

## Blog sobre Java EE

Estás aquí: [Inicio](#)/[Java EE](#)/[REST](#)/Arquitecturas REST y sus niveles

# Arquitecturas REST y sus niveles

22 marzo, 2018 por [Cecilio Álvarez Caules](#) — [Deja un comentario](#)

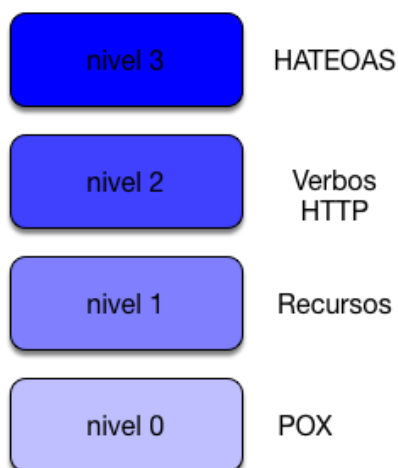


Are your binaries fine wine or expired milk?

SCAN & SEE



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.

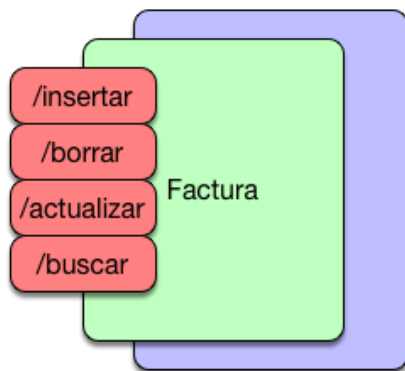
Este sitio web utiliza cookies para que usted tenga la mejor experiencia de usuario. Si continúa navegando está dando su consentimiento para la aceptación de las mencionadas cookies y la aceptación de nuestra [política de cookies](#), pinche el enlace para mayor información.

[plugin cookies](#)[ACEPTAR](#)

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.

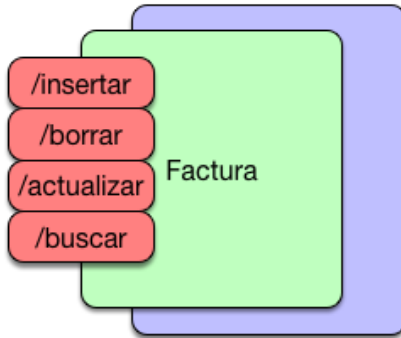
## Nivel 1 (Recursos)

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.**

Este sitio web utiliza cookies para que usted tenga la mejor experiencia de usuario. Si continúa navegando está dando su consentimiento para la aceptación de las mencionadas cookies y la aceptación de nuestra [política de cookies](#), pinche el enlace para mayor información.

[plugin cookies](#)[ACEPTAR](#)

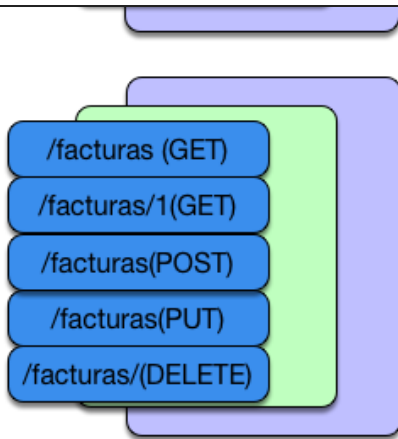
## Nivel 2 (Verbos Http)

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:

