CEU | Universidad San Pablo

# Introducción a los Microservicios

Sistemas Web II

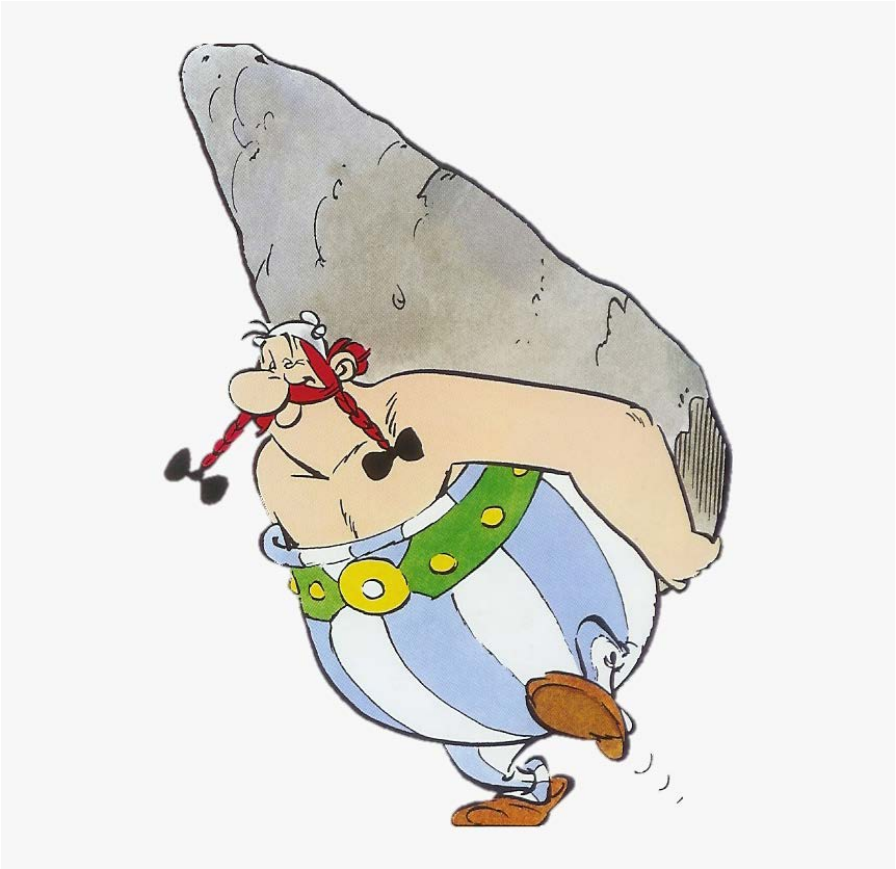
Grado en Ingeniería de Sistemas de Información
Álvaro Sánchez Picot
alvaro.sanchezpicot@ceu.es

v20230509

# ¿Qué son los microservicios?

# Conceptos

- Cohesión
  - Dependencia entre los elementos dentro de un módulo
  - "The code that changes together stays together"
  - Relacionado con el "Single Responsibility Principle":
    - "A module should be responsible to one, and only one, actor."
  - Mejor cuanto mayor sea
- Acoplamiento (coupling)
  - Dependencia entre varios módulos
  - Mejor cuanto menor sea



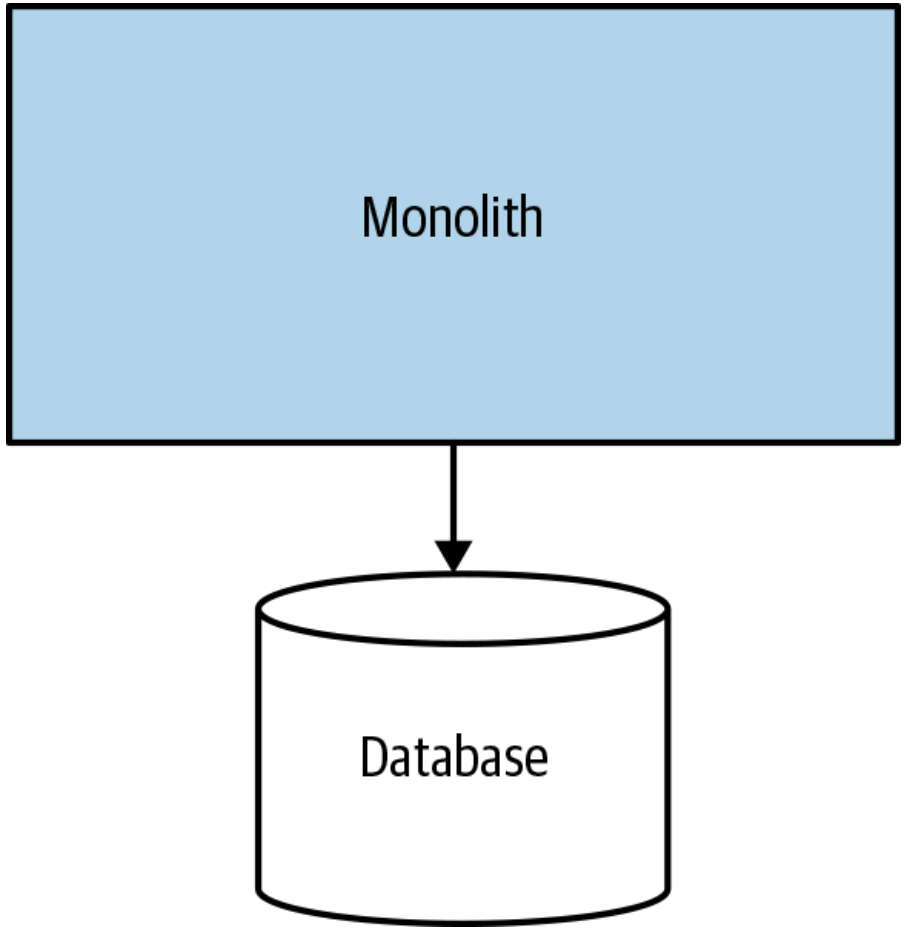
# APLICACIONES MONOLÍTICAS

# Aplicaciones monolíticas

- Una unidad de despliegue
- Comprende todas las funcionalidades del sistema
- Tipos:
  - Single-process system
  - Distributed monolith
  - Third-party black-box system

