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

AUTORIZACIONES DTE – PROYECTO 3

201900131 – Elías Abraham Vasquez Soto

Resumen

Autorizaciones DTE, es un software diseñado para la Superintendencia de Administración Tributaria (SAT), que es capaz de recibir datos, solicitando la autorización de Documentos Tributarios Electrónicos (DTE) emitidos por contribuyentes, respondiendo con números únicos de autorización, en caso las solicitudes hayan sido validadas como correctas.

La resolución del proyecto significa optimizar en gran medida los procesos que a diario lleva a cabo la Superintendencia de Administración Tributaria, automatizando los procesos de recolección de solicitudes, así como de la revisión de estas, y su respectiva aprobación, por lo que el potencial que este proyecto posee es considerable, al estar enfocado a que los procesos que la entidad guatemalteca realiza sean más eficientes.

Para la realización del software, se desarrollaron conocimientos como la utilización del paradigma de programación orientado a objetos (POO), el uso de frameworks como Django y Flask, y la manipulación de archivos XML y su utilización como medio de traslado de información.

Palabras clave
API, backend, clases, framework, frontend, XML.

Abstract

DTE Authorizations is a software designed for the Superintendence of Tax Administration (SAT), which can receive data, requesting the authorization of Electronic Tax Documents (DTE) issued by taxpayers, responding with unique authorization numbers, in case the requests have been validated as correct.

The resolution of the project means optimizing to a great extent, the processes that the Superintendence of Tax Administration carries out daily, automating the processes of collecting requests, as well as the review of these, and their respective approval, so the potential that this project has is considerable, as it is focused on making the processes that the Guatemalan entity carries out more efficient.

For the realization of the software, knowledge was developed such as the use of the object-oriented programming paradigm (OOP), the use of frameworks such as Django and Flask, and the manipulation of XML files and their use as a means of transferring information.

Keywords
API, backend, classes, framework, frontend, XML.

Introducción

El presente proyecto se centra en la creación de un software, que sea capaz de recibir solicitudes DTE, entregadas en formato XML, para determinar si estas son válidas o no, y así, generar un número único de autorización a las que hayan sido aprobadas. Además, es posible visualizar toda esta información en un sitio web donde, además, el usuario podrá ver resúmenes de movimientos de IVA por NIT, o bien, montos totales en un rango de fechas, de forma gráfica.

Para el desarrollo del programa, se utilizó el lenguaje de Python, el cual soporta el paradigma de programación orientado a objetos. Dicho paradigma se utilizó para la creación de diversas clases, cuyas instancias, representan las solicitudes DTE que ingresan al software, las autorizaciones por fecha que se generan, y las aprobaciones, que representan las facturas que han sido aprobadas, y que cuentan con el número de autorización.

En este ensayo, se busca familiarizar al lector con las clases y objetos propios del POO, así como del uso de frameworks para el desarrollo del Backend y Frontend del software, todo esto para llegar a la resolución del problema principal planteado.

Desarrollo del tema

Para el desarrollo del programa, fue necesaria la aplicación de múltiples conocimientos en diversos temas, tales como manejo de archivos XML, manejo del paradigma de programación orientado a objetos, uso de frameworks, entre otros. A continuación, se detallan los conocimientos adquiridos y aplicados en este programa, además de la lógica implementada para ciertos métodos propios de las clases creadas.

Estos conocimientos son:

a. Backend

Backend es la capa de acceso a datos de un software o cualquier dispositivo, que no es directamente accesible por los usuarios, además contiene la lógica de la aplicación que maneja dichos datos. El Backend también accede al servidor, que es una aplicación especializada que entiende la forma como el navegador solicita cosas.

