

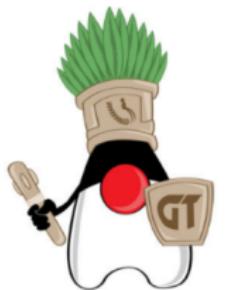
Microservicios funcionales con Java 8 y Java EE

Víctor Orozco - @tuxtor

30 de agosto de 2017

Java Cloud Day Mexico 2017

Acerca de



NABENIK

Esta presentación se basa exclusivamente en mi búsqueda diaria por trabajar menos, proyectos fracasados, lecciones aprendidas, poco capital y mucho código ;-)

Advertencia 1

¿Estado zen del arquitecto de software?



Advertencia 1

Estado zen del arquitecto de software

- Productividad
- Recurso humano
- Predictibilidad y estabilidad
- Escalabilidad
- Costos

Advertencia 1

Estado zen del arquitecto de software

- Productividad - Java 8
- Recurso humano - POO -> Funcional
- Predictibilidad y estabilidad - Java EE 7
- Escalabilidad brutal - Microservicios
- Costos - Todas las anteriores

Advertencia 1

Estado zen del arquitecto de software

- **Productividad** - Java 8
- Recurso humano - POO -> Funcional
- **Predictibilidad y estabilidad** - Java EE 7
- **Escalabilidad brutal** - Microservicios
- Costos - Todas las anteriores

TL;DR Microservicio creado con
programación funcional

TL;DR Microservicio creado con Java 8, algunas APIs de JavaEE y Payara Micro

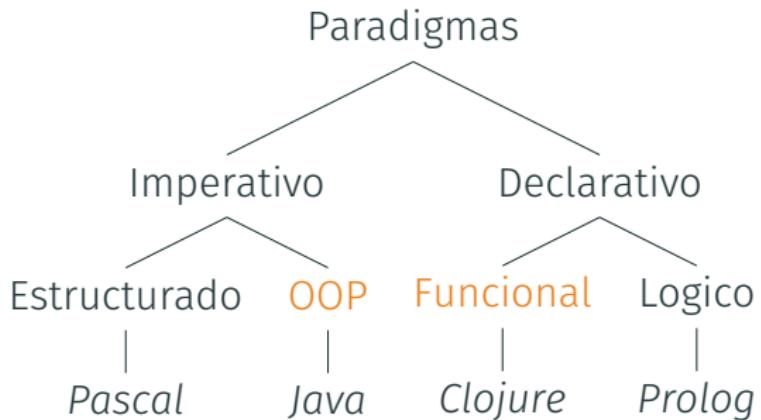
Java 8



- Nashorn
- Date/Time API
- Compact Profiles
- Type Annotations
- Default methods
- Streams
- Lambda Expressions



Paradigmas (Simplificación)



Programación funcional

- Computación = Evaluación de funciones matemáticas
(calculo de lambdas)
- NO cambios en estado
- NO mutar datos
- Declarativo → Expresiones

Un lenguaje de programación orientada a objetos con características funcionales.



¿Porque programación funcional?

- Paralelismo
- Multicore, multicpu
- Elegancia

Programación funcional en Java 8

- Java no es un lenguaje funcional puro (Clojure)
- Otras opciones JVM (Scala, Kotlin, Ceylon)
- Java soporta programación funcional a través de **bibliotecas**

Código funcional

PF en Java (Consecuencias)



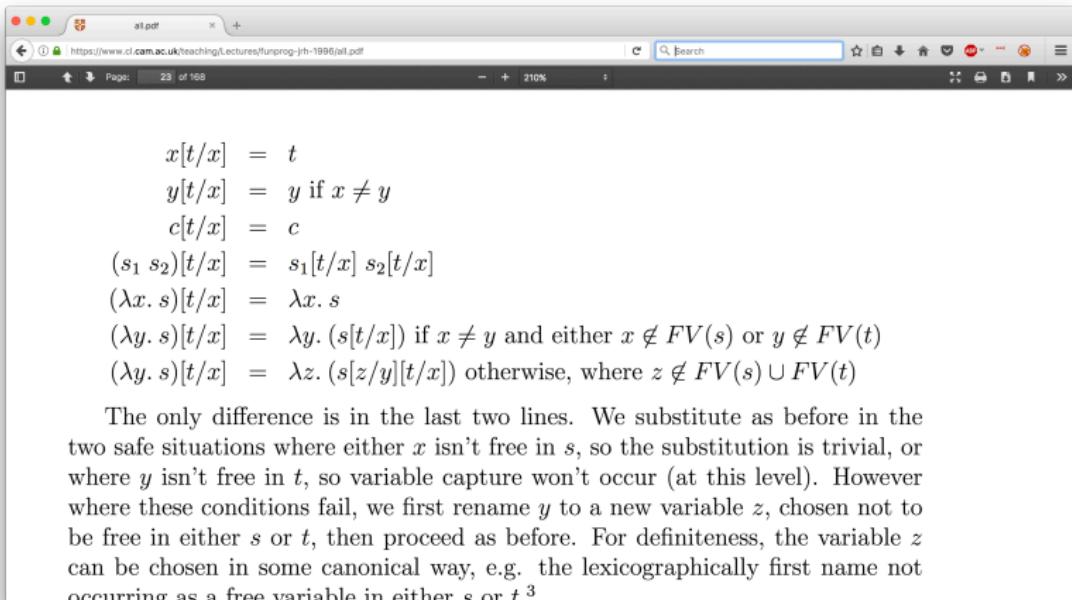
Java 8 generó un nuevo ecosistema de bibliotecas funcionales, declarativas y reactivas . . .

