Caso de éxito: SOA & Java

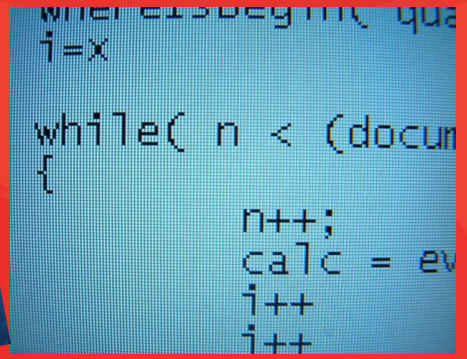
Kevin Lajpop

Java Day 2016

Acerca de ...

Kevin Lajpop

- Pensum cerrado en Ing. en Ciencias y Sistemas
- Programador
- Guitarrista y bajista por afición
- PC gamer
- Algún día fotografo



Problema y necesidad (I)

- Software poco mantenible debido a poca documentación técnica ni de procesos, cero versionamiento, pero sobre todo no existía una arquitectura definida.
- Un cambio requería que se modificara varias partes de código del sistema dado que no estaba generalizado las funcionalidades.

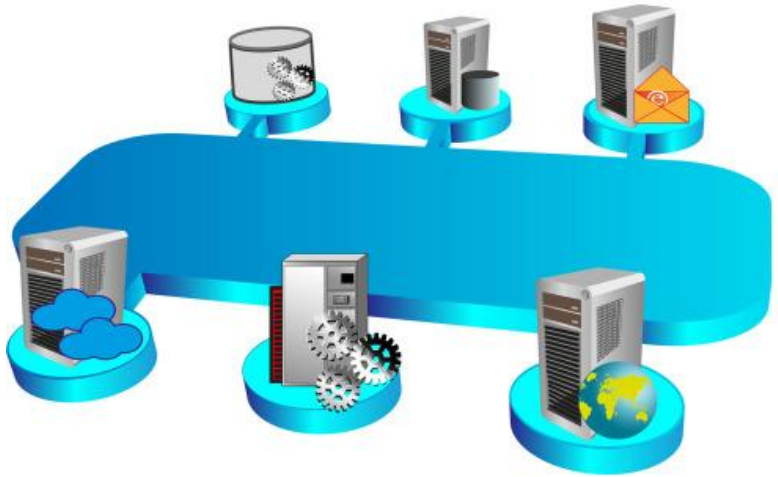
Problema y necesidad (II)

- El sistema no era tan escalable porque nuevas funcionalidades requerían demasiado tiempo.
- Todos los módulos hacen de todo, no hay especialidades.

La solución, ¿SOA?

NECESITAMOS

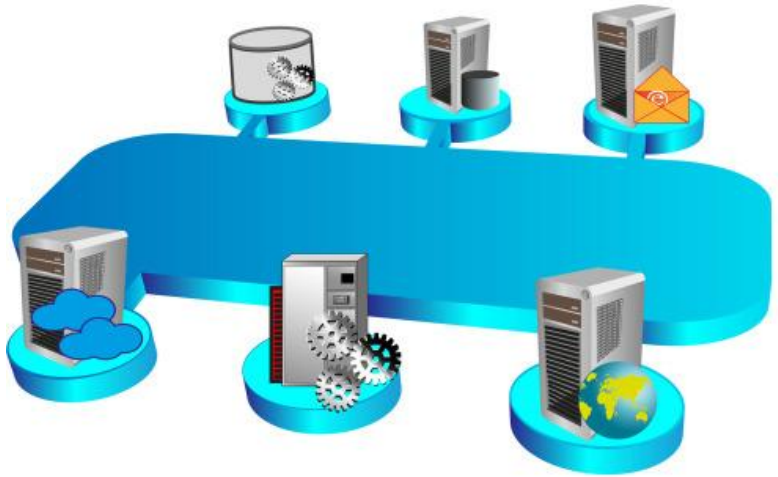
- Productividad del equipo de desarrollo.
- Escalabilidad de nuestros sistemas.
- Soluciones que sean independientes de las tecnologías o de las plataformas finales.



La solución, ¿SOA?

NECESITAMOS

- Tener un control total sobre los sistemas.
- Seguridad para nuestro sistema y sobre todo para nuestros datos.
- Pero sobre todo: **tener una arquitectura definida.**



Arquitectura orientada a servicios

Definición: *“SOA es un concepto de arquitectura de software que define la utilización de los servicios para dar soporte a los requisitos del negocio.”*

Pero… usar servicios (cualquier tipo de servicio) no significa que sea una arquitectura orientada a servicios.

Servicios

¿Cómo debe de ser un servicio?

- Definido
- Implementado
- Desplegado
- Manejado
- Reusable

- Comunicativo
- Abstracto
- Orquestado
- Granularidad
- Desacoplado
- Tiene que tener estado.

¿Por qué Java?

¿Por qué no tener servicios en otro lenguaje?

- Java nos brinda portabilidad.
- Java se presta al uso de buenas prácticas de programación con esto aumenta la buena productividad del equipo.

¿Por qué Java?

¿Por qué no tener servicios en otro lenguaje?

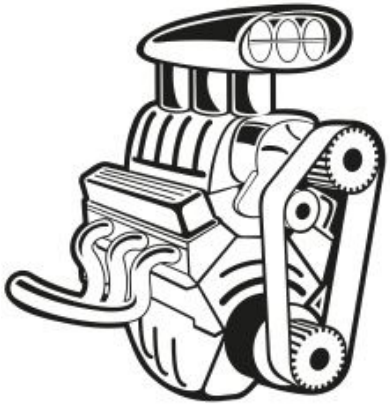
- El manejo de procesos por parte de JVM nos hace tener el control total sobre el servicio.
- El servidor ejecuta no compila.
- La principal ventaja de Java para nosotros fue su máquina virtual.

