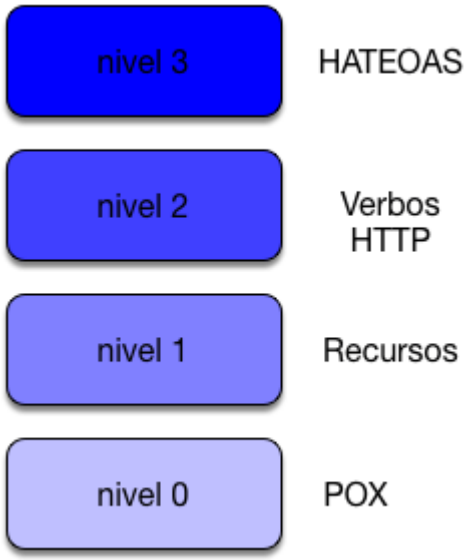
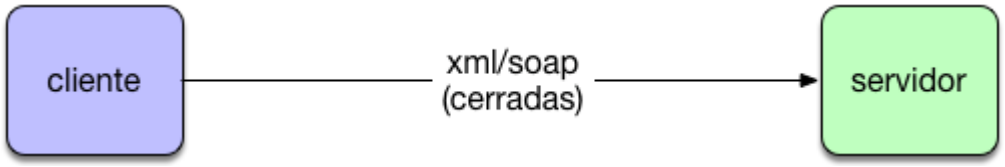
El uso de Arquitecturas REST es a día de hoy muy común. Bien es cierto que poco a poco otras soluciones se abren camino como es el caso de GraphQL . Sin embargo en muchas ocasiones nos encontraremos a día de hoy construyendo soluciones basadas en arquitecturas REST. Estas arquitecturas soportan varios niveles:



Arquitecturas REST ,Nivel 0 (Swamp of POX)

Este nivel hace referencia a cuando se construye una arquitectura basada en XML (POX= Plain Old XML) . En este tipo de arquitecturas disponemos de URLS de acceso que utilizando el método POST de HTTP intercambian mensajes en formato XML entre cliente y servidor un ejemplo clásico puede ser SOAP.

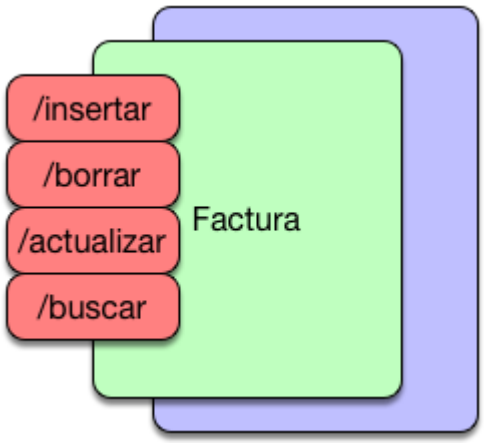


Estas arquitecturas en su momento se usaron mucho y tienen su utilidad .Sin embargo tienen algunos problemas . Uno de ellos es su complejidad ya que el protocolo SOAP genera mensajes complejos. Por otro lado hay que destacar que son fuertemente cerradas es decir

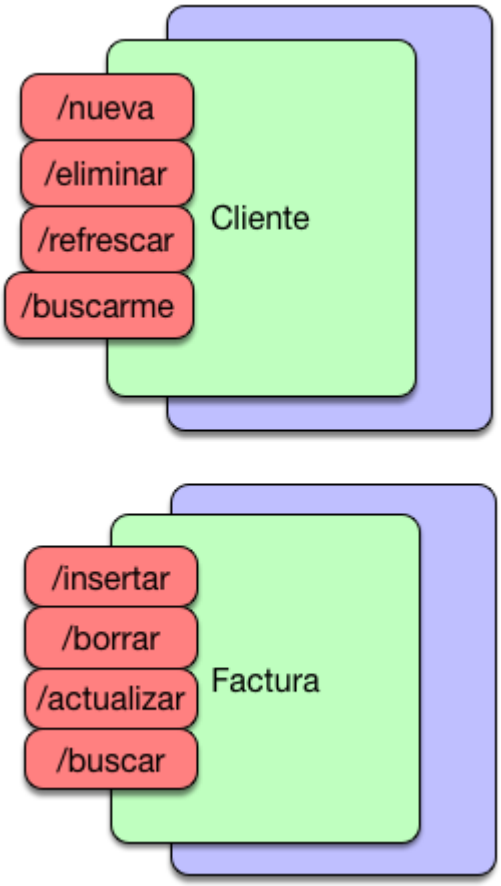
hay que conocer la URL de destino y de una forma muy clara como construir el cliente que se conecta o proxy ya que los mensajes son complejos para ello nos ayudan los ficheros WSDL.

