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. José Manuel Ruiz Juárez
Ing. Edwin Estuardo Zapeta Gómez
Ing. Fernando José Paz González

Tutores de curso:
Andrea María Cabrera Rosito
Josué Alfredo González Caal
Paula Gabriela García Reynoso
Mario César Morán Porras
Denilson Florentín De León Aguilar

PROYECTO No. 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 empresa “Industria Típica Guatemalteca, S.A.” (ITGSA) ha decidido expandir su negocio creando un canal de distribución denominado “On-Line”.

En el canal de distribución On-Line, la empresa ha creado una solución web denominado “Tienda virtual” para que los clientes adquieran productos y servicios que ITGSA ofrece. Una vez que el cliente ha seleccionado los productos y servicios que desea adquirir, se debe generar una factura por el monto total que el cliente debe pagar. La solución Web incluye un portal Web, aplicaciones móviles para dispositivos móviles Android y para dispositivos móviles con iOS.

Adicionalmente, ITGSA ha realizado un convenio con los bancos de Guatemala, de tal manera que los clientes, utilizando la banca en línea de su banco favorito, pueden realizar pagos que se trasladan directamente a las cuentas de ITGSA. Una vez el pago es trasladado a la cuenta de ITGSA, la cantidad pagada debe ser abonada de tal forma que la factura más antigua que posea el cliente sea abonada o pagada en su totalidad. Si un cliente paga más de lo que debe, podría quedar con un saldo a su favor, el cual se aplicará automáticamente cuando el cliente realice otra compra en el sitio web.

Usted ha sido contratado para crear el backend del sistema de ITGSA, de tal manera que pueda llevar control de las facturas generadas por el sitio WEB y de los pagos que los clientes realicen en las distintas bancas en línea. Adicionalmente, deberá crear una aplicación frontend que se utilizará para crear los clientes y bancos con que trabaja ITGSA, además de gestionar las operaciones de facturación y pagos. La figura No. 1 ejemplifica el ecosistema tecnológico de ITGSA.