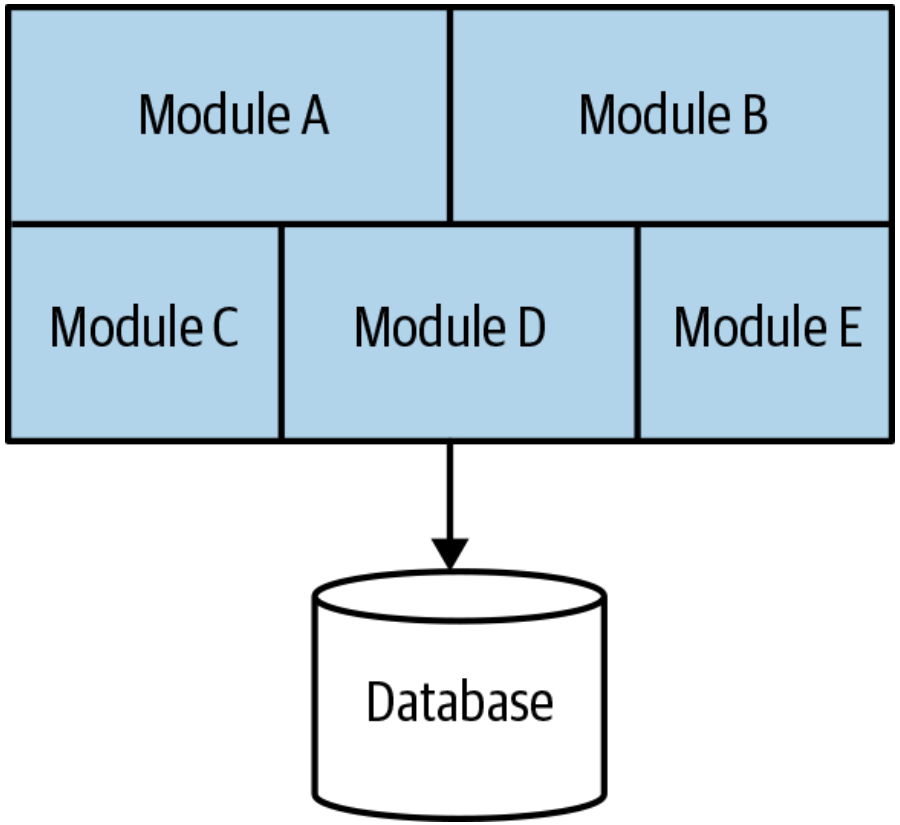
# Single process Monolith

- El código del sistema se despliega como un único proceso
- Puede haber múltiples instancias
  - Robustez
  - Escalado



# Single process Monolith – Modular

- Subset del single process monolith
- El único proceso consiste en módulos separados
  - Cada módulo trabaja de forma independiente
  - Todo el sistema tiene que desplegarse a la vez



# Distributed monolith

- Múltiples servicios
- Por algún motivo todo el sistema tiene que desplegarse junto
- Tiene las desventajas de ambos mundos
  - No suele tener suficientes ventajas

# Third-party black-box systems

- Software desarrollado por otras personas
  - No tienes la capacidad para cambiar el código
- Ejemplos:
  - Payroll system
  - Customer Relationship Management (CRM) system
  - Human Resource (HR) system

# Aplicaciones monolíticas – Desafíos

- Vulnerable a los problemas de acoplamiento
- Conflicto entre desarrolladores trabajando en lo mismo
- Problemas de acoplamiento en el despliegue
- Escalado
- Delivery contention
  - Problemas de pertenencia (ownership)
  - Problemas de toma de decisiones

# Aplicaciones monolíticas – Ventajas

- Topología de despliegue
- Flujo de desarrollo
- Monitorización
- Gestión de problemas
- End-to-end testing
- Reutilización de código

# Aplicaciones monolíticas

- Es una arquitectura más
- Muy apropiado en empresas pequeñas
- Ejemplos en empresas:
  - Shopify (vídeo, blog)

# MICROSERVICIOS

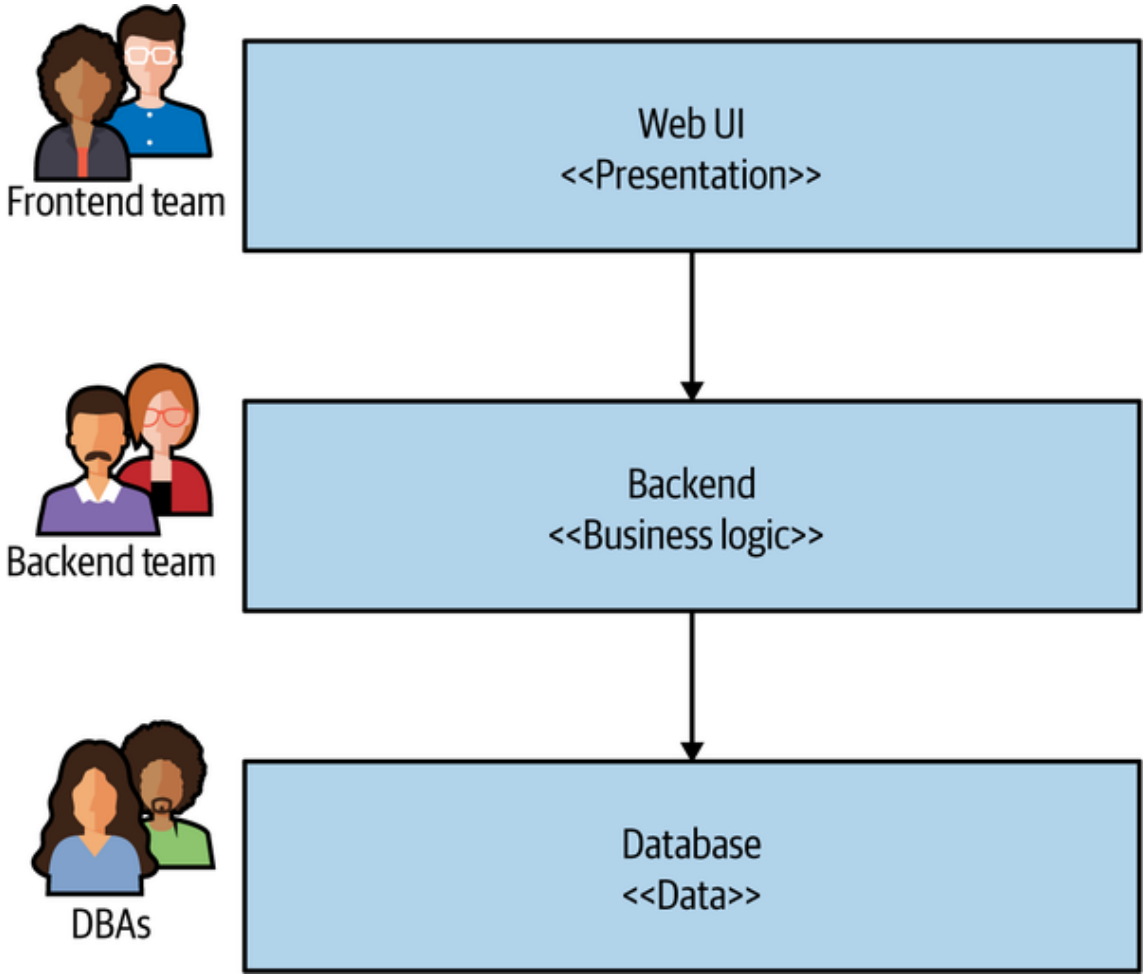
# Microservicios

- Un tipo de Service-oriented architecture (SOA)
- Independently releasable services (independent deployability)
  - Encapsulan funcionalidad
  - Accesible a otros
  - Technology agnostic
- Modelados en torno a un dominio de negocio
- Vistos desde el exterior son una caja negra
  - No se conocen los detalles de implementación

