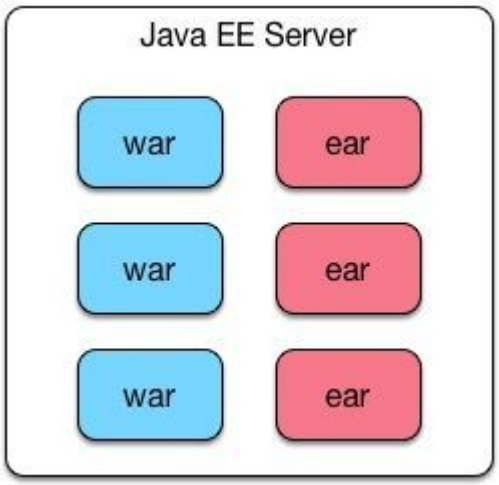
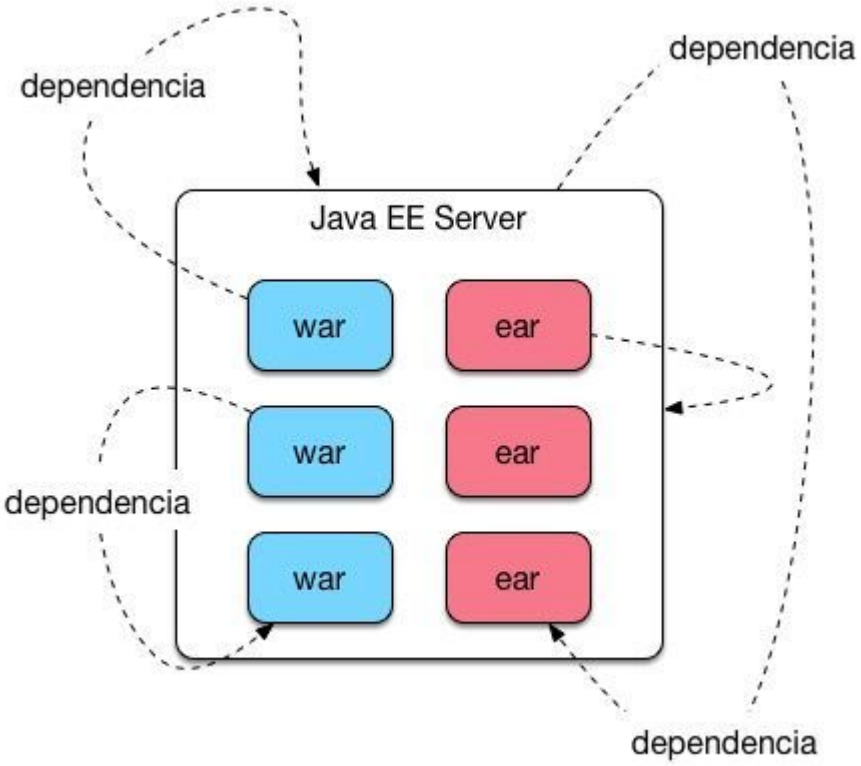
MicroServicios , una de las palabras más de moda en estos días. Todo el mundo quiere entender como funcionan estas nuevas arquitecturas. Hasta hace no mucho tiempo la mayor parte de los desarrollos se han hecho usando métodos muy standard. En el caso de Java desplegando aplicaciones a través de EARs y WARs. Cada una de estas aplicaciones era independiente. Podemos usar nuestro servidor y borrar y volver a desplegar la aplicación.



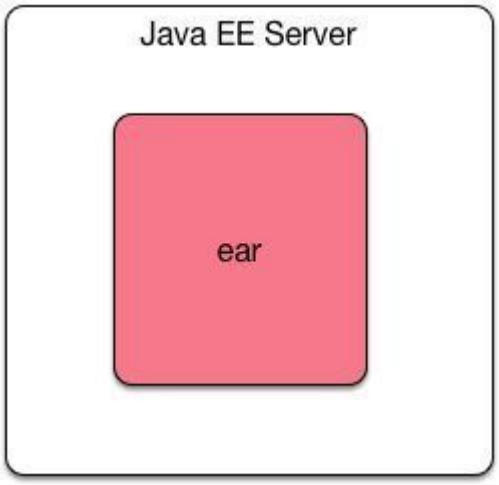
¿Son realmente independientes las aplicaciones?