Calentando motores ...

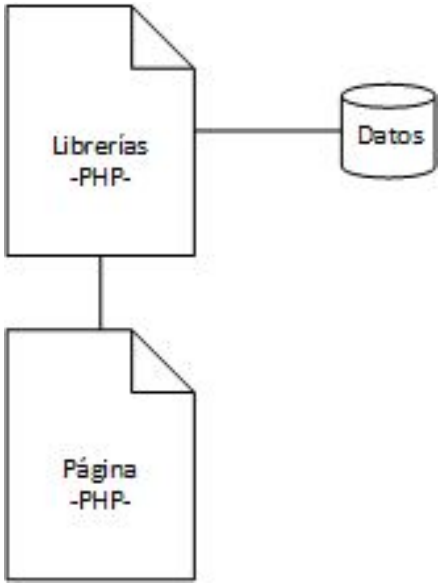
- Vamos a utilizar SOA y Java ¿qué sigue? la definición de la arquitectura.

Para aplicar una arquitectura orientada a servicios necesitamos (como mínimo) varias cosas:

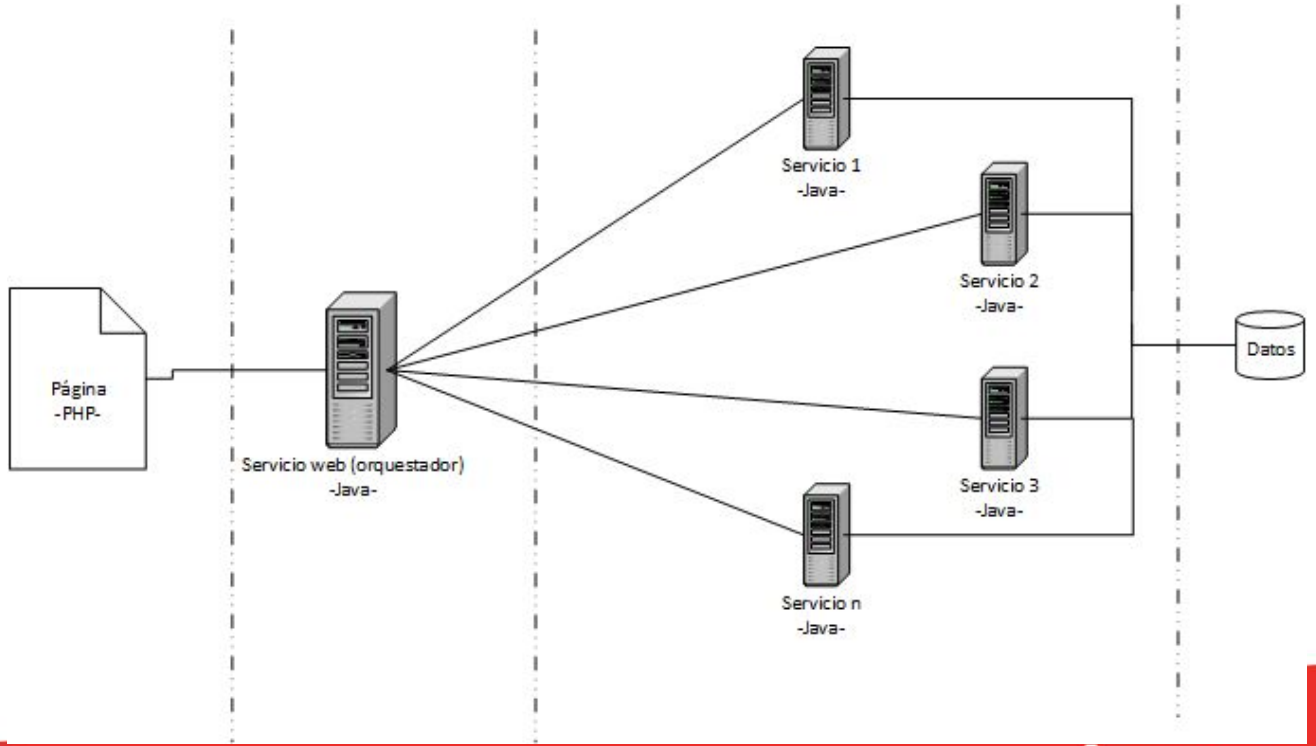
- Orquestación
- Servicios
- Consumidor



Arquitectura - Antes



Arquitectura - Después



Java y SOA (I)

Tecnologías utilizadas:

- Servicios Web (SOAP).
- Mensajes mediante colección de datos JSON.
- Orquestador en Java, servicios primitivos en Java, cliente PHP.

Con una arquitectura definida, lógica de negocio definida y con tecnologías definidas lo único que queda es: programar.

Java y SOA (II)

Los resultados de implementar una arquitectura orientada a servicios se ven desde en todo el ciclo de desarrollo, por ejemplo:

- La independencia de los servicios hace que el desarrollo sea independiente.
- Cada desarrollador se centra en su trabajo, no hay “todologos”.
- En la etapa de pruebas es más fácil encontrar bugs.

Resultados

- Pasamos de no tener arquitectura a una arquitectura sólida y escalable.
- Gracias a la JVM, que nos dá el *control* del servicio, tenemos el monitoreo de nuestros sistemas y sobre todo de nuestros procesos.
- Del lado de programador, Java es un lenguaje más amigable y se presta más a buenas prácticas.

Resultados

- Toda la carga de los sistemas están sobre los servicios aprovechando la robustez de Java.
- Obtuvimos un sistema distribuido.
- Pero sobre todo aprendimos.

¿Dudas?