# Microservicios

- Information hiding
  - Ocultar tanta información como sea posible
  - Exponer lo menos posible a través de las interfaces
- Buscamos alta cohesión y bajo acoplamiento

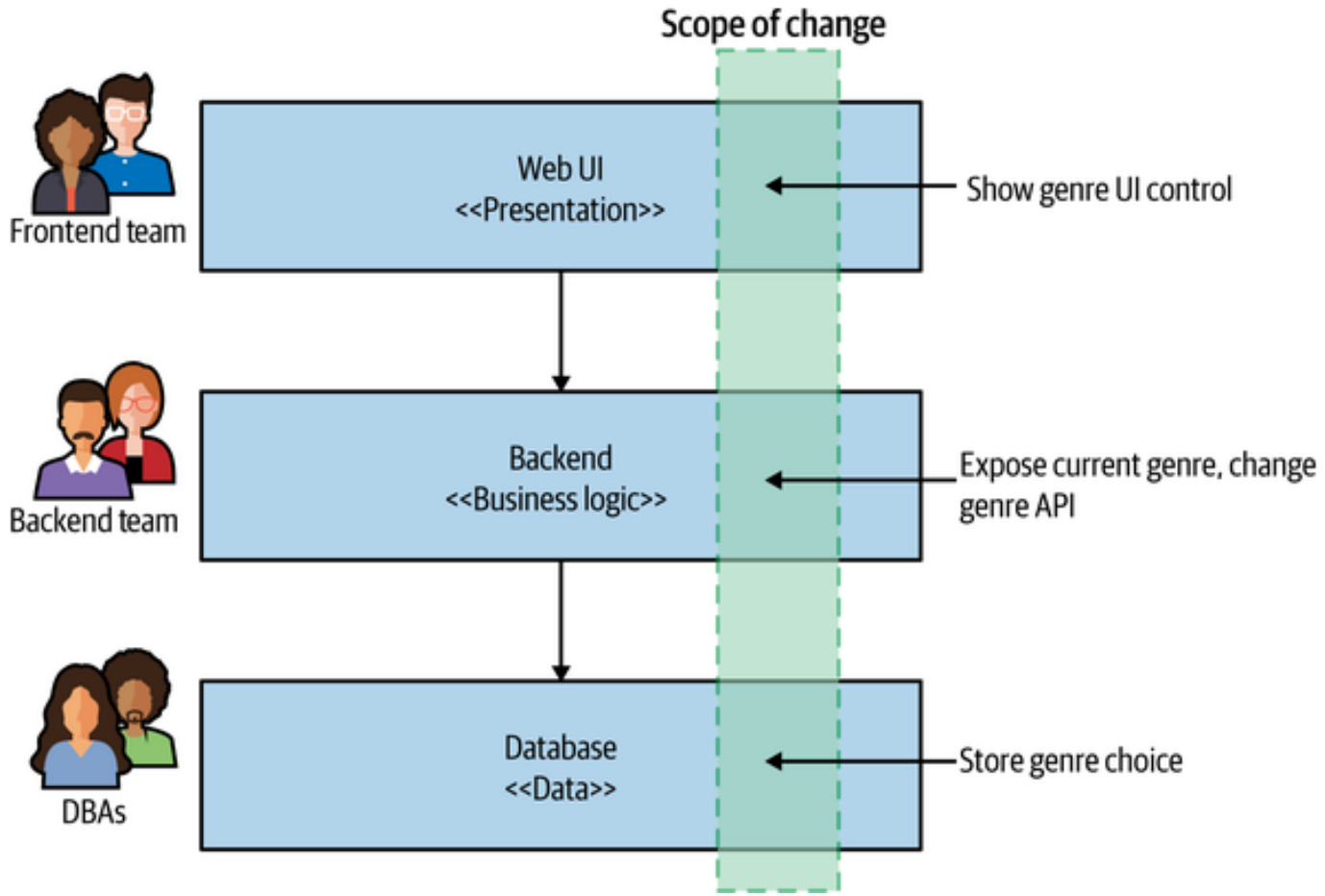
# Microservicios – Arquitectura tradicional



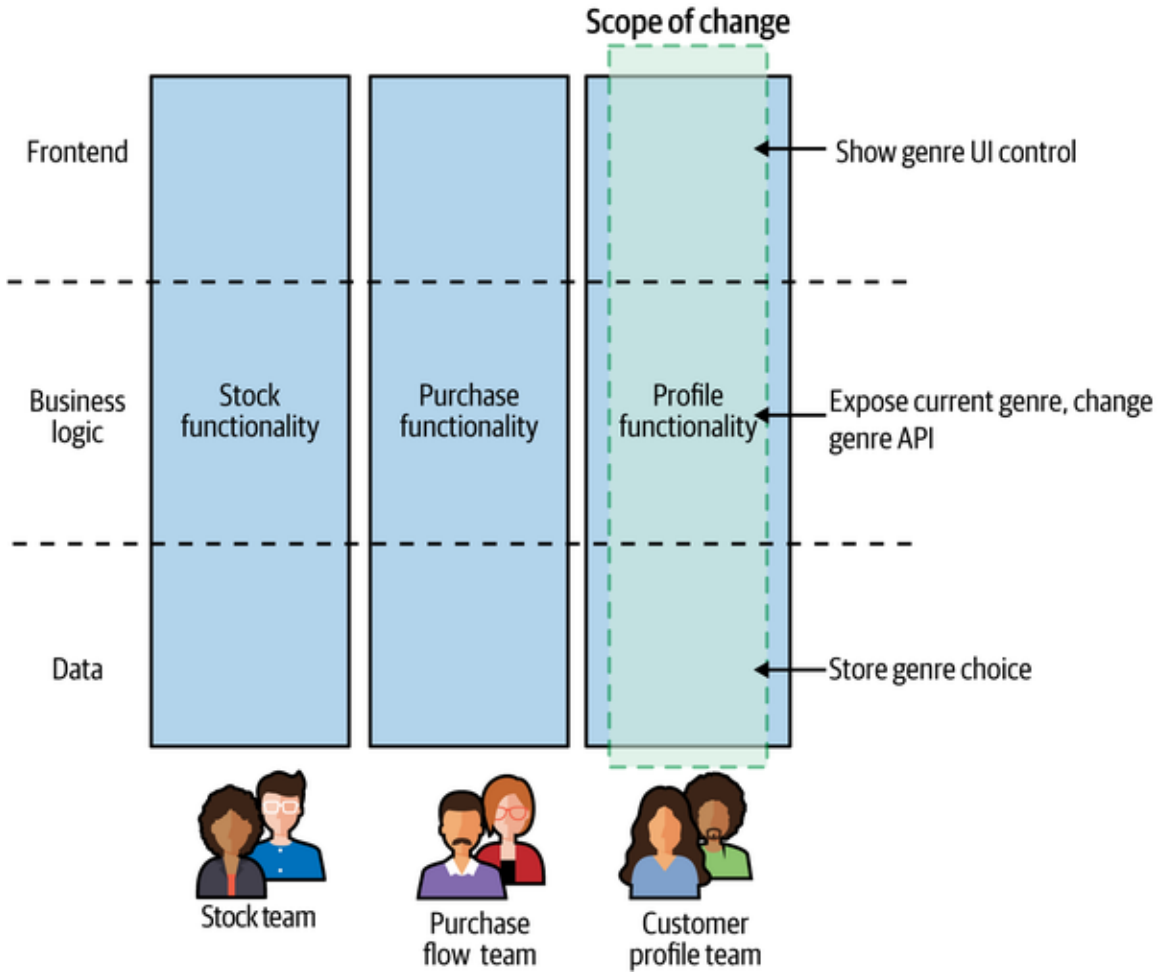
# Microservicios

- Tenemos una tienda de música online
- Queremos añadir una nueva funcionalidad:
  - El cliente puede especificar su género de música favorito

