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.

PROYECTO 3

201901974 – Joseph Raphael Gomez Tzorin

Resumen

El proyecto está enfocado en 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, esto fue solicitado por la Superintendencia de Administración Tributaria (SAT).

Para el desarrollo del Backend 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.

El Frontend 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).

Palabras clave

1. Backend
2. Frontend
3. Métodos
4. Simulación

Abstract

The project is focused on building software that can be consumed from the Internet as a service. This software will receive a message with the data to request the authorization of an Electronic Tax Document (DTE) issued by a taxpayer and in response it will issue a unique authorization number, this was requested by the Superintendency of Tax Administration (SAT).

For the development of the Backend, the student must define on their own the methods they will need to carry out this service. This means that you must implement as many methods as you need to consume the API.

The Frontend consists of a Web application and it will consist of a simulator of the main application, it will contain only the functionalities necessary to test the proper functioning of the API (Service 2), in this application the events that will be processed and the statistical data can be displayed that were stored in the output XML database).

Keywords

1. Backend
2. Frontend
3. Methods
4. Simulation

Introducción

Con Django, puede llevar las aplicaciones web desde el concepto hasta el lanzamiento en cuestión de horas. Django se encarga de gran parte de la molestia del desarrollo web, por lo que puede concentrarse en escribir su aplicación sin necesidad de reinventar la rueda. Es gratis y de código abierto, fue diseñado para ayudar a los desarrolladores a llevar las aplicaciones desde el concepto hasta su finalización lo más rápido posible.
Flask es un marco de aplicación web WSGI ligero . Está diseñado para que la puesta en marcha sea rápida y sencilla, con la capacidad de escalar a aplicaciones complejas. Comenzó como una simple envoltura alrededor de Werkzeug y Jinja y se ha convertido en uno de los marcos de aplicaciones web de Python más populares.

Flask ofrece sugerencias, pero no impone ninguna dependencia o diseño del proyecto. Depende del desarrollador elegir las herramientas y bibliotecas que desea utilizar.

Desarrollo del tema

API

api.py :

esta es el main en donde corre el servidor de flask (backend) donde su función principal es procesar los datos recogidos del programa 1 que es el frontend, posee una serie de métodos GET y POST que se encargan de verificar que información posee el request y realizar un determinado procedimiento.

METODOS

Get_data:

Este método se encarga de leer el archivo xml creado con todas las validaciones hechas y devolverlo como tal para su posterior presentación en el html que se indica por medio de django.

Post_data: este método se encaga de recibir el archivo en formato json y de realizar todas las validaciones necesarias para devolver un xml con la información necesaria que solicita la SAT.

Resumen_iva: 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.
Resumen_rango: 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)

```python
app = Flask(__name__)
cors=CORS(app , resources={r"/*":{"origin":"*"}})
@app.route('/datos', methods=['GET'])
def get_data():
    save_file=open('archi.json', 'r+')

    return Response(status=200,
    response=save_file.read(),
    content_type='text/plain')

@app.route('/datos', methods=['POST'])
def post_data():
    str_file= request.data.decode('utf-8')
    save_file=open('archi.json','w+')
    save_file.write(str_file)
    save_file.close()
    return Response(status=204)
```

Figura 1. Ejemplo de métodos en flask.

Fuente: elaboración propia

Proyecto1

Esta carpeta contiene todo lo necesario para representar el frontend ya que este posee toda la información de nuestro frontend en donde debemos configurar varias cosas antes de empezar a trabajar como por ejemplo cada que creemos una aplicación nueva debemos agregarla en la carpeta settings.py en el apartado que indica INSTALLED_APPS como se muestra a continuación:

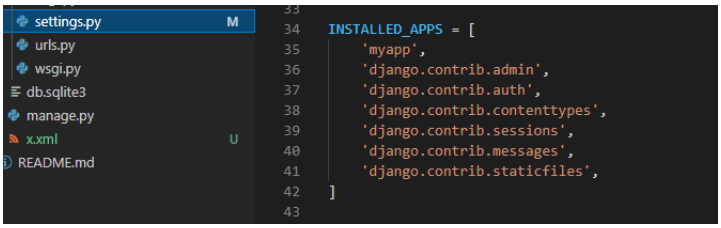


Figura 2. Agregar apps

Fuente: elaboración propia

Luego debemos agregar las urls de dos formas, la primera es agregándolas directamente a la carpeta urls o desde la carpeta de nuestra app crear otra carpera urls y agregar todas nuestras urls y luego agregar esta carpeta a la carpeta original de urls:

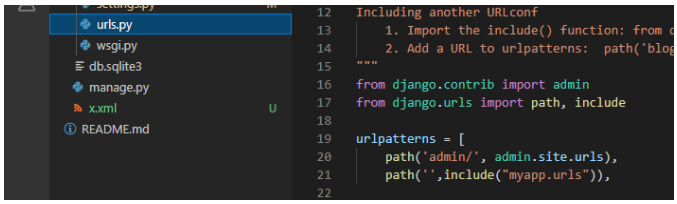


Figura 3. Agregar urls

Fuente: elaboración propia

Myapp

Es la carpeta que contiene toda las clases y métodos para el desarrollo de nuestra aplicación a continuación se explicaran las clases y los métodos que poseen.

