Arquitectura de Integración

Clase 4 y 5: Arquitectura y diseño de APIs

Alexander Chaparro Romero
alexander.chaparro@javerianacali.edu.co

Agenda

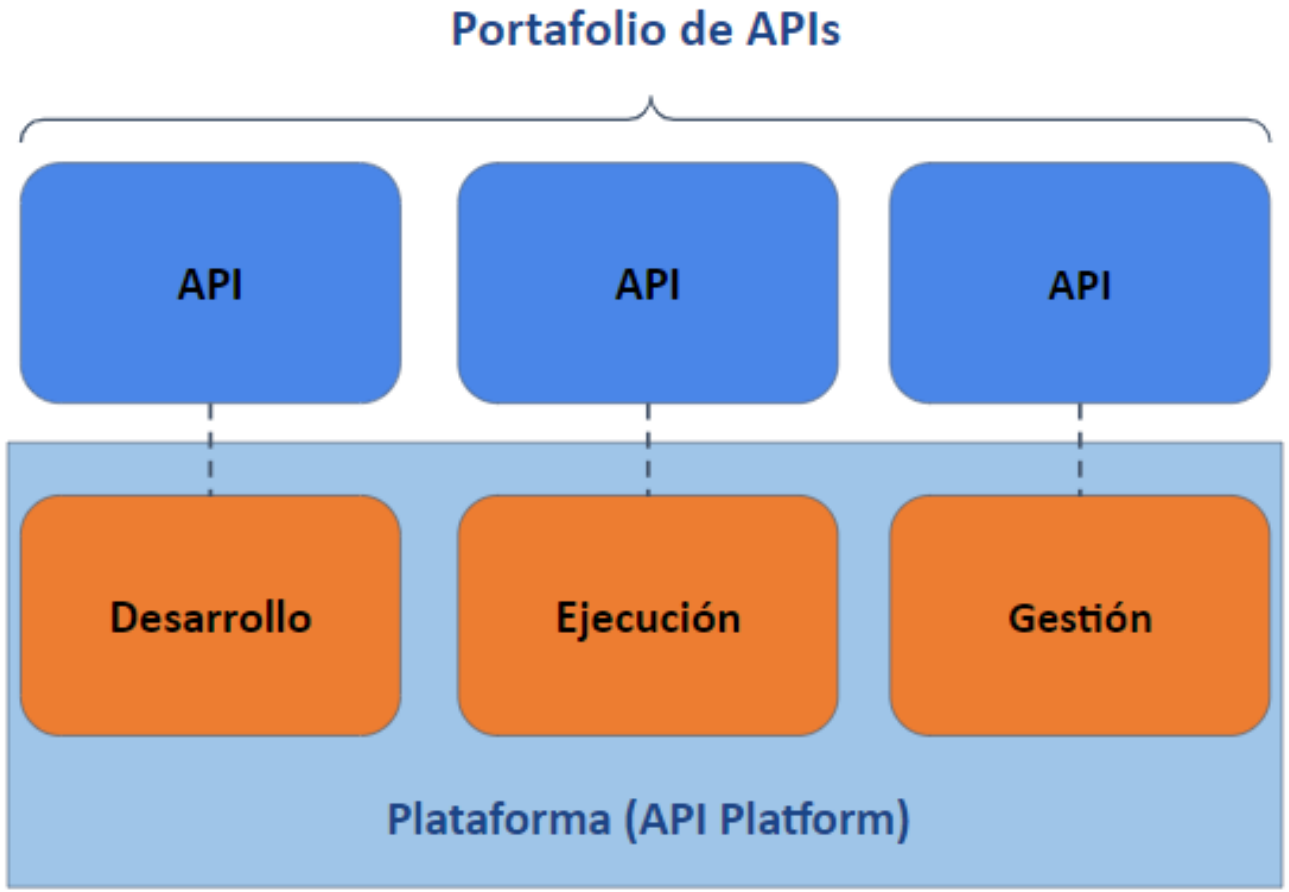
1. Arquitectura de APIs
   - Plataforma de desarrollo
   - Plataforma de despliegue
   - Plataforma de gestión
2. Portafolio de APIs
3. Patrones de arquitectura
4. Tipos de comunicación
   - Request reply
   - RPC
   - Basada en eventos
5. Topologías EDA
   - Broker
   - Mediator
   - Transferencia de estado
   - Colaboración
   - Streaming

