Introducción a Microservicios con Spring Cloud

Por Geovani de León

Geovani de León (Yova)

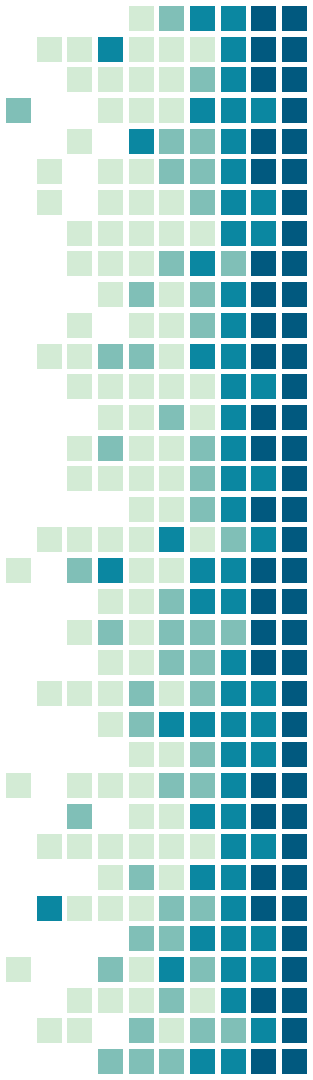
Ingeniero de Software en Pensotec

Lider de Microsoft Dev Group Cobán

Docente Universitario

https://github.com/yovafree

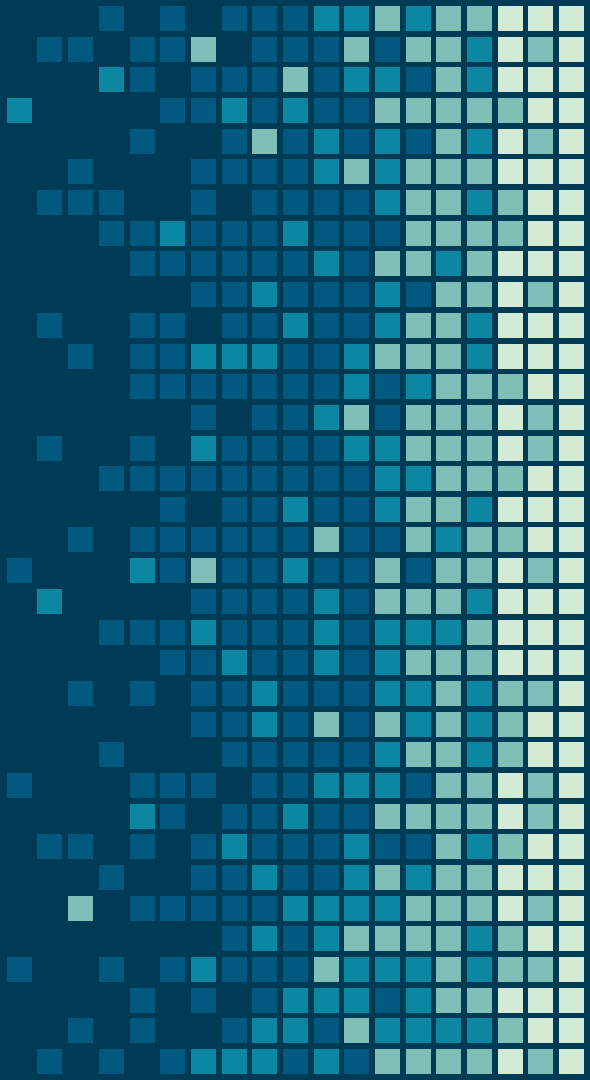
https://yovadeleon.dev



Agenda

- Arquitectura monolítica
- Arquitectura basada en microservicios
- Retos y Soluciones de Microservicios
- Arquitectura Monolítica vs Microservicios
- Características de Microservicios
- Buenas prácticas en Microservicios
- Spring Cloud
- Demo
- Preguntas y respuestas

Arquitectura de Software



“Es un conjunto de patrones que proporcionan un marco de referencia necesario para guiar la construcción de un software.

Arquitectura Monolítica

En la ingeniería de software, una aplicación monolítica describe una única aplicación de software en niveles en los que la interfaz de usuario y código de acceso a datos se combinan en un solo programa de una plataforma única.

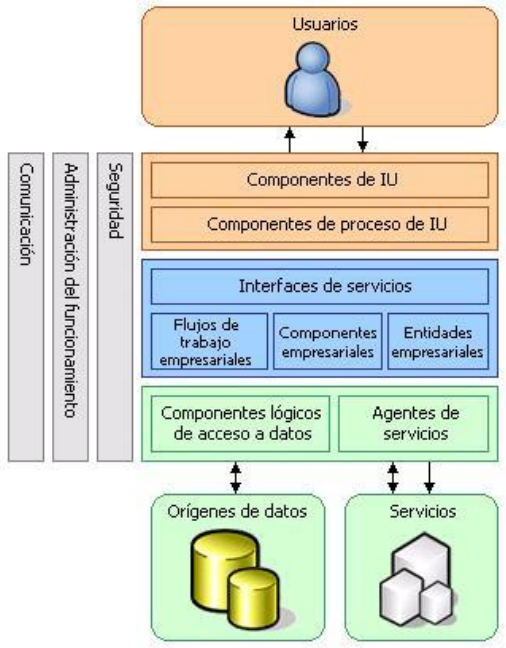
Arquitectura Monolítica

Pros

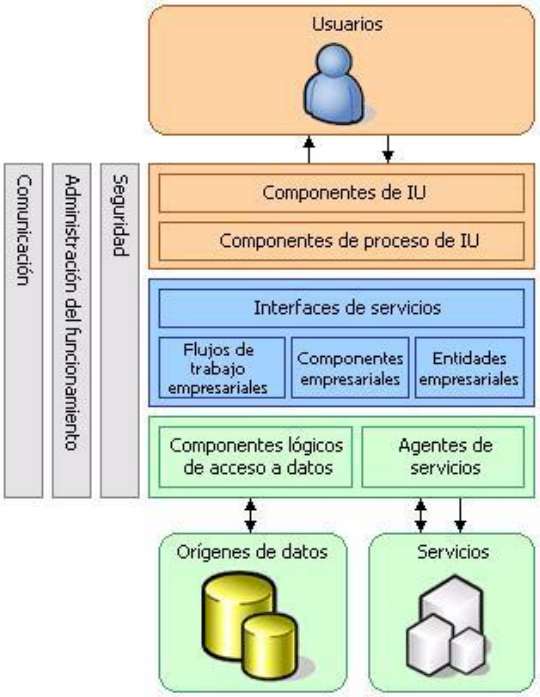
- Fácil de desarrollar
- Fácil de entender
- Baja complejidad