Algunos de los lenguajes de programación para Backend son Python, Node.js, PHP, Go, Ruby y C#. Y así como en el frontend, todos estos lenguajes tienen diferentes **frameworks** que permiten trabajar mejor según el proyecto que se esté desarrollando, como Django, Flask, Express.js, Laravel, y muchos más (un framework son herramientas que nos dan un esquema de trabajo y una serie de utilidades y funciones que nos facilita y nos abstrae de la construcción de páginas web dinámicas). Para este proyecto, se utilizó el framework **Flask**.

b. Flask

Flask es un “micro” Framework escrito en Python y concebido para facilitar el desarrollo de Aplicaciones Web bajo el patrón MVC (Modelo-Vista-Controlador).

La palabra “micro” es porque, 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.

c. API

El término API es una abreviatura de *Application Programming Interfaces*, que significa interfaz de programación de aplicaciones. Se trata de un conjunto de definiciones y protocolos que se utiliza para desarrollar e integrar el software de las aplicaciones, permitiendo la comunicación entre dos aplicaciones de software a través de un conjunto de reglas. Así pues, una API es una especificación formal que establece cómo un módulo de un software se comunica o interactúa con otro para cumplir una o muchas funciones.

Para este proyecto, al API (desarrollada en Python) utiliza el protocolo HTTP para la manipulación de información entre Backend y Frontend.

d. Protocolo HTTP

HTTP es el protocolo que permite enviar documentos de un lado a otro en la web. Un protocolo es un conjunto de reglas que determina qué mensajes se pueden intercambiar y qué mensajes son respuestas apropiadas a otros.

En HTTP, hay dos funciones diferentes: servidor y cliente. En general, el cliente siempre inicia la conversación; El servidor responde. HTTP está basado en texto; Es decir, los mensajes son esencialmente bits de texto, aunque el cuerpo del mensaje también puede contener otros medios. El uso del texto facilita el monitoreo de un intercambio HTTP.

Los mensajes HTTP se hacen de un encabezado y un cuerpo. El cuerpo a menudo puede permanecer vacío; Contiene los datos que desea transmitir a través de la red, con el fin de utilizarlo de acuerdo con las instrucciones en el encabezado. El encabezado contiene metadatos, como la información de codificación; Pero, en el caso de una solicitud, también contiene los métodos HTTP importantes.

Verbos HTTP:

- GET: Es el tipo más simple de método de solicitud HTTP. Indica al servidor que transmita los datos identificados por la URL al cliente. Los datos nunca deben ser modificados en el lado del servidor como resultado de una solicitud GET. Una petición GET es de sólo lectura.
- PUT: Se utiliza cuando se desea crear o actualizar el recurso identificado por la URL. Las peticiones PUT contienen los datos que se utilizarán para actualizar o crear el recurso en el cuerpo.
- DELETE: Esta petición debe realizar lo contrario de PUT. Debe utilizarse cuando desee eliminar el recurso identificado por la URL de la solicitud.
- POST: Se utiliza cuando el procesamiento que desea que suceda en el servidor debe repetirse, si la solicitud POST se repite. PUT se utiliza fácilmente en lugar de solicitudes POST, y viceversa. Algunos sistemas utilizan sólo uno, algunos utilizan POST para crear operaciones y PUT para operaciones de

actualización, algunos incluso utilizan POST para actualizaciones y PUT para crear.

Para el presente proyecto, los métodos utilizados son: GET (para obtener los datos), POST (para la creación de datos) y DELETE (para la eliminación, cuando el usuario lo requiera).

e. Frontend

El frontend de una web describe la parte que el usuario puede ver. Incluye todo el contenido que se muestra y que es visible para el público.

El frontend es a menudo llamado GUI (*Graphical User Interface*) porque es la interfaz que los visitantes pueden ver y usar. El frontend se utiliza principalmente para mostrar varios tipos de contenido y hacer que la entrada del usuario esté disponible para el backend. El contenido mostrado incluye la estructura básica del sitio web, como la navegación. El frontend incluye textos, gráficos, vídeos y otros medios.

Así como existen diversos frameworks para desarrollo Backend, también los hay para la parte de Frontend. Para este proyecto, se utilizó el framework **Django**.

f. Django

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. Es gratuito y de código abierto, tiene una comunidad próspera y activa, una gran documentación y muchas opciones de soporte gratuito y de pago.

