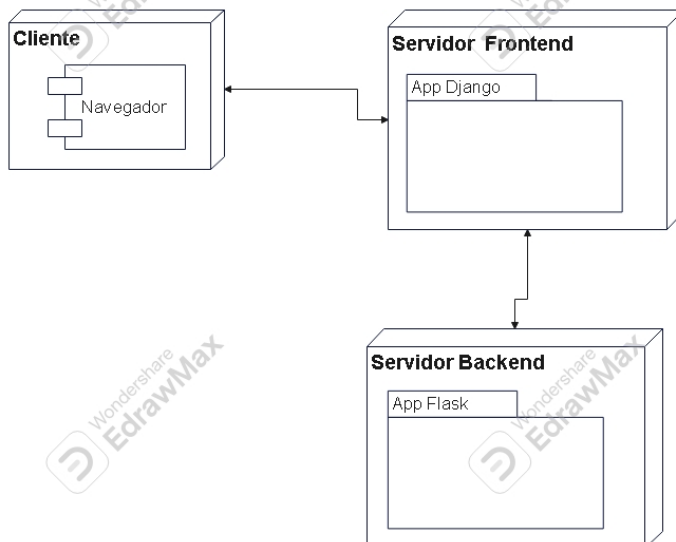
**Backed/Sistemas/SistemaCentral.py**

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



### DIAGRAMA DE DESPLIEGE



Estructura Clases

SistemaArchivo
-ruta -ListaDrones - Cola() -ListaPalantas - Lista() -ListaPlanes -Lista() -ColaInvernaderos - Cola -BanderaErrores - True/False
+leer_archivo() +convertir_xml_a_Dom() +segmentar_archivo()

SistemaArchivoHTML
-colainvernaderos -ruta -txthtml -idmovimientos -idacordion
+crearchivoHTML(ruta) +guardarHTML() +crearinvernadero(Inv,numero) +HTMLListamovimientos(Inv,numero) +HTMLDronesResumen(Inv,numero) +HTMLTitulosResumen(Inv,numero) +HTMLencabezado +HTMLpie

Sistema Salida archivos XML

Sistema genera todo lo relacionado con el archivo salida.xml

SistemaArchivoHTML

Sistema para generar reportes HTML

SistemaArchivoSalidaXML
-ruta -colainvernaderos -doc -root -listainvernaderos -listaplanes -invernaderoactual -nuevasinstruccionesinv
+ asignarcolainvernadero(colainver) +crear_archivo(contenido) +creararchivoDOC() +agregarinvernaderos() +obtenerinvernaderoactual(numero invernadero) +obtenerinstrucciones() +crear_plan(numeroinvernadero, nombreplan, numeroplan) +GuardarSalidaXML() +segmentar_archivo_XML()

Sistema Archivo TDA

Genera los graficos en graphviz

SistemaArchivoTDAs
-dot_text -colainstrucciones
+asignarcolainstrucciones() +crearTDAs(tiempo) +creararchivoDot()

SistemaRiegos

SistemaRiegos
-numeroinverndaerosel -numeroplanseleccionado -nombreplan -colainvernaderos -InvernaderoSel -PlanSel -ultimoriegotiempo #Datos invernadero evaluando -Invnombre -InvnumeroHilera -InvplantasXHilera -InvListaPlantas -InvListaDrones -InvInstrucciones  -Tiempoactual -Tiempomax -DronRegando = Flase/True -Colamovimientos -banderainstruccioncompletada -ColaHilerasIndividual - Cola() -rutaSalida -sistema_archivo_salida -Colainstrucciones
+obtenercolainvernaderos +desplegar +ColasInvernaderos +ColasPlanes +ReiniciarValores +ReiniciarDrones +ReiniciarAguayFertilizante +Obtenerinformacio(numinv, numplna) +ManejraAguayFertilizante(hilera,p osicion,dron) +Dron_Valida_Riego() +Dron_Mover_primer_posicon()

```
+Dron_Evaluar_movimiento(Rega
ndo,instruccion)
+Dron_Mover_adelante_esperar_a
tras
+Ejecutar_instruccion
+nuevohistoricoDrones
+Ejecutar_tiempo
+obtenerColainstruccion
+Guardarnhistorialmovimientos
+AlmacenrContenidoXML
+CrearListainvernaderoXML
+CrearXML
+GuardarSalidaXML
+CrearArchivoXML
+obtenercolainvernaderos
+obtenerplanes
```

