Universidad de San Carlos de Guatemala
Escuela de Ingeniería en Ciencias y Sistemas, Facultad de Ingeniería
Introducción a la programación y computación 2, 1er. Semestre 2021.

SOFTWARE CONSTRUIDO PARA ,MANEJAR LAS AUTORIZACIONES DE DOCUMENTOS TRIBUTARIOS ELECTRÓNICOS Y EMISIÓN DE REPORTES SOBRE LAS AUTORIZACIONES.

202004750 Pedro Alejandro Zetino Páez

Resumen

Para iniciar se desarrolló una aplicación web, que simulará la aplicación principal, esto con la finalidad de testear el buen funcionamiento de la API, mostrando los eventos a ser procesados y los datos estadísticos que se almacenan en un archivo xml utilizado como base de datos. El segundo servicio desarrollado fue el backend, en el cual mediante el uso del protocolo HTTP procesa los datos recibidos del primer desarrollo, para luego decidir cuales serán almacenados en archivos xml, asimismo este podrá devolver datos para que sean mostrados en el frontend. La conexión entre frontend y backend fue realizada mediante el framework de flask, se implementaron varios métodos, siendo estos ejecutados al momento de interactuar con la api, estos mueven, procesan y almacenan los datos, entre el backend y el frontend. La versión de prueba de la aplicación puede ser consumida tanto en el localhost como en un cliente distinto como postman, esto permitiendo realizar pruebas en simultaneo.

Palabras clave
- APLICACIÓN WEB
- FRONTEND
- BACKEND

Abstract

We've started developing a web application that will simulate the principal app, this giving us the opportunity to test the functionalities of the API, showing the events being processed, the statistical data that will be stored in a xml file, that will be utilized as a data base. The second service developed was the backend, in the which by using the HTTP protocol will processes the data from the first service, then it will decide which elements will be stored in xml files, it also can return the data to be showed in the frontend. The connection between the frontend and the backend was created by using flask, this framework allowed us to create some methods that when executed move, process and store data between the backend and the frontend. The trial version of the application can be consumed both on localhost and on a different client such as postman for the purpose of simultaneous testing.

Keywords:

- *WEB APPLICATION*
- *FRONTEND*
- *BACKEND*

Introducción

En este informe será planteada la solución al problema planteado en el proyecto, al cual se le dio solución mediante el uso de una aplicación web, levantada con Django, Flask y programada en Python, tomando como pilar la programación orientada a objetos y el manejo de datos entre clases, así como la creación de archivos que funcionaron como bases de datos.

La POO (programación orientada a objetos) nos permitió la creación de varias clases, entre las cuales se manejaron los datos recibidos por la aplicación mediante un archivo xml, permitiéndonos esto leer los archivos, separar los datos, clasificarlos y crear bases de datos en archivos xml.

Se implementaron varias clases, entre las cuales podemos encontrar el main, una clase facturas y una clase para la gestión de los datos.

Desarrollo del tema

Se nos presenta la siguiente problemática:

"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, mostrada en la figura 1"

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

Figura 1. Información a recibir

Fuente: Proporcionada por la empresa.

Para la solución de este problema iniciamos diseñando una interfaz web con código HTML.

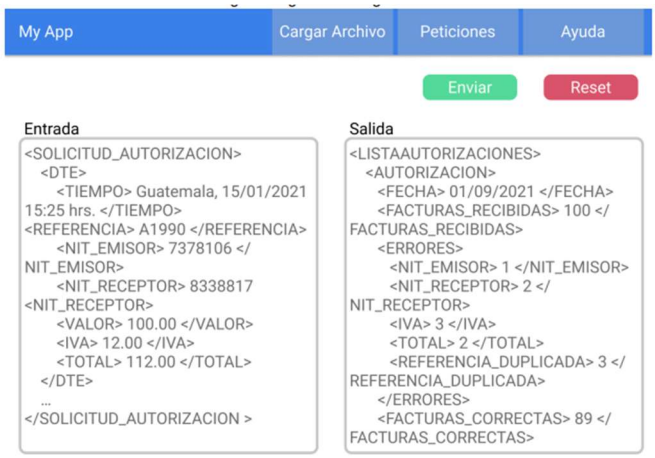

Figura 2. Interfaz gráfica.

Fuente: Sugerida por la empreza.

