

# **microservicios**

**Introducción y conceptos generales**



**¿Por que hablar de microservicios en FLISoL?**



**Comunidad**

**Aprender y Compartir**

**Software Libre**

# ¿Qué es la arquitectura de microservicios?

- Es una **interpretación** de Service Oriented Architecture para construir **sistemas distribuidos**.
- A diferencia de SOA surge como resultado de la **experiencia** y no solo como una teoría.

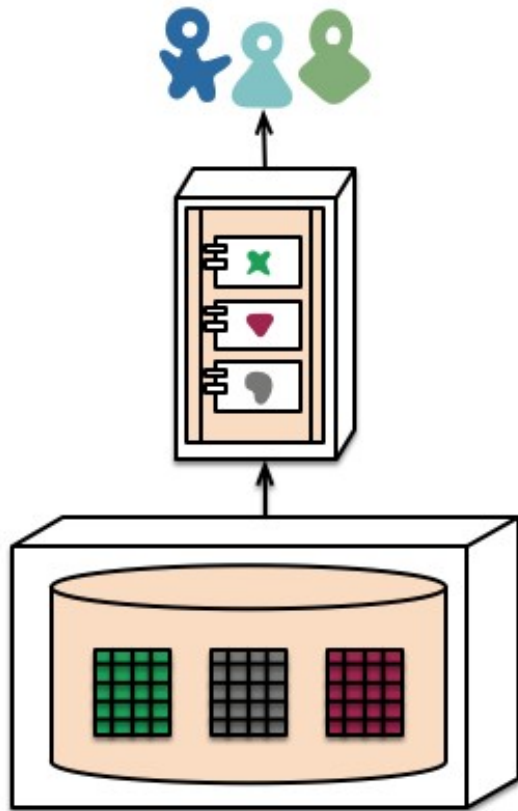
# ¿Qué es la arquitectura de microservicios?

## **Estructura:**

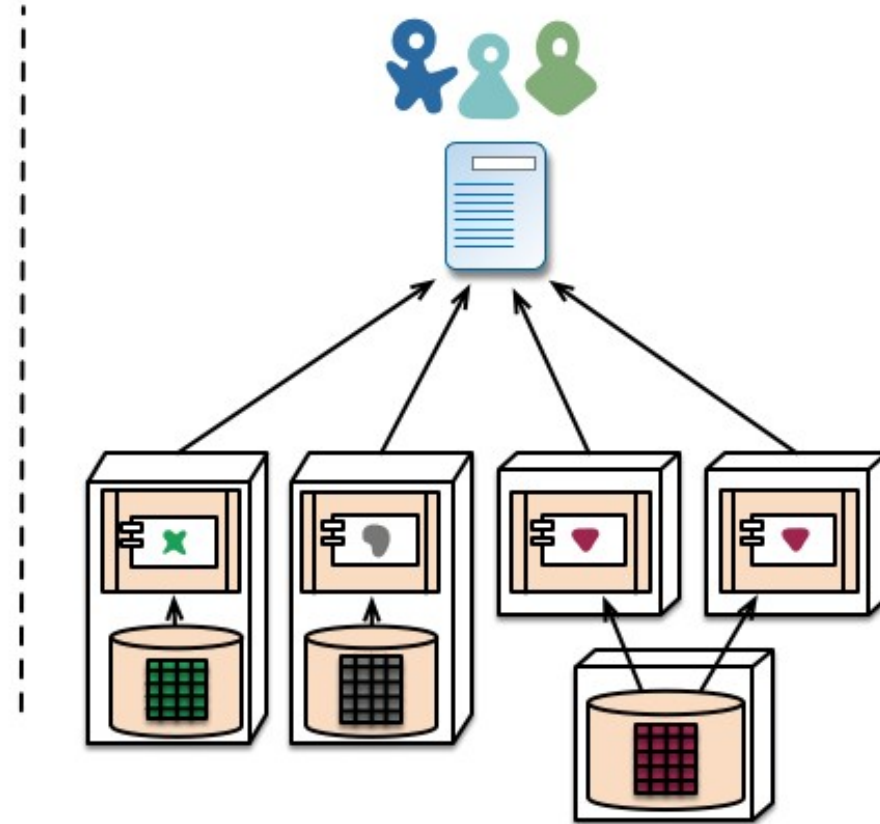
Conjunto de servicios desacoplados.