PF en Java (Consecuencias)

Google: Como me vuelvo el champion de la programación funcional

PF en Java (Consecuencias)

Google: Como me vuelvo el champion de la programación funcional



The screenshot shows a web browser window displaying a PDF document titled "all.pdf" from the URL <https://www.cl.cam.ac.uk/teaching/Lectures/funprog-jh-1996/all.pdf>. The browser interface includes a search bar, a page number indicator (23 of 168), and various navigation and zoom controls.

The PDF content contains the following definitions of substitution:

$$\begin{aligned}x[t/x] &= t \\y[t/x] &= y \text{ if } x \neq y \\c[t/x] &= c \\(s_1 s_2)[t/x] &= s_1[t/x] s_2[t/x] \\(\lambda x. s)[t/x] &= \lambda x. s \\(\lambda y. s)[t/x] &= \lambda y. (s[t/x]) \text{ if } x \neq y \text{ and either } x \notin FV(s) \text{ or } y \notin FV(t) \\(\lambda y. s)[t/x] &= \lambda z. (s[z/y][t/x]) \text{ otherwise, where } z \notin FV(s) \cup FV(t)\end{aligned}$$

The only difference is in the last two lines. We substitute as before in the two safe situations where either x isn't free in s , so the substitution is trivial, or where y isn't free in t , so variable capture won't occur (at this level). However where these conditions fail, we first rename y to a new variable z , chosen not to be free in either s or t , then proceed as before. For definiteness, the variable z can be chosen in some canonical way, e.g. the lexicographically first name not occurring as a free variable in either s or t .³

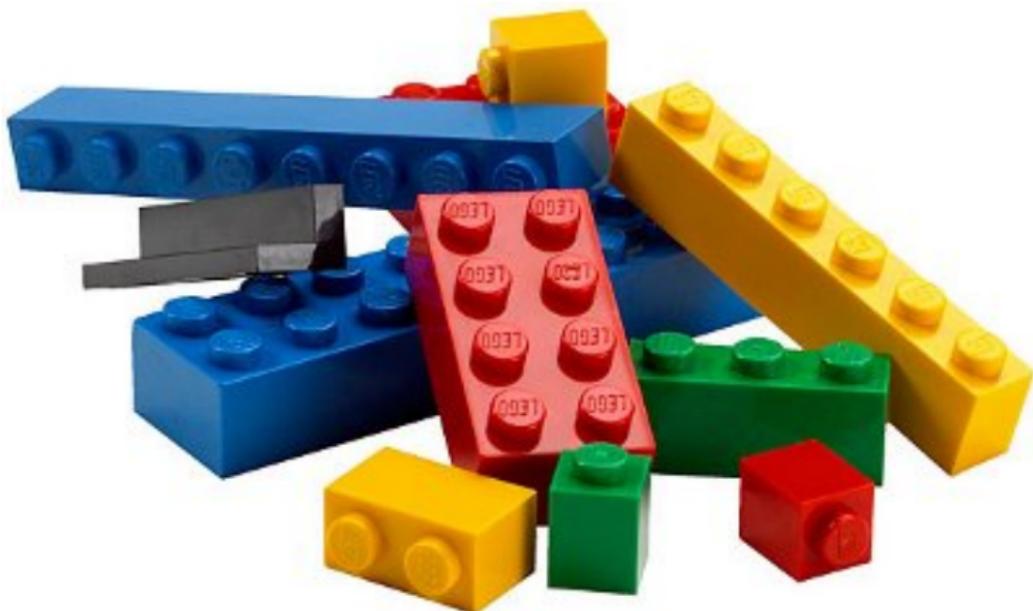
Pregunta - ¿Como lo alcance?