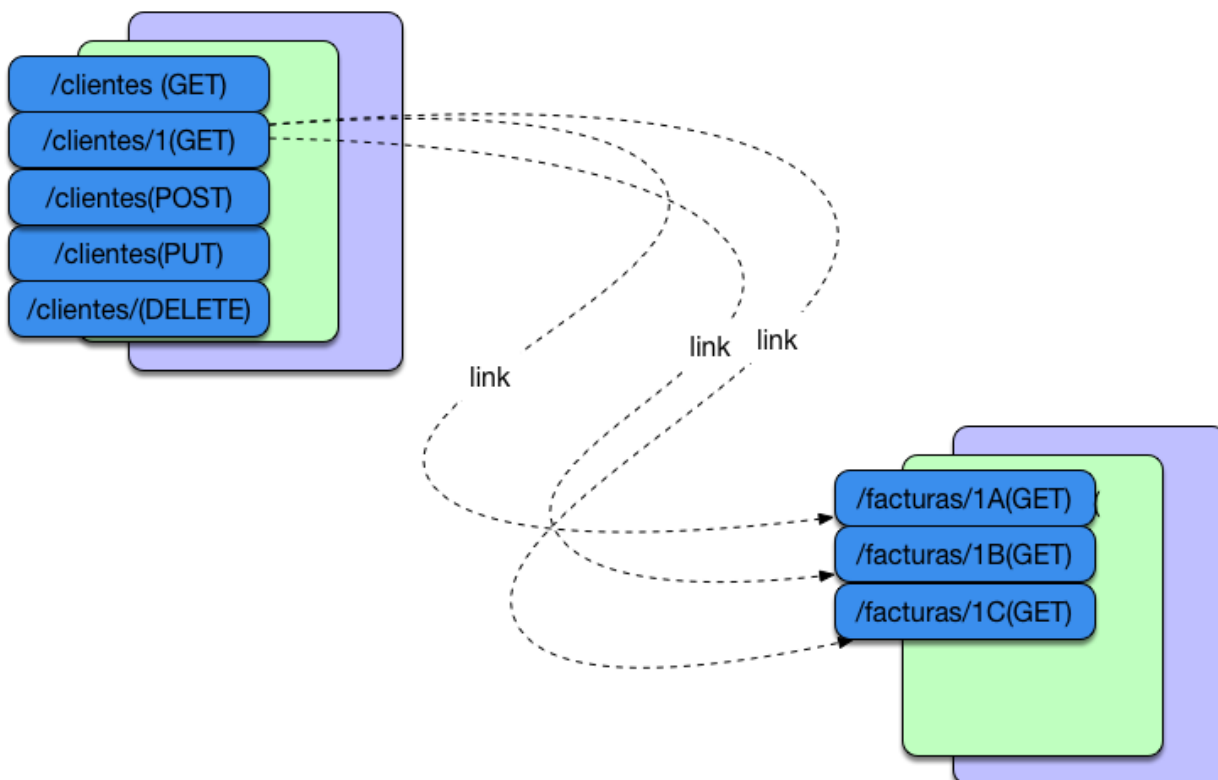
Este sitio web utiliza cookies para que usted tenga la mejor experiencia de usuario. Si continúa navegando está dando su consentimiento para la aceptación de las mencionadas cookies y la aceptación de nuestra [política de cookies](#), pinche el enlace para mayor información.

[plugin cookies](#)[ACEPTAR](#)

## Nivel 3 (HATEOAS)

[JAVA SE](#) [SPRING](#) [JAVA EE](#) [JAVASCRIPT](#) [FRAMEWORKS JS](#) [ARQUITECTURA](#) [MIS LIBROS](#) [MIS CURSOS](#)

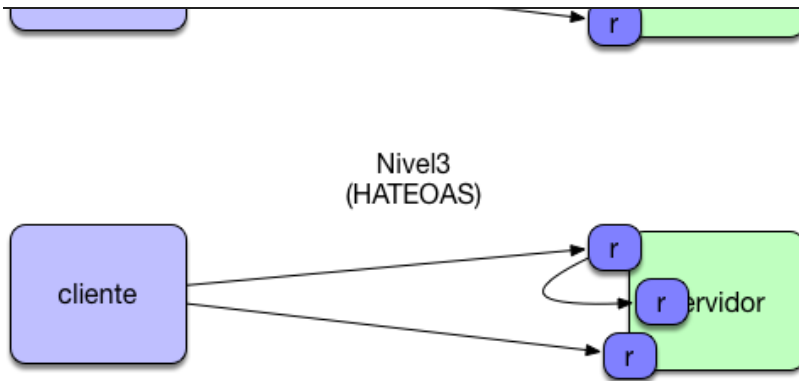
El nivel 3 de HATEOAS ( Hypermedia as the Engine of Application State) . Es el último a nivel de arquitecturas REST. En esta caso los Recursos pueden estar relacionados **entre ellos a través del uso de links** . De tal forma que cuando solicitamos la lista de Clientes esta lista puede incluir links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



Este sitio web utiliza cookies para que usted tenga la mejor experiencia de usuario. Si continúa navegando está dando su consentimiento para la aceptación de las mencionadas cookies y la aceptación de nuestra [política de cookies](#), pinche el enlace para mayor información.

