Universidad San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas
Introducción a la Programación y Computación 2

Inga. Claudia Liceth Rojas Morales
Ing. Marlon Antonio Pérez Türk
Ing. Byron Rodolfo Zepeda Arevalo
Ing. José Manuel Ruiz Juárez
Ing. Edwin Estuardo Zapeta Gómez

Tutores de curso:
Oscar René Jordán Orellana, Juan Pablo Osuna de León
Astrid Edith Hernández González, Henry Adolfo Gálvez
César Dionicio Sazo Mayen, Oscar Alejandro Rodríguez Calderón
Edwar Everaldo Zacarias, Javier Estuardo Lima Abrego
Mónica Raquel Calderón Muñoz, Marco Antonio López Grajeda

PROYECTO 3

OBJETIVO GENERAL

Desarrollar una solución integral que implemente un API que brinde servicios utilizando el Protocolo HTTP bajo el concepto de programación orientada a objetos (POO) y el uso de bases de datos.

OBJETIVOS ESPECÍFICOS

- Implementar un API a través de lenguaje Python que pueda ser consumida utilizando el protocolo HTTP.
- Utilizar el paradigma de programación orientada a objetos para construir software.
- Utilizar bases de datos para almacenar información de forma persistente.
- Utilizar archivos XML como insumos para la comunicación con el API desarrollado.
- Utilizar expresiones regulares para extraer contenido de texto.

ENUNCIADO

La Superintendencia de Administración Tributaria le ha solicitado construir un software que pueda ser consumido desde Internet como un servicio. Este software recibirá un mensaje con los datos para solicitar la autorización de un Documento Tributario Electrónico (DTE) emitido por un contribuyente y como respuesta emitirá un número único de autorización, este número será un correlativo que iniciará con el valor 1 cada día y no deberá repetirse de nuevo en ese día. La estructura del número de autorización es la siguiente: yyyymmdd########, donde:

yyyy corresponde al año en que se solicita la autorización

mm corresponde al mes en que se solicita la autorización

dd corresponde al día en que se solicita la autorización

######## corresponde al correlativo precedido por la cantidad de 0's necesarios para mantener el formato: Ej. Para el correlativo 1: 00000001.

El mensaje que el proveedor enviará contendrá la siguiente información:

```
LUGAR Y FECHA: Guatemala, dd/mm/yyyy hh24:mi
REFERENCIA: A3038
NIT EMISOR: XXXXXXV
NIT RECEPTOR: YYYYYYYV
VALOR: ########.##
IVA: #######.##
TOTAL: #######.##
```

Reglas y consideraciones del formato para cada mensaje recibido:

- **FECHA:** Fecha en formato dd/mm/yyyy hh24:mi (horas en formato 24 horas – 0 a 23 horas)
- **REFERENCIA:** Código interno enviado por el emisor. Es un texto único de máximo 40 posiciones.
- **NIT EMISOR:** XXXXXXV cada X corresponde a un valor entre 0 ... 9. V debe ser la validación del NIT ingresado. La cantidad de valores X podrá ser desde 1 hasta 20.
- **NIT RECEPTOR:** YYYYYYV cada Y corresponde a un valor entre 0 ... 9. V debe ser la validación del NIT ingresado. La cantidad de valores Y podrá ser desde 1 hasta 20.
- **VALOR:** Valor real positivo de 2 posiciones decimales.
- **IVA:** Es el resultado de la operación REDONDEAR(VALOR * 0.12, 2).
- **TOTAL:** Es el resultado de la operación VALOR + IVA

NOTA: La validación de NIT utiliza el siguiente algoritmo:

1. Multiplique cada carácter por su posición respectiva (siendo la posición 1 el carácter más a la derecha), excepto el primero.
2. Sume todos los resultados.
3. Obtenga el módulo 11 de la sumatoria.
4. A 11 réstale el resultado obtenido en el punto 3.

5. Calcule el módulo 11 del resultado obtenido en el punto No. 4, si este resultado es 10, entonces el dígito verificador del NIT deberá ser K.

