Programación Distribuida

Microservicios

JAIME SALVADOR

AGENDA

- **Introducción**
- Principios
- Características
- Ejemplo

Introducción

Introducción

Introducción

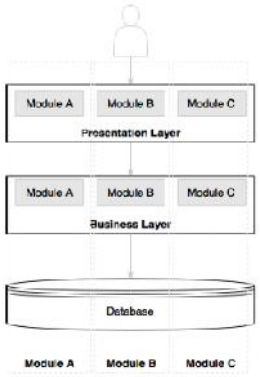
Microservicios

Es un estilo de arquitectura utilizado por organizaciones para conseguir un alto grado de agilidad, velocidad y escalamiento

Aplicaciones modulares, los módulos pueden ser separados físicamente

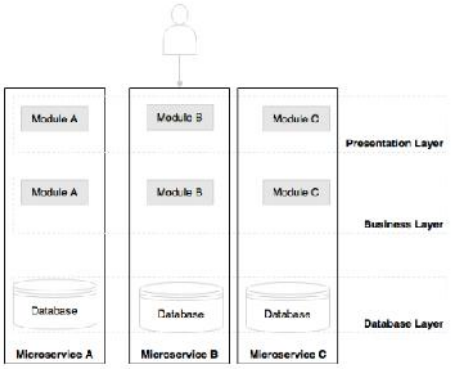
Introducción

Arquitectura tradicional



Introducción

Arquitectura basado en microservicios



AGENDA

- Introducción
- **Principios**
- Características
- Ejemplo

Principios

Single responsability (responsabilidad simple)

Una unidad de trabajo debe tener solo una responsabilidad

Principios

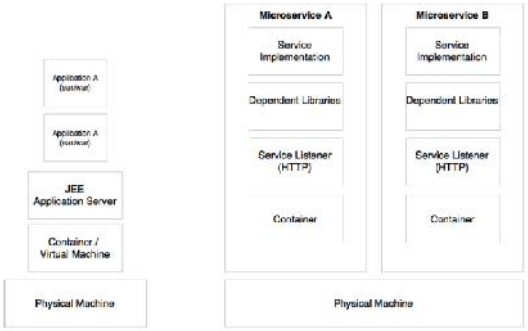
Autonomous (autonomía)

- Autocontenidos
- Unidades independientes instalables (deployables)
- Servicios autónomos

(f*uncionalidad y ejecución de una parte de la lógica de negocio)*

Principios

Autonomous (autonomía)



AGENDA

- Introducción
- Principios
- **Características**
- Ejemplo

Características

Servicios: First-class citizens

- Contrato (contract)
- Bajo acoplamiento (loose coupling)
- Abstracción (abstraction)
- Reutilización (reuse)
- Sin estado (stateless)
- Descubribles (discoverable)
- Interoperables (interoperability)
- Agrupables (composeability)

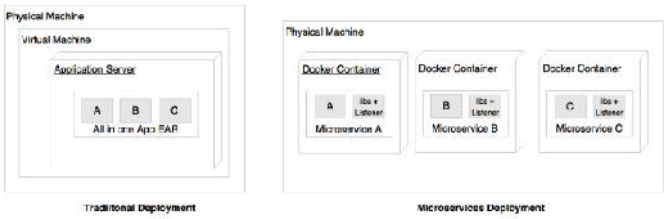
Características

Ligeros (lightweight)

- El microservicio realiza (implementa) una sola función
- El contenedor también debe ser ligero
  - Tomcat, Jetty
  - Docker

Características

Ligeros (lightweight)



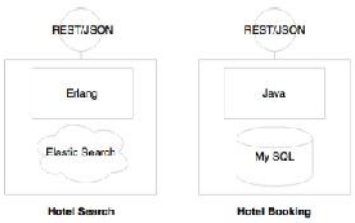
Características

Interoperabilidad (polyglot architecture)

- El microservicio debe ser autónomo y abstraer todo el funcionamiento (API?)
- Es posible tener diferentes arquitecturas para diferentes microservicios

