PROYECTO 2 - CesevXml

201901973 – Raudy David Cabrera Contreras

Resumen

En la realización de este proyecto se utilizó programación web, usando el lenguaje de programación Python, creando una API con flask para el back-end y Django para el front-end del proyecto, utilizando programación orientada a objetos para el manejo de datos e interpretación de estos.

Este proyecto consiste en la carga de archivos tipo CSV, donde la parte visual se usará la librería de Django, donde se analizarán los archivos uno por uno, analizando que los archivos estén de la manera correcta, ya que si el analizador encuentra un error ya no se podrá seguir con el funcionamiento del programa. Una vez comprobado que los archivos estén de manera correcta, se procederá a crear un archivo XML, donde este será mostrado para que el cliente pueda agregar o quitar elementos de este para luego dar la confirmación de envió del XML a la API, donde usando flask se analizara de manera más detallada los archivos, para luego poder mostrar graficas de los elementos que conforman los archivos cargados al principio.

Palabras clave

HTTP, Framework, Xml

Abstract

In the realization of this project, web programming was used, using the Python programming language, creating an API with flask for the back-end and Django for the front-end of the project, using object-oriented programming for data management and interpretation.

This project consists of loading CSV files, where the visual part will be used the Django library, where the files will be scanned one by one, analyzing that the files are in the correct way, since if the analyzer finds an error, it will no longer be ossible to continue with the operation of the program. Once verified that the files are correct, an XML file will be created, where this will be shown so that the client can add or remove elements from it and then confirm the sending of the XML to the API, where using flask the files will be analyzed in a more detailed way, to then be able to show graphs of the elements that make up the files loaded at the beginning.

Keywords

HTTP, Framework, Xml.

Introducción

Este proyecto 2 tiene una arquitectura Cliente-Servidor, donde se utilizo el lenguaje de programación Python, creando una API apoyado de la librería de Flask, un framework para el desarrollo web, utilizando este como el Back-End. Y usando la librería de Django para la interfaz grafica en la web, utilizada como Front-End.

En la aplicación se podrá hacer la carga de 4 archivos tipo csv, donde serán analizados y serán convertidos a un archivo XML, mostrandolo en pantalla al usuario en una caja de texto, donde podrá ver y cambiar la estructura del archivo para luego ser enviado al servidor. Donde la API volverá analizar los archivos para crear graficas que se le mostraran al usuario.

Desarrollo del tema

Para el desarrollo de este proyecto se implementara un API a través de lenguaje Python que pueda ser consumida utilizando el protocolo HTTP, con el que se utilizaron diversas librerías para la manipulación de datos, crear la API del proyecto y el servidor para usarlo en la web:

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

Django: es un framework web de alto nivel que permite el desarrollo rápido de sitios web seguros y mantenibles.

ElementTree: La biblioteca incluye herramientas para analizar XML usando APIs basadas en eventos y documentos, buscando documentos analizados con expresiones XPath, y creando nuevos o modificando documentos existentes.

Minidom: es una implementación mínima de la interfaz Document Object Model (Modelo de objetos del documento), con una API similar a la de otros lenguajes. Está destinada a ser más simple que una implementación completa del DOM y también significativamente más pequeña.

En la pagina principal de la aplicación se podrán cargar 4 CSV, donde en cada de estos archivos vendrán separados por coma los atributos. Los archivos CSV son:

1. Clientes: Almacena los datos relevantes de cada uno de los clientes, donde posee las siguientes columnas:

- Nombre
- Apellido
- Edad
- Fecha cumpleaños
- Fecha primer compra

2. Mejores Clientes: Almacena los datos referentes a los clientes que más compras han realizado a la fecha, donde posee las siguientes columnas:

- Nombre cliente
- Fecha ultima compa
- Cantidad juegos vendidos
- Cantidad en quetzales gastado

3. Juegos mas vendidos: Almacena la información relacionada a los juegos más vendidos en la tienda, donde posee las siguientes columnas:

- Nombre juego
- Fecha ultima compra
- Copias vendidas
- Cantidad en stock

4. Juegos: Almacena la información de los juegos que posee Chet en su catálogo y posee las siguientes columnas:

- Nombre
- Plataforma
- Año de lanzamiento
- Clasificación

Una vez sean cargados los archivos CSV, se analizará cada archivo, línea por línea, verificando que tenga la estructura correcta, sin ningún error, ya que si posee alguno no se podrá seguir con el transcurso normal del programa. Si no existe ningún error en los archivos, por parte del Back-End, se convertirán los archivos a XML, utilizando las librerías de ElementTree y MiniDom. Para luego de ser creado el archivo XML se muestre en un campo de texto al usuario, donde el usuario puede ver y modificar el XML generado de los archivos CSV.

El usuario luego de ver o modificar el XML, dará la orden de 'Enviar' donde el texto del cuadro se enviará a la API, que esta siendo creada con la librería de Flask, una vez el archivo se reciba en la API, se volverá analizar, esta vez será para crear datos que servirán para crear graficas de los datos, que se mostraran a través de la página web.