# Microservicios



# Microservicios



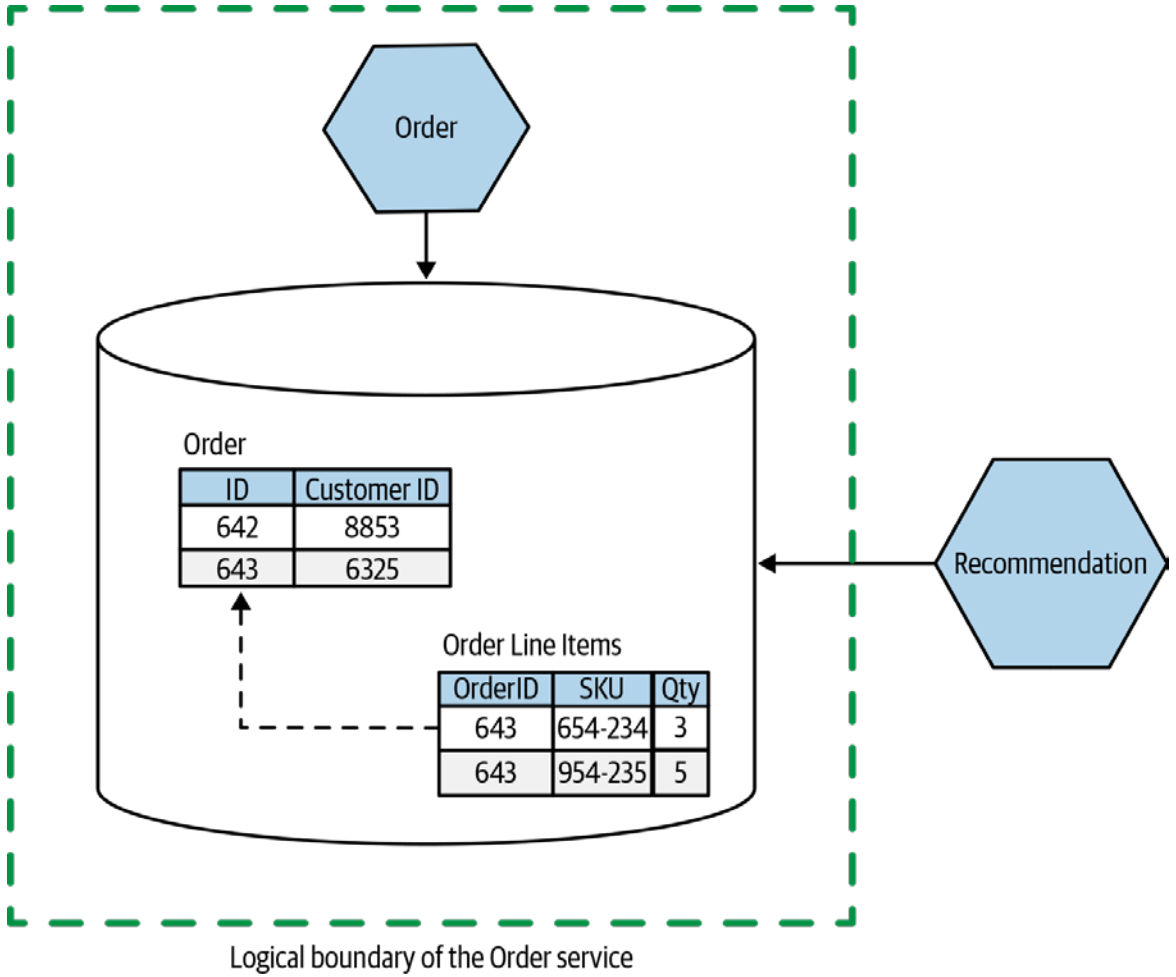
# Microservicios

- Tipos de acoplamiento:
  - Implementation coupling
  - Temporal coupling
  - Deployment coupling
  - Domain coupling