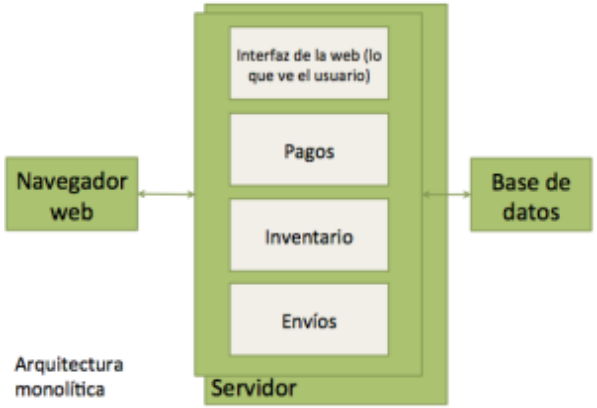
Contras

- Baja especialización
- Recursos altos para escalar
- Nuevos cambios, nueva versión!



Arquitectura Monolítica - Ejemplo





Arquitectura basada en Microservicios

Es un estilo de arquitectura y un modo de programar software. Las aplicaciones se dividen en componentes más pequeños, y son independientes entre sí.

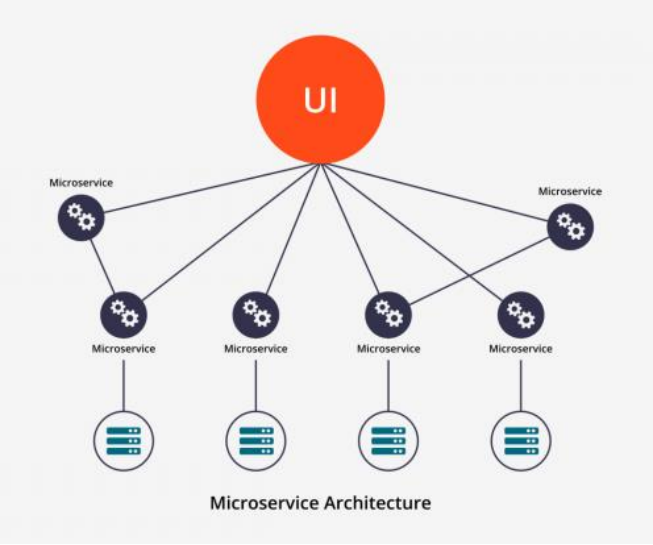
Cada uno de estos elementos es un microservicio. Este enfoque privilegia el nivel de detalle, la sencillez y la capacidad de compartir un proceso similar en varias aplicaciones.

Es un enfoque fundamental hacia un modelo nativo de la nube.

Arquitectura basada en Microservicios

Pros

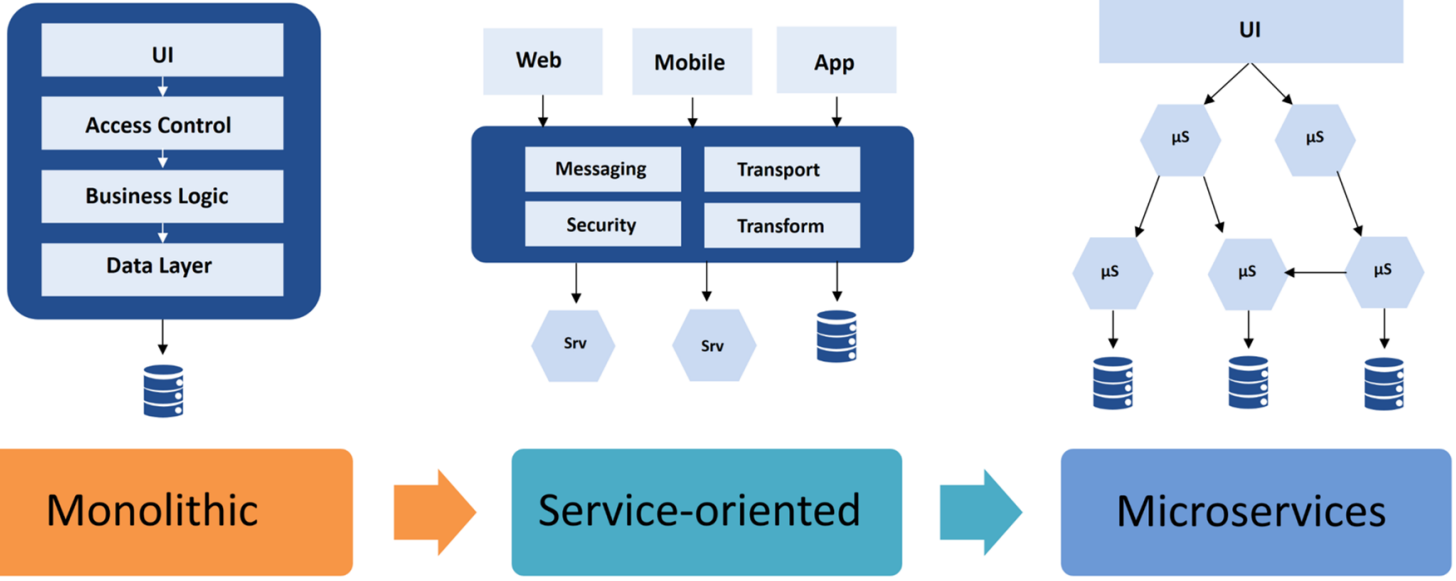
- Alta granularidad
- Alta especialización
- Escalabilidad
- Facilidad de combinar con diversas tecnologías
- Resiliencia a fallos