En un sitio web tradicional basado en datos, una aplicación web espera peticiones HTTP del explorador web (o de otro cliente). Cuando se recibe una petición la aplicación elabora lo que se necesita basándose en la URL y en la información incluida en los datos POST o GET. Dependiendo de qué se necesita, podrá entonces leer o escribir información desde una base de datos o realizar otras tareas requeridas para satisfacer la petición. La aplicación devolverá a continuación una respuesta al explorador web, con frecuencia creando dinámicamente una página HTML para que el explorador la presente insertando los datos recuperados en marcadores de posición dentro de una plantilla HTML.

g. XML - ElementTree

XML es un lenguaje de marcado similar a HTML. Significa Extensible Markup Language (Lenguaje de Marcado Extensible). Esto significa que, a diferencia de otros lenguajes de marcado, XML no está predefinido, por lo que debes definir tus propias etiquetas. El propósito principal del lenguaje es compartir datos a través de diferentes sistemas, como Internet.

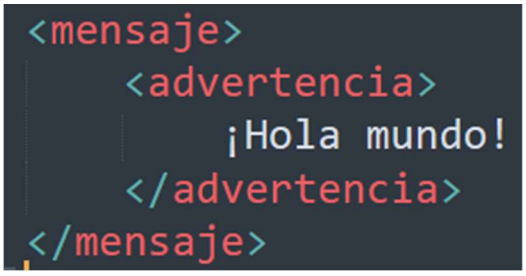

Figura 1. Ejemplo práctico XML.

Fuente: Elaboración propia.

XML posee diversas características:

- Está compuesto por múltiples pares de etiquetas
- Las etiquetas pueden tener atributos
- Los datos correspondientes a la etiqueta se colocan en el medio del par de etiquetas.
- Pueden anidarse etiquetas dentro de otras

Element Tree, el cual es un módulo de Python, es un procesador XML simple y liviano para crear, analizar y procesar datos XML. Las clases diseñadas en Element Tree son:

- ElementTree: Representa toda la jerarquía XML.
- Element: Representa todos los nodos principales en la estructura de árbol.
- SubElement: Representa todos los nodos secundarios en la estructura de árbol.

La manera en que maneja Element Tree las distintas características en un archivo XML, y que se usaron en este proyecto son:

Tabla I.

Manejo de XML con Element Tree.

CARACTERÍSTICA	ATRIBUTO
Nombre de la etiqueta	tag
Atributos	attrib
Valor de la etiqueta	text
Y la próxima etiqueta	tail

Fuente: Elaboración propia.

h. Paradigma de programación orientada a objetos

El paradigma de programación utilizado para la resolución de este problema fue el orientado a objetos. Este facilita el desarrollo, al facilitar la reutilización, creando la cantidad de objetos que sean necesarios sin límite, de manera relativamente sencilla. Para comprender este paradigma, se deben hablar de dos componentes importantes del mismo, los objetos y las clases.

Un objeto posee identidad (un identificador único), estado (propiedades o atributos) y comportamiento (un conjunto de operaciones o métodos). A lo largo del presente ensayo, se detallan la identidad, el estado y el comportamiento de los distintos tipos de objetos programados.

Una clase es la representación de la estructura y comportamiento de un objeto. Es un patrón para la definición de atributos y métodos para un tipo particular de objetos. Una característica de las clases es que todos los objetos que provengan de una clase dada son idénticos en estructura y comportamiento, pero son únicos, aunque posean los mismos valores en sus atributos.

Parafraseando lo anterior expuesto, una clase es un patrón o plantilla, de donde crearemos a todos nuestros objetos.

Existen 4 pilares de la programación orientada a objetos: la abstracción, el encapsulamiento, la herencia y el polimorfismo. Para el programa desarrollado, no fue necesario implementar la herencia ni el polimorfismo, por lo que todas las instancias que se crearon, aunque algunas se encuentran anidadas dentro de otras, ninguna es “hija” de otra.

i. Software utilizado

El proyecto se realizó en lenguaje Python, en su versión más actual a la fecha de redacción de este ensayo (3.9.7). El mismo se ejecuta por medio de una interfaz gráfica.