[plugin cookies](#)

ACEPTAR



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

```

1  {
2  "dni":"123456778A",
3  "nombre":"pedro"
4  "links": [
5  { "línea": "http://dominio/facturas/1A/lineas/1"} ,
6  { "línea": "http://dominio/facturas/1A/lineas/2"} ,
7  { "línea": "http://dominio/facturas/1A/lineas/3"} ,
8  ]
9  }
```

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](#)



1

1
COMPARTI

JVM DEDICADA

**HOSTING JAVA APPS**

Tomcat, Glassfish/Payara y WildFly desde sólo 79€/Año

MÁS INFO

www.anw.es

Archivada en: [REST](#)

Start the discussion

Este sitio web utiliza cookies para que usted tenga la mejor experiencia de usuario. Si continúa navegando está dando su consentimiento para la aceptación de las mencionadas cookies y la aceptación de nuestra [política de cookies](#), pinche el enlace para mayor información.

[plugin cookies](#)

ACEPTAR

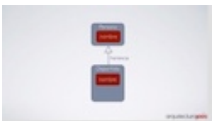
 [Subscribe](#) ▼

BUSCAR

Buscar en este sitio ...

Mis Cursos de Java Gratuitos

Java Herencia



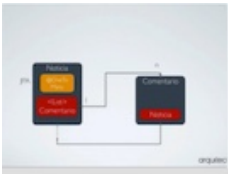
Java JDBC



Servlets



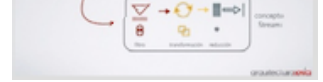
Introduccion JPA



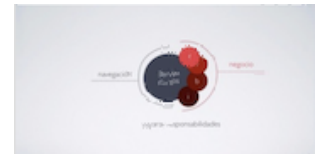
Mis Cursos de Java

Programación Orientada a Objeto en Java

Este sitio web utiliza cookies para que usted tenga la mejor experiencia de usuario. Si continúa navegando está dando su consentimiento para la aceptación de las mencionadas cookies y la aceptación de nuestra [política de cookies](#), pinche el enlace para mayor información.

[plugin cookies](#)[ACEPTAR](#)

## Java Web



## Pack Java Core



## Arquitectura Java Solida con Spring



## POPULAR

[Maven Parent POM y uso de librerías](#)[Spring REST Client con RestTemplates](#)[Java Generic Repository y JPA](#)[Nuevo Curso:Arquitectura Java Sólida con Spring 4.3 y Anotaciones](#)[Java Interfaces y el concepto de simplicidad](#)[PostMan App con Spring REST](#)[Java 9 Modules y el concepto de modularidad](#)[Angular 5 Hello World y su funcionamiento](#)[Spring Boot Properties utilizando @Value](#)

---

Este sitio web utiliza cookies para que usted tenga la mejor experiencia de usuario. Si continúa navegando está dando su consentimiento para la aceptación de las mencionadas cookies y la aceptación de nuestra [política de cookies](#), pinche el enlace para mayor información.

[plugin cookies](#)[ACEPTAR](#)

---

[¿Qué es Spring Boot?](#)[Arquitecturas REST y sus niveles](#)[Java Constructores this\(\) y super\(\)](#)[Usando Java Session en aplicaciones web](#)[Java Iterator vs ForEach](#)[¿Cuales son las certificaciones Java?](#)[Introducción a Servicios REST](#)[¿Qué es Gradle?](#)[Java Override y encapsulación](#)[Ejemplo de Java Singleton \(Patrones y ClassLoaders\)](#)[REST JSON y Java](#)[Ejemplo de JPA , Introducción \(I\)](#)[Usando el patron factory](#)[Uso de Java Generics \(I\)](#)[Mis Libros](#)[¿Qué es un Microservicio?](#)[Nuevo Curso:Arquitectura Java Sólida con Spring 4.3 y Anotaciones](#)[Comparando java == vs equals](#)[Spring Web\\_INITIALIZER ,eliminando el web.xml](#)[Spring MVC Configuración \(I\)](#)

---

---