PROYECTO 3: INFORMACION ESTADISTICA

201900603 – Karen Michelle Gatica Arriola

Resumen

Se utiliza un XML como una base de datos de todos aquellos casos reportados por fallos de cualquier tipo, la tarea del software es organizar esos datos y presentarlos de manera ordenada en un documento llamado: estadísticas.xml, tomando en cuenta que la vista del documento original y el modificado deben aparecer una al lado de la otra.

Se hará uso de herramientas web, desarrolladas en Python como lo son: Django y Flask que son dos Frameworks los cuales nos permitirán alojar los elementos de Frontend como Backend además de unificarlo para que este pueda trabajar en conjunto.

Se obtendrán las respuestas a través de un servidor de manera local dentro de la computadora, sin embargo, este tendrá varios clientes que pueden realizar peticiones a la misma dirección de acceso a la cual accede el servidor local, dicho cliente será: Postman.

La información puede ser filtrada a conveniencia del usuario, por código, nombre y fecha.

Palabras clave

API, Flask, Django, XML, Programación Orientada a Objetos

Abstract

An XML is used as a database of all those cases reported by failures of any kind, the task of the software is to organize this data and present it in an orderly way in a document called: statistics.xml, considering that the view of the document original and modified must appear side by side.

It will make use of web tools, developed in Python such as: Django and Flask, which are two Frameworks which will allow us to host Frontend elements as Backend as well as unify it so that it can work together.

The responses will be obtained through a server locally within the computer; however, this will have several clients that can make requests to the same access address that the local server accesses, said client will be: Postman.

The information can be filtered at the user's convenience, by code, name, and date.

Keywords

API, Flask, Django, XML, Object-Oriented Paradigm

Introducción

Se requiere de servicios web para poder realizar un sitio web completo, para este proyecto se hizo de un framework con una metodología MVC (modelo vista controlador) el cual permite trabajar con distintos elementos como HTML, CSS, Python, entre otros.

Se utiliza un archivo XML como base de datos, el cual obtiene la información de manera ordenada y la transmite, permitiendo filtrar la información a conveniencia del cliente.

Por medio de la conexión de Flask y Django es posible realizar consultas por más de un cliente, ya que se pueden hacer consultas a través de otro cliente el cual consume la API, ese cliente será Postman. Cada consulta realizada puede ser visualizada dentro de la interfaz y al mismo tiempo se puede ver el archivo XML original, la interfaz además tiene botones los cuales permitirán la carga del archivo, aquel que envirará los datos al servidor y aquel que limpiará la interfaz.

Desarrollo del tema

API: se define como una especificación de posibles interacciones con un componente de software una cosa a tener en cuenta es que el nombre de algunas API se usa a menudo para referirse tanto a la especificación de las interacciones como al componente de software real con el que interactúa.

Se accede a los objetos como variables con una estructura interna compleja, y en muchos lenguajes son efectivamente punteros, que sirven como referencias reales a una sola instancia de dicho objeto en la memoria dentro de un montón o pila. Proporcionan una capa de abstracción que se puede utilizar para separar el código interno del externo.

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. Los objetos se crean llamando a un tipo especial de método en la clase conocido como constructor. Un programa puede crear muchas instancias de la misma clase mientras se ejecuta, que operan de forma independiente.

XML: son las siglas de Extensible Markup Language. Es un lenguaje de marcado basado en texto derivado del lenguaje de marcado estándar generalizado. Las etiquetas XML identifican los datos y se utilizan para almacenar y organizar los datos, en lugar de especificar cómo mostrarlos como etiquetas HTML, que se utilizan para mostrar los datos. XML no reemplazará al HTML en un futuro cercano, pero presenta nuevas posibilidades al adoptar muchas características exitosas de HTML.

- ✓ **XML es extensible:** XML le permite crear sus propias etiquetas autodescriptivas, o lenguaje, que se adapte a su aplicación.

- ✓ **XML lleva los datos, no los presenta**: XML le permite almacenar los datos independientemente de cómo se presenten.

- ✓ **XML es un estándar público:** XML fue desarrollado por una organización llamada World Wide Web Consortium (W3C) y está disponible como estándar abierto.

Usos Principales de un XML:

- XML puede funcionar entre bastidores para simplificar la creación de documentos HTML para sitios web grandes.
- XML se puede utilizar para intercambiar información entre organizaciones y sistemas.
- XML se puede utilizar para descargar y recargar bases de datos.
- XML se puede utilizar para almacenar y organizar los datos, lo que puede personalizar sus necesidades de manejo de datos.
- XML se puede combinar fácilmente con hojas de estilo para crear casi cualquier resultado deseado.