Agenda

6. Domain Driven Design
    - Dominio del problema y de la solución
    - Diseño estratégico
    - Diseño táctico
7. Microservicios
    - Sistemas distribuidos
    - Mecanismos de comunicación
    - Arquitectura hexagonal
8. Patrones
    - Database per service
    - API Gateway / Backends for Frontends
    - API Composition

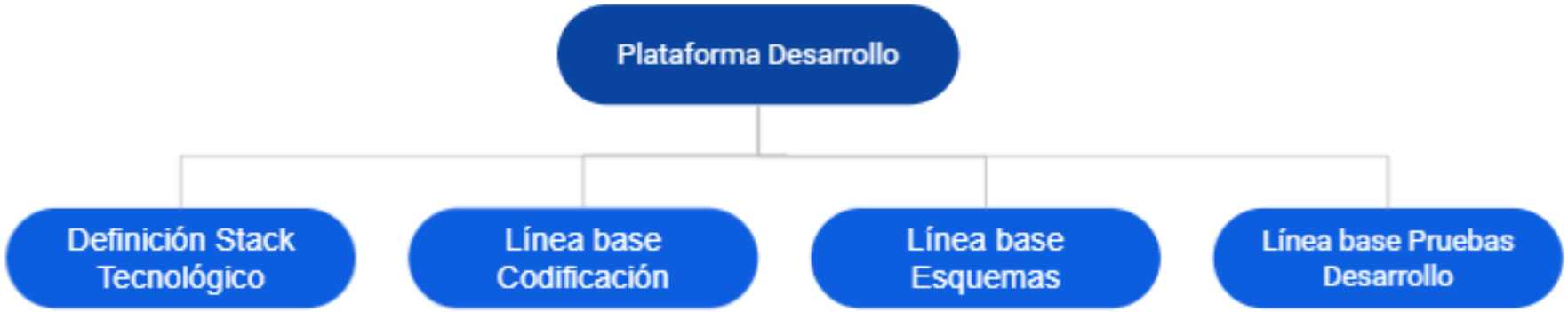
1. Arquitectura de APIs

Contexto general

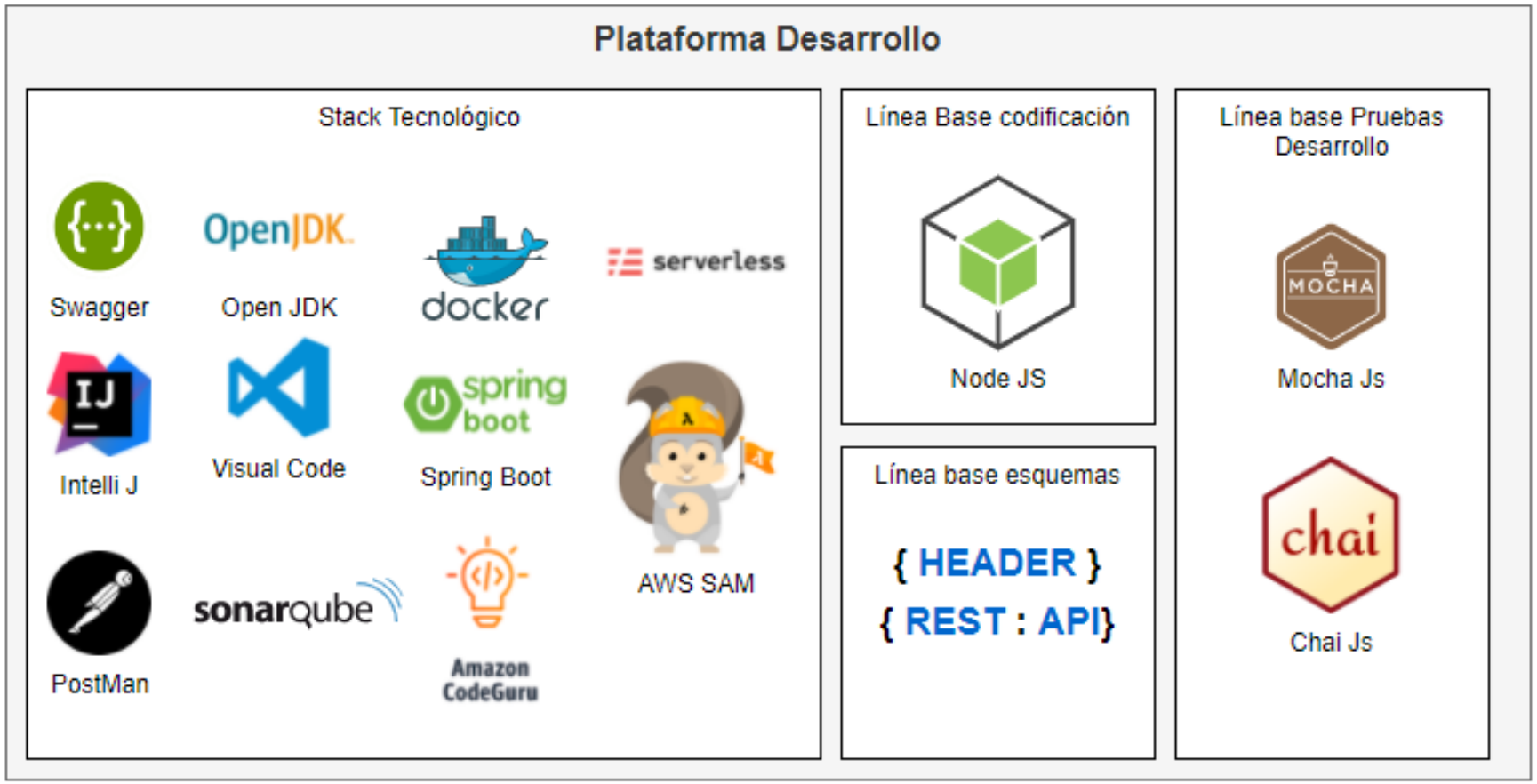


Plataforma de desarrollo

La plataforma de desarrollo es uno de los tres (3) pilares establecidos dentro de la arquitectura que busca facilitar la construcción de APIs, haciendo énfasis en aspectos como la rapidez y la calidad. Brindando todas las herramientas necesarias para el diseño y desarrollo de las APIs.



Plataforma de desarrollo – Footprint



Plataforma de despliegue

*La plataforma de despliegue proporciona la **estructura**, los **patrones**, las **buenas prácticas** y **lineamientos** para desarrollar e implementar los servicios, microservicio y APIs en el marco de la **arquitectura de APIs**.*