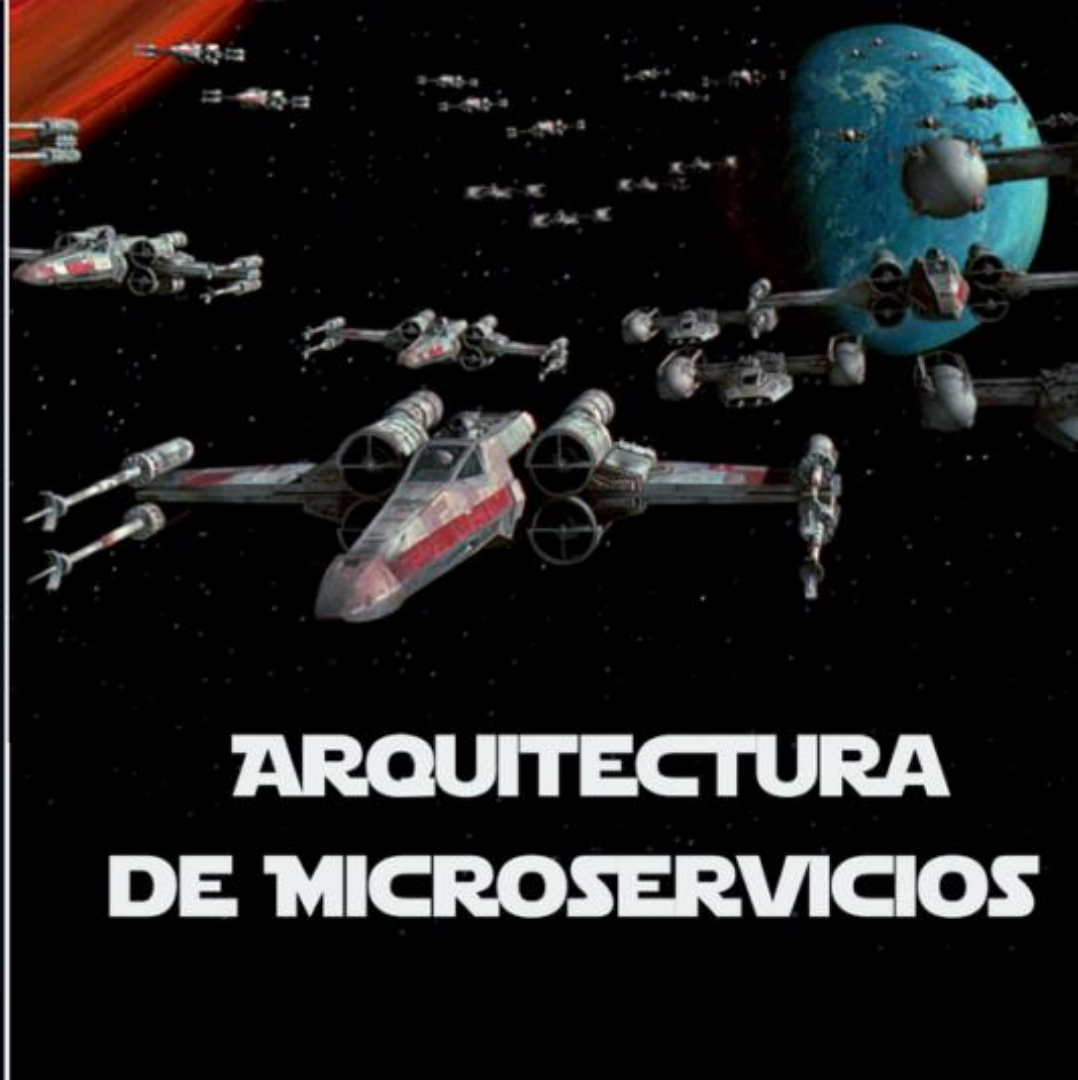
Contras

- Alta complejidad
- Requiere buena comunicación del equipo de desarrollo
- Requiere alto dominio de patrones de diseño y arquitectura

Evolución de la Arquitectura de Software



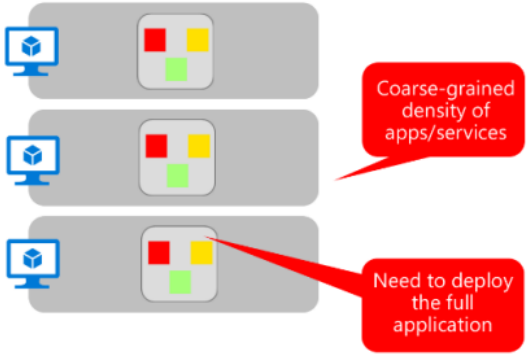

ARQUITECTURA
MONOLÍTICA


ARQUITECTURA
DE MICROSERVICIOS

Enfoque Monolítico vs Microservicios

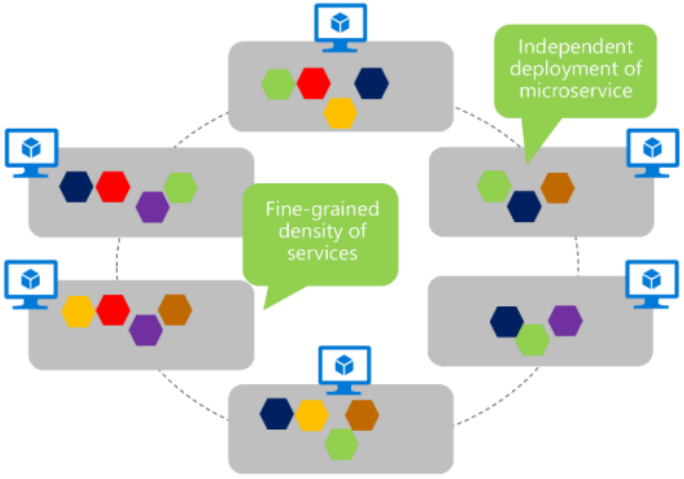
Monolithic deployment approach

- A traditional application has most of its functionality within a few processes that are componentized with layers and libraries.
- Scales by cloning the app on multiple servers/VMs



