PROYECTO 3

201902714 – KEVIN JOSUÉ CALDERÓN PERAZA

Resumen (150 palabras)

El elemento principal en el que se basan estos servicios son las URLs. En líneas generales podemos decir que estos servicios consisten en URLs a las que podemos acceder, por ejemplo, mediante protocolo HTTP, para obtener información o realizar alguna operación. El formato de la información que se intercambie con estas URLs ya se decidió como se iban a llamar. Este tipo de servicios acercan los Servicios Web al tipo de arquitectura de la web, con ella se pudo tener tomar en cuenta que existen patrones de arquitectura para el área del Frontend, para este caso es el MVC. Aplicando siempre los estándares del json y xml para el manejo y la persistencia de los datos.

Palabras clave

Protocolo HTTP, Servidor, Cliente, Peticiones, json

Abstract

The main element on which these services are based are the URLs. In general terms, we can say that these services consist of URLs that we can access, for example, through the HTTP protocol, to obtain information or perform an operation. The format of the information that is exchanged with these URLs has already been decided as they were going to be called. This type of service brings Web Services closer to the type of architecture of the web, with it it could be taken into account that there are architecture patterns for the Frontend area, for this case it is the MVC. Always applying the json and xml standards for the handling and persistence of the data.

Keywords

Protocole HTTP, Server, Client, requests, json.

Introducción

El desarrollo de servicios web REST es similar al desarrollo de aplicaciones web. Sin embargo, la diferencia fundamental entre el desarrollo de aplicaciones web tradicionales y las más modernas es cómo pensamos sobre las acciones a realizar sobre nuestras abstracciones de datos. De forma más específica, el desarrollo actual está centrado en el concepto de nombres (intercambio de recursos); el desarrollo tradicional está centrado en el concepto de verbos (acciones remotas a realizar sobre los datos). Con la primera forma, estamos implementando un servicio web RESTful; con la segunda un servicio similar a una llamada a procedimiento remoto- RPC). Y lo que es más, un servicio RESTful modifica el estado de los datos a través de la representación de los recursos (por el contrario, una llamada a un servicio RPC, oculta la representación de los datos y en su lugar envía comandos para modificar el estado de los datos en el lado del servidor).

Desarrollo del tema

En la actualidad podemos usar APIs para hacer multitud de sitios web. Por ejemplo, podríamos usar el API REST de Twitter para crear un servicio basada en esa red o el API de Youtube para crear un sitio web que muestra vídeos de distintas maneras.
Pero para aplicar estos conceptos, inicialmente se tomó en cuenta otras herramientas para la persistencia de los datos. Debido a que el XML proporcionado había que analizarlo detenidamente para obtener la información correctamente y necesaria.
XML es el nuevo estándar universal para intercambio electrónico de datos. Este estándar es un metalenguaje que puede ser utilizado para describir la estructura lógica y el contenido de una gran variedad documentos, y puede ser adaptado para satisfacer una gran cantidad de aplicaciones.

Estos atributos de ser universal y extensible abre un rango ilimitado de usos para el XML, desde procesadores de texto, páginas web, el comercio electrónico, hasta las más complejas soluciones de almacenamiento en bases de datos.

Debido a la forma en que la información se encuentra en el XML, para este caso no se utilizó las librerias que nos ofrecen Python, si no que, el documento XML se tuvo que leer como si fuera texto plano. Y por medio de expresiones regulares (REGEX) obtener la información deseada a las propiedades como usuario y reportados.

Una expresión regular, o expresión racional, también son conocidas como regex o regexp, por su contracción de las palabras inglesas regular expression, es una secuencia de caracteres que conforma un patrón de búsqueda. Se utilizan principalmente para la búsqueda de patrones de cadenas de caracteres u operaciones de sustituciones.

En el área de la programación, las expresiones regulares son un método por medio del cual se pueden realizar búsquedas dentro de cadenas de caracteres. Sin importar la amplitud de la búsqueda requerida de un patrón definido de caracteres, las expresiones regulares proporcionan una solución práctica al problema. Adicionalmente, un uso derivado de la búsqueda de patrones es la validación de un formato específico en una cadena de caracteres dada, como por ejemplo fechas o identificadores.

Para poder utilizar las expresiones regulares al programar es necesario tener acceso a un motor de búsqueda con la capacidad de utilizarlas. Es posible clasificar los motores disponibles en dos tipos según

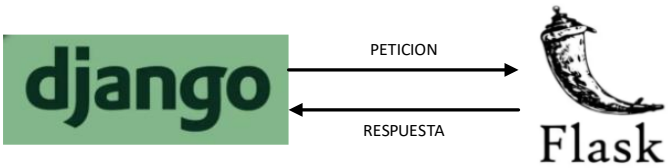
su uso: motores para el programador y motores para el usuario final.

Las expresiones regulares son patrones utilizados para encontrar una determinada combinación de caracteres dentro de una cadena de texto. Las expresiones regulares proporcionan una manera muy flexible de buscar o reconocer cadenas de texto.

Así como Django o angular etc., se comunican al servidor por medio de métodos HTTP, la representación de los recursos es lo que se envía entre los servidores y clientes. Una representación muestra el estado del dato real almacenado en algún dispositivo de almacenamiento en el momento de la petición. En términos generales, es un stream binario, juntamente con los metadatos que describen cómo dicho stream debe ser consumido por el cliente y/o servidor (los metadatos también pueden contener información extra sobre el recurso, como por ejemplo información de validación y encriptación, o código extra para ser ejecutado dinámicamente). A través del ciclo de vida de un servicio web, puede haber varios clientes solicitando recursos. Clientes diferentes son capaces de consumir diferentes representaciones del mismo recurso. Por lo tanto, una representación puede tener varias formas, como, por ejemplo, una imagen, un texto, un fichero XML, o un fichero JSON, pero tienen que estar disponibles en la misma URL.