Nivel 1 (Recursos)

En este nivel las cosas cambian y aparece el concepto de Recurso que ayuda a simplificar la forma que tenemos de acceder a los servicios. Un Recurso no es ni más ni menos que un concepto que las arquitecturas REST publican . Por ejemplo Facturas , Albaranes, Clientes etc. La arquitectura REST al publicar este concepto permite , inserción ,borrado , actualizaciones y búsquedas sobre ellos. En este nivel se utilizan peticiones HTTP Sencillas y es muy habitual utilizar JSON.



El problema que existe en este nivel es que cada URL puede tener un nombre arbitrario generando apis bastante caóticas si que es cierto que cada recurso tiene su grupo de URLs pero no existe una convención concreta.

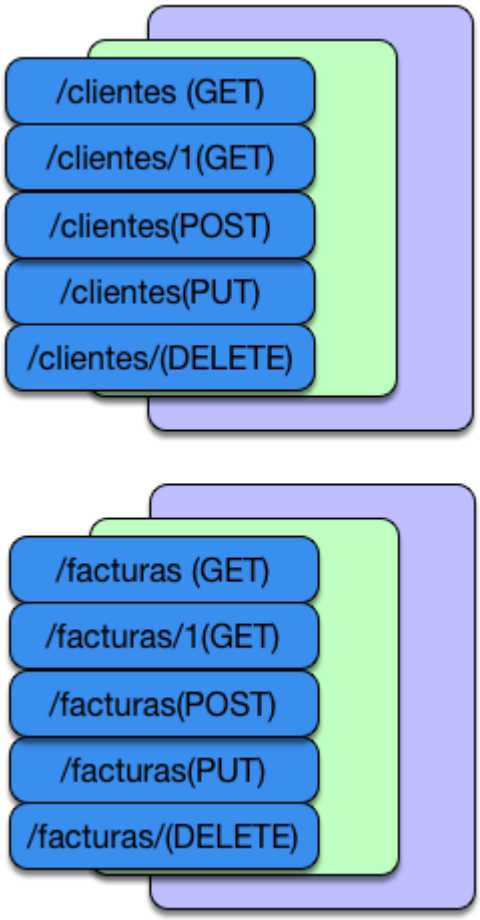


Nivel 2 (Verbos Http)

El nivel 2 aporta una convención concreta a nivel de nombre de URLs y los verbos HTTP que se usan para cada operación. La mayor parte de las operaciones se realizarán contra la URL general del recurso /facturas o /clientes. Los verbos fundamentales son :

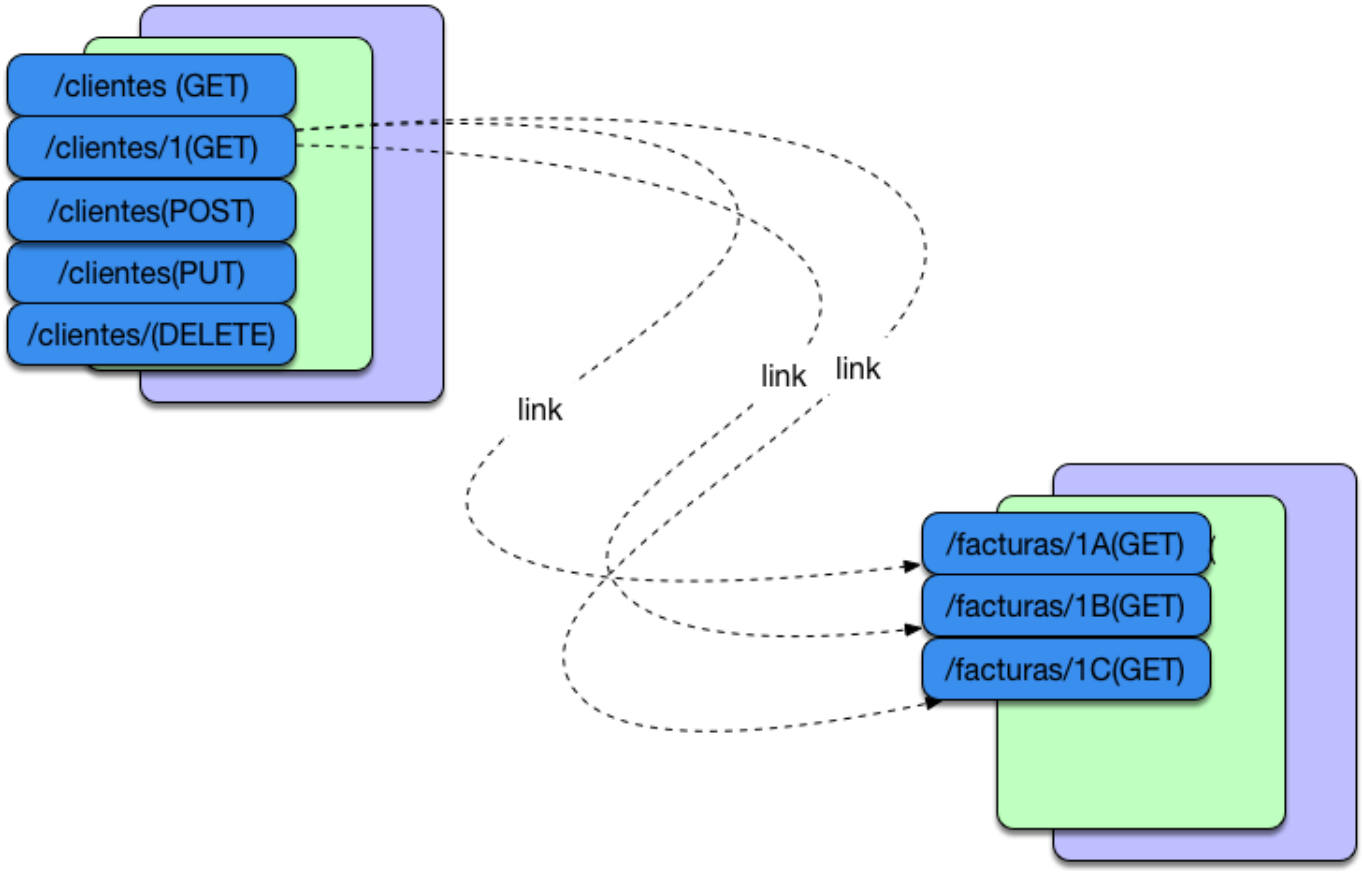
1. GET: Obtiene los datos. /facturas obtendrá la lista completa de Facturas (Recursos) , /facturas/1 obtendrá una factura en concreto.
2. POST: Cuando usemos POST insertaremos una nueva Factura en el sistema
3. PUT: Cuando utilicemos PUT actualizaremos una Factura
4. DELETE: Cuando utilicemos DELETE borraremos una Factura