Tras esto iniciamos a trabajar en VSCode creando nuestras clases, iniciamos con el main, en el cual correrá la aplicación, iniciamos programando el servidor con ayuda de la librería Flask,

posteriormente definimos los endpoints a utilizar, esto para poder empezar a implementar las funciones de cada uno, entre los endpoints encontraremos:

- /cargararchivo.
- /peticiones.
- /peticiones/consultar
- /peticiones/resumeninva
- /peticiones/resumenrango
- /peticiones/grafica
- /peticiones/reportes
- /ayuda
- /enviar
- /salida

Con el método de *"cargararchivo"* leemos los datos de un archivo xml, el cual es manejado en el main y distribuido en las demás clases, las cuales se presentan a continuación:

- Main.
- Facturas.
- Manage.

Al iniciar la ejecución del programa nos dirigiremos al local host, donde podremos empezar a testear los endpoints. En la figura 3 se puede observar el cliente de postman realizando un método "post" al endpoint "/cargararchivo".

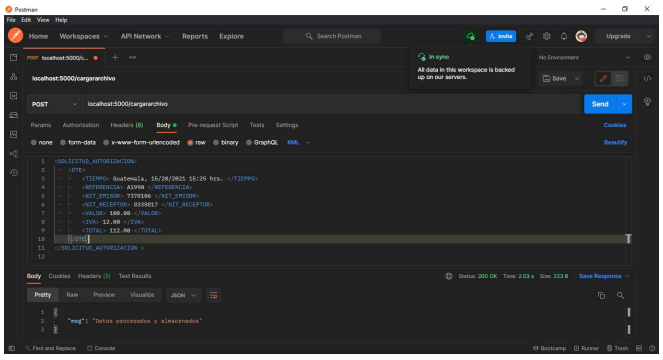

Figura 3. Método post

Fuente: Elaboración propia.

Una vez recibida la información, esta es mostrada en consola, para mostrar que el archivo fue leído, en caso de que alguno de los DTE venga incompleto será ignorado.

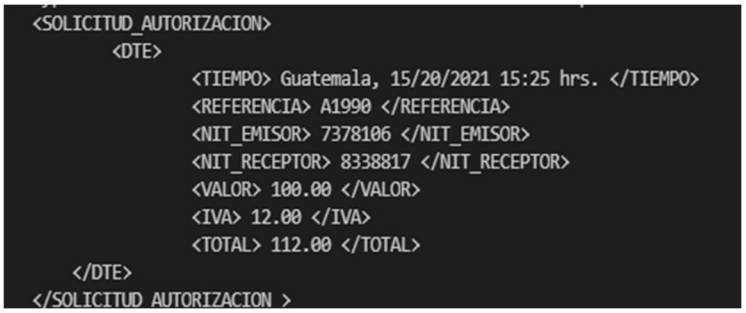

Figura 4. Archivo xml leído.

Fuente: Elaboración propia.

En el endpoint *"/ayuda"* podemos accesar a una breve descripción de los datos del estudiante desarrollador del programa, así como una copia en formato pdf de este reporte. Al acceder a este endpoint los archivos serán abiertos automáticamente, con la ayuda de la librería OS de Python.

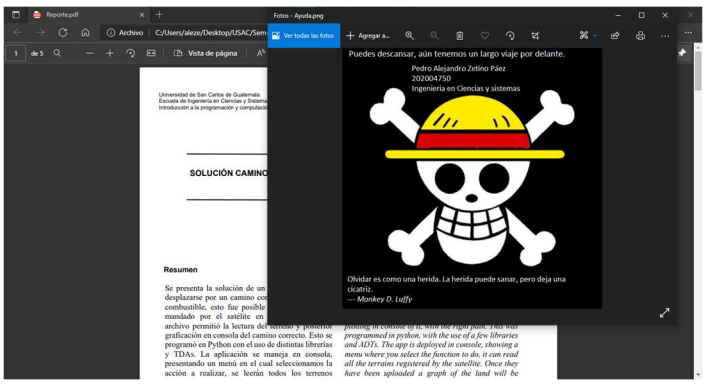

Figura 5. Archivos "/ayuda"

Fuente: Elaboración propia.

Conclusiones

El correcto uso del lenguaje de programación Python, junto al paradigma de la POO y los TDAs permite la solución de una infinidad de problemas, junto a la optimización de una gran cantidad de procesos. En este caso específico se realizó un algoritmo de que optimiza el ensamble de varios productos.

Un ejemplo más cercano a la realidad de este programa serían los programas que manejas las líneas de ensamblaje de carros.

Las ventajas presentadas por estas aplicaciones de la programación es la capacidad de llevar a la realidad algo que solamente imaginabas.

Algo importante a resaltar, es que para el manejo de los TDAs debemos comprender primero el concepto, de lo contrario no podremos utilizarlos de forma eficiente.

Referencias bibliográficas

Grinberg, M. (2018). Flask web development: developing web applications with python. " O'Reilly Media, Inc.".