La teoría nos dice que sí, que cada aplicación se ejecuta de forma independiente y es autónoma. La realidad es un poco más dura. Imaginemos que tenemos que desplegar una nueva aplicación y queremos compartir los jars para otras futuras aplicaciones. Tendremos que parar nuestro servidor y volverlo a lanzar para usar un shared lib, así que todas las aplicaciones pararán. Si una aplicación genera PDFs muy pesados quizás tengamos que cambiar los parámetros de la JVM y dar más memoria y volver a arrancar el servidor. Puede que nuestro servidor necesite mas hilos de ejecución porque una aplicación lo demande. Puede que la nueva aplicación que despleguemos se coma la memoria o la CPU de todas . Puede … pueden ocurrir muchas cosas. Así que nuestras aplicaciones no son tan independientes como parecen al estar albergadas en el mismo servidor.



Servidores y Aplicaciones

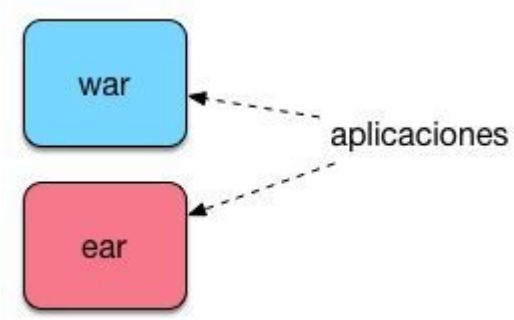
Cuando nuestras aplicaciones son sencillas y de poca importancia solemos tener varias en el mismo servidor de aplicaciones. Pero cuando las cosas se ponen serias y tenemos clientes con un alto nivel de exigencia las cosas cambian . No es fácil decir a alguien que hemos reiniciado un servidor por la aplicación de al lado y no la suya. Así pues en más de una ocasión me he encontrado en arquitecturas que tienen una aplicación grande por servidor y punto.



Es algo que al principio sorprende, pero con el paso de los años uno entiende que los administradores de sistemas tienen muchas, muchas tareas que hacer y necesitan reducir los riesgos y sobre todo aislarlos. Esa es una de lass claves , aislar las aplicaciones y hacer que cada aplicación sea completamente independiente y no influya en las otras. Es algo que suena sencillo , pero que no lo es. Tendremos muchas cosas que cambiar y habrá que ir con calma.

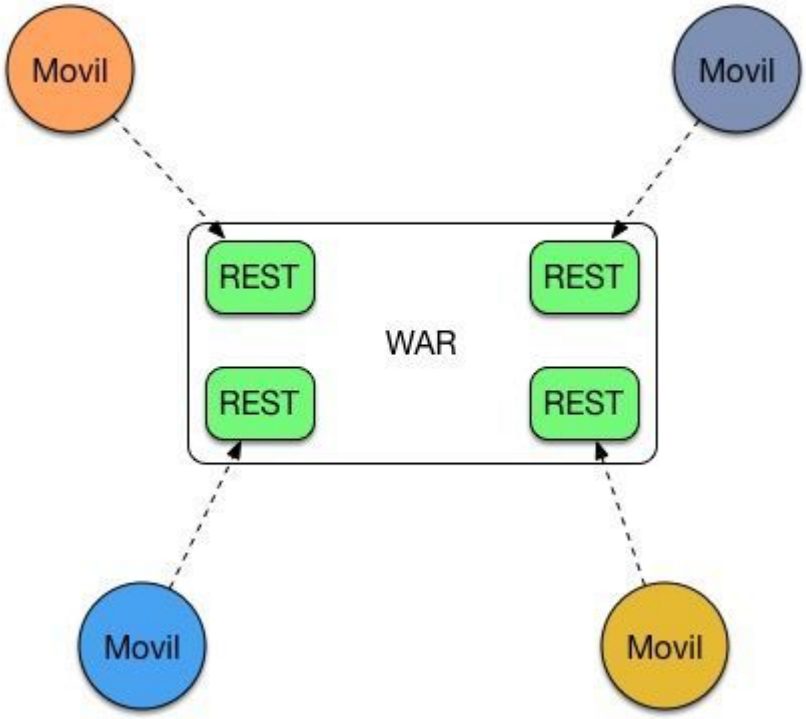
¿Qué es una aplicación?