El programa por desarrollar, luego de recibir el mensaje antes mencionado, deberá almacenar la información necesaria en un archivo de salida XML, para posteriormente emitir reportes y realizar consultas. El archivo de salida debe almacenarse con la siguiente información:

FECHA: dd/mm/yyyy
Cantidad total de facturas recibidas: Cantidad total de facturas enviadas al servicio
Listado de errores en facturas recibidas
NIT emisor inválido: cantidad errores
NIT receptor inválido: cantidad errores
IVA mal calculado: cantidad de errores
TOTAL mal calculado: cantidad de errores
REFERENCIA duplicada: cantidad de errores
Cantidad de facturas sin errores: Cantidad total de facturas enviadas sin errores al servicio.
Cantidad de emisores de facturas: Cantidad de distintos NIT's emisores de facturas
Cantidad de receptores de facturas: Cantidad de distintos NIT's receptores de facturas
…
FECHA: dd/mm/yyyy
…

La estructura completa del archivo de salida está descrita en la siguiente sección.

ARCHIVOS DE ENTRADA Y SALIDA

Archivo de entrada

Si el archivo de entrada posee un error de formato con respecto al valor de las etiquetas XML, entonces, la solicitud de autorización de DTE con el error será ignorada y se continuará con el análisis de las solicitudes restantes.

```
<SOLICITUD_AUTORIZACION>
   <DTE>
      <TIEMPO> Guatemala, 15/01/2021 15:25 hrs. </TIEMPO>
      <REFERENCIA> A1990 </REFERENCIA>
      <NIT_EMISOR> 7378106 </NIT_EMISOR>
      <NIT_RECEPTOR> 8338817 <NIT_RECEPTOR>
      <VALOR> 100.00 </VALOR>
      <IVA> 12.00 </IVA>
      <TOTAL> 112.00 </TOTAL>
   </DTE>
   …
</SOLICITUD_AUTORIZACION >
```

Archivo de salida

Nombre del archivo: autorizaciones.xml

```
<LISTAAUTORIZACIONES>
  <AUTORIZACION>
    <FECHA> 01/09/2021 </FECHA>
    <FACTURAS_RECIBIDAS> 100 </FACTURAS_RECIBIDAS>
    <ERRORES>
      <NIT_EMISOR> 1 </NIT_EMISOR>
      <NIT_RECEPTOR> 2 </NIT_RECEPTOR>
      <IVA> 3 </IVA>
      <TOTAL> 2 </TOTAL>
      <REFERENCIA_DUPLICADA> 3 </REFERENCIA_DUPLICADA>
    </ERRORES>
    <FACTURAS_CORRECTAS> 89 </FACTURAS_CORRECTAS>
    <CANTIDAD_EMISORES> 10 </CANTIDAD_EMISORES>
    <CANTIDAD_RECEPTORES> 88[1] </CANTIDAD_RECEPTORES>
    <LISTADO_AUTORIZACIONES>
      <APROBACION>[2]
        <NIT_EMISOR ref="A1990"> 7378106 </NIT_EMISOR>
        <CODIGO_APROBACION> 2021090100000001 </CODIGO_APROBACION>
      </APROBACION>
      …
      <TOTAL_APROBACIONES> 89 </TOTAL_APROBACIONES>[3]
    </LISTADO_AUTORIZACIONES>
  </AUTORIZACION>
  …
</LISTAAUTORIZACIONES>
```