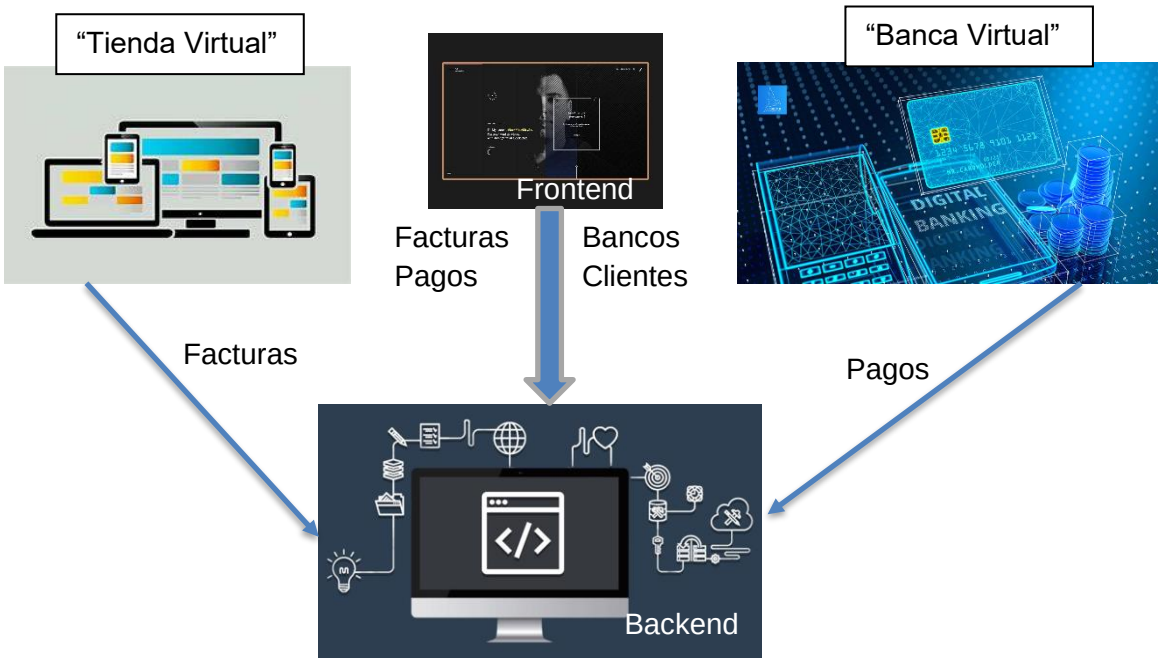


Fig. 1 – Ecosistema tecnológico de ITGSA

ARCHIVOS DE ENTRADA Y DE RESPUESTA

El frontend ha construir debe ser capaz de recibir 2 tipos de archivo de entrada, uno para configurar los bancos y los clientes del backend y otro para gestionar transacciones de facturación o pagos generadas por la “Tienda Virtual” o la “Banca Virtual” respectivamente.

Entrada: config.xml

Este archivo contiene el listado de clientes y bancos que trabajan con ITGSA. Debe tomar en cuenta que este archivo es incremental, es decir, se pueden ingresar varios archivos de configuración para ampliar la base de datos de clientes y bancos. Si se repite un NIT de cliente o un código de banco, los datos deben ser actualizados.

```
<?xml version="1.0"?>
<config>
        <clientes>
                <cliente>
                        <NIT> [valor_alfanumérico] </NIT>
                        <nombre> [valor_alfanumérico] </nombre>
                </cliente>
                ...
        </clientes>
        <bancos>
                <banco>
                        <codigo> [valor_numérico] </codigo>
                        <nombre> [valor_alfanumérico] </nombre>
                </banco>
                ...
        </bancos>
</config>
```

Como respuesta a cada archivo config.xml cargado al sistema, se deberá generar un XML con la siguiente estructura:

```
<?xml version="1.0"?>
<respuesta>
        <clientes>
                <creados> [valor_numérico] </creados>
                <actualizados> [valor_numérico] </actualizados>
        </clientes>
        <bancos>
                <creados> [valor_numérico] </creados>
                <actualizados> [valor_numérico] </actualizados>
        </bancos>
</respuesta>
```

Entrada: transac.xml

Este archivo contiene un listado de facturas generadas por la “Tienda Virtual” y un listado de pagos generados por las “Bancas virtuales”. Debe tomar en cuenta que este archivo es incremental, es decir, se pueden ingresar varios archivos de transacciones para crear nuevas facturas y nuevos pagos. En este caso, debe considerar que no puede haber actualizaciones de datos de facturas o de pagos.

```
<?xml version="1.0"?>
<transacciones>
        <facturas>
                <factura>
                   <numeroFactura> [valor_alfanumérico] </numeroFactura>
                        <NITcliente> [valor_alfanumérico] </NITcliente>
                        <fecha> [dd/mm/yyyy] </fecha>
                        <valor> [valor_numérico] </valor>
                </factura>
                ...
        </facturas>
        <pagos>
                <pago>
                        <codigoBanco> [valor_numérico] </codigoBanco>
                        <fecha> [dd/mm/yyyy] </fecha>
                        <NITcliente> [valor_alfanumérico] </NITcliente>
                        <valor> [valor_numérico] </valor>
                </pago>
                ...
        </pagos>
</transacciones>
```

Como respuesta a cada archivo transac.xml cargado al sistema, se deberá generar un XML con la siguiente estructura:

```
<?xml version="1.0"?>
<transacciones>
        <facturas>
                <nuevasFacturas> [valor_numérico] </nuevasFacturas>
                <facturasDuplicadas> [valor_numérico] </facturasDuplicadas>
                <facturasConError> [valor_numérico] </facturasConError>
        </facturas>
        <pagos>
                <nuevosPagos> [valor_numérico] </nuevosPagos>
                <pagosDuplicados> [valor_numérico] </pagosDuplicados>
                <pagosConError> [valor_numérico] </pagosConError>
        </pagos>
</transacciones>
```

