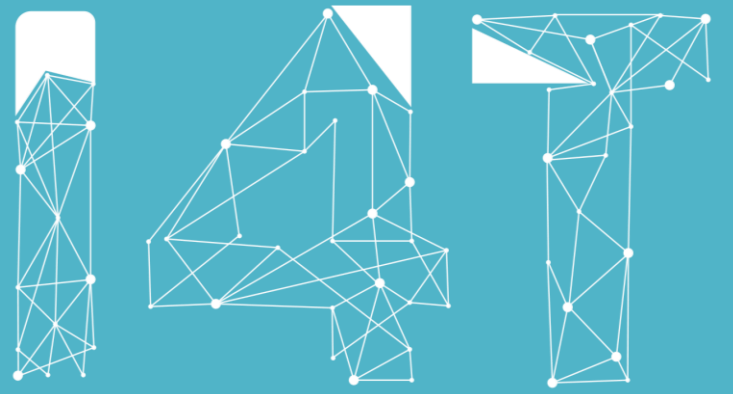
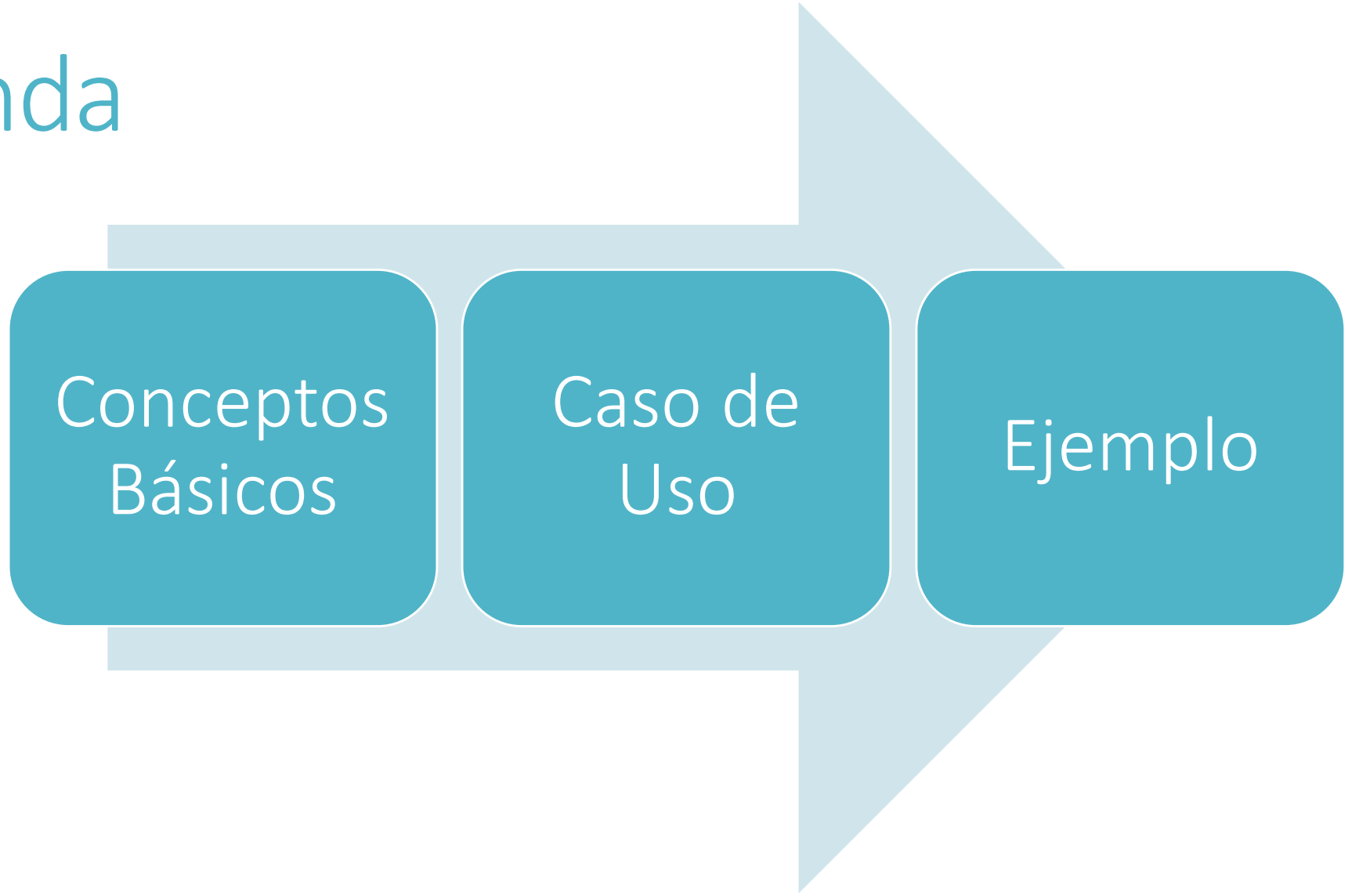