### Clases.py

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

InfoNodo
+desplegar() +EsigualALLave()

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

Almacena la informacion del dron

CDron
-id -nombre -hilera -planta -fertilizanteutilizado -aguautilizada
+asignaraguatutilizada(agua) +asignarfertilizanteutilizado(fretiliz ante) +asignarPlanta(planta) +asignarHilera(hilera) +desplegar() +EsIgualALLave(id)

Guarda la informacion de la planta

CPlanta
-hilera -posicion -litrosAgua -gramosFertilizante -nombre
+desplegar() +EsIgualALLave()

Guardar informacion plan

CAsignacionPlan
-hilera -planta
+asignarhilera(hilera) +asignarplanta(planta) +desplegar +EsIgualALLave(hilera)

Guarda toda la informacion del invernadero

CInvernadero
-nombre -numeroHilera -plantaXHilera -ListaPlantas -ListaPlanes -ListaDrones -colainstrucciones -tiempoOptimo -aguaRequerida -fertilizanteRequerido -historialmovimientos -historiatiempooptimo -historiaagua -historiafertilizante -historialdrones
+asignarcolainstrucciones(cola) +asignarhistorialmovimientos(cola) +asignarhistorialmovimientos(cola) +asignartiempoOptimo(tiempoO) +asignarAguaRequerida(agua) +asignarFertilizanteRequerido(fertilizante) +EslgualALLave(nombre) +desplegar()

Guarda informacion plan riego

CPlanRiego
-nombre -colaplan
+desplegar()

Guarda la informacion movimiento

Cmovimiento
-nombre -accion
+desplegar()

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

Cnombreinvernadero
-nombre -opcion
+desplegar() +EslgualALLave()

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

Cnombreplan
-nombre -opcion
+desplegar() +EslgualALLave(hilera)

Clase que almacena los movimientos

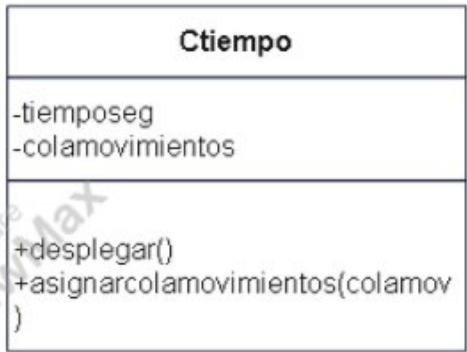
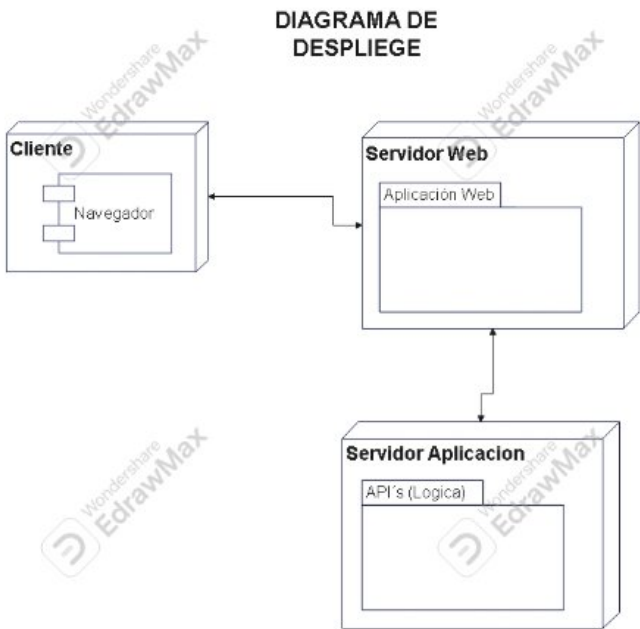


Diagrama de Despliegue aplicación.