## **Objetivo:**

Acelerar el desarrollo de software mediante la aplicación de un flujo completo de integración y despliegue continuos.



monolith - single database



microservices - application databases

# ¿Qué es un microservicio?

Es un artefacto de software que:

- Sigue la filosofía Unix **“Hace una sola cosa y la hace bien”**
  - Tiene una área de dominio bien delimitada
    - Es autónomo en su desarrollo
  - Se puede instalar de manera independiente
  - Automatiza cada etapa de su ciclo de vida  
(desarrollo, pruebas, empaquetado, despliegue y escala)

# Ventajas de usar microservicios

- Permiten la integración de sistemas con tecnología heterogénea.
- Aceleran el ritmo de desarrollo al reducir el impacto en otros componentes del sistema.



# Ventajas de usar microservicios

- Incrementan la resiliencia del sistema (capacidad de recuperarse ante un fallo).
- Adaptan su escala fácilmente a la demanda del sistema.

# Desventajas de usar microservicios

- Mantener un sistema distribuido no es una tarea trivial.
- La consistencia de los datos es asíncrona y no transaccional.
- Aumenta la complejidad para probar y monitorear.

# ¿Por dónde empiezo?

Analiza la estructura de tu organización

“Las organizaciones dedicadas al diseño de sistemas [...] están abocadas a producir diseños que son copias de las estructuras de comunicación de dichas organizaciones”

Melvyn Conway, 1967

# ¿Por dónde empiezo?

## Prepara a tu equipo

- Promueve una cultura de la automatización
- Potencia las habilidades existentes y desarrolla aquellas que hagan falta
- Implementa un flujo efectivo de Continuous Delivery



# ¿Por dónde empiezo?

Evalúa el estado de tu proyecto

- Coding (practices, patterns, etc)
- Testing (unit, acceptance, functional, integration, coverage, performance)
- Deployment (infrastructure as a code, containers, h/a)

# ¿Por dónde empiezo?

Define los objetivos del proyecto

- Objetivos estratégicos
  - Principios
  - Prácticas

# ¿Qué sigue?

- Definir los dominios del sistema
- Buscar los patrones de diseño y adecuarlos al diseño general  
(Comunicación, descubrimiento de servicios, consistencia de datos, seguridad, pruebas, etc.)
- Elegir las herramientas para implementar los componentes base del sistema  
(Infraestructura, pruebas, empaquetado, monitoreo, etc)

# Definiendo la estructura del sistema

Detalla el modelo general que gobernará la implementación:

- Definición de dominios (de acuerdo a la lógica de negocios)
- Estandarización de interfaces (reglas y contratos entre servicios)
  - Mecanismos de comunicación (REST, RPC, etc)