Microservicios utilizando Spring

Marvin Diaz- Is4Tech

Agenda



Entendiendo los Microservicios

No fueron inventados, han sido una evolución de arquitecturas previas.

Arquitectura Hexagonal, Alister Cockburn, 2005.
http://alistair.cockburn.us/Hexagonal+architecture

Influenciados por las tendencias en los en los negocios actuales y la evolución de la tecnología.

¿Qué son los microservicios?

El termino “Microservice Architecture” describe una forma particular de desarrollar aplicaciones de software como suites de servicios independientes, cada uno ejecutándose independientemente con su propio mecanismo de comunicación (regularmente un HTTP API). Los cuáles pueden ser escritos en diferentes lenguajes de programación y usar diferentes tecnologías de almacenamiento de datos.

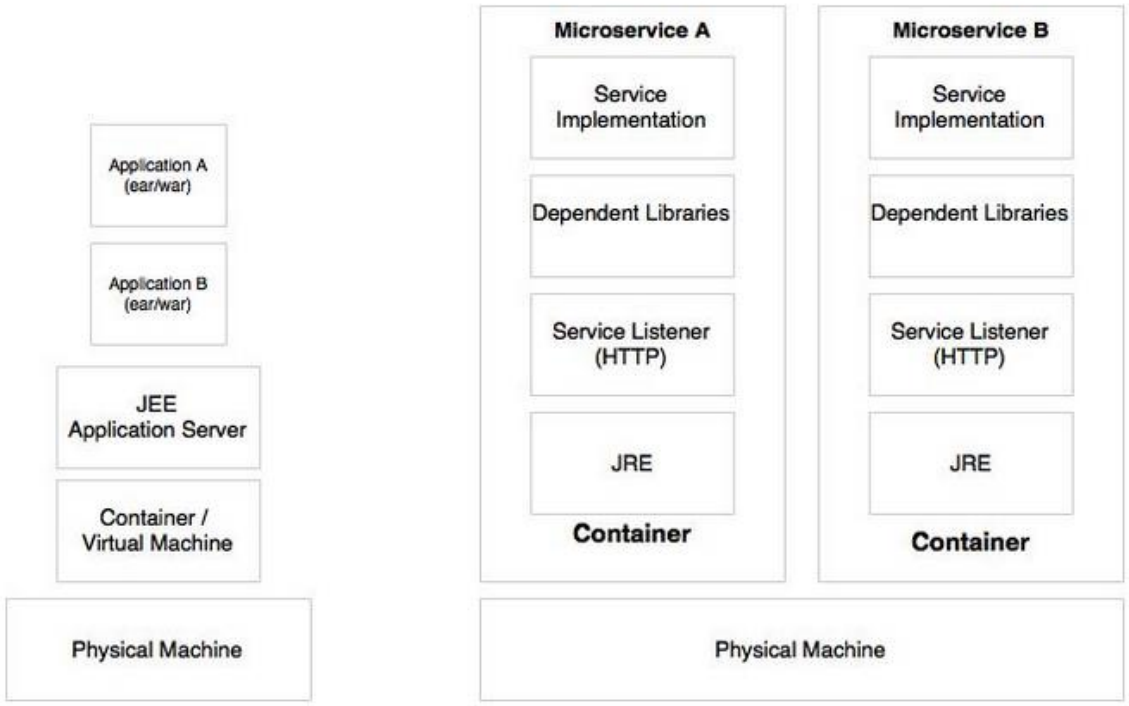
Martin Fowler https://www.martinfowler.com/articles/microservices.html