Esta es una pregunta interesante, simplificando mucho podríamos decir que para un desarrollador Java es un WAR o un EAR que se despliega y realiza una funcionalidad X . Para el usuario simplemente serán las pantallas que realizan esa funcionalidad X , no le es necesario conocer el concepto de WAR.



Aplicaciones y Movilidad

¿Realmente una aplicación es un WAR ?. Vamos a suponer que desplegamos nuestra aplicación WAR y como estamos en un desarrollo moderno dispone de varios servicios REST a los cuales ligamos aplicaciones móviles.

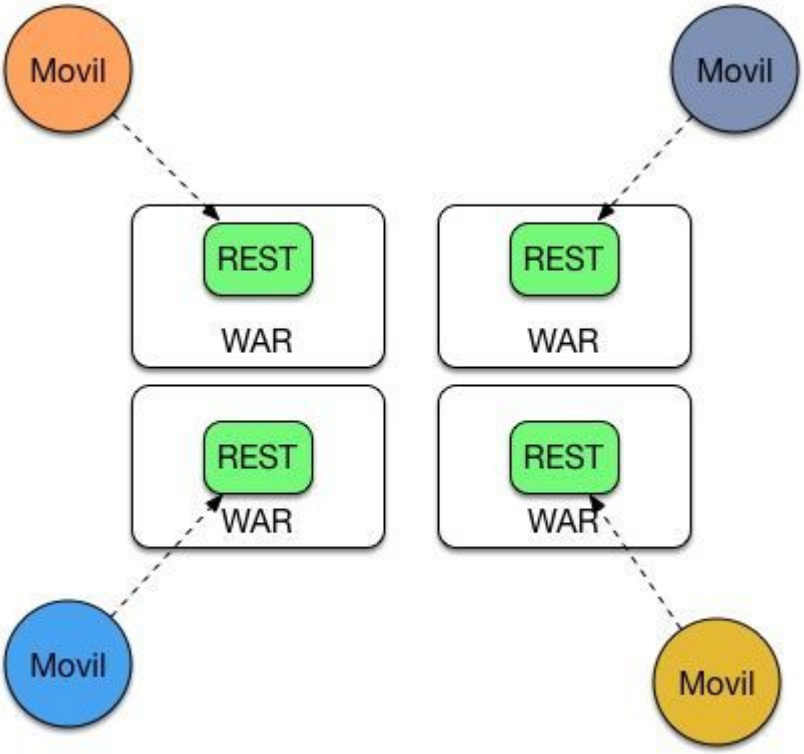


¿Cuantas aplicaciones tenemos? Bueno tenemos 5 aplicaciones 4 aplicaciones móviles y el WAR. ¿Son 5 de verdad?. Si preguntamos a los usuarios estos nos dirán que tenemos 4 aplicaciones móviles. Es un punto sobre el cual reflexionar porque para un usuario el war no existe. Las cuatro aplicaciones serán diferentes y no aceptará de muy buen grado que paremos la aplicación A porque hay que reinstalar la B ya que tenemos que redesplegar el WAR. Eso no es fácil de asumir .

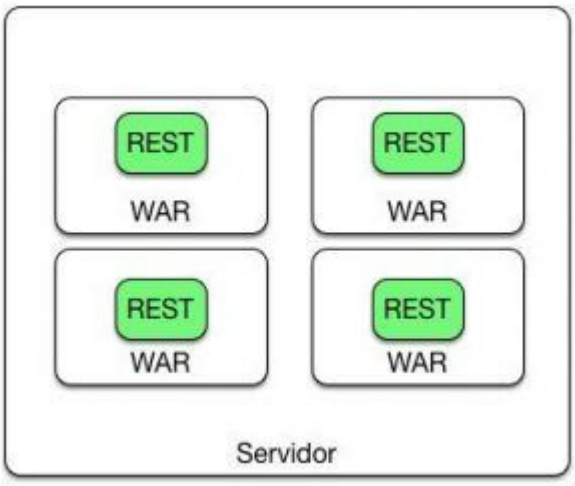
¿Qué es realmente una aplicación?. Una aplicación es un grupo de funcionalidad que genera un flujo de trabajo útil para el usuario y que se puede desplegar de forma aislada. Quizá no es la mejor definición pero cada día pienso que el concepto de “aislada” es CLAVE.