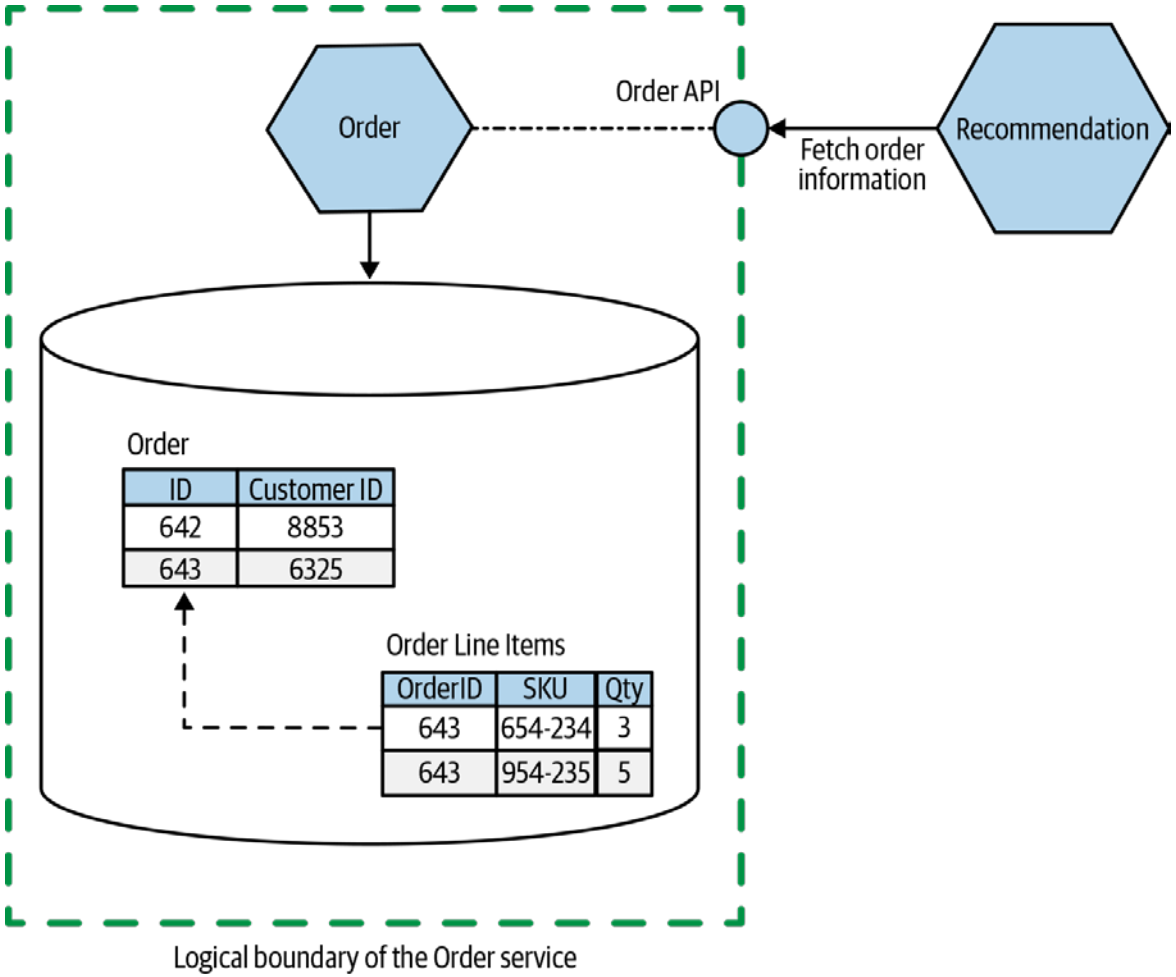
# Implementation coupling

- A acoplado a B → cambia la implementación de B, también cambia A
- El peor tipo de acoplamiento
- Fácil de solucionar
- Ej.: compartir una base de datos

# Implementation coupling

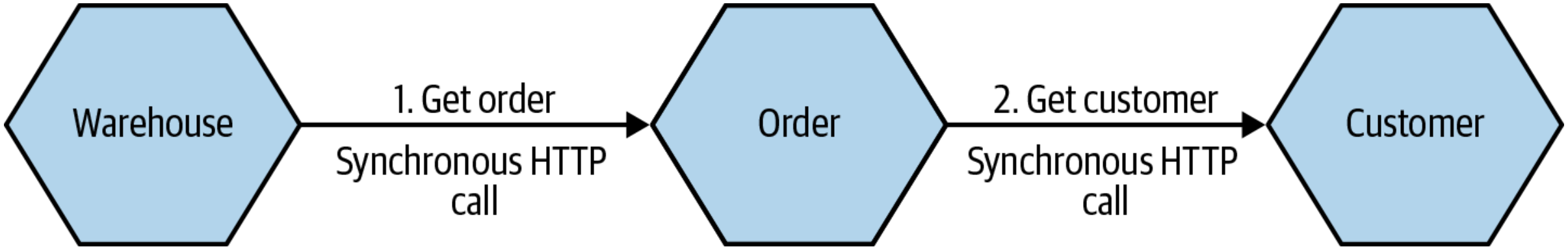


# Implementation coupling



# Temporal coupling

- Peticiones anidadas



# Temporal coupling

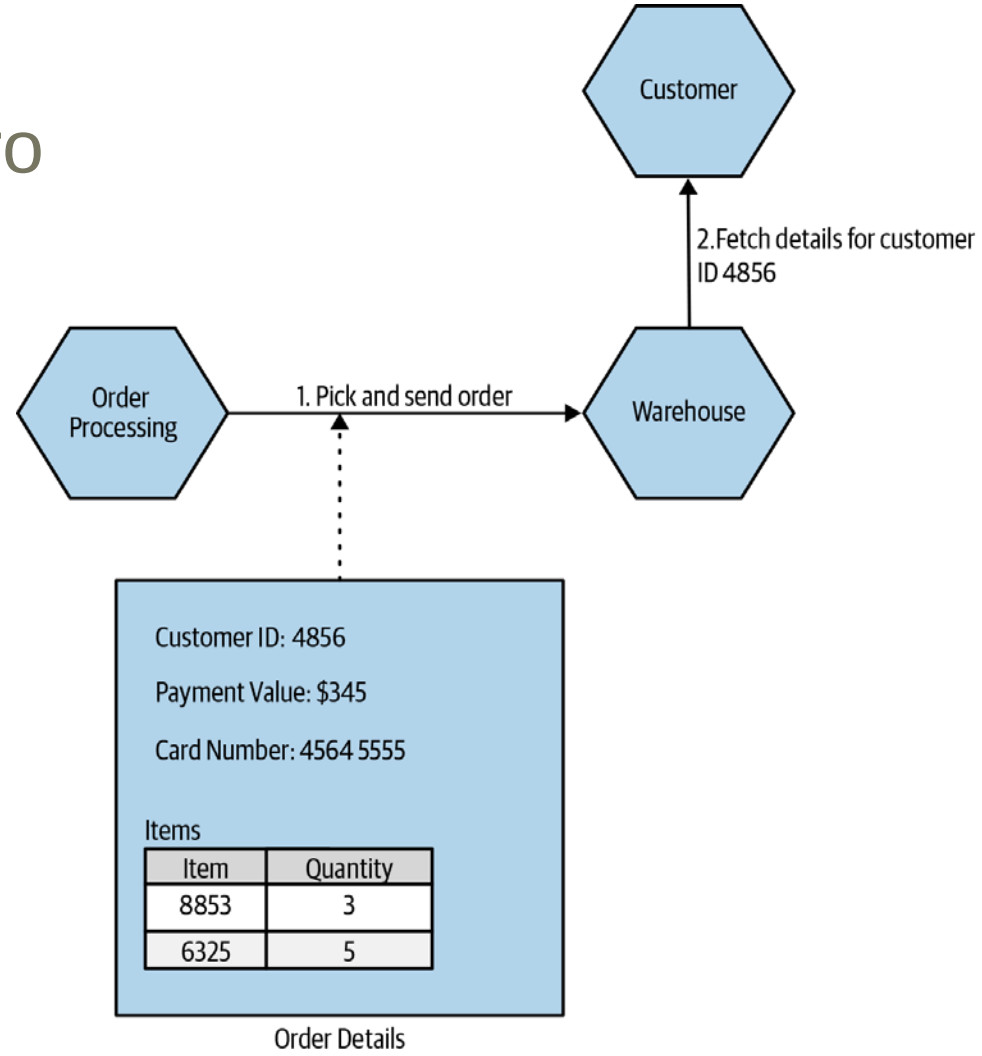
- Problemas:
  - Todos los servicios tienen que estar disponibles
- Soluciones:
  - Usar cacheado
  - Mensajes asíncronos