ARQUITECTURA

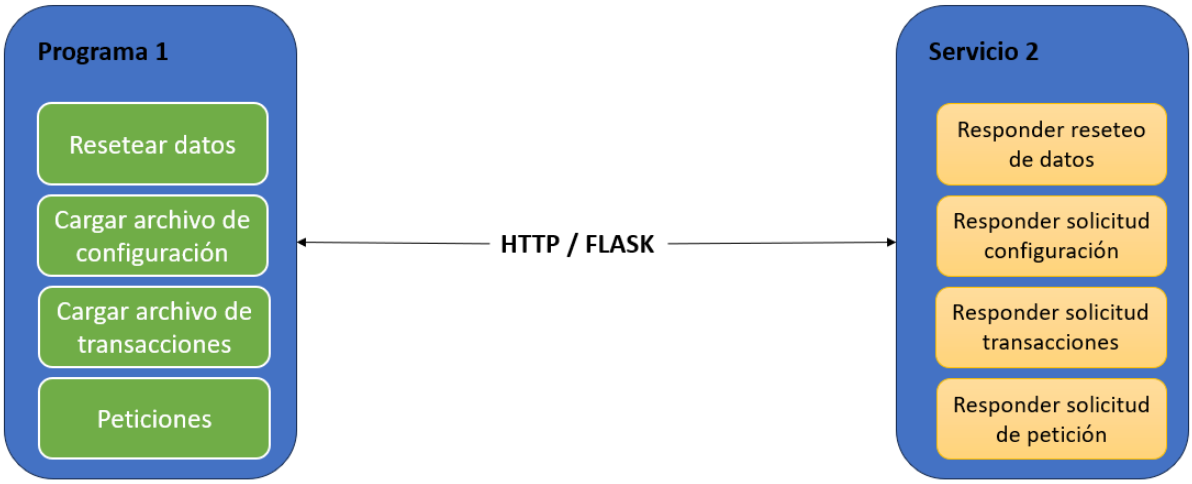


Fig. 2 – Arquitectura general de la aplicación

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.

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

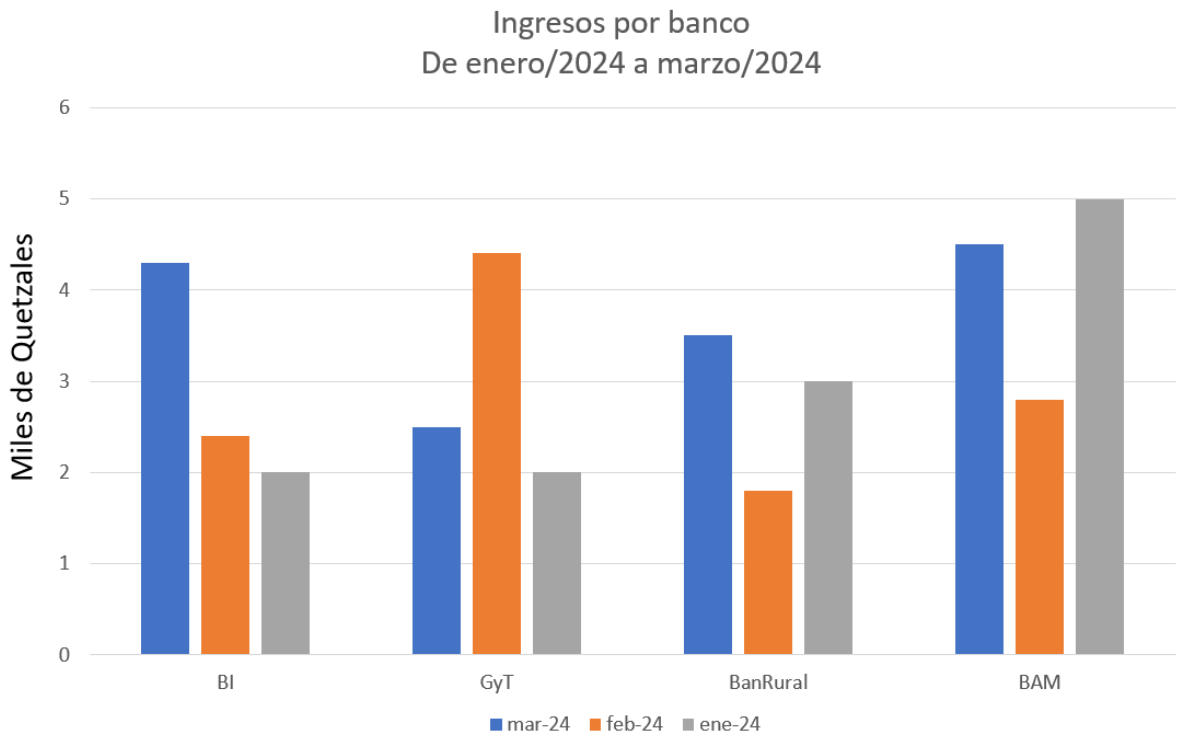
Componentes del Programa 1:

