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.

GENERADOR DE ESTADISTICAS DE REPORTES DE ERRORES

201906053 – Jose Rodrigo Rodas Palencia

Resumen

La aplicación presentada se basa en un servicio web el cual busca procesar archivos XML los cuales contienen reportes de errores por diferentes usuarios o usuarios iguales así como también reportes en diferentes fechas o fechas iguales.

La aplicación aplica el uso de dos servicios los cuales implican un front-end y un back-end, y estos usan cada uno un marco de trabajo, flask y django respectivamente.

La posición de utilizar dos frameworks implica el uso de más recursos pero a su vez saber cómo funciona la API al enviarle y recibir peticiones que realiza el front-end.

Palabras clave

- Front-End
- Back-end
- Rest API
- Eventos
- XML

Abstract

The aplication is based on a web service that search The maneg of XML files that contains error reports do it by diferents users or the same user and also reports do it in diferents dates or do it the same date.

The aplication aplicates the use of two services that are a front-end and a back-end and each of this use a diferent framework, flask and django repespectevely.

The desicion of using this two framework implicates the use of to much resources but also this gives the posibility of see how the API works when it send or recibe request from the front-end.

Keywords

- *Front-End*
- *Back-End*
- *Rest API*
- *Event*
- *XML*

Introducción

Un web services se define como una tecnología que brinda un servicio por medio de protocolos para realizar una comunicación entre servicios. El proyecto presentado utiliza los protocolos http para comunicar una API, la cual es un servicio el cual recibe peticiones y a la vez las envía al servicio el cual la consume, con el servicio que se le presenta al usuario que en este caso se le conoce como front-end el cual solo se encarga de presentar los datos en un navegador.

El proyecto se basa en recibir un archivo en formato XML el cual posee diferentes valores a procesar, por lo tanto el front-end es el encargado de procesar este archivo y enviárselo a la API para que esta procese los datos y genere todo lo necesario para la consulta y generación de datos y archivos de salida que se presentaran posteriormente.

Desarrollo del tema

Framework: se define como framework a un marco de trabajo que da diferentes posibilidades al momento de desarrollar software, se basa como un tipo de plantilla en la cual se pueden construir diferentes aplicaciones con una base ya echa. En este caso se utilizan dos frameworks que utiliza el lenguaje Python para el desarrollo web son los siguientes:

- **Flask**: se le conoce como un framework muy famoso ya que nos permite crear aplicaciones web sin la necesidad de importar demasiadas librerías si no que solo las necesarias para un buen desarrollo. Gracias a flask es posible construir una API la cual detona como nuestro back-end en el proyecto, este las peticiones POST y GET para recibir datos y realizar la comunicación con el front-end.
- **Django**: django es otro framework para el desarrollo de páginas web muy famoso el cual utiliza el patrón de diseño modelo-vista-controlador ya que este posee diferentes archivos para realizar las configuraciones del proyecto como lo son las vistas, los modelos, el reidreccionamiento de las vistas, la bases de datos, etc. En el presente proyecto django funciona como el front-end de la aplicación pero a su vez django utiliza los métodos GET y POST para la comunicación con la vista y el back-end y así retornar la vista al usuario.

Rest API: una api se define como un conjunto de requisiciones las cuales utilizan los protocolos http para realizar la comunicaciones básicas y manejo de dato entre aplicaciones. Las peticiones más utilizadas en el desarrollo de esta aplicación son:

- **POST**: crea datos en el servidor
- **GET**: lectura de datos en el host

En este caso la rest API fue construida con flask creando un método por cada petición la cual puede hacer la vista y esta retorna los datos en forma de texto, a diferencia de otros en este caso la comunicación entre el cliente y el servidor no se realiza por medio de json sino que se utiliza texto con etiquetas las cuales lo identifican.

BACK-END: La rest API construida consiste en diferentes métodos los cuales componen las peticiones las cuales puede realizar, este se encuentra alojado en el puerto 8800 de manera local, los más importantes son los siguientes:

- **Carga de archivo**: este método es una petición post por lo que recibe información en este caso un texto con sintaxis XML llamado “ourfile”, una vez recibida esta información cada línea se convierte en un elemento de una lista y se va validando cada línea con las expresiones regulares para validar la sintaxis de la entrada y con ello guardarlo en un objeto

evento y estos almacenarlos en la base de datos que en este caso será listas que componen el archivo de salida.

  - **Evento**: evento es una clase POO la cual posee los atributos con los cuales se escribe en el archivo de entrada un reporte de un error los cuales son la fecha, el usuario que realizo el reporte, la lista de afectados y el código de error.

Con todo esto realizado se retorna toda la información almacenada en formato de texto.

- **Generar Salida**: la opción generar salida se le hace un llamado GET y con la base de datos llena empieza a construir el archivo de salida con formato XML, creando un valor llamado estadística el cual posee la fecha, los usuarios que hicieron un reporte en esa fecha junto con el número de reportes que realizaron en esa fecha así como la lista de usuarios afectados y por ultimo una lista de los errores reportados en la fecha con el código de error y las veces que fue reportado, esta "estadística" se realiza por cada fecha que se encontró en el archivo de entrada, a su vez esta retorna el archivo de salida en formato de texto.

- **Graficar Errores**: con los proceso anteriores hechos esta función recibe un parámetro a través de un método POST recibiendo una cadena de texto con la etiqueta "codigo" con ello se recorren las listas que contienen la información de los errores las cuales están clasificadas por fecha y por código de error, con ello echo se hace el llamado a una librería la cual recibe dos listas, uno que será las fechas en que se realizó el reporte del error solicitado y la otra se compone del número de veces que se hizo un reporte del error en esa misma fecha, con esto echo se realiza una gráfica de barras la cual muestra los valores indicados, a su vez retorna un mensaje en el cual confirma que se realizó la gráfica.