The Honeycomb Analogy

Principios de los Microservicios

Responsabilidad única por servicio

Microservicios son autónomos



Características de los Microservicios

Service Contract (JSON/REST, Swagger)

Loose coupling (Events)

Services are discoverable

Interoperability (HTTP, Rest/Json, Protocol Buffers)

Lightweight

Beneficios

Arquitectura políglota (Java/Scala, DB/ACID, ElasticSearch)

Permite la experimentación e innovación

Selectividad y Escalabilidad (500:1 airline website)

Habilita la Substitución (Third-party service)

Permite la co-existencia de diferentes versiones (Gateway rules, Duolingo)

Event-Driven architecture (topic subscribe)

DevOps

Casos de uso

Migración de aplicaciones monolíticas (Netflix, Uber, Airbnb, Amazon, Twitter)

Aplicaciones ágiles que demandan velocidad de entrega e innovación de pilotos.

Requerimientos políglotas

Servicios autónomos por naturaleza

- Servicios de pago
- Autenticación
- Notificaciones

Casos de uso que debemos evitar

Políticas organizacionales (ESB)

Cultura y procesos basados en el modelo de cascada

Conway’s Law

http://www.melconway.com/Home/Conways_Law.html

Lineamientos de diseño

Definir límites
- Funciones autónomas
- Tamaño de la unidad desplegable
- Funciones apropiadas o sub-dominios (60% búsquedas / hotel)
- Funciones Políglotas (ACID, NoSQL, etc)

Estilo de comunicación (Síncrona o Asíncrona*)

¿Cuántos endpoints?

Selección de protocolos (Rest, JMS, AMQP, Protocol Buffers*)

Versionamiento (URI*, Headers)

*KISS (Keep It simple Stupid). *YAGNI (You Ain't Gonna Need It)

Capacidades en un ambiente utilizando MS

Desarrollando MS con Spring Boot

Starters (Dependencies)