ARQUITECTURA

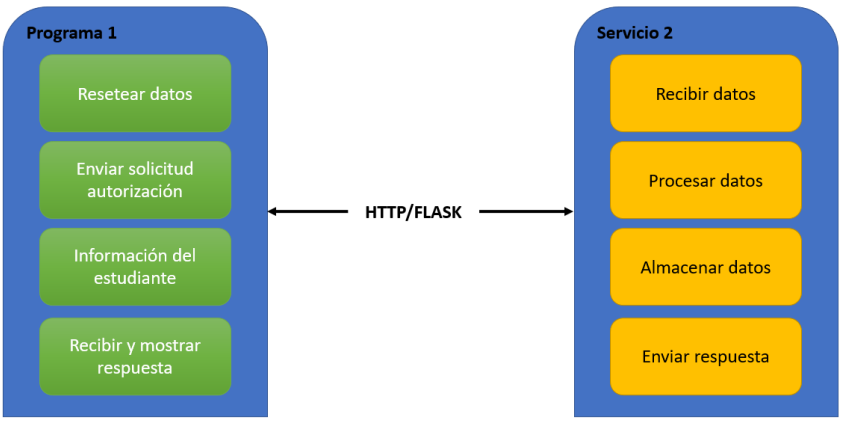

Fig. 1 – Arquitectura general de la aplicación

[1] Algunos NIT receptores recibieron dos facturas.
[2] Debe haber tantos tags APROBACION como FACTURAS_CORRECTAS
[3] Cantidad de tags APROBACION generados en una fecha dada

Programa 1 - Frontend

Este programa consiste en una aplicación Web y consistirá en un simulador de la aplicación principal, contendrá únicamente las funcionalidades necesarias para testear el buen funcionamiento de la API (Servicio 2), en esta aplicación se podrán mostrar los eventos que se procesarán y los datos estadísticos que fueron almacenados en la base de datos XML de salida).

Para realizar el frontend deberá utilizarse el framework **Django**, el cual trabaja con el patrón MVT (Modelo-Vista-Template).

Componentes:

- **Cargar Archivo:** Se desplegará una pantalla para gestionar la carga de los archivos de entrada con extensión .xml con una o varias solicitudes de autorización. Se especifica en la sección de archivos de entrada y salida.
- **Peticiones:** En este apartado se debe de tener las siguientes opciones:

    - **Consultar Datos:** Al seleccionar esta opción se deben de consultar los datos almacenados en el archivo autorizaciones.xml y se mostrarán los datos en un recuadro de texto de salida.
    - **Resumen de IVA por fecha y NIT:** Al seleccionar esta opción se podrá elegir la fecha por la cual se requiere filtrar y se debe de **mostrar gráficamente** un resumen de movimientos por NIT en esa fecha dada. Por cada NIT se debe apreciar el IVA emitido y el IVA recibido.
    - **Resumen por rango de fechas:** Al seleccionar esta opción se podrá elegir un rango de fechas por la cual se requiere filtrar, luego podrá elegir si desea ver los valores totales o los valores sin IVA. Se deberá **presentar gráficamente** por cada fecha el valor autorizado (Total o sin IVA).
    - **Gráfica:** Se deberán crear gráficas para los dos resúmenes anteriores. Un ejemplo se muestra en la figura 2.
    - **Reporte en PDF:** Deberá ser posible realizar un reporte de todas las peticiones anteriormente descritas.

- **Ayuda:** desplegará 2 opciones, una para visualizar información del estudiante y otra para visualizar la documentación del programa.
- **Botón Enviar:** Enviará las solicitudes de autorización del recuadro de texto a la Api para su procesamiento.
- **Botón Reset:** Este botón mandará la instrucción a la Api para devolver al estado inicial la Api, es decir, sin datos.