HTML: Se utiliza para diseñar páginas web utilizando un lenguaje de marcado. HTML es la combinación de hipertexto y lenguaje de marcado. El hipertexto define el vínculo entre las páginas web. Se utiliza un lenguaje de marcado para definir el documento de texto dentro de la etiqueta que define la estructura de las páginas web.

- **Elementos y etiquetas:** HTML utiliza etiquetas y elementos predefinidos que le indican al navegador cómo mostrar correctamente el contenido. Recuerde incluir etiquetas de cierre. Si se omite, el navegador aplica el efecto de la etiqueta de apertura hasta el final de la página.
- **Estructura de la página HTML:** la estructura básica de una página HTML se presenta a continuación. Contiene los elementos esenciales de los bloques de construcción sobre los que se crean todas las páginas web.

Ventajas:

- ✓ HTML se utiliza para crear sitios web.
- ✓ Es compatible con todos los navegadores.
- ✓ Se puede integrar con otros lenguajes como CSS, JavaScript, etc.

Desventajas:

- ✓ HTML solo puede crear páginas web estáticas.
- ✓ Para las páginas web dinámicas, se deben utilizar otros idiomas.
- ✓ Se debe escribir una gran cantidad de código para crear una página web simple.
- ✓ La característica de seguridad no es buena.

CSS: Las hojas de estilo en cascada, a las que se hace referencia cariñosamente como CSS, es un lenguaje de diseño simple destinado a simplificar el proceso de hacer que las páginas web sean presentables. CSS le permite aplicar estilos a las páginas web. Más importante aún, CSS le permite hacer esto independientemente del HTML que compone cada página web. CSS es fácil de aprender y comprender, pero proporciona un control poderoso sobre la presentación de un documento HTML.

Ventajas de CSS

- ✓ **CSS ahorra tiempo:** puede escribir CSS una vez y reutilizar la misma hoja en varias páginas HTML.
- ✓ **Fácil mantenimiento:** para realizar un cambio global, simplemente cambie el estilo y todos los elementos de todas las páginas web se actualizarán automáticamente.
- ✓ **Motores de búsqueda**: CSS se considera una técnica de codificación limpia, lo que significa que los motores de búsqueda no tendrán que esforzarse por "leer" su contenido.
- ✓ **Estilos superiores a HTML:** CSS tiene una gama de atributos mucho más amplia que HTML, por lo que puede darle un aspecto mucho mejor a su página HTML en comparación con los atributos HTML.
- ✓ **Navegación fuera de línea:** CSS puede almacenar aplicaciones web localmente con la ayuda de captura fuera de línea. Con esto podemos ver sitios web fuera de línea.

Django: es un marco web de Python de alto nivel que fomenta el desarrollo rápido y el diseño limpio y pragmático. Creado por desarrolladores experimentados, 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.

Características principales del framework:

- ✓ **Versátil:** usado para construir casi cualquier tipo de sitio web. Internamente, mientras ofrece opciones para casi cualquier funcionalidad: distintos motores de base de datos, motores de plantillas.
- ✓ **Seguro:** ayuda a los desarrolladores evitar varios errores comunes de seguridad al proveer un framework proporciona una manera segura de administrar cuentas de usuario y contraseñas, evitando así errores comunes de los programadores como colocar informaciones de sesión en cookies donde es vulnerable o almacena directamente las contraseñas en un hash.
- ✓ **Escalable:** en cuenta una clara separación entre las diferentes partes significa que puede escalar para aumentar el tráfico al agregar hardware en cualquier nivel: servidores de cache, servidores de bases de datos o servidores de aplicación
- ✓ **Mantenible:** está escrito usando principios y patrones de diseño para fomentar la creación de código mantenible y reutilizable
- ✓ **Portable:** Django cuenta con el respaldo de muchos proveedores de alojamiento web, y que a menudo proporcionan una infraestructura específica y documentación para el alojamiento de sitios de Django.

Infraestructura de Django

Django maneja una modelo vista controlador:

- **URLs:** Aunque es posible procesar peticiones de cada URL individual vía una función

individual, es mucho más sostenible escribir una función de visualización separada para cada recurso. Se usa un mapeador URL para redirigir las peticiones HTTP a la vista apropiada basándose en la URL de la petición. El mapeador URL puede también emparejar patrones de cadenas o dígitos específicos que aparecen en una URL y los pasan a la función de visualización como datos.

- **Vista:** es una función de gestión de peticiones que recibe peticiones HTTP y devuelve respuestas HTTP. Las vistas acceden a los datos que necesitan para satisfacer las peticiones por medio de modelos, y delegan el formateo de la respuesta a las plantillas.

- **Modelos:** son objetos de Python que definen la estructura de los datos de una aplicación y proporcionan mecanismos para gestionar (añadir, modificar y borrar) y consultar registros en la base de datos.

