Universidad Mariano Gálvez De Guatemala

**Imagen que contiene Logotipo

Descripción generada automáticamenteSede De Zacapa**

**Programación III**

**Ing. Vinicio Peña.**

**Análisis de API REST**

**Marco Antonio Salvatierra Salas**

**1190-22-10134**

**FECHA DE ENTREGA: 13/04/2024**

**Análisis de Servicio REST**

Los servicios REST (Representational State Transfer) son una forma común de comunicación entre aplicaciones a través de la web. Aquí hay algunos pasos que puedes seguir para analizar un servicio REST:

**Identificar el Servicio REST**:

Comienza por comprender qué servicio REST deseas analizar. ¿Tienes una URL específica o un punto final en mente?

**Documentación del Servicio**:

Busca la documentación oficial del servicio REST. Esto puede proporcionarte información sobre los puntos finales disponibles, los métodos HTTP admitidos (como GET, POST, PUT, DELETE), los parámetros requeridos y las respuestas esperadas.

**Exploración con Herramientas**:

Utiliza herramientas como **Postman**, **curl** o incluso un navegador web para realizar solicitudes al servicio REST.

Envía solicitudes GET para obtener datos y observa las respuestas. Asegúrate de comprender la estructura de los datos devueltos (generalmente en formato JSON o XML).

**Autenticación y Autorización**:

Verifica si el servicio requiere autenticación o autorización. Algunos servicios pueden usar tokens de acceso o claves API.

Asegúrate de incluir cualquier encabezado o parámetro necesario para autenticarte correctamente.

**Pruebas de Rendimiento**:

Realiza pruebas de rendimiento para evaluar la velocidad de respuesta del servicio. ¿Cuánto tiempo tarda en responder una solicitud?

Observa si hay cuellos de botella o retrasos significativos.

**Manejo de Errores**:

Intencionalmente envía solicitudes incorrectas (por ejemplo, con parámetros faltantes o valores no válidos) para ver cómo maneja el servicio los errores.

Verifica si el servicio proporciona códigos de estado HTTP apropiados (como 200 OK, 404 Not Found, 500 Internal Server Error).

**Seguridad**:

Evalúa la seguridad del servicio. ¿Utiliza HTTPS? ¿Hay vulnerabilidades conocidas?

Considera la posibilidad de realizar pruebas de seguridad, como pruebas de penetración.

**Monitoreo y Registro**:

Investiga si el servicio proporciona registros o métricas para el monitoreo.

¿Cómo se registran las solicitudes y las respuestas? ¿Hay umbrales de tiempo o límites de uso?

**Escalabilidad**:

Piensa en la escalabilidad del servicio. ¿Puede manejar una gran cantidad de solicitudes simultáneas?

Investiga si hay límites de uso o cuotas.

**Documenta tus Hallazgos**:

Escribe un informe que incluya tus observaciones, recomendaciones y cualquier problema identificado.

**Qué significa REST**

REST es una interfaz para conectar varios sistemas basados en el protocolo HTTP (uno de los protocolos más antiguos) y nos sirve para obtener y generar datos y operaciones, devolviendo esos datos en formatos muy específicos, como XML y JSON.

El formato más usado en la actualidad es el formato JSON, ya que es más ligero y legible en comparación al formato XML. Elegir uno será cuestión de la lógica y necesidades de cada proyecto.

REST se apoya en HTTP, los verbos que utiliza son exactamente los mismos, con ellos se puede hacer GET, POST, PUT y DELETE. De aquí surge una alternativa a SOAP.

Cuando hablamos de SOAP hablamos de una arquitectura divididas por niveles que se utilizaba para hacer un servicio, es más complejo de montar como de gestionar y solo trabajaba con XML.

Ahora bien, REST llega a solucionar esa complejidad que añadía SOAP, haciendo mucho más fácil el desarrollo de una API REST, en este caso de un servicio en el cual nosotros vamos a almacenar nuestra lógica de negocio y vamos servir los datos con una serie de recursos URL y una serie de datos que nosotros los limitaremos, es decir, será nuestro BACKEND nuestra lógica pura de negocios que nosotros vamos a utilizar.

## Por qué debemos utilizar REST

REST no es solo una moda, y es por las siguientes razones que esta interfaz está teniendo tanto protagonismo en los últimos años:

* Crea una petición HTTP que contiene toda la información necesaria, es decir, **un REQUEST a un servidor tiene toda la información necesaria y solo espera una RESPONSE**, ósea una respuesta en concreto.
* **Se apoya sobre un protocolo que es el que se utiliza para las páginas web, que es HTTP**, es un protocolo que existe hace muchos años y que ya está consolidado, no se tiene que inventar ni realizar cosas nuevas.
* Se apoya en los métodos básicos de HTTP, como son:
  + **Post**: Para crear recursos nuevos.
  + **Get**: Para obtener un listado o un recurso en concreto.
  + **Put**: Para modificar.
  + **Patch**: Para modificar un recurso que no es un recurso de un dato, por ejemplo.
  + **Delete**: Para borrar un recurso, un dato por ejemplo de nuestra base de datos.

Todos**los objetos se manipulan mediante URI, por ejemplo**, si tenemos un recurso usuario y queremos acceder a un usuario en concreto nuestra URI seria /user/identificadordelobjeto, con eso ya tendríamos un servicio USER preparado para obtener la información de un usuario, dado un ID.