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
</dependency>
```

Embedded Web Server (jar)

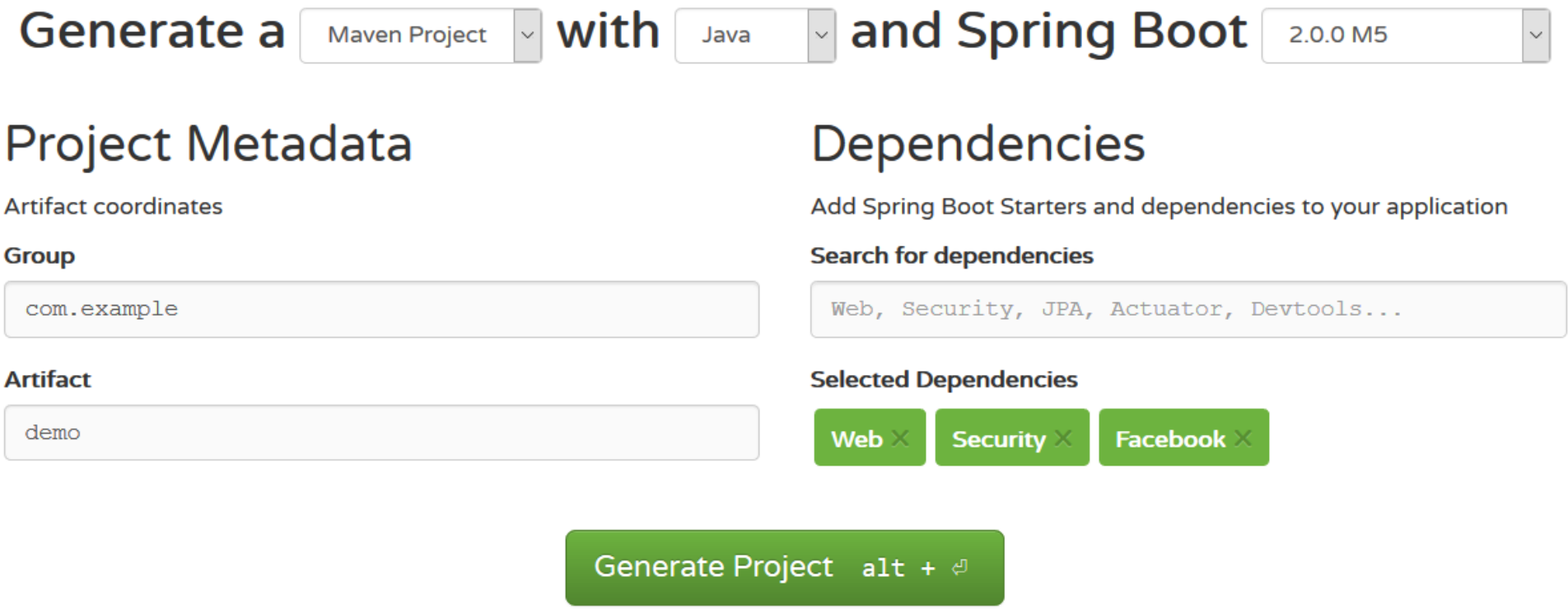
- Tomcat
- Jetty

@AutoConfiguration 80/20

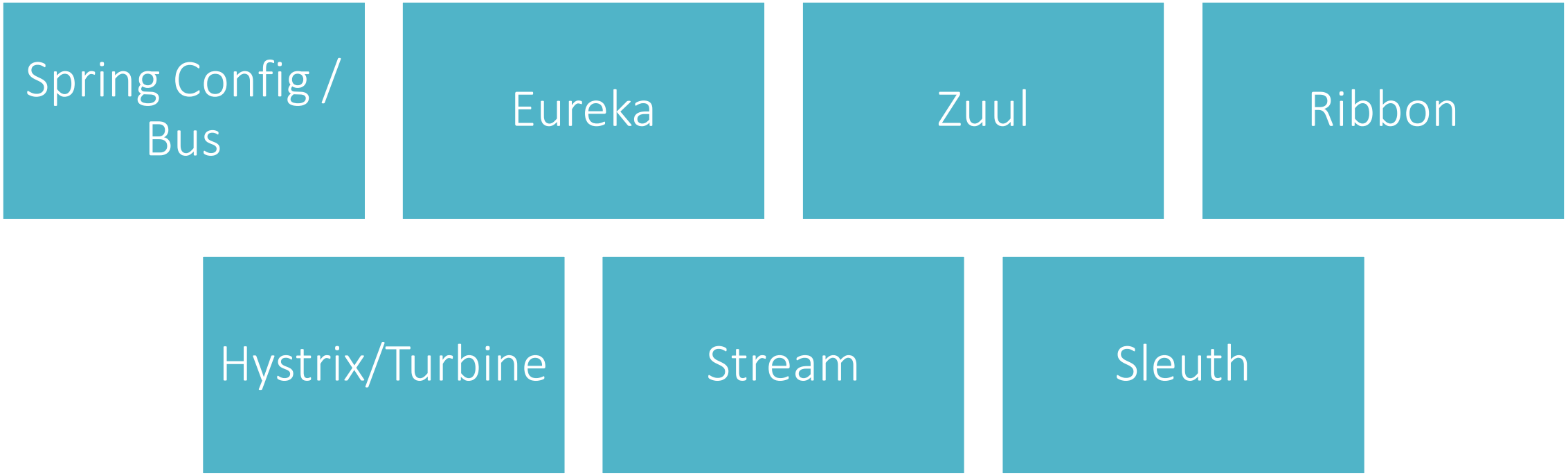
```
@SpringBootApplication
@RestController
@EnableResourceServer
@EnableSwagger2