- **Resetear datos:** Esta opción mandará la instrucción a la Api para devolver al estado inicial la Api, es decir, sin datos.
- **Cargar archivo de configuración:** Se desplegará una pantalla para gestionar la carga de los archivos de entrada config.xml con la configuración de clientes y bancos que podrán realizar transacciones de facturación y pagos (ver archivos de entrada config.xml). Luego de cargar el archivo deberá presentar la respuesta apropiada.
- **Cargar archivo de transacciones:** Se desplegará una pantalla para gestionar la carga de los archivos de entrada transac.xml con uno o varias operaciones de facturación y de pago (ver archivos de entrada transac.xml). Luego de cargar el archivo deberá presentar la respuesta apropiada.
- **Peticiones:** En este apartado se debe de tener las siguientes opciones:
  - **Consultar estado de cuenta:** Al seleccionar esta opción se deben de solicitar un NIT de cliente, o si desea ver a todos los clientes ordenados por NIT y presentar la siguiente información de forma amigable:
    Cliente: 399399-K – Almacén Guatemalteco
    Saldo actual: Q. 0.00
    Transacciones (ordenadas de la más reciente a la más antigua)

Fecha	Cargo	Abono
31/03/2024		Q 300.00 (BI)

28/03/2024 Q. 300.00 (Fact. # 10)
…
…

- **Consultar ingresos:** Al seleccionar esta opción se deben de solicitar un mes y mostrar una gráfica con la siguiente información de forma amigable:
  Mes elegido: marzo/2024 (se presentan los últimos 3 meses de forma descendente)



- **Ayuda:** desplegará 2 opciones, una para visualizar información del estudiante y otra para visualizar la documentación del programa.

NOTA 1: Todos los reportes y gráficas deben poder ser generados en formato .pdf
NOTA 2: El Frontend deberá mostrar los datos recuperados del consumo del servicio que se describe en la siguiente sección.

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 de respuesta[1], este servicio también tiene la funcionalidad de devolver los datos que fueron procesados para que

[1] El estudiante puede definir otros archivos que sean útiles para mostrar los resultados solicitados por el frontend.

sean mostrados como se indica en las respuestas en la sección “Archivos de entrada y de respuesta”.

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 y recibirá una respuesta XML con los datos necesarios para cada funcionalidad de la sección “peticiones” del programa 1 – Frontend.