Respuesta - Construyendo una escalera



1 - Bloques



- Lambda expressions
- Functional interface
- High order functions
- Complements (predicates, method reference)

2 - Java Developer Kit Funcional



2 - Java Developer Kit Funcional

- Funciones predefinidas en Java 8 (`java.util.function`)
- Más de 40 interfaces funcionales
- Streams API

3 - Bibliotecas



3 - Bibliotecas

Catedral

- Java EE (Reactivos)
- Spring (Reactivos)
- Lagom (Reactivos)

Bazar

- **Funcionalidad:** vavr, jOOQ/jOOL, EclipseCollections, FunctionalJava
- **Arquitectura:** RxJava, Akka, Vert.x (Interactions - Architecture)

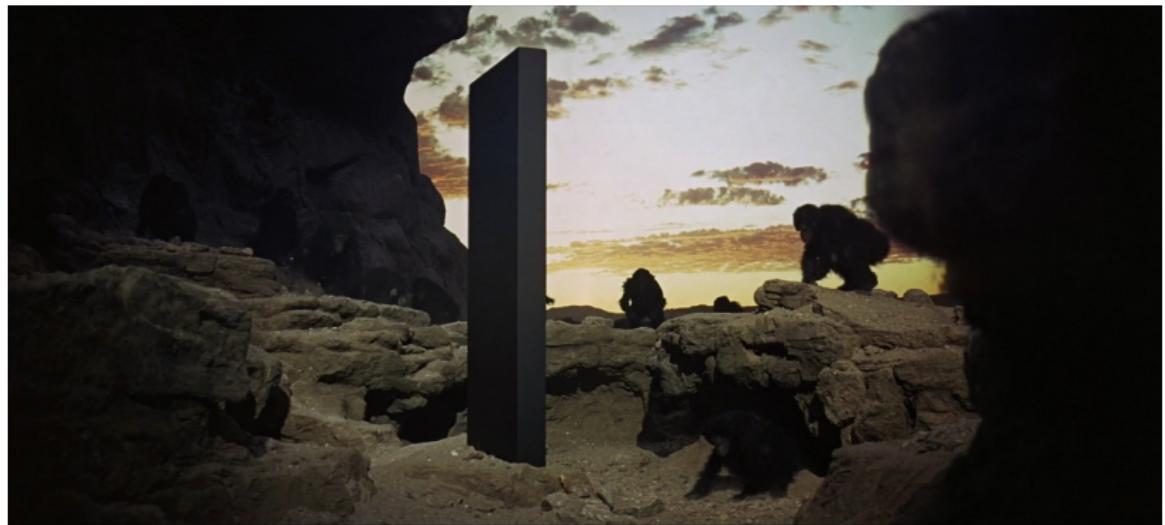
Solución

Solución

Estado zen del arquitecto de software

- Productividad - Java 8
- Recurso humano - POO -> Funcional
- ¿Predictibilidad y estabilidad?
- ¿Escalabilidad?
 - App Server - Vertical y luego horizontal
 - Microservicios - Horizontal
- Costos - Todas las anteriores