Así pues la construcción de URLs sería:

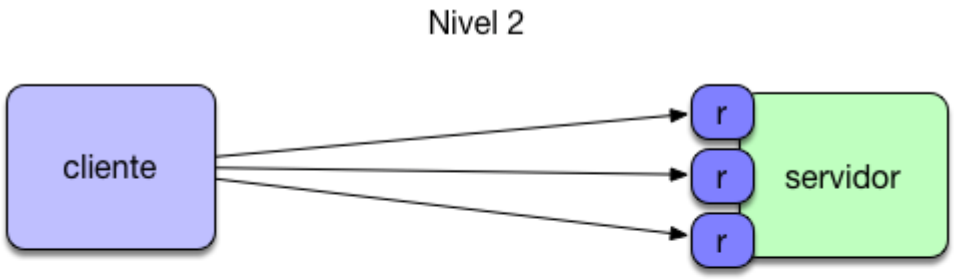


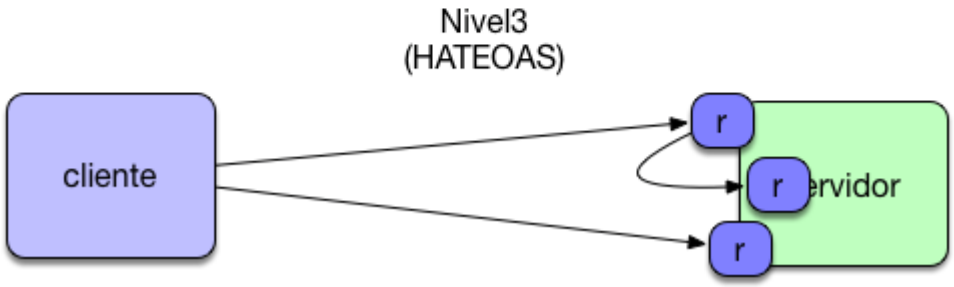
Nivel 3 (HATEOAS)

El nivel 3 de HATEOAS (Hypermedia as the Engine of Application State) . Es el último a nivel de arquitecturas REST. En esta caso los Recursos pueden estar relacionados entre ellos a través del uso de links . De tal forma que cuando solicitamos la lista de Clientes esta lista puede incluir links a las facturas de cada uno de ellos.



Esto ayuda a reducir el acoplamiento entre cliente y servidor ya que el cliente queda enlazado de forma directa a un menor conjunto de recursos.





Vamos a ver un ejemplo en código que nos ayude a clarificar lo que implica el uso de HATEOAS:

```
{
“dni”:”123456778A”,
“nombre”:”pedro”
"links": [
{  "línea": "http://dominio/facturas/1A/lineas/1"}  ,
{  "línea": "http://dominio/facturas/1A/lineas/2"}  ,
{  "línea": "http://dominio/facturas/1A/lineas/3"}  ,
  ]
}
```

Acabamos de ver un pequeño resumen de los diferentes niveles de las Arquitecturas REST.

1. PostMan App con Spring REST
2. ¿ Que es REST ?
3. Introducción a Servicios REST
4. Arquitecturas MVC y REST
5. Rest WikiPedia

Cecilio Álvarez Caules
Cecilio Álvarez Caules Oracle Java Certified Architect