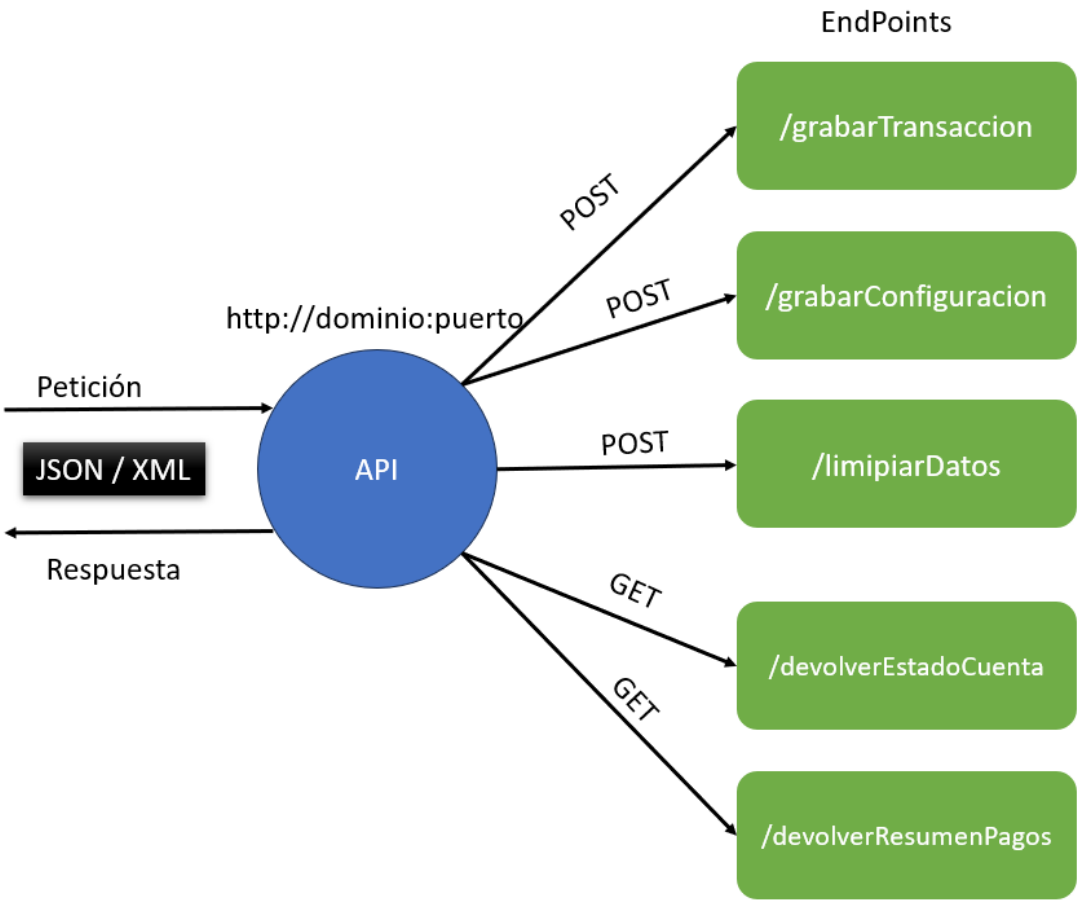


Fig. 3 - Sugerencia de la estructura de un API

CONSIDERACIONES

En las solicitudes de procesamiento 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 considerados errores, solamente debe considerarse la información útil para el proyecto, este mismo principio aplica para los NITs.

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 sugiere desarrollar un Release por semana). Se deberá agregar a sus respectivos auxiliares como colaboradores del repositorio (**Github a utilizar: AndreaCabrera01**). 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 puede tener un mínimo de 4 y un máximo de 7 páginas de contenido, este máximo no incluye los apéndices o anexos donde se pueden mostrar modelos y diseños utilizados para construir la solución. Es obligatorio incluir el diagrama de clases del diseño del software desarrollado, el diagrama entidad relación del modelo de datos implementado en XML y se recomienda incluir el modelo conceptual y los diagramas de actividades que modelan la solución de software presentada por el estudiante.

RESTRICCIONES

- Solo se permitirá la utilización de los IDEs discutidos en el laboratorio.
- Uso obligatorio de programación orientada a objetos (POO). De no cumplir con esta restricción, no se tendrá derecho a calificación
- El nombre del repositorio debe de ser **IPC2_Proyecto3_#Carnet.**
- El estudiante debe entregar la documentación solicitada para poder optar a la calificación.
- Los archivos de entrada no podrán modificarse.
- Los archivos de salida (o respuestas) deben llevar la estructura mostrada en el enunciado obligatoriamente.
- Deben existir 4 releases mínimo, podría ser uno por cada semana, de esta manera se corrobora el avance continuo del proyecto.
- Se calificará de los cambios realizados en el último release hasta la fecha de entrega. Los cambios realizados después de ese release no se tomarán en cuenta.
- Cualquier caso de copia parcial o total tendrá una nota de 0 y será reportada a Escuela.
- 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 martes **30 de abril,** a más tardar a las 11:59 pm.
- 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.