Microservices application approach

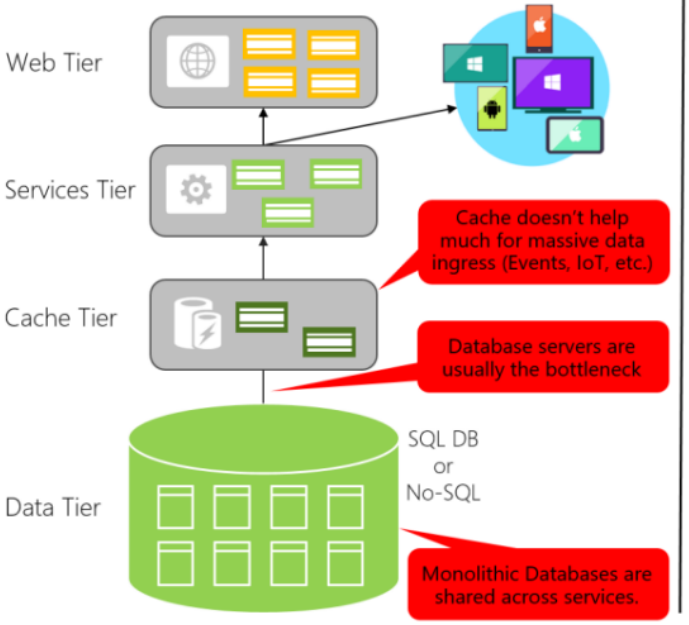
- A microservice application segregates functionality into separate smaller services.
- Scales out by **deploying each service independently** with multiple instances across servers/VMs



Enfoque Monolítico vs Microservicios

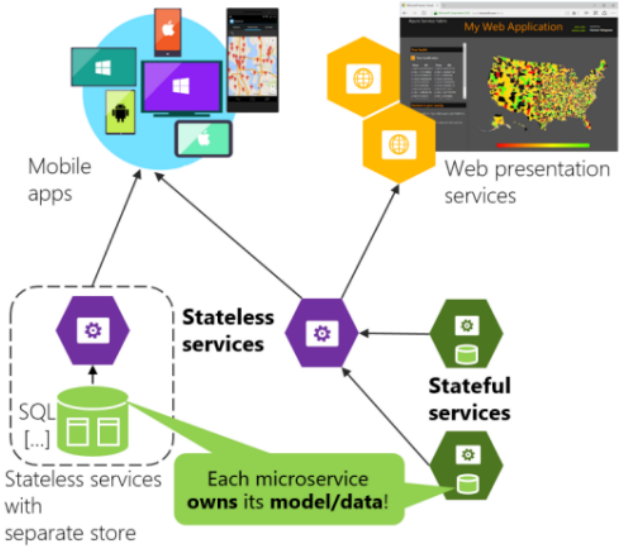
Data in Traditional approach

- Single monolithic database
- Tiers of specific technologies

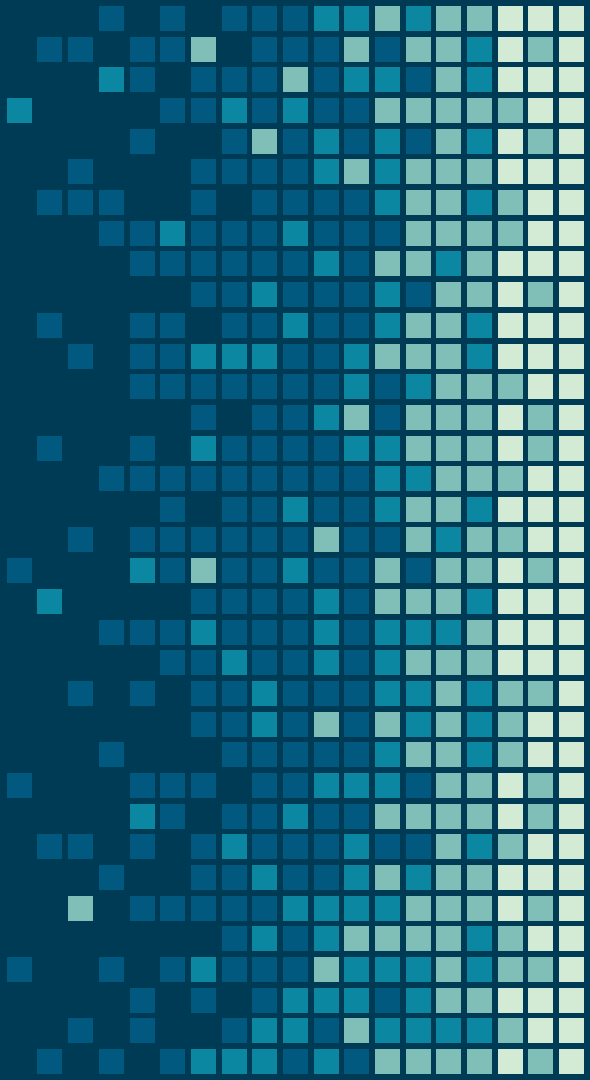


Data in Microservices approach

- Graph of interconnected microservices
- State typically scoped to the microservice
- Remote Storage for cold data



Retos al abordar el enfoque en Microservicios



Reto #1: Cómo definir los límites de cada microservicio

- Se debe intentar identificar las islas de datos desacopladas y los diferentes contextos dentro de la aplicación.