# Deployment coupling

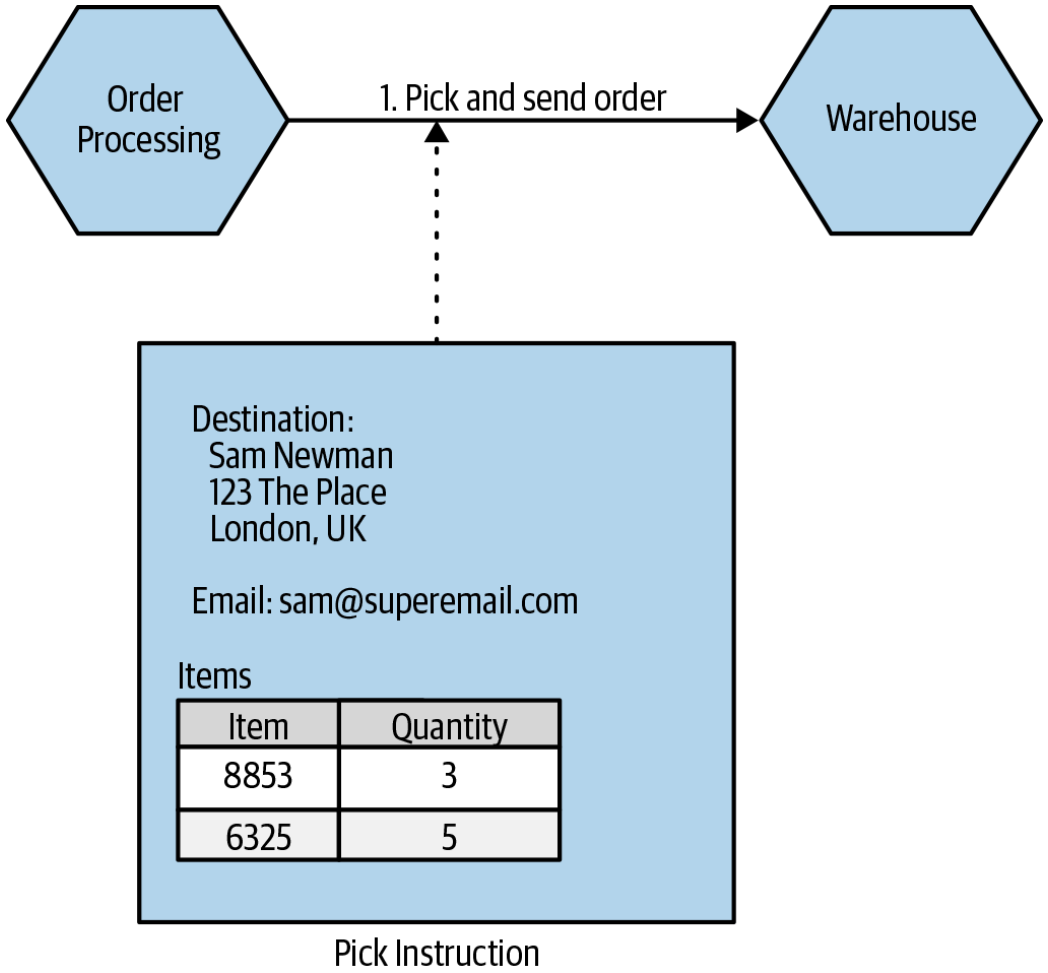
- Un pequeño cambio implica desplegar todo, aunque el resto no haya cambiado
- Soluciones:
  - Descomponer los procesos más grandes en microservicios independientes
- Ventajas:
  - En vez de despliegues fijos (release train), continuous delivery

# Domain coupling

- Interacción entre servicios en nuestro dominio
- En cierto modo inevitable
  - Se puede reducir la información compartida



# Microservicios



# Microservicios – ventajas

- Technology Heterogeneity
  - La tecnología apropiada para cada caso
- Robustness
  - Degradar la funcionalidad
- Scaling
- Ease of Deployment
  - Continuous delivery

# Microservicios – ventajas

- Organizational Alignment
  - Equipos más pequeños trabajando en código concreto
- Composability (componibilidad)
  - La funcionalidad se puede consumir para diferentes propósitos

# Microservicios - desventajas

- Developer Experience
  - ¿Cuántos microservicios puedo desplegar localmente?
- Technology Overload
- Cost
  - Más procesos, máquinas, red, almacenamiento…
- Reporting
  - La información está esparcida en múltiples sitios

# Microservicios - desventajas

- Monitoring and Troubleshooting
- Security
- Testing
- Latency
- Data Consistency
  - Bases de datos distribuidas

# Comunicaciones

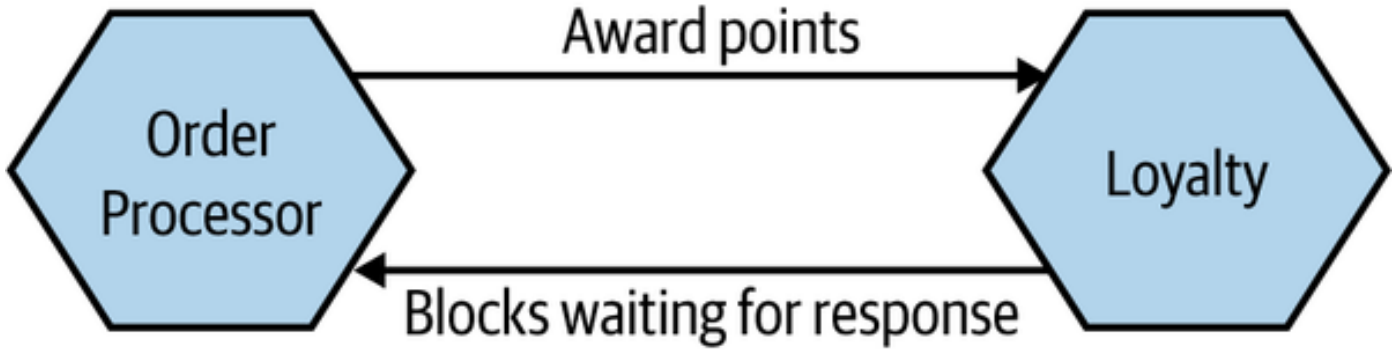
- From in-process to inter-process
  - Rendimiento:
    - No hay optimizaciones del compilador
    - Retardo
    - Cuidado con el número de llamadas
    - Overhead
  - Cambio de interfaces
  - Gestión de errores

# Comunicaciones

- Synchronous blocking
- Asynchronous nonblocking
- Request-response
- Event-driven
- Common data

# Synchronous blocking

- La llamada está bloqueada hasta que se complete
- Ventajas:
  - Familiaridad
- Desventajas:
  - Acoplamiento temporal