Aplicaciones y MicroServicios

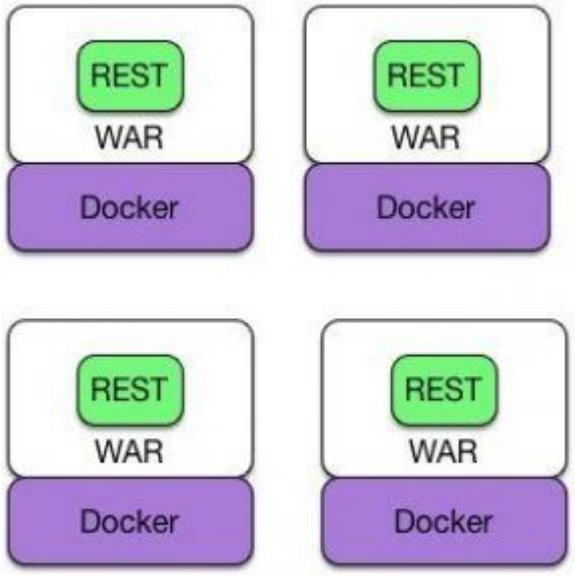
Si queremos solventar el problema anterior y contentar a nuestros usuarios deberemos desplegar 4 WARS independientes a los cuales nuestras aplicaciones móviles se conecten.



Sin embargo con esta solución tampoco tendremos todo resuelto ya que todos los wars dependen del mismo servidor y como antes comente no son tan independientes como parecen en un primer momento.



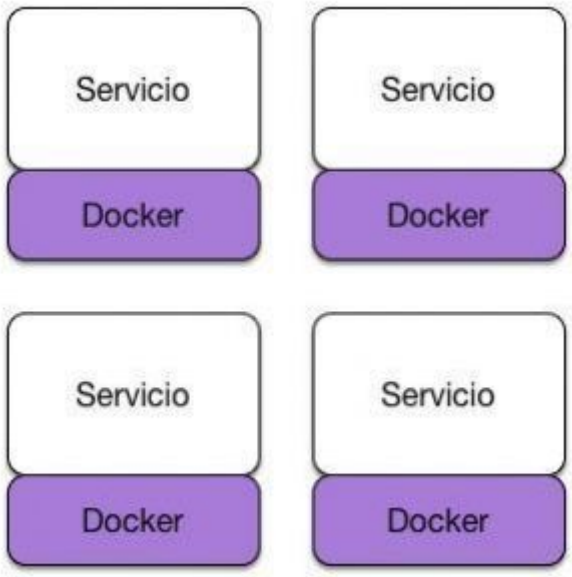
Si queremos tener servicios realmente independientes tendremos que desplegar una solución como Docker basada en contenedores y en cada contenedor un WAR. De esta forma conseguiremos un aislamiento total.



MicroServicios y problemas

Este es una de las cosas que hará que cada día pensemos más en el mundo de los

Microservicios. Las aplicaciones móviles presionarán sobre la evolución de nuestra arquitectura. Estas aplicaciones dependen de servicios y estos servicios deben de poderse desplegar de forma aislada. Sin este enfoque las labores de sistemas se complicarán cada día más.



Los MicroServicios aportan mucho pero también implican abordar muchos retos. El primer problema importante es que no es solo una decisión de si Spring MVC o JSF. Esas decisiones siempre han pertenecido al mundo del desarrollo. Ahora las decisiones afectaran de una forma importante al departamento de sistemas que es quien despliega y es muy posible que ellos sean los que tengan que tomar la decisión final . Los MicroServicios tienen muchos temas todavía que clarificar , transaccionalidad , modularidad , servicios transversales etc. Pero cada día esta más claro que han venido para quedarse.

Otros artículos relacionados : Java EE MicroServices con Payara ,Reactive MicroServices y Arquitectura , JAX-RS Client y JSON