public class ResourceServerApplication
```

http://start.spring.io/

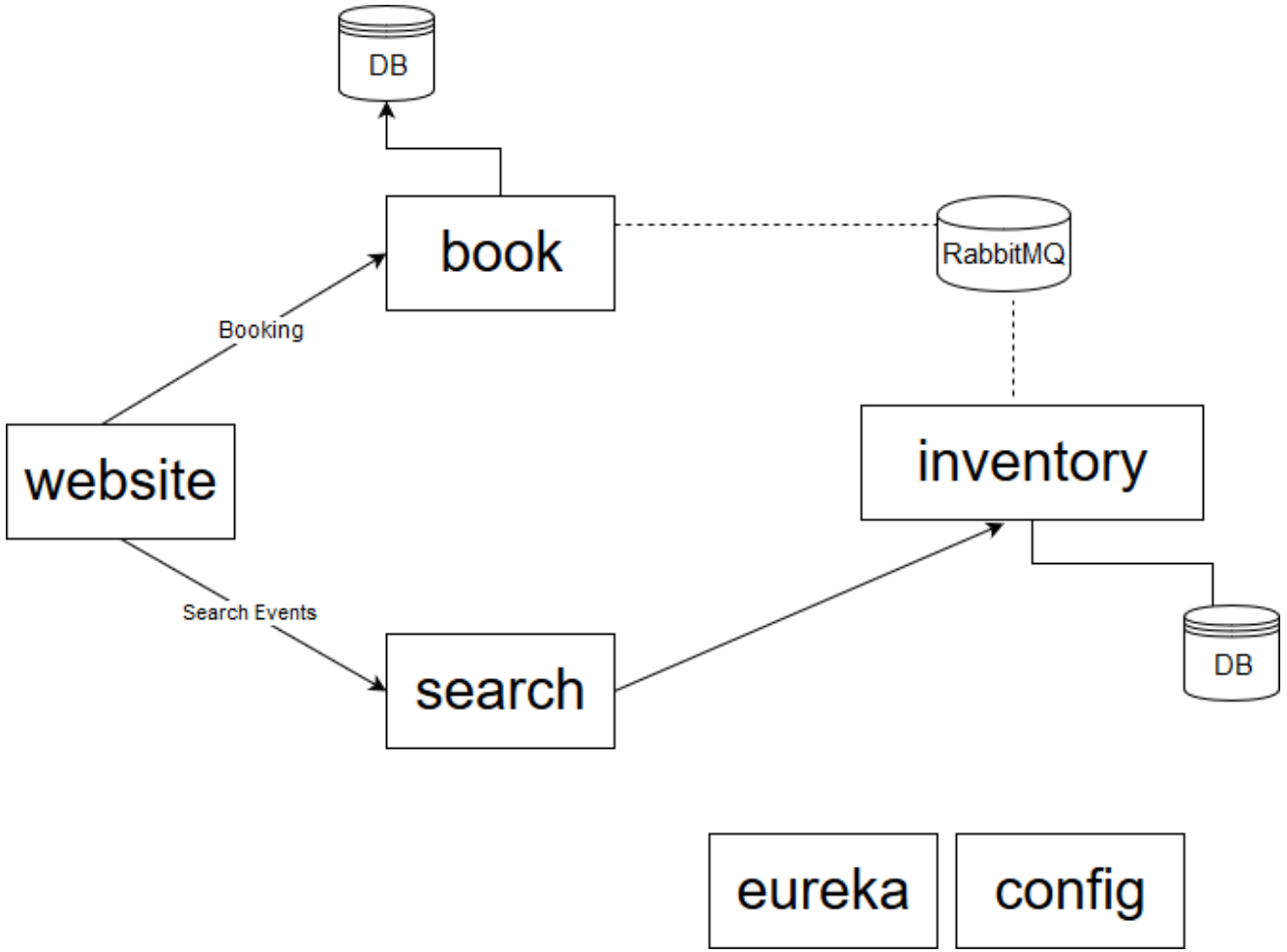
Desarrollando MS con Spring Boot



Spring Cloud



Caso de Uso


DB
book
RabbitMQ
Booking
website
inventory
Search Events
search
DB
eureka
config

Ejemplo

https://github.com/marvindaviddiaz/JavaDay2017