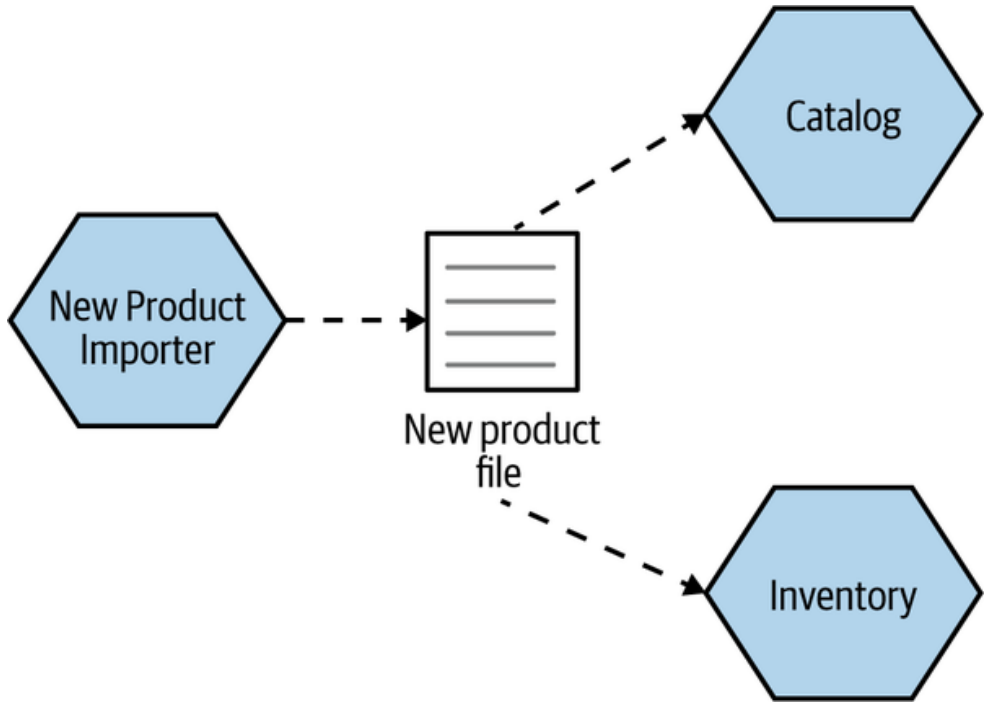


# Asynchronous Nonblocking

- La comunicación no bloquea el microservicio que hace la llamada
- Ventajas:
  - Desacoplo temporal
  - Útil para procesos que pueden llevar mucho tiempo
- Desventajas
  - Complejidad
  - Diferentes opciones

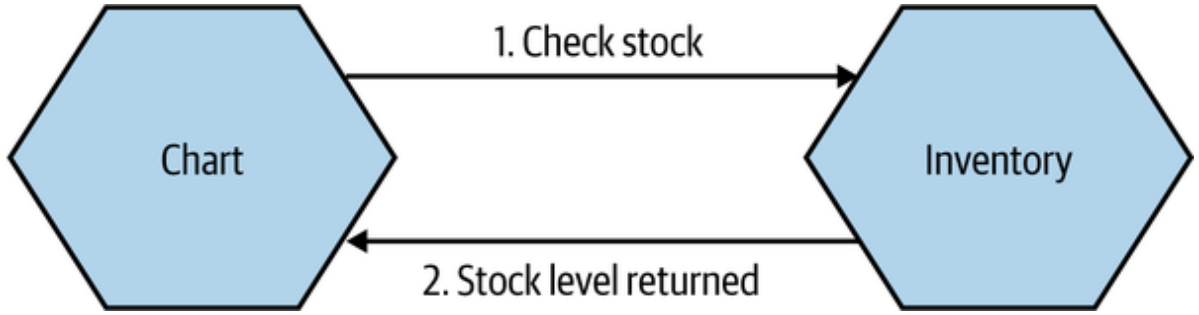
# Common data

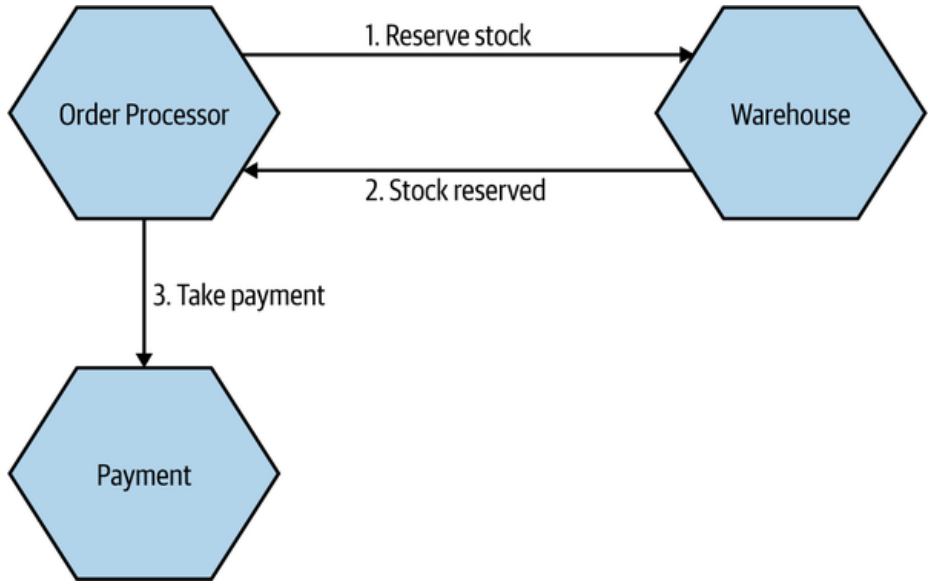
- Un microservicio pone la información en una localización
- El otro microservicio tiene que escanear la localización
- Asíncrono
- Lugares:
  - Fichero
  - Data lake
  - Data warehouse



# Request-Response

- Un microservicio envía una petición y espera una respuesta
  - Puede ser solicitud de información
  - Puede ser para realizar una acción
- Puede ser síncrono o asíncrono