El IDE se dejó a criterio del desarrollador, por lo que se escogió Visual Studio Code versión 1.60.1, por la facilidad de personalización y múltiples extensiones que vuelven el desarrollo amigable a la vista.

Conclusiones

En la actualidad, son cada vez más las instituciones que optan por automatizar sus procesos, con el fin de ser más eficientes. La automatización de gestiones administrativas es la utilización de sistemas con el fin de hacer más fácil, efectivo y eficiente el funcionamiento de una institución. En este caso, se realizó esta automatización para procesos realizados por la Superintendencia de Administración Tributaria. Para tal efecto, fue necesaria la implementación de una solución integral, implementando diversos conocimientos, para el desarrollo de Backend y Frontend de la aplicación.

En el desarrollo Backend, conlleva dominar el paradigma de programación orientado a objetos, que fue la base del desarrollo del programa, además de la implementación del framework Flask, propio de Python.

Además, el manejo de archivos XML es un tema que se encuentra presente en el manejo de datos, principalmente en Internet, por lo que se vuelve de vital importancia, que se aprenda a procesar los datos de dichos archivos de manera correcta (tal aprendizaje se dio en el desarrollo del presente proyecto).

Por último, se recalca la importancia de que el programador desarrolle la habilidad de diseñar interfaces gráficas amables e intuitivas al usuario (Frontend), en donde se busca que la curva de aprendizaje en ella se nula, otorgándole al usuario la capacidad de dominar el programa al poco tiempo de interactuar con ella.

Referencias bibliográficas

Archivo de Python XML y procesamiento de archivos de configuración. Página web: https://programmerclick.com/article/4079327273/

D. Gilberto. *Listas Enlazadas.* Universidad de los Andes, Facultad de Ingeniería, Escuela de Sistemas.

Guía para principiantes de HTTP y REST. Página web: https://code.tutsplus.com/es/tutorials/a-beginners-guide-to-http-and-rest--net-16340 Ludovico Fisher.

Introducción a Django. Página web: https://developer.mozilla.org/es/docs/Learn/Server-side/Django/Introduction. MDN Web Docs.

O. José, (2004). *Conceptos básicos de la Orientación a Objetos.* Documentación de javaHispano.

O. José, (2004). *Paradigmas de la Orientación a Objetos.* Documentación de javaHispano.

Apéndices

Autómata implementado para reconocer los distintos componentes de las fechas de las solicitudes DTE.

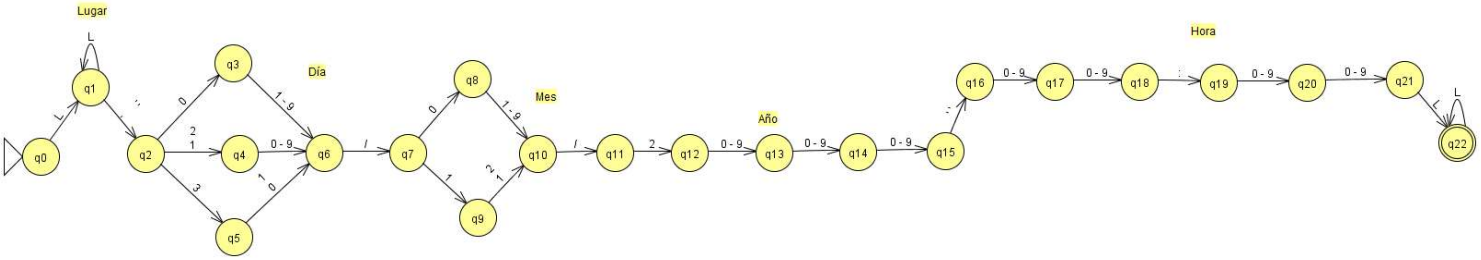

Figura 2. Autómata que reconoce Lugar, Fecha y Hora de la solicitud.

Fuente: Elaboración propia