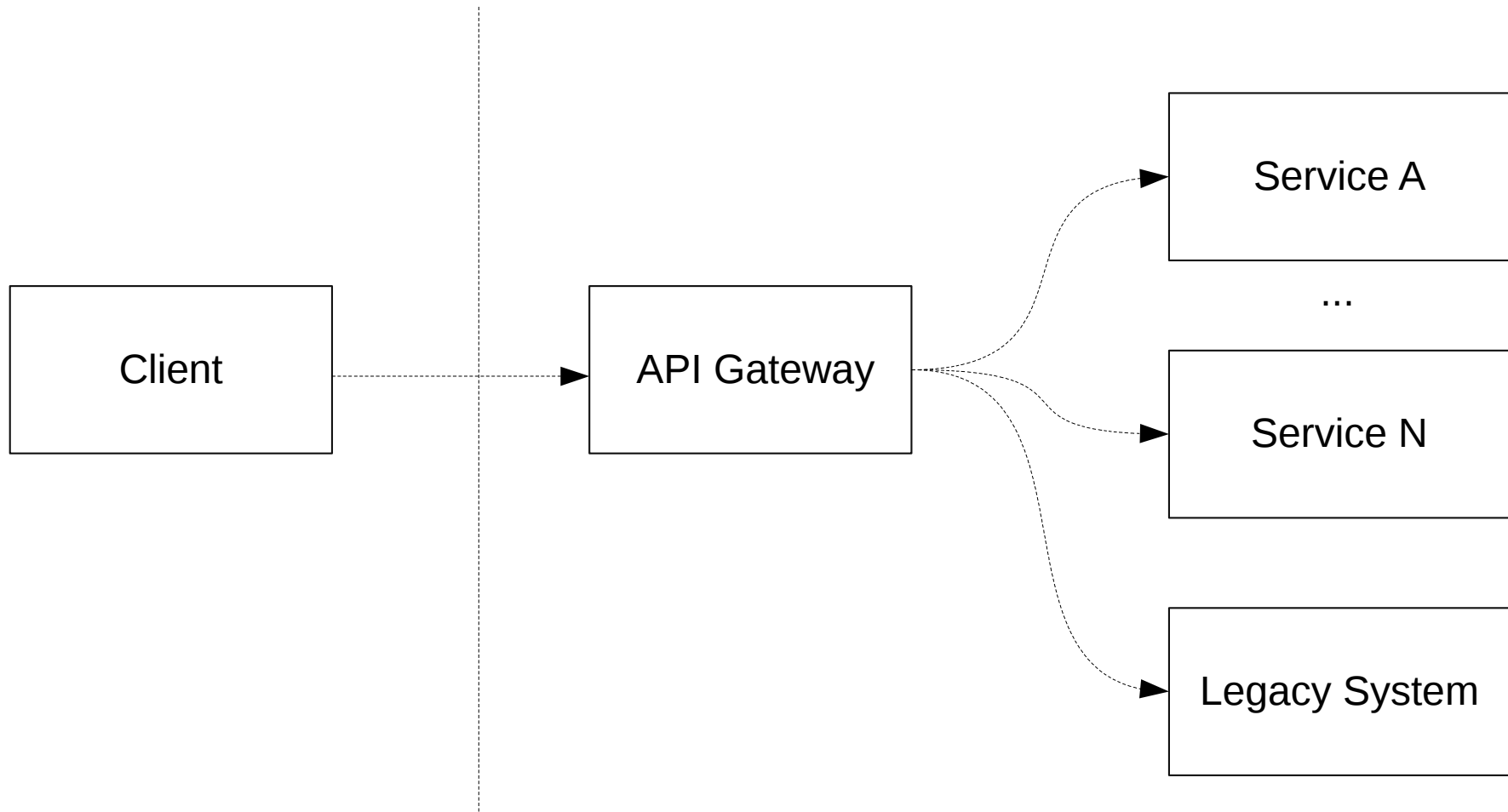
# Definiendo la estructura del sistema

- Identificar dominios (de acuerdo a la lógica y estructuras de negocio)
- Estandarizar interfaces (reglas y contratos entre servicios)
- Establecer mecanismos de comunicación (REST, RPC, etc)

# Buscando patrones de diseño

- Ruteo (API Gateway)
- Descubrimiento de servicios (Service Registry)
- Consistencia de datos (Eventually consistent approach)
  - Seguridad (Access Token)