- **Graficar Fecha**: con los proceso anteriores hechos esta función recibe un parámetro a través de un método POST recibiendo una cadena de texto con la etiqueta "fecha" con ello se recorren las listas que contienen la información de las fechas las cuales están clasificadas por fecha y por usuario que reporto un error, con ello echo se hace el llamado a una librería la cual recibe dos listas, uno que será los usuarios que realizaron el reporte de un error en la fecha solicitada y la otra se compone del número de veces que se hizo un mismo usuario un reporte de error en esa misma fecha, con esto echo se realiza una gráfica de barras la cual muestra los valores indicados, a su vez este devuelve un mensaje en el cual se confirma la generación de la gráfica.

- **Reset**: el método reset únicamente hace un llamado GET y limpia toda la información almacenada al momento del programa dejando así como si el servidor hubiera realizado un reinicio, a su vez envía un mensaje en el cual confirma la eliminación de todos los datos.

FRONT-END: en el proyecto se aplicó el uso del framework django con el cual se construyó un servicio el cual únicamente retorna vistas agradables al usuario, este se encuentra almacenado en el puerto 8000 de manera local. Algunas de las vistas más importantes son las siguientes:

- **Home**: la vista home es la primera vista con la cual el usuario se encontrara al momento de acceder al servicio, este le mostrara una serie de botones los cuales cada uno tiene una función diferente. Como primer punto se tiene un input para el archivo de entrada el cual lo redireccionarlo a la ventana archivo. Otra de

las funcionalidades es el botón enviar el cual redirige a la vista archivo. El botón consulta que redirige a la vista consultar. El botón filtros que redirige a la vista filtros. El botón reset que redirige a la vista reset. El botón docu que redirige a la vista docu.

- **Archivo**: al momento de redirigirse a esta vista se hace una petición POST con un archivo seleccionado en la vista home con el cual se realiza una copia del archivo cargado en la carpeta "media" del proyecto y con ello poder leer el contenido dentro de el para guardarlo en la bd, que en este caso es una lista, para su posterior envió. Esta vista devuelve la misma plantilla que home con el cambio de que se muestra el contenido del archivo cargado en una caja de texto.

- **Enviar**: al momento de redirigirse a esta vista se toma el valor almacenado en la lista bd y se hace una petición a la API de cargar archivo, luego se devuelve la misma vista que la vista archivo.

- **Consultar**: al momento de redirigirse a esta vista se llama a la API y se le hace la petición de generar salida con ello echo se recibe el archivo de salida en forma de texto y se envía como parámetro a la plantilla como también se guarda en la lista bd. Se devuelve la vista con el archivo de entrada y de salida mostrándose en cajas de texto.

- **Filtros**: al momento de redirigirse a esta vista se devuelve una vista con el cambio de que ahora se muestran dos campos de texto con un botón cada uno, desde estas cajas de texto se toma el parámetro que se le será enviado a la API como petición de generar error o bien generar fecha por lo que los botones indican que campo de texto es el correspondiente, con esto echo se le envía la información a la API y esta muestra en una ventana emergente las gráficas solicitadas.

- **Docu**: al momento de redirigirse a esta vista se abre en la misma ventana este archivo en formato pdf.

- **Reset**: al momento de redirigirse a esta vista se llama a la API y se le hace una petición de reset para borrar toda la información y a su ve esta vista limpia su lista bd, echo esto se redirige a la ventana home.

- **Ayuda**: únicamente al dar en este botón se llama a un pequeño script que muestra una alerta con los datos del desarrollador.

Requests: para la comunicación entre django y flask, front-end y back-end respectivamente, se utiliza la librería requests la cual consiste en una librería que realiza peticiones http en una línea de código recibiendo los parámetros de la url a la cual se le harán las peticiones y un parámetro data si este es un método POST, esta librería tiene la posibilidad de recibir respuestas en formato json como en formato de texto de una manera rápida y sencilla.

Matplotlib: esta librería es una librería externa a Python con la cual se pueden realizar diferentes graficas de diferentes tipo y con objetivos muchos más grandes como lo son el mundo de las estadísticas y de las matemáticas, para el proyecto únicamente se aplica su grafica de barras la cual solo necesita dos listas de igual tamaño una con leyendas y otras con valores numéricos para la construcción de la misma de una manera automática sin definir dimensiones ni ejes.

Conclusiones

- Se logra el uso y comunicación entre dos frameworks distintos.

- Se identifica un uso inadecuado de recursos al no trabajar únicamente con una herramienta.

- Se logra comprender el uso de la comunicación http para aplicaciones web.

- Se consigue el cometido de construir dos servicios diferentes que únicamente son unidos a través de protocolos http.

Referencias bibliográficas

Holovaty, A., & Kaplan-Moss, J. (2009). The definitive guide to Django: Web development done right. Apress.

He, J. S. K. T. G., & Naughton, C. Z. D. D. J. (2008). Relational databases for querying XML documents: Limitations and opportunities. In Proceedings of VLDB (pp. 302-314).

Grinberg, M. (2018). Flask web development: developing web applications with python. " O'Reilly Media, Inc.".

Pop, D. P., & Altar, A. (2014). Designing an MVC model for rapid web application development. Procedia Engineering, 69, 1172-1179.