PROYECTO 2 INTRODUCCIÓN A AL PROGRAMACIÓN 2*

Luciano Isaac Xiquín Ajpop, 201800632[1, **]

[1] *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 2021.*

I. RESUMEN

El siguiente proyecto utiliza dos Frameworks bastante utiles para la creación de páginas web en la actualidad, pese a su versatilidad para manejar bases de datos, en este proyecto se usa un archivo xml para recolectar y guardar la información.

De la misma manera de forma poco ortodoxa se utilizara Django exclusivamente para el frontend.

II. ABSTRACT

The following project uses two very useful Frameworks for creating web pages today, despite their versatility to handle databases, in this project an xml file is used to collect and save it the information.

In the same unorthodox way, Django will be used exclusively for the frontend.

III. PALABRAS CLAVES

XML, Djano, Flask, Desarrollo Web, Backend

IV. KEYWORDS

XML, Djano, Flask, Web Development, Backend

V. MARCO TEÓRICO

Django es un framework web de alto nivel que permite el desarrollo rápido de sitios web seguros y mantenibles. Desarrollado por programadores experimentados, Django se encarga de gran parte de las complicaciones del desarrollo web, por lo que el usuario puede concentrarse en escribir su aplicación sin necesidad de reinventar la rueda. Es gratuito y de código abierto, tiene una comunidad próspera y activa, una gran documentación y muchas opciones de soporte gratuito y de pago. Django puede ser (y ha sido) usado para construir casi cualquier tipo de sitio web — desde sistemas manejadores de contenidos y wikis, hasta redes sociales y sitios de noticias. Puede funcionar con cualquier framework en el lado del cliente, y puede devolver contenido en casi cualquier formato (incluyendo HTML, RSS feeds, JSON, XML, etc). ¡El sitio que estás leyendo actualmente está basado en Django!

Internamente, mientras ofrece opciones para casi cualquier funcionalidad que desees (distintos motores de base de datos , motores de plantillas, etc.), también puede ser extendido para usar otros componentes si es necesario.

Por otra parte Flask

Flask es un “micro” Framework escrito en Python y concebido para facilitar el desarrollo de Aplicaciones Web bajo el patrón MVC.

La palabra “micro” no designa a que sea un proyecto pequeño o que nos permita hacer páginas web pequeñas sino que al instalar Flask tenemos las herramientas necesarias para crear una aplicación web funcional pero si se necesita en algún momento una nueva funcionalidad hay un conjunto muy grande extensiones (plugins) que se pueden instalar con Flask que le van dotando de funcionalidad.

De principio en la instalación no se tienen todas las funcionalidades que se pueden necesitar pero de una manera muy sencilla se pueden extender el proyecto con nuevas funcionalidades por medio de plugins.

---

* ***

** e-mail: 3257440841401@ingenieria.usac.edu.gt

VI. DISEÑO EXPERIMENTAL

El siguiente programa fue desarrollado y examindo con las siguientes herramientas:

Equipo

1. Visual Studio Code 2019
2. Python 3.9
3. matplotlib
4. Pydot
5. Procesador i-7
6. Windows 10

Objetivos del programa

1. Ejemplificar el uso de Django como Frontend
2. Ejemplificar el uso de Flask como Backend
3. Control de Facturas

VII. DESARROLLO

El desarrollo de la API fue la primera en ser realizada ya que sera el esqueleto donde nuestra página web se sostendra. La aplicación fue desarrollada en Flask, la permite un control de los endpoints más preciso y simple. Dentro de Flask se usaron Blueprint, para evitar que todas las funciones y endpoint estén en un solo lugar. Los Endpoint utilizados en este proyecto fueron los siguientes

Endpoints

1. Upload
2. Parse Upload
3. Resumen
4. Graph NIT
5. resumen IVA
6. Graph NIT
7. Reset
8. PDF

El primero de estos métodos es el método Upload que tiene el end point /upload. Este a diferencia de los siguientes endpoints de este programa Upload no esta dentro de el archivo factura.py, si no en app.py la raíz de nuestra API

Los siguientes métodos están dentro de el archcivo factura.py y tienen como prefijo /api antes de llamar la ruta de los diferentes métodos

El siguiente método que se utilizara es el endpoint de analizar el XML, para este paso primero se creara un Objeto Xmltodict el cual nos creara un diccionario y a partir de este se obtendra la imformación para el archivo de salida. La función asociada a este paso es usada más adelante para el resumen de IVA por NIT, con la diferencia que en este paso es explicito la creación de un archivo de sálida.