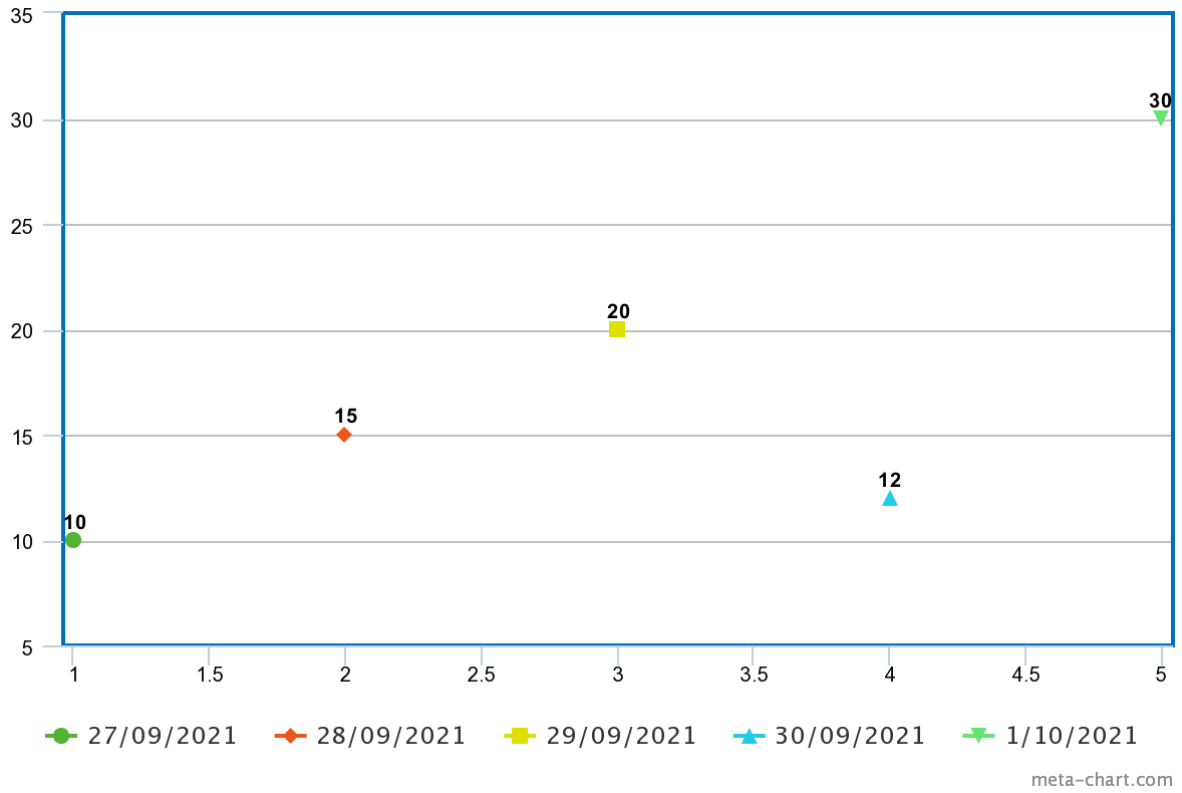

Fig. 2 - Sugerencia de gráfica.