- “un Usuario puede referirse a un usuario en el contexto de identidad o membresía, a un cliente en el contexto de CRM, a un comprador en el contexto de pedidos y así sucesivamente”

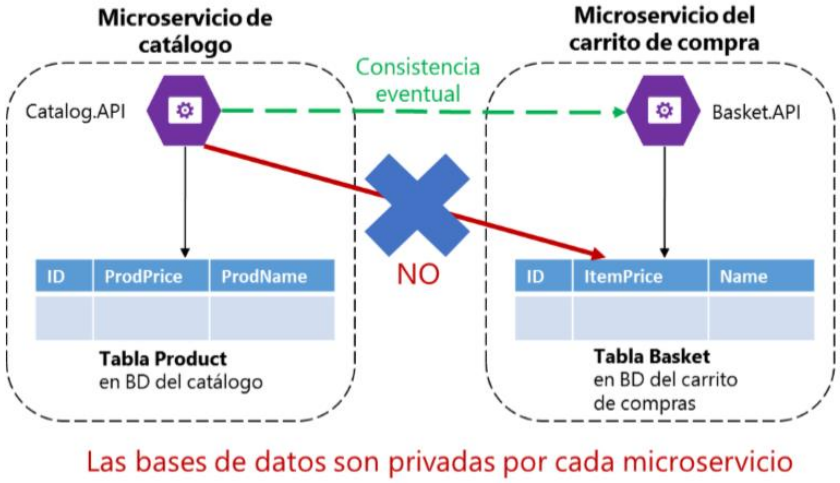
Reto #2: Cómo crear consultas que recuperan datos de varios microservicios

- Tomar en cuenta que esta base de datos centralizada, se usará sólo para consultas e informes que no necesitan datos en tiempo real. Las actualizaciones y transacciones originales, como su fuente original, tienen que estar en las bases de datos de los microservicios

- **API Gateway**
- **CQRS**
- **Datos fríos**

Reto #3: Cómo lograr consistencia entre múltiples microservicios

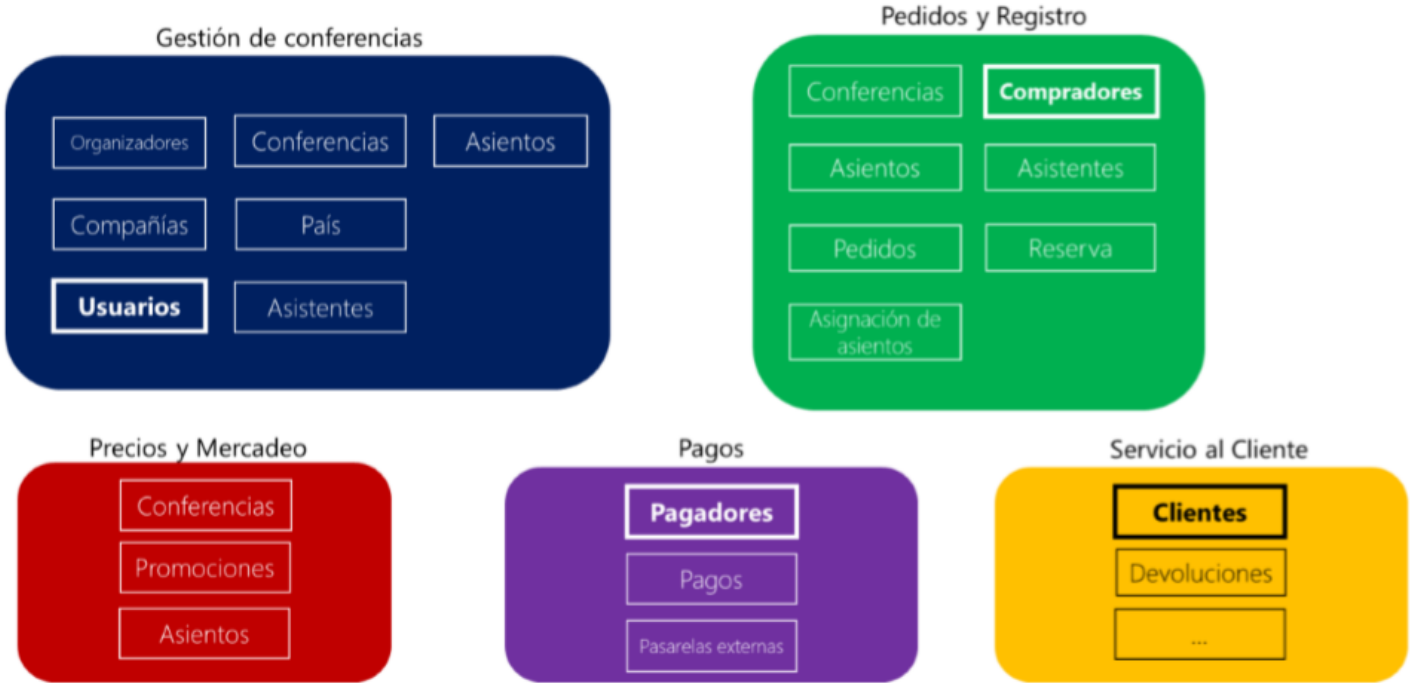
- los datos de cada microservicio deben ser privados y sólo se debe poder acceder a ellos usando el API del microservicio



Reto #4: Cómo diseñar comunicaciones a través de los límites de los microservicios