Características

Interoperabilidad (polyglot architecture)



Características

Automatización

- PROBLEMA: La aplicación se divide en muchos microservicios
- Automatización desde el desarrollo hasta la puesta en producción

Características

Automatización

- Desarrollo
  - Git + Continuous Integration (CI)
  - CI: Jenkins -> build, deploy
- Pruebas
  - Selenium, Cucumber

Características

Automatización

- Infraestructura
  - Docker Container
- Deployment
  - Spring Cloud
  - Apache Mesos

Características

Ecosistema

- DevOps
- Log centralizado
- Registro
- API Gateways
- Monitoreo
- Routing
- Flow control

Características

Ecosistema de soporte

- DevOps
- Log centralizado
- Registro
- API Gateways
- Monitoreo
- Routing
- Flow control

Características

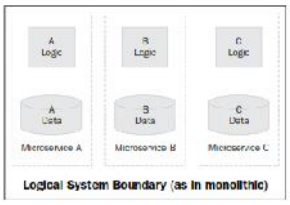
Distribuidos y dinámicos

- Encapsulan datos y lógica de negocio
- Datos + lógica de negocio distribuido
- Descentralizan los datos y la lógica de negocio

Características

Distribuidos y dinámicos

- Encapsulan datos y lógica de negocio
- Datos + lógica de negocio distribuido
- Descentralizan los datos y la lógica de negocio



Características

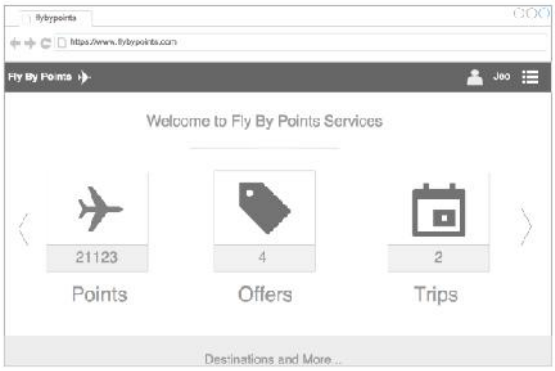
Fail fast – self-healing

- Fail fast: sistemas que esperan fallas.
  - Cuando el sistema falla, qué tan rápido se recupera de la falla?
  - Métricas: Mean Time Between Failures, Mean Time To Recover
- Self-healing: el sistema aprende de las fallas y se auto-regula

AGENDA

- Introducción
- Principios
- Características
- **Ejemplo**

Ejemplo

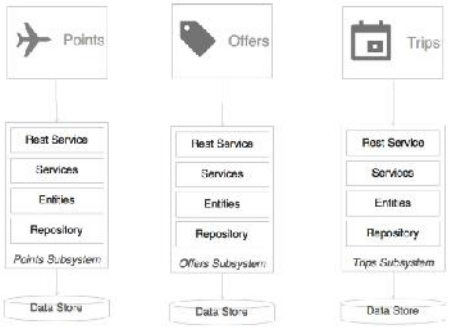


Ejemplo

Arquitectura tradicional

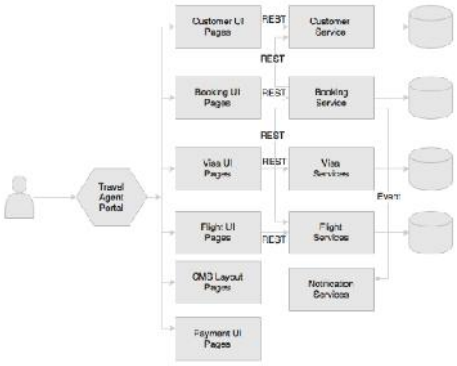
Ejemplo

Arquitectura basada en microservicios



Ejemplo

Agencia de viajes



Frameworks

WildFly SWARM

spring boot

WSO2 Microservices Framework for Java

PREGUNTAS?