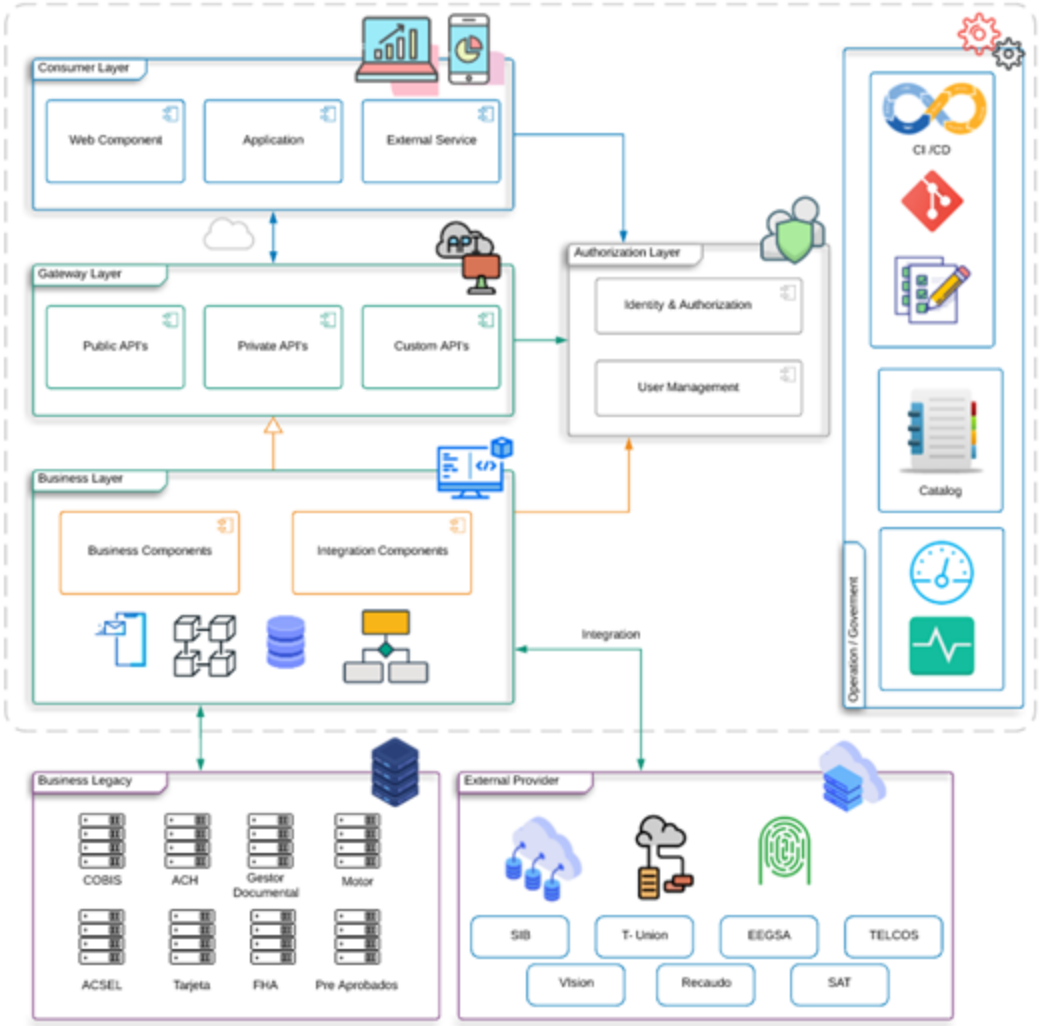
Los dominios de la arquitectura contemplados en este pilar son:

- *Consumidores*
- *Proveedores*
- *Mediación*
- *Eventos síncronos*
- *Eventos asíncronos*
- *Monitoreo*
- *Auditoría*
- *Seguridad*
- *Api Gateway*

Plataforma de despliegue - Contexto

En la arquitectura tiene como objetivos:

- ***Contar con un lenguaje común:*** *Permite a todas las partes implicadas comunicarse claramente y sin ambigüedades.*
- ***Mantener la uniformidad:*** *Proporciona una implementación homogénea de la tecnología en toda la empresa.*
- ***Establecer una estandarización:*** *Motiva el cumplimiento de especificaciones y patrones comunes para abordar cualquier decisión técnica.*