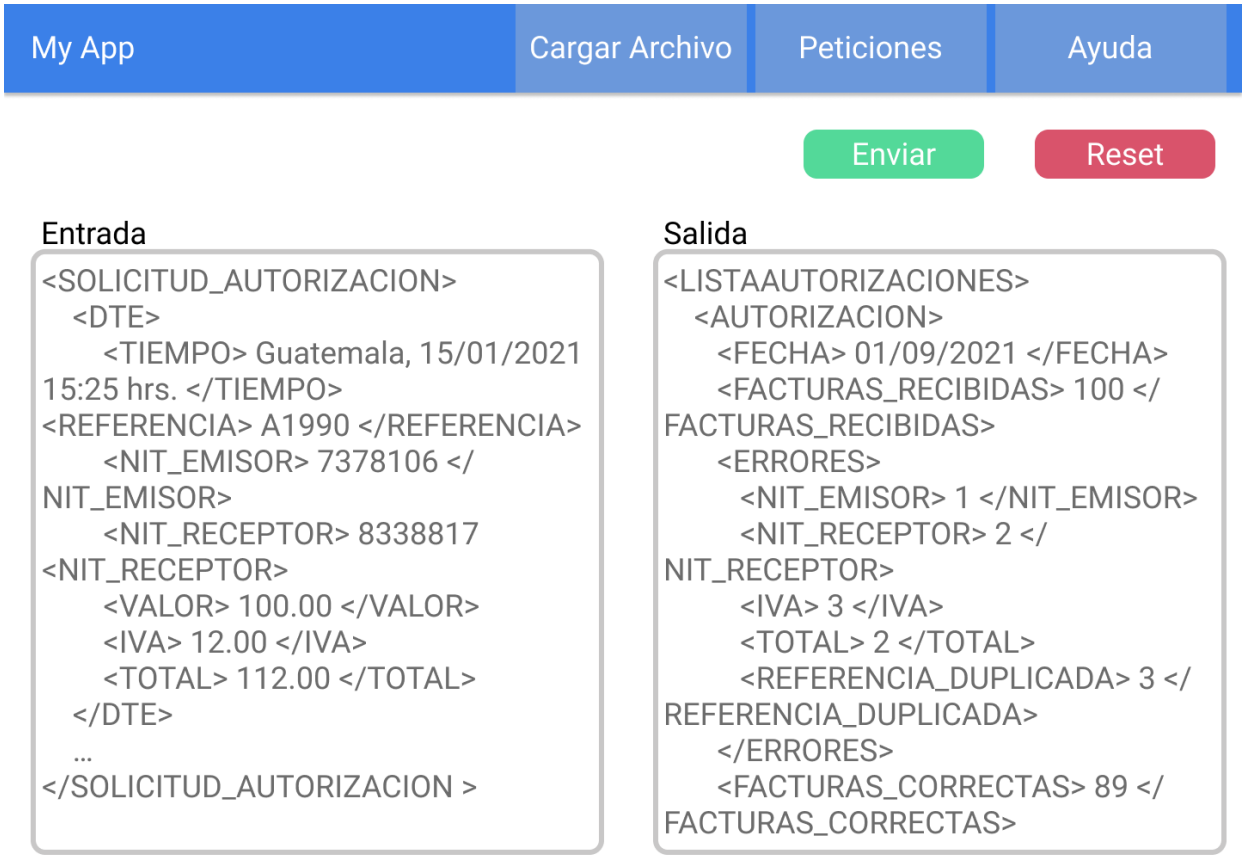

Fig. 3 - Sugerencia de interfaz.

Servicio 2 - Backend

Este servicio consiste en una API que brindará servicios utilizando el protocolo HTTP, su funcionalidad principal es procesar los datos recibidos del programa 1, luego de procesar los datos es necesario que estos sean almacenados en uno o varios archivos xml, algunos de estos archivos están especificados en la sección de archivos de entrada y salida[4], este servicio también tiene la funcionalidad de devolver los datos que fueron almacenados para que sean mostrados como se indica en la sección “Programa 1 – Frontend / Componentes”.

Para la realización de este servicio debe utilizarse el framework **Flask**. El estudiante deberá definir por su propia cuenta los métodos que necesitará para la realización de este servicio. Esto significa que debe implementar tantos métodos como necesite para consumir la API.

NOTA: Durante la calificación de este proyecto, el Servicio 2 podrá ser consumido desde otro cliente, por ejemplo, Postman.