Solución



Monolito

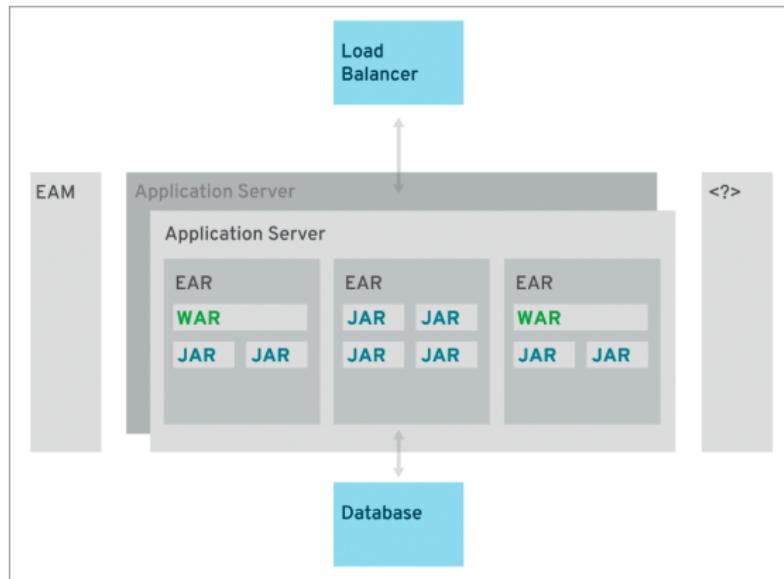


Figura 1: Arquitectura monolitica - Creditos: Markus Eisele

ESB

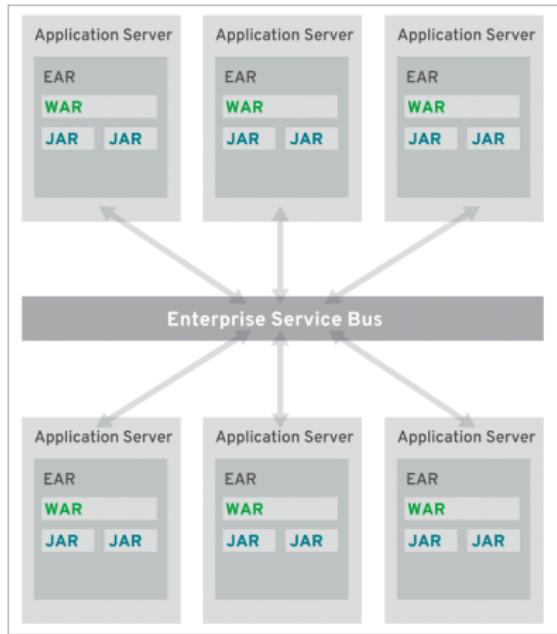


Figura 2: Arquitectura Esb - Creditos: Markus Eisele

Microservicios

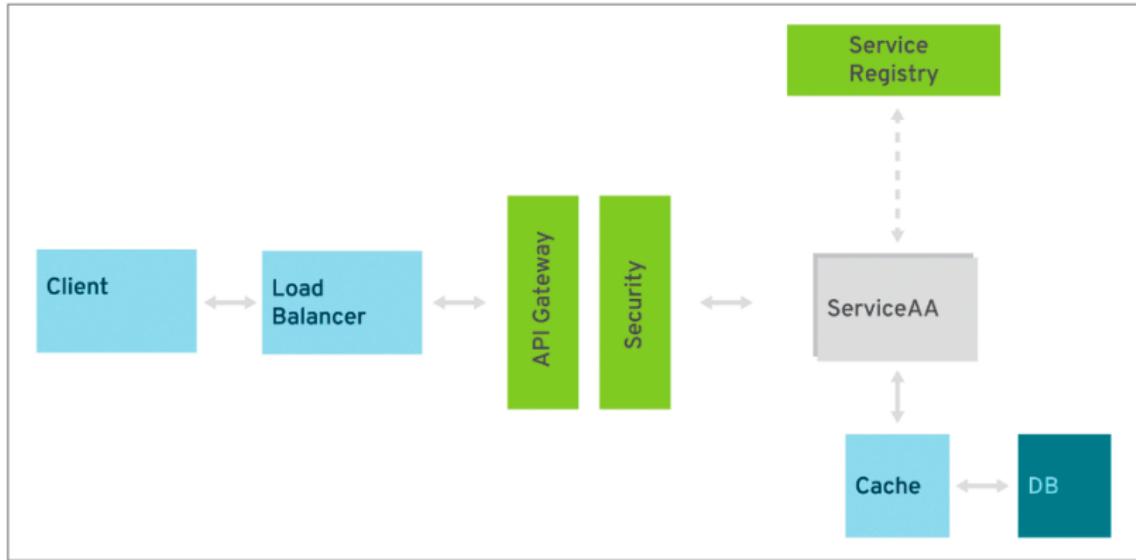


Figura 3: Arquitectura Microservicio - Creditos: Markus Eisele

Microservicios

Ventajas

- Bases de código pequeñas
- Buenas prácticas de programación
- Tolerancia a fallas
- Escalabilidad

Desventajas

- Tooling overhead
- Debug
- Transacciones distribuidas
- Latencia
- Interdependencia
- Falacias de la computación distribuida

Desventaja

Hype Driven Development

Microservicios funcionales con JavaEE



NABENIK

JavaEE el framework más anti-hype de la tierra

J2EE 1.2 (Diciembre 12, 1999)

Implementación

- Refactoring iterativo
- Refactoring practico
- Nuevos servicios

Implementación

- Refactoring iterativo
- Refactoring practico
- Nuevos servicios

Subset de Api en JavaEE

- JAX-RS
- JSON-P
- CDI, EJB
- JCache, JPA

Nuevas formas de despliegue

- Versiones micro JavaEE (CLI)
- Uber-Jar/Fat-Jar/Serverless

Opciones

- Wildfly Swarm
- KumuluzEE
- Dropwizard
- WebSphere Liberty
- Payara Micro



Microservicios - Payara Micro

- Glassfish Embedded (Web Profile)
- War en CLI
- FatJar en `main()`
- Compatible con Microprofile (JAX-RS, CDI, Config, JSON-P)
- JCache y clustering - Hazelcast

Demo



NABENIK

Payara Micro - Demo

- JAX-RS, CDI
- DeltaSpike, vavr
- JCache

Fin



Gracias

- me@vorozco.com
- <http://vorozco.com>
- <http://github.com/tuxtor/slides>



This work is licensed under a Creative Commons
Attribution-ShareAlike 3.0 Guatemala License.