Se crean lo siguiente una carpeta templates donde se guardarán todos nuestros html a utilizar, como anteriormente se explica se crea una nuevo archivo urls.py para el almacenamiento de nuestra rutas a utilizar en nuestra app, views.py que es en donde se realizaran las vistas para nuestra app este accede a la carpeta templates por medio de una url y muestra el html que le indiquemos.

Views.py:

Este contendrá todos los métodos para representar la información procesada en el servicio 2, como las graficas y los reportes solicitados.

```python
from django.shortcuts import redirect, render
import requests
import simplejson
import xmltodict

endpoint =' http://127.0.0.1:5000{}'

def cargar(request):
    if request.method == 'GET':
        url=endpoint.format('/datos')
        data=requests.get(url)
        context = {
            'data': data.text,
        }
        return render(request, 'myapp/Cargar.html', context)

    elif request.method == 'POST':
        docs = request.FILES['document']
        data= docs.read()
        str_file=data.decode('utf-8')
        save_file= open('x.xml','w+')
        save_file.write(str_file)
        save_file.close()
        with open("x.xml") as xml_file:
            data_dict=xmltodict.parse(xml_file.read())
            xml_file.close()
            json_data=simplejson.dumps(data_dict)
```

Figura 4. Métodos de views

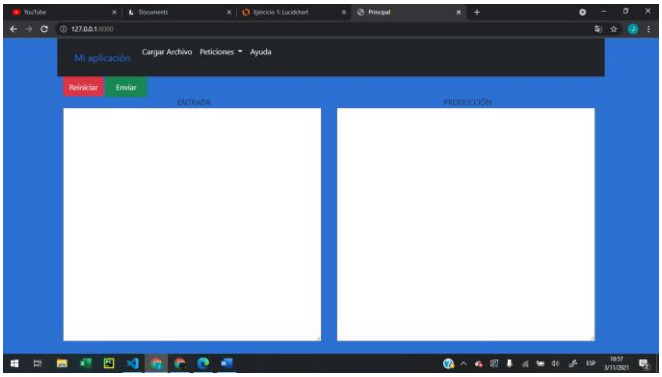
Fuente: elaboración propia

Conclusiones

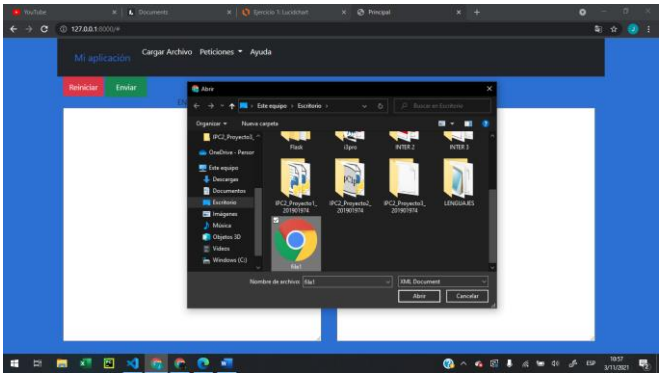
El uso de la POO es de gran ayuda ya que no ahorra una gran cantidad de código ya que al crear una clase y usar todos sus métodos y atributos por medio de un objeto facilita el desarrollo del programa.

Django incluye docenas de extras que puede usar para manejar tareas comunes de desarrollo web, se encarga de la autenticación de usuarios, la administración de contenido, los mapas del sitio y muchas otras cosas más que son de gran ayuda en estos proyectos, además se toma la seguridad muy en serio y ayuda a los desarrolladores a evitar muchos errores de seguridad comunes, como la inyección de SQL, las secuencias de comandos entre sitios, la falsificación de solicitudes entre sitios y el secuestro de clics.

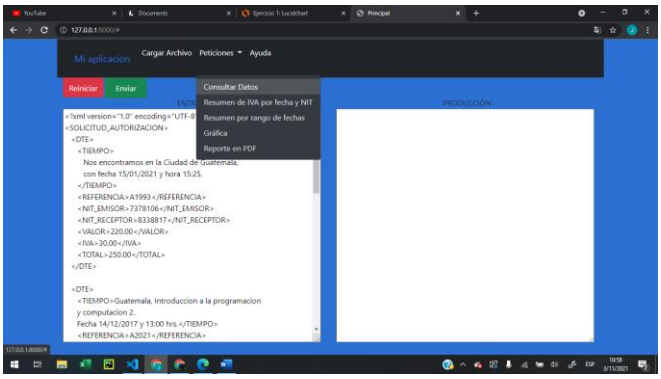
Anexos



Cargar Archivo



Mostrar informacion



Referencias bibliográficas

1. Alchin, M., Kaplan-Moss, J. y Vilches, G. (2013). *Pro Django* (Vol. 2, pág. 28). Presione.

2. Equipo Vértice. (2009). *Diseño básico de páginas web en HTML* . Editorial Vértice.

3. Forcier, J., Bissex, P. y Chun, WJ (2008). *Desarrollo web Python con Django* . Addison-Wesley Professional.

4. Grinberg, M. (2018). *Flask web development: desarrollo de aplicaciones web con python* . "O'Reilly Media, Inc.".

5. Pinkham, A. (2015). *Django desatado* . Sams Publishing.