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 DE AUTORIZACIÓN DE DOCUMENTOS TRIBUTARIOS ELECTRÓNICOS (DTE) PARA LA SUPERINTENDENCIA DE ADMINISTRACIÓN TRIBUTARIA

202003381 – Luisa María Ortíz Romero

Resumen

Se desarrolla una aplicación web para la Superintendencia de Administración Tributaria, que permita procesar la autorización de Documentos Tributarios Electrónicos (DTE). La aplicación permite ingresar solicitudes que siguen un formato de un archivo XML, las solicitudes son validadas y almacenadas. Toda la información recibida por fecha es almacenada en un archivo XML que funciona como base de datos.

La solución se divide en FrontEnd y BackEnd. La parte Front es realizada con el framework Django, el cual trabaja con el patrón MVT (Modelo-Vista-Template) y permite probar todas las funciones realizadas en el BackEnd (API), este desarrollado en Python con el framework Flask, el cual se destaca por la implementación de los métodos CRUD (créate, read, update, delete), en esta parte se realiza toda la lógica y métodos necesarios para cumplir con los requerimientos presentados. Es posible consumir la API desde otro cliente, como Postman, que permite enviar peticiones al servidor como se haría desde la interfaz de usuario.

Palabras clave

DTE, Framework, API, Frontend, Petición

Abstract

A web app is developed for the Superintendency of Tax Administration, which allows the authorization of Electronic Tax Documents (ETD) to be processed. The appl allows to enter requests that follow a format of an XML file, the requests are validated and stored. All the information received by date is stored in an XML file that works as a database.

The solution is divided into FrontEnd and BackEnd. The Front part is made with the Django framework, which works with the MVT pattern (Model-View-Template) and allows testing all the functions performed in the BackEnd (API), is developed in Python with the Flask framework, which is It stands out for the implementation of the CRUD methods (create, read, update, delete), in this part are all the logic and methods necessary to comply with the requirements presented. It is possible to consume the API from another client, such as Postman, which allows to send requests to the server as it would be done from the user interface.

Keywords

ETD, Framework, API, Frontrnd, Request

Introducción

Las aplicaciones web son de mucha demanda en la actualidad, esto debido que se reduce el almacenamiento y requerimientos del usuario ya que todo se aloja en un servidor. El usuario mediante la interfaz web realiza peticiones al servidor, el cual manda una respuesta a la solicitud y se muestra al usuario.

Para permitir la comunicación del front con el servidor, se utiliza el framework Django, el cual gracias a su modelo MVT (Modelo-Vista-Template) permite mostrar la interfaz y enviar peticiones del usuario, las peticiones son recibidas por una API trabajada con el framework Flask, el cual recibe las peticiones y envía respuestas para que puedan ser mostradas al usuario.

La mayor ventaja de este tipo de aplicaciones es que permiten a cualquier persona con un navegador acceder a las funcionalidades, además se evita la instalación de archivos ejecutables y permite a los desarrolladores encargados acceder al código fuente fácilmente.

Desarrollo

a. Django

Django es un framework que tiene por característica principal facilitar la tarea del programador o desarrollador de software. Tal es el caso que su diseño se hace evidente debido a que proporciona una serie de características que facilitan la creación o desarrollo ágil de páginas orientadas a contenidos. Algunos autores mencionan que Django crea una página administrativa o aplicación incorporada para la administración de contenidos y páginas; esta página permite crear, actualizar y eliminar objetos del contenido, pero sin duda lleva un registro de cada acción que realice, proporciona de igual forma una interfaz para administrar los usuarios y grupos de usuarios teniendo en cuenta los permisos que se puede asignar.

Django sigue la arquitectura Modelo Vista Template (MVT). Debido a las capas que tiene el framework Django, permite que los programadores se dediquen a construir únicamente los objetos y la lógica para mostrar o presentar su trabajo y también desde luego el control para ellos.