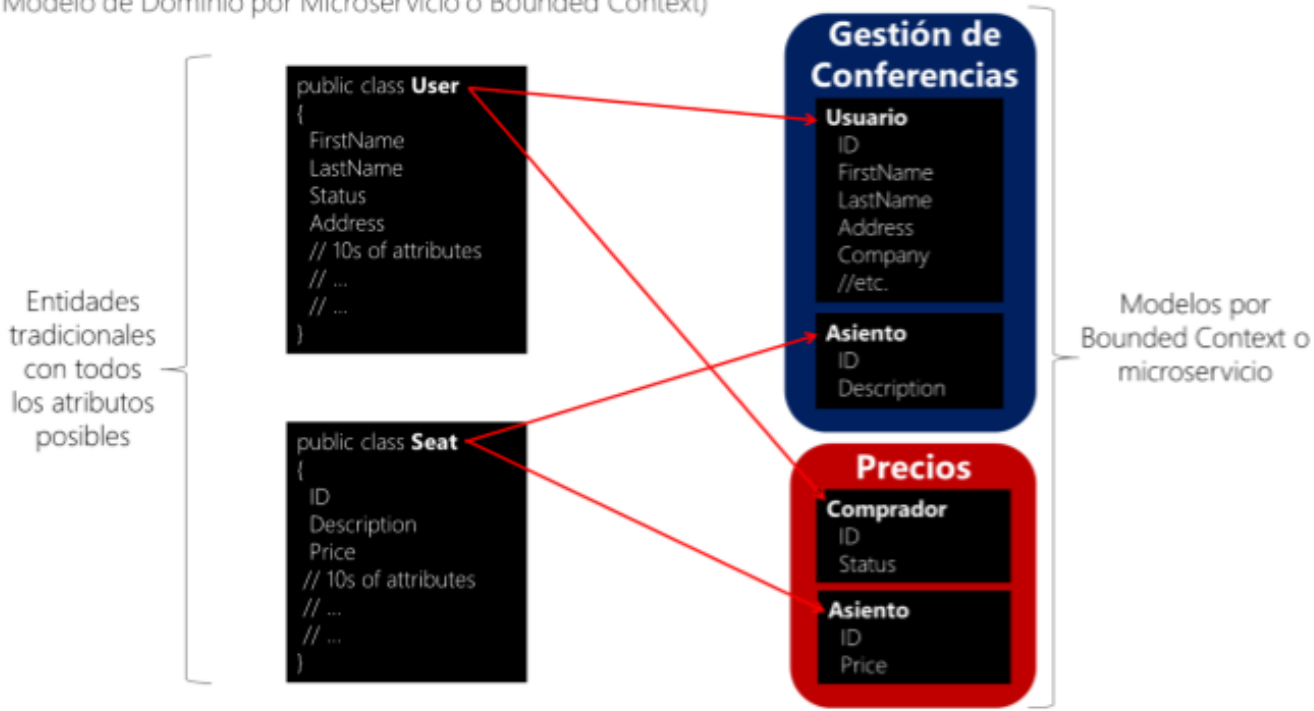
- La comunicación a través de los límites de los microservicios es un verdadero desafío
- La comunicación no se refiere a qué protocolo usar (HTTP y REST, AMQP, mensajes, etc.)
- Un enfoque popular es implementar microservicios basados en HTTP (REST), debido a su simplicidad

Identificando un modelo de dominio por microservicio o Bounded Context



Descomponiendo un modelo de datos tradicional en múltiples Modelos de Dominio

(Un Modelo de Dominio por Microservicio o Bounded Context)



Spring Cloud

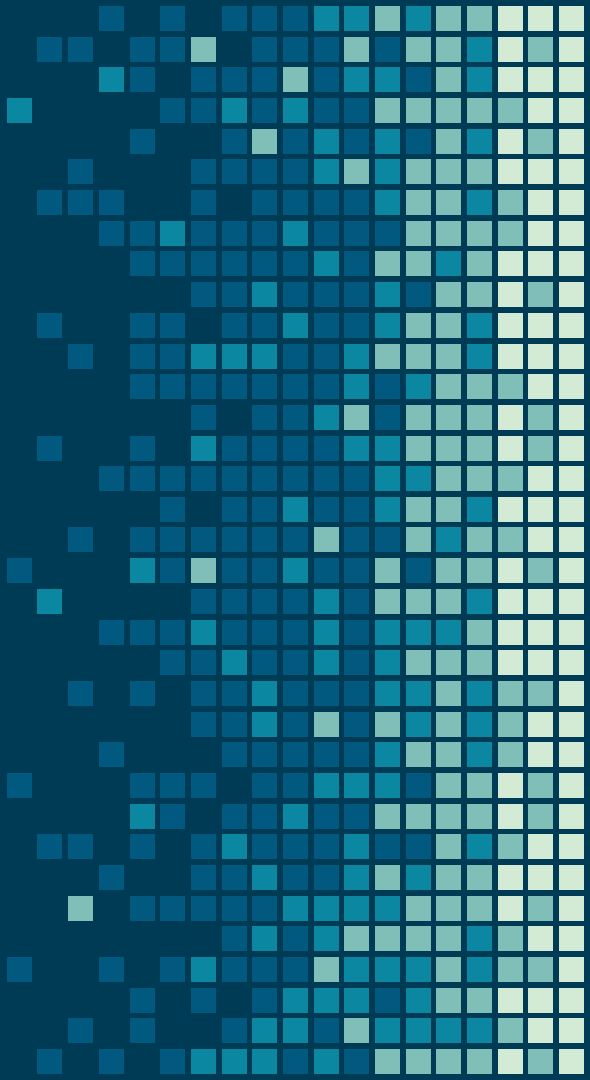
- Spring-cloud permite que las aplicaciones descubran su propia información/configuración en ejecución, además ofrece un mecanismo de extensión para tabajar en múltiples clouds y servicios Cloud.

- Los conceptos principales de Spring Cloud son:
- Cloud Connector: interfaz que un proveedor Cloud puede implementar para permitir que el sistema funcione al modo PaaS.