Pese al nombre tan generico el endpoint /resumen es un endpoint para la creacion de un informe sobre los movimientos como EMISOR y RECEPTOR durante una fecha en especifico por un numero de NIT. El formato para llamar a este de método es el de /api/FECHA/NIT. Estos parametros no tienen que tener ningún otro simbolo más que números. El formato de la fecha en especifico es 'ddmmyyyy'

GRAPH NIT es el siguiente endpoint. Esta ruta está directamente relacionada con el endpoint anterior, es escencialmente el mismo metodo con la direferencia de que aquí llamara otro método para crear una gráfica de pie. En este método se utiliza la librería MatPlotLib para la creación de estás gráficas.

Al igual que el metodo anterior toma dos parametros en su URL

Resumen IVA es un método que creara un resumen de las transacciones hechas durante un lapso de tiempo. La URL toma 3 parametros, 2 de ellos son fechas y la ultima un string para regresar el valor completo o el valor sin IVA. El formato de los dos primeros es ddmmyyyy y el tercero es una unica letra.

Graph IVA es la función relacionada del resumen de IVA que creara una grafica de barras que utiliza el mismo método de la grafica anterior. Y toma los mismos parametros pese a que no va a utilizar el ultimo de estos.

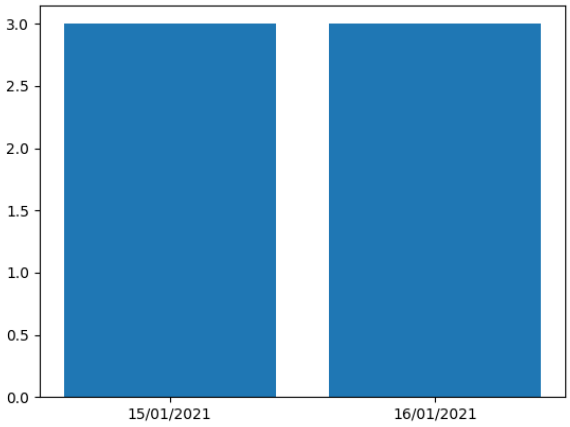

Figura 1: Ejemplo de Grafica de IVA

Figura 2: Ejemplo de Grafica de NIT por Día por NIT

El siguiente método es el de reiniciar la base de datos, o en nuestro caso crear un documento vacio en el archivo XML. El metodo no toma ningun parametro. Es un simple método para eliminar cualquier subElemento del archivo

Finalmente el método final es de la creación de un PDF de todas las peticiones que le hemos hecho a la API. Este metodo utiliza una lista que se ha creado y ha estado guardando todos las peticiones y de que tipo es. Además utiliza la librería FPDF para crear la estructura del archivo. Este es un simple documento que usas celdas para organizar la lista de peticiones.

EJEMPLO DE ARCHIVO DE SALIDA

$$< LISTAAUTORIZACIONES > \quad (1)$$

$$< AUTORIZACION > \quad (2)$$

$$< FECHA > 15/01/2021 < /FECHA > \quad (3)$$

$$< ERRORES > \quad (4)$$

$$< NIT_EMISOR > 1 < /NIT_EMISOR > \quad (5)$$

$$< NIT_RECEPTOR > 1 < /NIT_RECEPTOR > \quad (6)$$

$$< IVA > 0 < /IVA > \quad (7)$$

$$< TOTAL > 0 < /TOTAL > \quad (8)$$

$$< /ERRORES > \quad (9)$$

VIII. FRONTEND

Como se menciono al inicio de este documento la forma en la que se ha creado el FrontEnd de la página es mediante el Framework Django. En adición de la utilización de Django se ha usado Bootstrap para crear la parte gráfica del Proyecto.

Siendo Bootstrap una librería para CSS de lo más versatil para la creación de una página web, es natural que se utilice en conjunto con Bootstrap para hacer más eficiente la completación de este proyecto.

El front en tiene un barra superior que tienen todas las opciones de las peticiones y la visualización de la documentación y información del autor. **Opciones de la Barra superior**

1. Cargar archivo
2. Peticiones
3. Ayuda: usada para ver información de usuario y documentación
4. Y volver al inicio

La interfaz debería ser capaz de subir los archivos desde la computadora y mostrar sus contentido antes y después. Ademas de mostrar cosas como las imagenes creadas por la API y que fueron pedidas desde la interfaz misma.

Todo esto es possible gracias al documento views.py, la cual se encarga de renderisar archivos HTML creado especificamente para el programa con una sintaxis propia para la correcta modulazacion de las vistas. Este programa está relacionado con el archivo urls.py que crea las rutas que deben ser utilizadas para llegar a estos lugares además de poder llamar al json creado por la API.

Esto debido a que internamente Django no está pensado en usar XML como formato predilecto para renderinzar información.

IX. ANEXOS

Figura 3: Barra superior