**¿QUÉ ES REST?**

REST forma parte de una arquitectura de software que nos permite construir APIs para generar entre el servidor y sus clientes.

"Representational State Transfer" o traducido a "Transferencia de presentación de estado" es lo que se domina a REST. ¿**Y eso es**?, una técnica de arquitectura de software usada para construir APIs que permitan comunicar a nuestro servidor con sus clientes usando el protocolo HTTP mediante URIs lo suficientemente inteligentes para poder satisfacer la necesidad del cliente.

* REST es **STATELESS**, es decir que cada petición que reciba nuestra API **debe perecer**. Por ejemplo, **no podemos RECORDAR** un usuario logeado en el API usando una sesión, esto es un PECADO ya que agotaría la memoria RAM de nuestro servidor (10 mil usuarios conectados a nuestra API). Lo que correcto es pasar un TOKEN para cada petición realizada al API, y el API deberá validar si esta es correcta o no (por ahora no vamos hablar de técnicas para generar el TOKEN, pero lo más común es usar una COOKIE).
* Se implementan**RECURSOS**para generar comunicación, es decir crea URIs únicas que permiten al cliente entender y utilizar lo que está exponiendo. Por ejemplo:
  + <https://rickandmortyapi.com/api/character>
  + [rickandmortyapi.com/api/character/32](https://rickandmortyapi.com/api/character/32)

Cada petición realizada a nuestra API responde a un verbo, y dicho verbo a una operación en común. Mediante los métodos HTTP hacemos las peticiones, lo común es GET y POST, PUT y DELETE.

* + **POST (create):**cuando mandamos información para insertar por ejemplo un registro en la base de datos. La información es enviado en el cuerpo de la petición, es decir que los datos no son visibles al usuario.
  + **GET (read)**: es usado para modo lectura, por ejemplo: cuando queremos listar a todos los usuarios de nuestra base de datos. Los parámetros son enviados por la URL.
  + **PUT (update):**cuando queremos actualizar un registro. Actualizar la información de un usuario X.
  + **DELETE (delete):**cuando queremos eliminar un registro. Borrar un usuario X de nuestra base de datos.

Con esto hemos mencionado algunas características básicas de lo que es REST, la cual podríamos decir que es un estándar para crear una REST Api o RESTFul.

**¿QUÉ ES RESTFUL?**

Es un servicio que disponemos al público usando REST. REST es el concepto, RESTFul es la implementación y al crear un RESTFul creamos una API, la cual una API es un conjunto de funciones o procedimientos para que sea utilizado por otro software.

**Ejm:** la Api de Google Maps, Youtube, Facebook, etc ..

**¿BAJO QUÉ CIRTUSTANCIAS DEBERÍAMOS IMPLEMENTAR UN SERVICIO RESTFUL?**

Se dice que hoy en día debemos pensar orientado al servicio. Yo recomendaría trabajar directamente con una REST Api bajo estas situaciones.

* Si vamos a trabajar una aplicación del tipo SPA, si o si debemos crear una REST Api. Un framework recomendado puede ser AngularJS, EmberJS, backbone entre otros.
* Si es una Web comercial y vas a exponer recursos al público, es muy recomendable crear una REST Api.
* Si tu sistema no solo va a ser accedido desde una PC, también deberíamos pensar en crear una REST Api, ya que permite crear escalabilidad.

En resumen, crear una REST Api es una manera muy profesional de desacoplar tu sistema de la capa de persistencia.

**PUNTOS A FAVOR :)**

* **Separación entre el cliente y el servidor**  
  Nuestros proyectos se vuelven autónomos, por lo tanto, no interesa si dicha tecnología es compatible con la otra ya que usaremos como un medio de comunicación JSON.
* **No importa la tecnología**  
  Si eres PHP, .NET, Java, Ruby, etc da igual, al final solo necesitas saber cómo consumir/responder al servicio.
* **Escalabilidad, Flexibilidad**  
  Realiza los cambios que quieras dentro de tu Api, lo que interesa es que se respete el mismo mensaje o respuesta que le brindas al cliente para mantener la misma lógica.
* **¿Mejora de recursos consumidos por el servidor?**
  + REST no debe usar sesiones, por lo tanto, disponemos de más memoria RAM.
  + Lo correcto es trabajar con formatos estandarizados como JSON no haremos uso de HTML para responder al cliente. En este caso ganamos velocidad.

**PUNTOS EN CONTRA :(**

* Mayor tiempo en desarrollo debido que hay que plantear y estandarizar las respuestas de nuestra API para que se tornen amigables para quien la consume.
* Curva de aprendizaje incrementada, ya que como muchos están acostumbrados a trabajar enviado HTML por las peticiones AJAX, podría resultar un poco tedioso trabajar con JSON puro, entre ellas, entender el uso del framework que implemente la API o tener claro en el concepto si queremos desarrollar un framework desde cero.

En las siguientes publicaciones vamos a implementar un REST Api para PHP y .NET usando algún framework.

**MATERIAL ADICIONAL**

**CÓDIGOS DE ESTADO HTTP**

Cada petición realizada esta acompañada por un estado, este estado es representado por un entero.

* 401: acceso no autorizado
* 404: página no encontrada
* 500: error de servidor