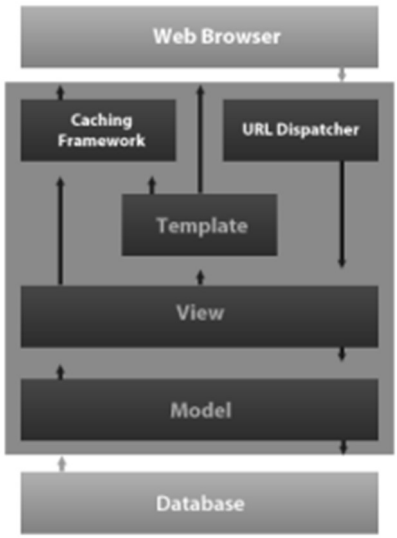

Figura 1. Arquitectura de Django.

Fuente:http://blog.chattyhive.com/wp-content/uploads/2014/01/Django_mvc.png

b. Flask

Flask es un framework 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, actualmente 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. Hay muchas extensiones proporcionadas por la comunidad que facilitan la adición de nuevas funciones.

c. Arquitectura Cliente - Servidor

c.1. Cliente

Programa ejecutable que participa activamente en el establecimiento de las conexiones. Envía una petición al servidor y se queda esperando por una respuesta. Su tiempo de vida es finito una vez que son servidas sus solicitudes, termina el trabajo. El usuario mediante su navegador es un cliente, pues gracias a la guía de la interfaz de usuario, es capaz de enviar peticiones al servidor fácilmente por medio de botones y formularios. Existen otros clientes, como el programa Postman, que permite testrar las funcionalidades implementadas en el backend sin desarrollar una interfaz.

c.2. Servidor

Es un programa que ofrece un servicio que se puede obtener en una red. Acepta la petición desde la red, realiza el servicio y devuelve el resultado al solicitante. Al ser posible implantarlo como aplicaciones de programas, puede ejecutarse en cualquier sistema donde exista TCP/IP y junto con otros programas de aplicación. El servidor comienza su ejecución antes de comenzar la interacción con el cliente, pues está listo para recibir. Algunas aplicaciones inician mostrando la base de datos existente, por lo que es necesario que el servidor esté en ejecución antes de que el usuario decida comenzar su interacción.

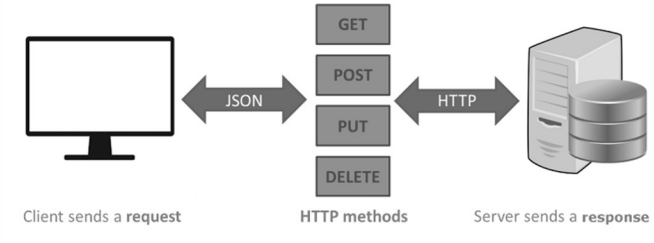

Figura 2. Arquitectura Cliente-Servidor.

Fuente: https://hevodata.com/learn/crud-vs-rest/

Conclusiones

La arquitectura Cliente-Servidor es utilizada para desarrollar aplicaciones web en las que el usuario solo necesita de un navegador para acceder a las funcionalidades, lo cual facilita su uso y expande el alcance de estas.

Los framework son herramientas que ayudan a los desarrolladores a desplegar sus aplicaciones gracias a los métodos y funciones que estas implementan, logrando soluciones funcionales, además de contar con la documentación de cada framework par la resolución de problemas.

El Backend es sin duda la parte principal de las aplicaciones, pues se realizan las funciones que dan solución a los requerimientos planteados, sin embargo, un usuario si conocimientos avanzados de informática no podrá utilizarlas, por lo que es muy significativo el apoyo que los frameworks brindan para la comunicación entre la interfaz de usuario y el servidor.

Referencias bibliográficas

Flask. (s. f.). Pallets. Recuperado 3 de noviembre de 2021, de https://palletsprojects.com/p/flask/

Guerrero Benalcazar, R. I. (2016). *Estudio comparativo de los frameworks Ruby on Rails y Django para la implementación de un sistema informático de control y administración de network marketing*.

Lizama, O., Kindley, G., & Ignacio, J. (2016). Redes de computadores Arquitectura Cliente-Servidor. *Universidad Tecnica Federico Santa Maria*, 1-8.