- **Plantillas:** una plantilla es un fichero de texto que define la estructura o diagrama de otro fichero, con marcadores de posición que se utilizan para representar el contenido real. Una vista puede crear dinámicamente una página usando una plantilla, rellenándola con datos de un modelo.

Flask: es un marco de aplicación web escrito en Python. Fue desarrollado por Armin Ronacher, quien dirigió un equipo de entusiastas internacionales de Python llamado Poocco. Flask se basa en el kit de herramientas Werkzeg WSGI y el motor de plantillas Jinja2, ambos proyectos de Pocco.

El matraz se denomina a menudo microframework. Está diseñado para mantener el núcleo de la aplicación simple y escalable. En lugar de una capa de abstracción para el soporte de la base de datos, Flask admite extensiones para agregar tales capacidades a la aplicación.

Contiene los siguientes componentes

- ✓ **WSGI:** la Interfaz de puerta de enlace del servidor web (Interfaz de puerta de enlace del servidor web, WSGI) se ha utilizado como estándar para el desarrollo de aplicaciones web Python.

- ✓ **Werkzeug:** es un conjunto de herramientas WSGI que implementa solicitudes, objetos de respuesta y funciones de utilidad. Esto permite construir un marco web sobre él. El marco Flask utiliza Werkzeg como una de sus bases.

- ✓ **jjinja2:** es un motor de plantillas popular para Python. Un sistema de plantillas web combina una plantilla con una fuente de datos específica para representar una página web dinámica.

HTTP: El Protocolo de transferencia de hipertexto () es un protocolo a nivel de aplicación para sistemas de información hipermedia distribuidos y colaborativos. HTTP es un protocolo genérico y sin estado que se puede usar para otros fines, así como extensiones de sus métodos de solicitud, códigos de error y encabezados.

Básicamente, HTTP es un protocolo de comunicación basado en TCP / IP, que se utiliza para entregar datos archivos HTML, archivos de imagen, resultados de consultas. Proporciona una forma estandarizada para que las computadoras se comuniquen entre sí. La especificación HTTP especifica cómo se construirán y enviarán los datos de solicitud de los clientes al servidor, y cómo los servidores responderán a estas solicitudes.

- ✓ **HTTP no tiene conexión:** el cliente HTTP, es decir, un navegador inicia una solicitud HTTP y después de que se realiza una solicitud, el cliente espera la respuesta. El servidor procesa la solicitud y envía una respuesta, después de lo cual el cliente desconecta la conexión. Por lo tanto, el cliente y el servidor se conocen entre sí solo durante la solicitud y la respuesta actuales.

- ✓ **HTTP es independiente de los medios:** es decir, cualquier tipo de datos puede ser enviado por HTTP siempre que tanto el cliente como el servidor sepan cómo manejar el contenido de los datos.

- ✓ **HTTP no tiene estado:** como se mencionó anteriormente, HTTP no tiene conexión y es un resultado directo de que HTTP es un protocolo sin estado. El servidor y el cliente se conocen entre sí solo durante una solicitud actual. Debido a esta naturaleza del protocolo, ni el cliente ni el navegador pueden retener información entre diferentes solicitudes en las páginas web.

Para implementar la solución a la interfaz propuesta, se propuso un diseño acorde a las necesidades que debían ser cubiertas, este incluye una barra de opciones la cual incluye la interfaz de peticiones y la pestaña de ayuda, la cual despliega dos opciones más, las cuales con: Información del Estudiante la cual puede ser accedida desde la barra o a través de: http://127.0.0.0/estudiante y Documentación la cual puede ser accedida desde la barra o a través de: http://127.0.0.0/documentacion las cuales son dos páginas estáticas en las cuales se da información y se presenta este reporte.

Para la pestaña de Peticiones la cual puede ser accedida desde la barra con un clic o desde la dirección: http://127.0.0.0/peticiones, la vista fue realizada con HTML agregando CSS dentro del mismo que viene de contiene un botón el cual cargará el archivo desde el directorio que el usuario prefiera, al cargar el archivo este será desplegado dentro del área de texto el cual presentará del lado izquierdo el archivo original. Para comunicarse con el servidor será necesario enviarlos por medio del botón enviar el cual contiene instrucciones dentro de JavaScript, con una función llamada: EnviarXML(), el cual envia los datos al servidor.

El servidor devolverá al usuario un archivo modificado, el encargado de realizar la transformación es Python por medio la función: transformarXML() y para devolver el archivo de salida con el nombre de estadísticas.xml: salidaXML(), para poder recibir el archivo y devolverlo dentro del textarea es CargarXML(), el cual devolverá el archivo ordenado.

Las gráficas se realizarán con Graphviz con forma de árbol para cada una de las consultas.