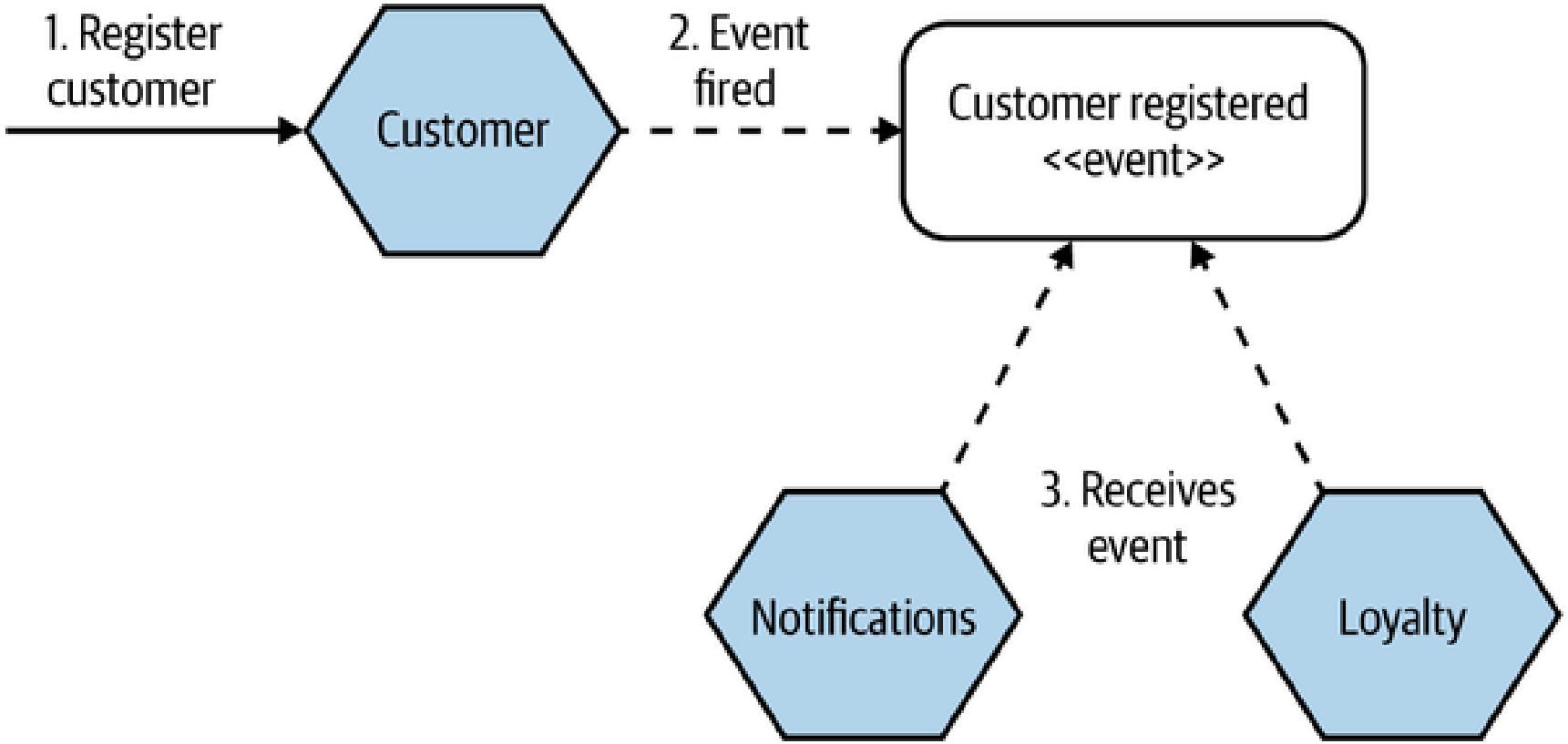


# Event driven

- Asíncrono
- Evento: afirmación de que algo ha sucedido
- Un microservicio emite eventos (productor)
  - Pueden o no ser escuchados por otros (cliente)
  - El productor no sabe lo que van a hacer los clientes con la información

# Event driven



# Asynchronous Nonblocking – Event driven

Implementación:

- Uso de message brokers (middleware) (RabbitMQ, Kafka)
- Los clientes se subscriben al bróker para recibir los eventos
- Los productores usan una API para publicar un evento en el bróker

# Microservicios – Tecnologías

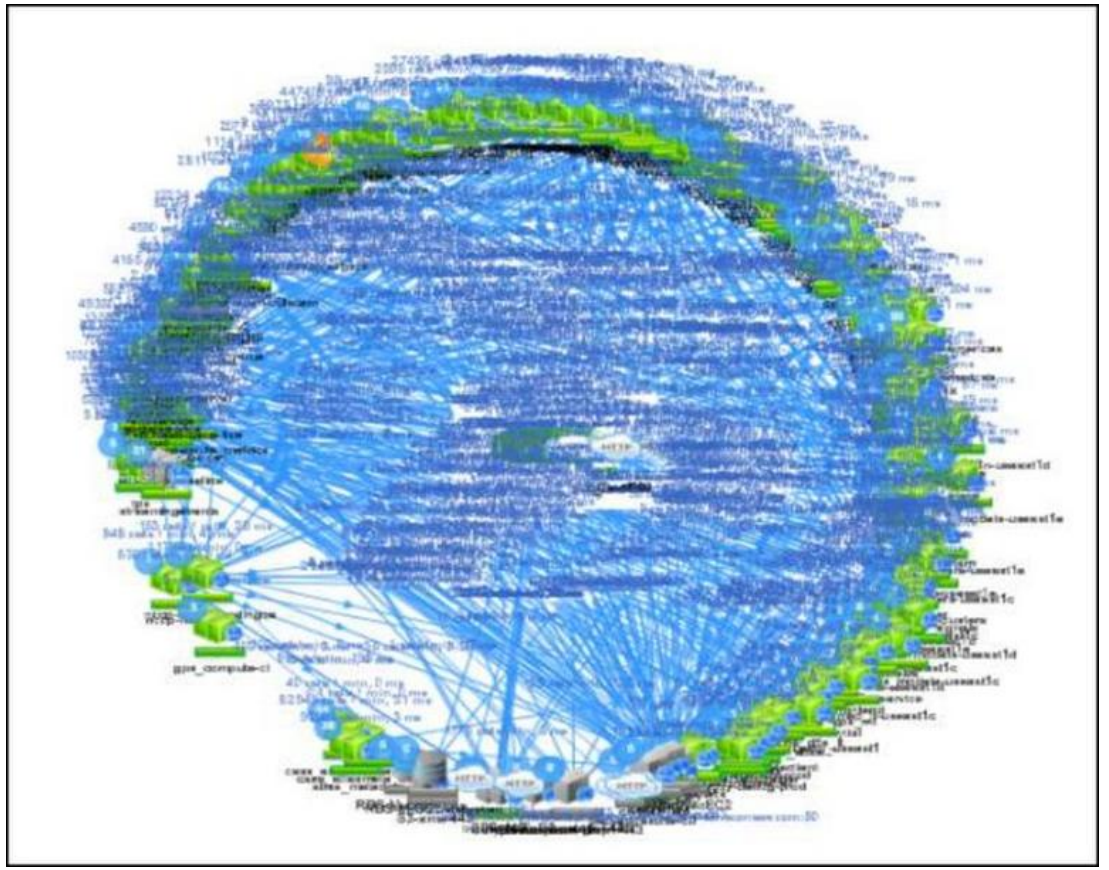
- Remote procedure calls (RPC)
  - SOAP
  - gRPC
- REST
- GraphQL
- Message brokers

# Microservicios – Despliegue

- Queremos cada microservicio desplegado como proceso
- Contenedores
  - Kubernetes
- Function as a Service (FaaS)
  - Serverless architecture
  - Cuando sucede esto ejecuta este código
  - Ej.: AWS Lambda

# Microservicios – Ejemplos

- Netflix
  - Más de mil microservicios
  - Desplegado en AWS
  - Netflix Tech Blog
- Ebay
  - Más de mil microservicios



# Microservicios – Problemas

- No es la solución a todo
- Es una arquitectura más
- Hay que considerar en cada caso si merece la pena
- A veces conviene volver a la arquitectura monolítica:
  - Scaling up the Prime Video audio/video monitoring service and reducing costs by 90%

# Referencias


- Introduction to microservices
- Building Microservices: Designing Fine-Grained Systems 2nd Edition, O'Reilly, 2021, ISBN: 978-1492034025
- Monolith to Microservices: Evolutionary Patterns to Transform Your Monolith, O'Reilly, 2019, ISBN: 978-1492047841