**Rest - Conceptos avanzados**

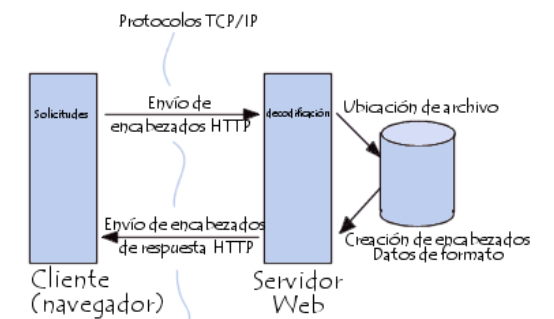
Los sistemas que siguen los principios REST se denominan a menudo como RESTful.

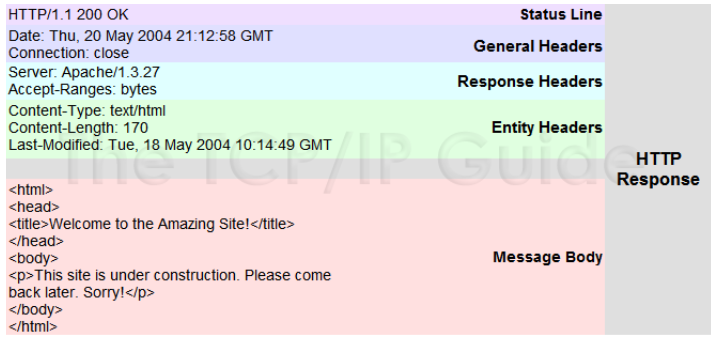
Los principios fundamentales de REST son:

* Un protocolo cliente/servidor [sin estado](https://www.google.com/url?q=http://es.wikipedia.org/wiki/Protocolo_sin_estado&sa=D&ust=1503593632209000&usg=AFQjCNHiJVWdu4phTjEsVKDVqK_l2xTPJw" \t "_blank) (Stateless): cada mensaje HTTP contiene toda la información necesaria para comprender la petición. Como resultado, ni el cliente ni el servidor necesitan recordar ningún estado de las comunicaciones entre mensajes.
* Un conjunto de operaciones bien definidas que se aplican a todos los recursos de información: HTTP define un conjunto de operaciones, las más importantes son POST,GET, PUT y DELETE. Con frecuencia estas operaciones se equiparan a las operaciones [CRUD](https://www.google.com/url?q=http://es.wikipedia.org/wiki/CRUD&sa=D&ust=1503593632210000&usg=AFQjCNFQwNcZSGyZu-676Z9a4HIMKjZg1Q) en bases de datos (ABMC en castellano: Alta, Baja, Modificación y Consulta) que se requieren para la persistencia de datos. Los métodos GET y POST son conocidos por los desarrolladores por los formularios web (<form method="get"/>). Sin embargo, Rest utiliza más: Get, Post, Delete y Put. Cada uno tiene una tarea específica:

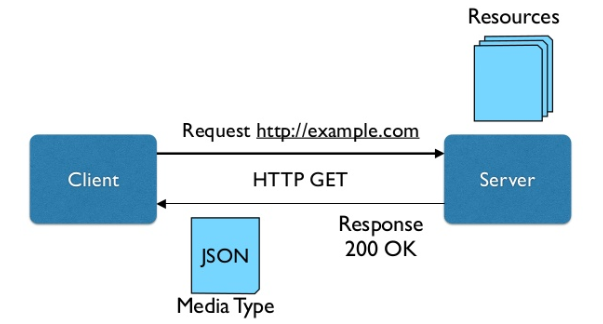
1. GET: Para obtener un valor. Puede ser un listado de objetos
2. POST: Para guardar un nuevo objeto en la aplicación
3. DELETE: Para eliminar un objeto
4. PUT: Para actualizar un objeto.

* Una sintaxis universal para identificar los recursos. En un sistema REST, cada recurso es direccionable únicamente a través de su URI.





* El uso de hipermedios, tanto para la información de la aplicación como para las transiciones de estado de la aplicación: la representación de este estado en un sistema REST son típicamente [HTML](https://www.google.com/url?q=http://es.wikipedia.org/wiki/HTML&sa=D&ust=1503593632213000&usg=AFQjCNEbNliCRhzwPhkOVS5H5mi422b9Yw) o [XML](https://www.google.com/url?q=http://es.wikipedia.org/wiki/XML&sa=D&ust=1503593632214000&usg=AFQjCNHijWG7NT3GLsi_hAkKmwVLWVHDFA). Como resultado de esto, es posible navegar de un recurso REST a muchos otros, simplemente siguiendo enlaces sin requerir el uso de registros u otra infraestructura adicional.



**Jersey Java Client**

La implementación Jersey  de Jax-RS  nos brinda una API de clases para generar clientes de un web service REST desde clases Java, a continuación se observan ejemplos básico del uso de esta API, la documentación completa de la API se puede obtener en:

[https://jersey.java.net/documentation/latest/client.html](https://www.google.com/url?q=https://jersey.java.net/documentation/latest/client.html&sa=D&ust=1503593632215000&usg=AFQjCNHlD9D8Auh2fPbiTpCZfH8GgYzh0Q" \t "_blank)

En conjunto con las clases generadas por la API de Jersey se puede utilizar la herramienta XJC como vimos en las clases anteriores para generar las clases de negocio a partir del XSD que se observa en el WADL.