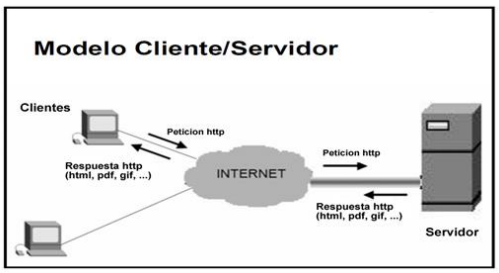
HTTP:

Hypertext Transfer Protocol (HTTP) (o Protocolo de Transferencia de Hipertexto en español) es un protocolo de la capa de aplicación para la transmisión de documentos hipermedia, como HTML. Fue diseñado para la comunicación entre los navegadores y servidores web, aunque puede ser utilizado para otros propósitos también. Sigue el clásico modelo cliente-servidor, en el que un cliente establece una conexión, realizando una petición a un servidor y espera una respuesta del mismo. Se trata de un protocolo sin estado, lo que significa que el servidor no guarda ningún dato (estado) entre dos peticiones. Aunque en la mayoría de los casos se basa en una conexión del tipo TCP/IP, puede ser usado sobre cualquier capa de transporte segura o de confianza, es decir, sobre cualquier protocolo que no pierda mensajes silenciosamente, tal como UDP.[1]

Modelo cliente servidor:

La arquitectura cliente-servidor es un modelo de aplicación distribuida en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes. Un cliente realiza peticiones a otro programa, el servidor, quien le da respuesta. Esta idea también se puede aplicar a programas que se ejecutan sobre una sola computadora, aunque es más ventajosa en un sistema operativo multiusuario distribuido a través de una red de computadoras.

Algunos ejemplos de aplicaciones computacionales que usen el modelo cliente-servidor son el Correo electrónico, un Servidor de impresión y la World Wide Web.[2]

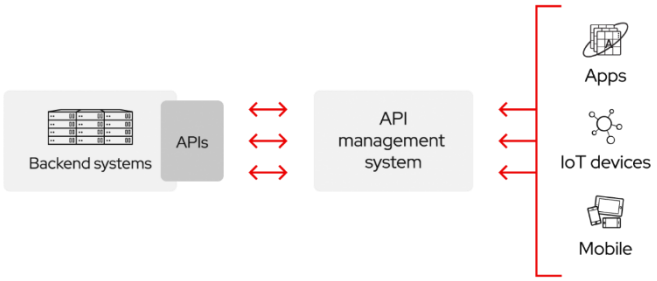


API:

Una API es un conjunto de definiciones y protocolos que se utiliza para desarrollar e integrar el software de las aplicaciones. API significa interfaz de programación de aplicaciones.

Las API permiten que sus productos y servicios se comuniquen con otros, sin necesidad de saber cómo están implementados. Esto simplifica el desarrollo de las aplicaciones y permite ahorrar tiempo y dinero. Las API le otorgan flexibilidad; simplifican el diseño, la administración y el uso de las aplicaciones, y proporcionan oportunidades de innovación, lo cual es ideal al momento de diseñar herramientas y productos nuevos (o de gestionar los actuales).

A veces, las API se consideran como contratos, con documentación que representa un acuerdo entre las partes: si una de las partes envía una solicitud remota con cierta estructura en particular, esa misma estructura determinará cómo responderá el software de la otra parte.[3]



Framework:

El concepto framework se emplean muchos ámbitos del desarrollo de sistemas software, no solo en el ámbito de aplicaciones Web. Podemos encontrar frameworks para el desarrollo de aplicaciones médicas, de visión por computador, para el desarrollo de juegos, y para cualquier ámbito que pueda ocurrírsenos.

En general, con el término framework, nos estamos refiriendo a una estructura software compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación. En otras palabras, un framework se puede considerar como una aplicación genérica incompleta y configurable a la que podemos añadirle las últimas piezas para construir una aplicación concreta.

Los objetivos principales que persigue un framework son: acelerar el proceso de desarrollo, reutilizar código ya existente y promover buenas prácticas de desarrollo como el uso de patrones.

Un framework Web, por tanto, podemos definirlo como un conjunto de componentes (por ejemplo, clases en java y descriptores y archivos de configuración en XML) que componen un diseño reutilizable que facilita y agiliza el desarrollo de sistemas Web.[4]

Conclusiones

Cuando se trabaja la parte del Front-End se debe tener mucha documentación, ya que se tiene que estar viendo cómo se trasladan los archivos y datos de un html a la librería de Django, es un poco complicado, pero una vez se tiene la idea de cómo implementar eso, será más fácil manejar los datos. Conectar el

Back-End con el Front-End, es de lo mas difícil, ya que no se tiene documentación de cómo se podría implementar los POST y los GET, pero una ves conectadas los framework, se podrá trabajar ya con la orientación a objetos que se viene trabajando desde los principios de los cursos de programación.

Anexos:

Repositorio del proyecto, ubicado en GitHub: https://github.com/201901973-RaudyCabrera/IPC2_Proyecto2_201901973_Junio2021

Referencias bibliográficas

1. https://developer.mozilla.org/es/docs/Web/HTTP
2. https://redespomactividad.weebly.com/modelo-cliente-servidor.html
3. https://www.redhat.com/es/topics/api/what-are-application-programming-interfaces
4. http://www.lsi.us.es/~javierj/investigacion_ficheros/Framework.pdf