Para en este caso, se utilizó django por el lado del frontend y es el que realiza las peticiones por medio del protocolo http y la vista. Para tener una vista amigable con el usuario se utilizó Bootstrap como framework de diseño. Para la comunicación con el servidor se realizó por medio de la librería request. Importándola de la siguiente manera y utilizando sus métodos get, y post. Transfiriendo la comunicación por medio de json.

“import request”
“import json”



Y para el backend se utilizó de igual manera el lenguaje Python con ayuda de la librería flask. Y las respectivas clases Evento y Afectado. Se utilizaron métodos http como POST y GET ya que necesitábamos recibir y enviar un archivo json que anteriormente fue procesado y evaluado con la información correcta. Igualmente, con la ayuda de la librería request, response y json, para mantener comunicado el servidor con el cliente. Ya con el archivo procesado se procedió a crear un archivo XML para tener de mejor forma presentado la información.

“from flask import Flask, request, response”
“import json”
“import xml.etree.cElementTree as ET”

También se tomó en cuenta el versionamiento en este proyecto. Se utilizó git para el control de versiones del proyecto y GitHub para tener almacenar en la nube el proyecto. Las releases de los proyectos se hicieron por medio de tags para tener un control de versión especifica dependiendo el tipo de cambio que se hizo.

Por ejemplo: v.X.Y.Z
La X marcará la versión mayor. Es una versión que incluye cambios importantes en el funcionamiento de la aplicación.

La Y marcará una versión media. Es una versión menor que añade funcionalidades de mantenimiento o actualización de funcionalidades que ya existían.

La Z marcará una versión menor. Es una versión que esta incluye corrección menor que no afectan el funcionamiento actual del sistema, pero tampoco es una actualización que mejoré el proyecto.

Conclusiones

1. El uso de archivos XML, simplifica y separa datos de manera que al compartir información e intercambiar datos, hace que los datos estén disponibles más fácilmente.

2. El transporte de información por medio de un archivo XML y JSON hace que el proyecto sea escalable.

3. Al ser sistemas independientes (solo se comunican con un lenguaje de intercambio como JSON) puedes desarrollarlos proyectos autónomos, equipos autónomos. Al cliente le da igual cómo está hecha la API y al servidor le da igual qué vas a hacer con los datos que te ha proporcionado.

4. A la hora de ejecutar la aplicación también tienes una flexibilidad mucho mayor. Las páginas del front pueden enviar desde unos servidores y las API pueden estar alojadas en servidores independientes, tantos como necesites.

5. Cuando haces una solicitud al servidor lo que tienes como respuesta son datos planos, que requieren tiempos de transferencia menores que si esos mismos datos los recibieras mezclados con el HTML/CSS de la presentación.

6. En este tipo de aplicaciones web no necesitas recargar la página, aunque esto no es una ventaja específica del desarrollo basado en REST, sino del uso de Ajax en general, con el que podemos conseguir aplicaciones web que se asemejan más a aplicaciones de escritorio

Anexos

Diagrama de conexión cliente/servidor

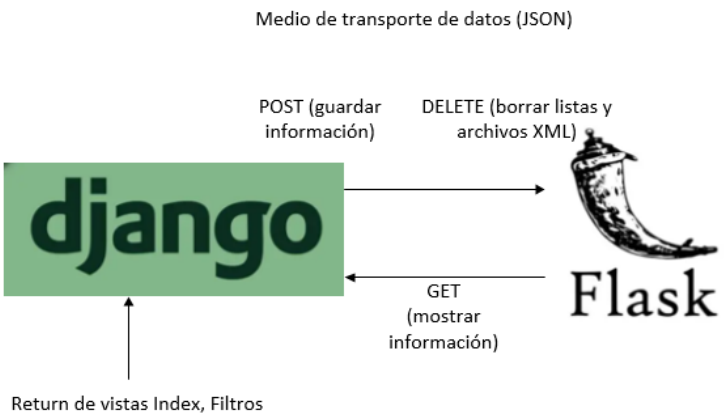
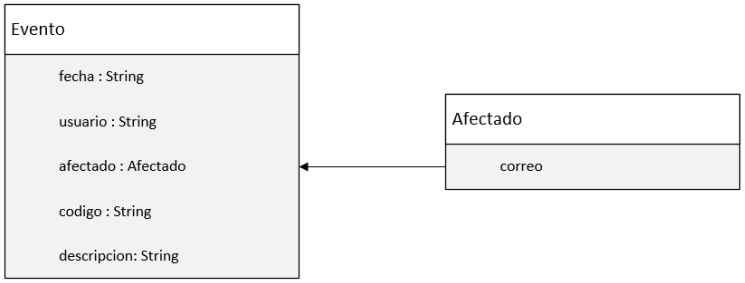


Diagrama de clases



Objetos fueron agregados a una listas de python temporalmente. Ya que todo fue almacenado en un XML como base de datos.

Referencias bibliográficas

1. Chakray. (2021 de 05 de 06). *Chakray*. Obtenido de Chakray: https://www.chakray.com/es/cuales-son-las-ventajas-de-una-api-rest/

2. *Grimpi IT Blog*. (22 de 11 de 2008). Recuperado el 07 de 05 de 2021, de https://grimpidev.wordpress.com/2008/11/22/patrones-de-acceso-a-datos-active-record/

3. Service, B. A. (2021 de Mayo de 06). *BbvaApiMarket*. Obtenido de https://www.bbvaapimarket.com/es/mundo-api/api-rest-que-es-y-cuales-son-sus-ventajas-en-el-desarrollo-de-proyectos/