# API Gateway (Patrón de diseño)



# API Gateway (Herramientas)



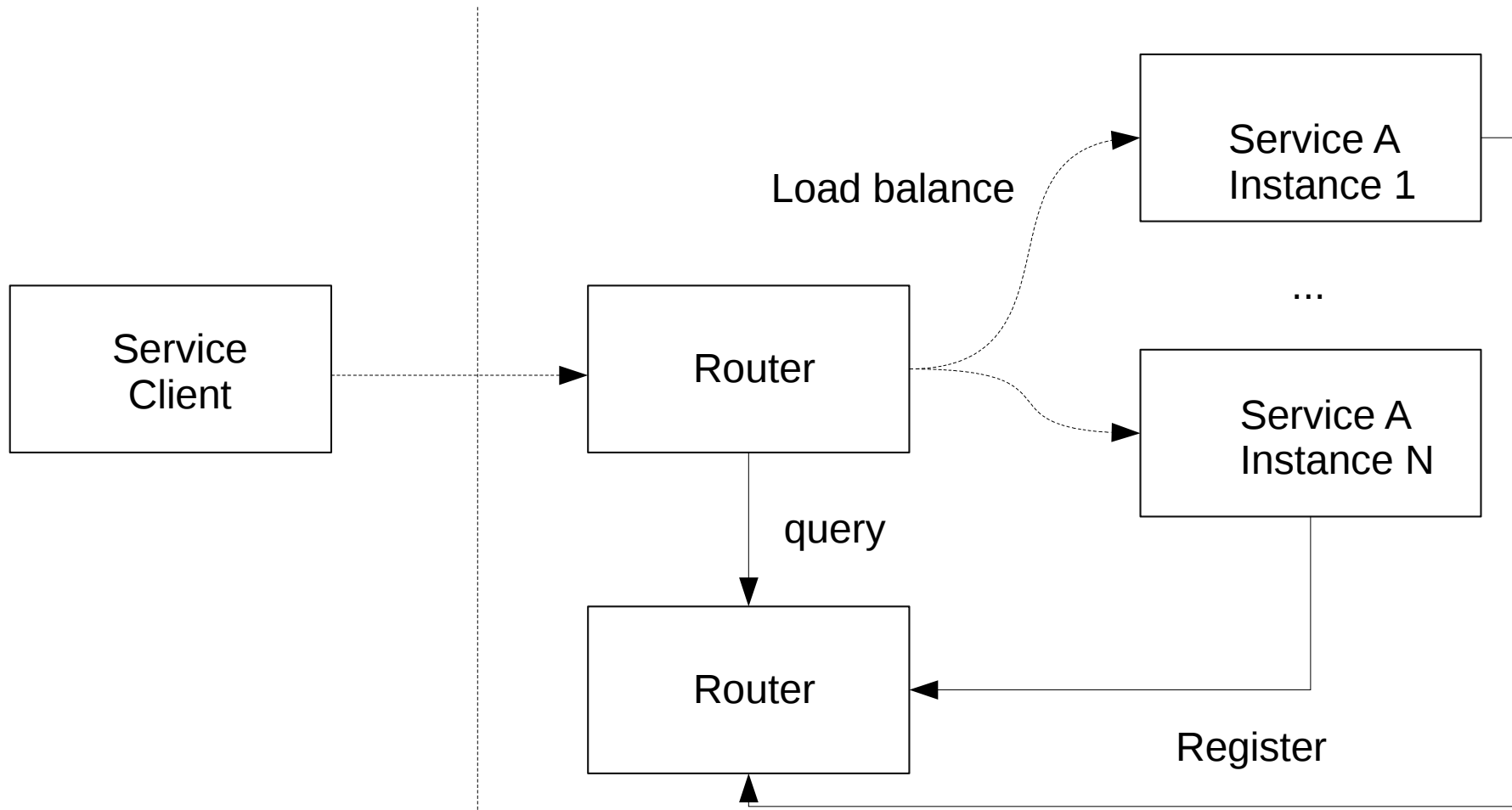
- Netflix / zuul

- Kong

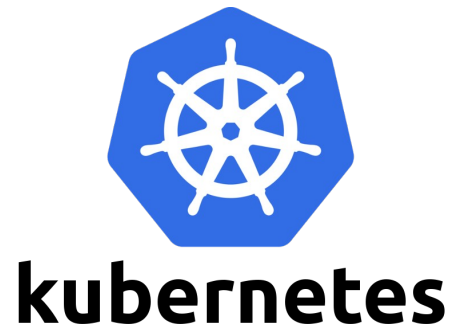
**NGINX**

- Solución a la medida

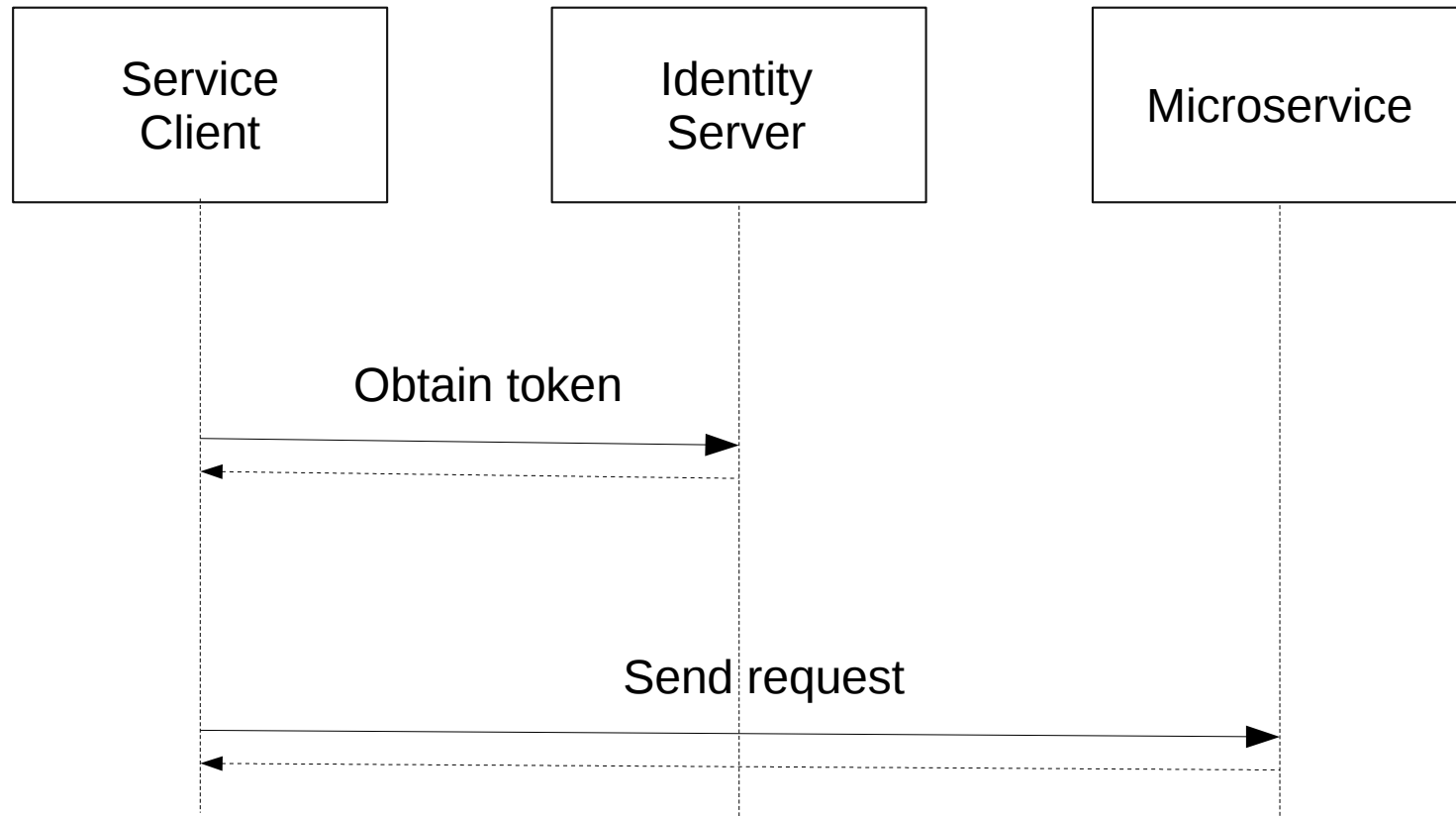
# Service Discovery (Patrón de diseño)



# Service Discovery (Herramientas)



# Service Authentication (Patrón de diseño)



# Service Authentication (Herramientas)



(XML  
Assertions)



(JSON tokens firmados)



OpenID Connect

(Misma funcionalidad que OID2  
pero amigable con los APIs)



# Antipráticas

- Intentar usar la última tecnología sin tener una razón para ello
- Intentar crear un sistema desacoplado antes de tener claros los dominios de cada aplicación

“If you can't build a well structured monolith what makes you think microservices are the answer?”

# Antipráticas

- Mantener más de un servicio en un mismo repositorio
- Usar una misma base de datos para mas de un servicio

# Antipráticas

- Exponer todos los componentes del servicio de manera pública
- Tener dependencias entre despliegues de diferentes servicios

# Antipráticas

- Pensar que la automatización viene después

# Referencias

- NEWMAN, Sam (2015), *Building Microservices*, USA: ed. O'Reilly
- NADAREISHVILI, et al (2016), *Microservices Architecture*, USA: ed. O'Reilly
- RICHARDSON, C., *A pattern language for microservices*, <http://microservices.io/patterns> Última consulta 21 Abr. 2017.
- FOWLER, M., *Microservices Resource Guide*, <https://www.martinfowler.com/microservices/> Última consulta 21 Abr. 2017.

**2017.04**

**mimex@fipasoft.mx**