Anexos

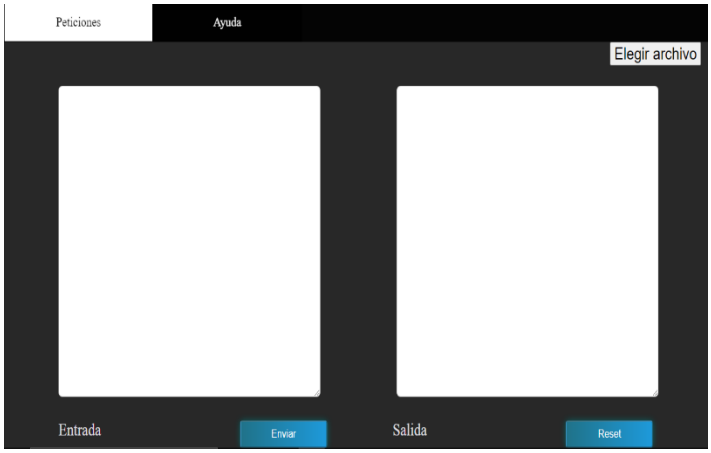

Figura 1: Vista de Petición

Fuente: elaboración propia, 2021

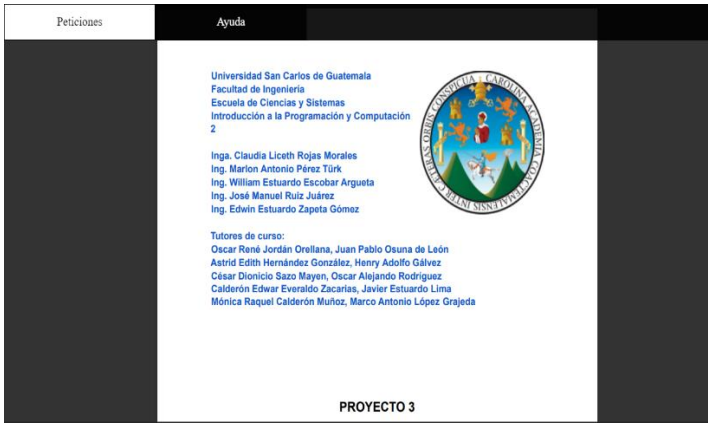

Figura 2: Documentación del Proyecto

Fuente: elaboración propia, 2021

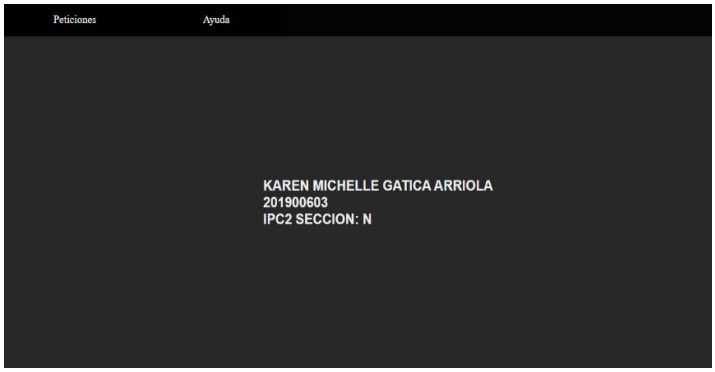

Figura 3: Información del Estudiante

Fuente: elaboración propia, 2021

Conclusiones

1. Django es un framework muy importante para el desarrollo de aplicaciones Cliente-Servidor, el cual contiene un patrón de diseño sumamente eficiente de Modelo Vista Controlador.

2. Flask se adapta muy bien debido a su diseño minimalista y rápida respuesta ante las peticiones realizadas a su servidor, además de ser muy versátil en cuanto a conexión con otros frameworks como Django.

3. XML es un formato muy importante para el almacenamiento de información de manera ordenada y por etiquetas.

4. Al utilizar POO fue más sencillo implementar clases y acceder a sus métodos, para realizar cada una de las acciones requeridas para llegar a ordenar la información de manera más eficiente.

Referencias bibliográficas

1. Berners-Lee, T., Fielding, R., & Frystyk, H. (1996). Hypertext transfer protocol--HTTP/1.0.

2. Bowers, M. (2007). *Pro CSS and HTML design patterns*. Apress.

3. Forcier, J., Bissex, P., & Chun, W. J. (2008). *Python web development with Django*. Addison-Wesley Professional.

4. Idris, N., Foozy, C. F. M., & Shamala, P. (2019). A Generic Review of Web Technology: DJango and Flask. *International Journal of Engineering Information Computing and Application*, *1*(1).

5. Ong, S. P., Cholia, S., Jain, A., Brafman, M., Gunter, D., Ceder, G., & Persson, K. A. (2015). The Materials Application Programming Interface (API): A simple, flexible and efficient API for materials data based on REpresentational State Transfer (REST) principles. *Computational Materials Science*, *97*, 209-215.

.