Holovaty, A., & Kaplan-Moss, J. (2009). The definitive guide to Django: Web development done right. Apress.

Lutz, M. (2001). Programming python. " O'Reilly Media, Inc.".

Anexos

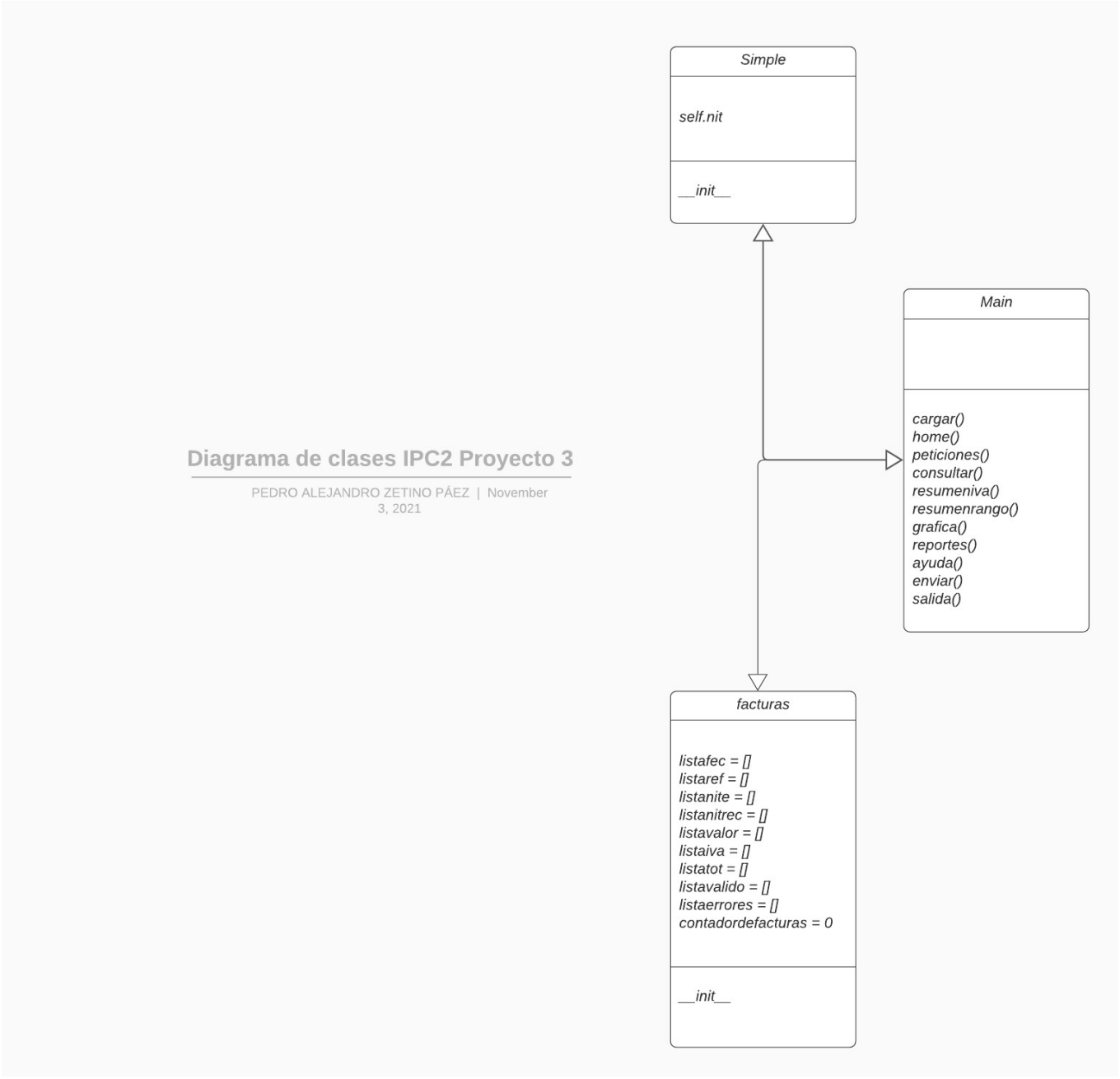
Diagrama de clases IPC2 Proyecto 3
PEDRO ALEJANDRO ZETINO PÁEZ | November 3, 2021
Simple
self.nit
__init__
Main
cargar()
home()
peticiones()
consultar()
resumeniva()
resumenrango()
grafica()
reportes()
ayuda()
enviar()
salida()
facturas
listafec = []
listaref = []
listanite = []
listanitrec = []
listavalor = []
listaiva = []
listatot = []
listavalido = []
listaerrores = []
contadordefacturas = 0
__init__