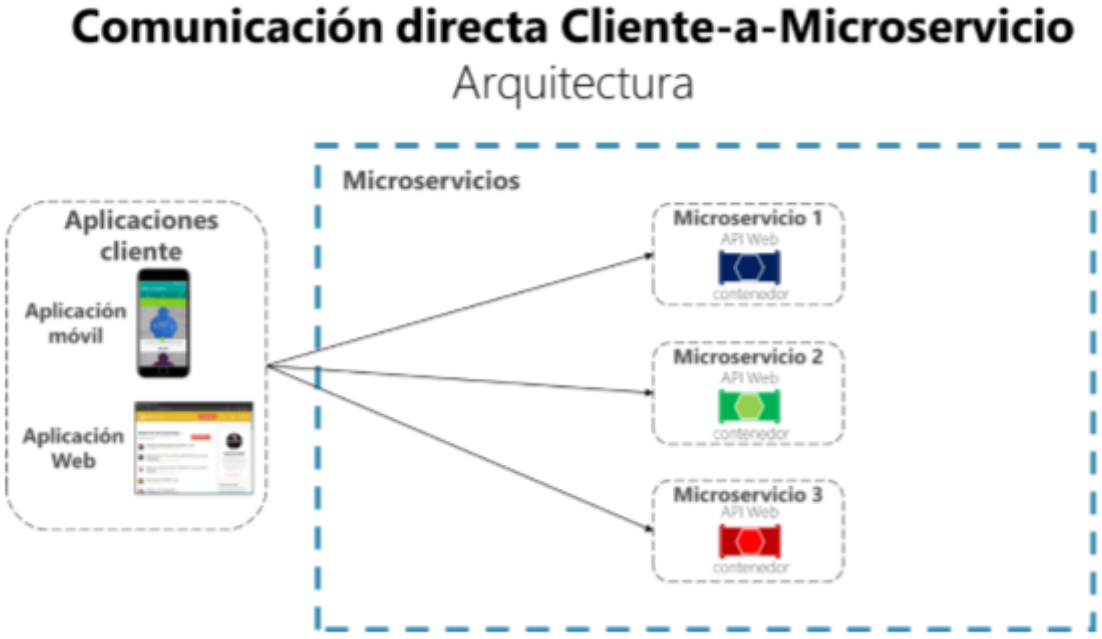
Spring Cloud

- Service Connector: un objeto (como javax.sql.DataSource) que representa una conexión a un servicio
- Service information: Información sobre el Servicio subyacente (host, Puerto, credenciales,…)
- Application information: Información sobre la aplicación e instancia en el que las librerías están embebidas.