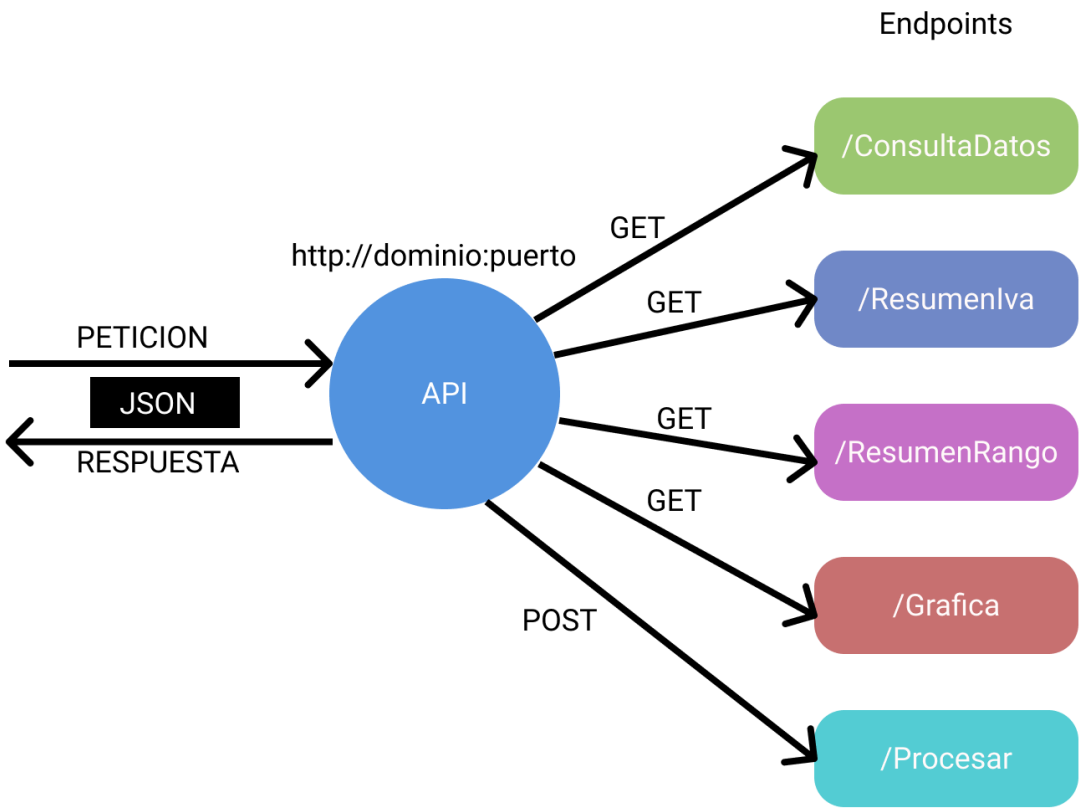

Fig. 4 - Sugerencia de la estructura de un API

CONSIDERACIONES

En las solicitudes de autorización del archivo de entrada debe descartarse cualquier información que no sea necesaria para la elaboración de las gráficas y archivos de salida, por ejemplo, si las fechas traen nombres de lugares u otra información, éstos no deben ser

[4] El estudiante puede definir otros archivos que sean útiles para mostrar los resultados de las solicitudes de autorización atendidas.

considerados errores, solamente debe considerarse la información útil para el proyecto.

Debe utilizarse versionamiento para el desarrollo del proyecto. Se utilizará la plataforma Github en la cual se debe crear un repositorio en el que se gestionará el proyecto. Se deben realizar 4 releases o versiones del proyecto (se recomienda realizar una por semana del tiempo disponible). Se deberá agregar a su respectivo auxiliar como colaborador del repositorio. El último release será el release final y se deberá de realizar antes de entregar el proyecto en la fecha estipulada.

DOCUMENTACIÓN

Para que el proyecto sea calificado, el estudiante deberá entregar la documentación utilizando el formato de ensayo definido para el curso. En el caso del proyecto, el ensayo debe tener entre 4 y 7 páginas de contenido, este máximo no incluye los apéndices o anexos donde se pueden mostrar modelos y diseños utilizados para construir la solución.

RESTRICCIONES

- Solo se permitirá la utilización de los IDEs discutidos en el laboratorio.
- Uso obligatorio de programación orientada a objetos (POO).
- El nombre del repositorio debe de ser **IPC2_Proyecto3_#Carnet.**
- El estudiante debe entregar la documentación solicitada para poder optar a la calificación.
- Se calificará de los cambios realizados en el cuarto release. Los cambios realizados después de ese release no se tomarán en cuenta.
- Para dudas concernientes al proyecto se utilizarán los foros en UEDI de manera que todos los estudiantes puedan ver las preguntas y las posteriores respuestas.
- **NO HABRÁ PRÓRROGA.**

ENTREGA

- La entrega será el **2 de noviembre** a las 23:59 como máximo.
- La entrega será por medio de la UEDI.
- La documentación debe estar subida en el repositorio en una carpeta separada.
- Para entregar el proyecto en UEDI se deberá subir un archivo de texto con el link del repositorio.