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

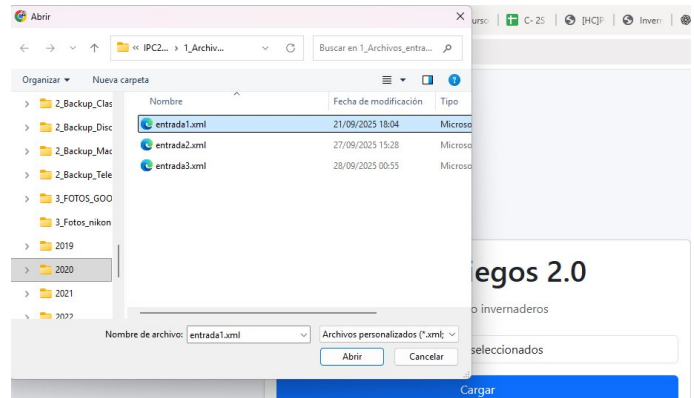
Paso 3: Colocar en las lineas de comando el siguiente ocmando “**python app.py**” luego presiona Enter.

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 seleccionar archivo y sube el archivo

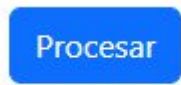


Paso 6: Preciona el boton cargar





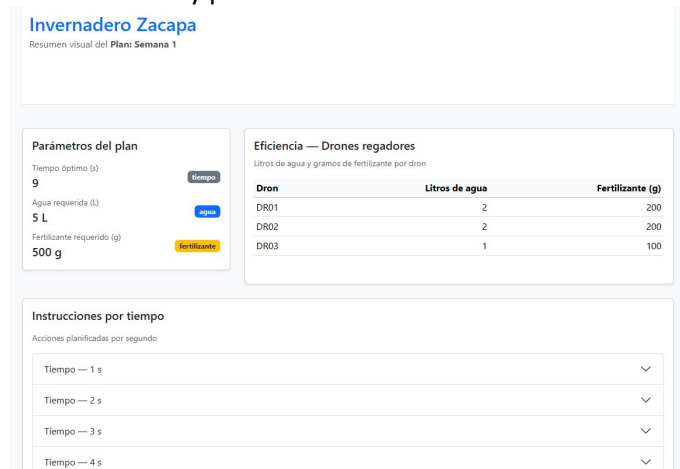
Paso 7: Luego click en procesar



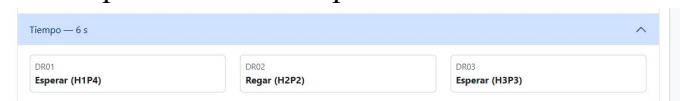
Paso 8: Para ver resumen del archivo click en ver resumen



Redirecciona a la siguiente ventana con la informacion del invernadero 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 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)"/>
            </tiempo>
          </instrucciones>
        </plan>
      </listaPlanes>
    </invernadero>
  </listaInvernaderos>
</datoSalida>
```

## 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 aplicación de los conceptos de listas, colas y nodos resultó muy enriquecedor, pero su nivel de complejidad va aumentando a la hora de almacenar muchos nodos dentro de ellas, el manejo de la información parece un poco repetitivo y no se pueden crear funciones para todo porque se termina formando duplicados de las mismas y las modificaciones se generan imposibles si no se usa bien los try exception para identificar 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](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) Documentación para uso framework

Flask

<https://flask.palletsprojects.com/en/stable/tutorial/>

Graphviz. (s.f.). Documentación para crear gráficas.

<https://graphviz.org/documentation>