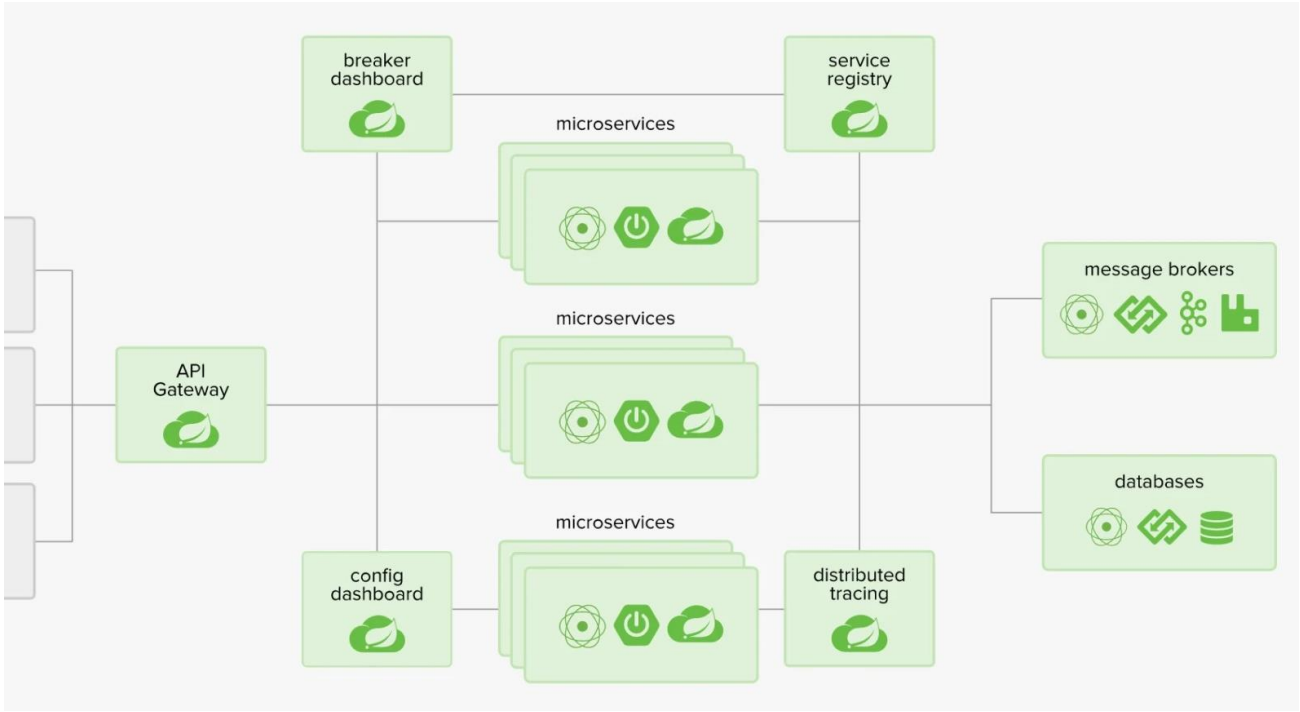
DEMO



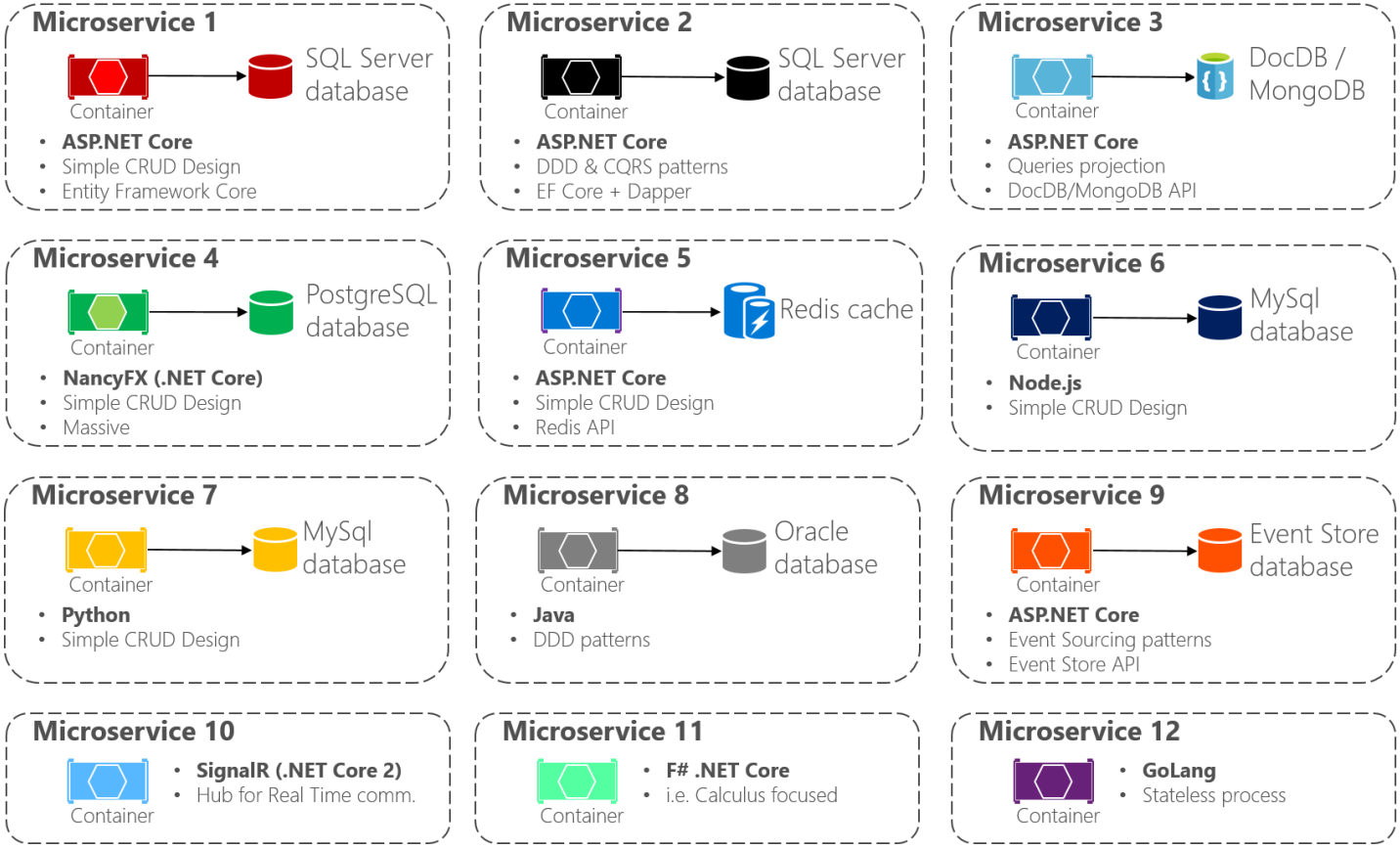
Enfoque de comunicación entre Microservicios



Enfoque de comunicación entre Microservicios



The Multi-Architectural-Patterns and polyglot microservices world



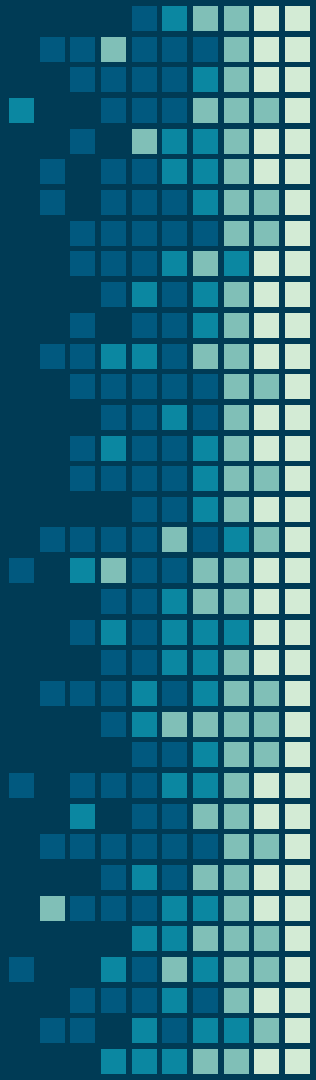
Enfoque de comunicación entre Microservicios

- Async pub/subs communication
  - Event Bus, RabbitMQ <- Mecanismos de cola y mensajería para manejar solicitudes.
- Health Check
  - Docker <- Sanidad del contenedor. Traces
- Resilent Cloud Applications ->
- API Gateways vs Direct Communication
- Orchestrators: Scale-out & dev Collaboration

GRACIAS!

¿Preguntas?

Puedes comunicarte conmigo:
egdeleon@pensotec.com
Github.com/YovaFree



TECH COMMUNITY DAY [2020]

All the things about tech:
By the community, for the community

JUNIO 12-13
9:00 - 19:00

Organizado por:

SQL SERVER USER GROUP • GUATEMALA

Microsoft Dev Group (Cobán)
Comparte | Aprende | Desarrolla

docker guatemala

kubernetes Guatemala CNCF Meetup

Conferencistas Internacionales Invitados:

NIGEL POULTON
Docker Captain

STEVE JONES
SQL Server Central

EDWIN SARMIENTO
Microsoft MVP

PABLO FREDRIKSON
Lead SRE, Youtuber

MANUEL PAIS
IT Consultant

GERMÁN KÜBER
.NET Software Architect

VITOR FAVA
Data Platform MVP

DEVLIN DULDULAO
Microsoft MVP

WARNER CHAVES
Pythian Consultant

LUIS SUAREZ
Front-End Developer

MARCELO ADADE
Microsoft MVP, Oracle OCP

SEBASTIÁN PÉREZ
Microsoft Azure MVP

Conferencias gratuitas sobre:

Nube e Infraestructura . Desarrollo . DevOps . Plataforma de Datos

Librerías y Productos API Gateway

- Ocelot (.NET)
- Amazon API Gateway: https://aws.amazon.com/es/api-gateway/
- Spring Cloud Gateway: https://spring.io/projects/spring-cloud-gateway
- Zuul: https://github.com/Netflix/zuul

Librerías y Productos para Service Discovery

- Eureka: https://spring.io/projects/spring-cloud-netflix
- Consul: https://www.consul.io/
- ZooKeeper: https://zookeeper.apache.org/
- Traefik: https://docs.traefik.io/providers/overview/

Referencias

https://martinfowler.com/articles/microservices.html
[Martin Fowler]

https://microservices.io/
[Chris Richardson]