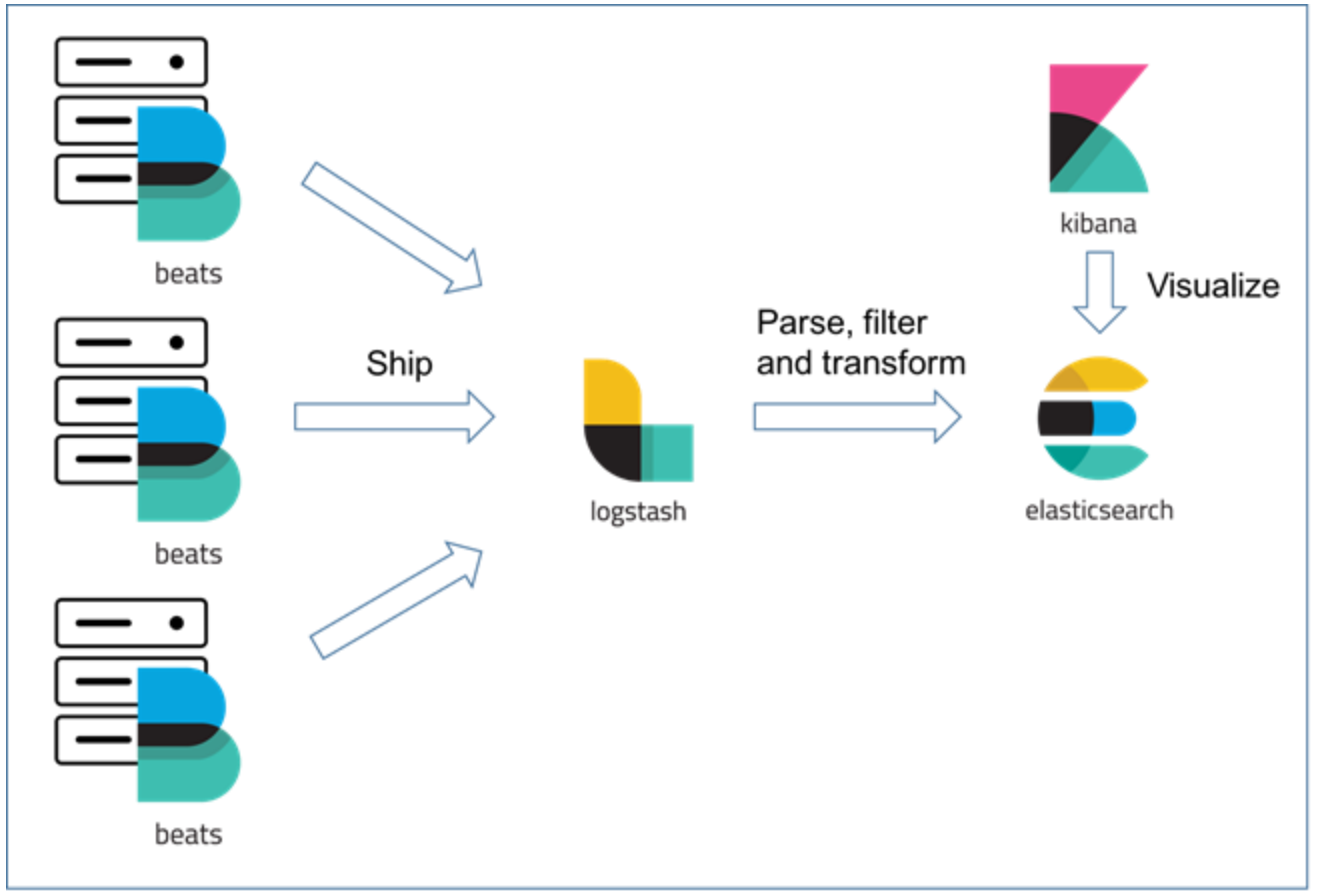
Plataforma de gestión

La plataforma de gestión permite que se pueda realizar monitoreo preventivo para apalancar la continuidad del negocio y tener visibilidad del consumo de cada una de las APIs.

Algunos tableros comunes son :

- *Errores de APIs VS Respuestas correctas*
- *Tiempos de respuesta de APIs y sus agregados*
- *Tiempos de respuesta bases de datos*
- *Tiempos de respuesta Servicios Externos (Core , entre otros)*
- *Uso de recursos (CPU, Memoria, Red)*

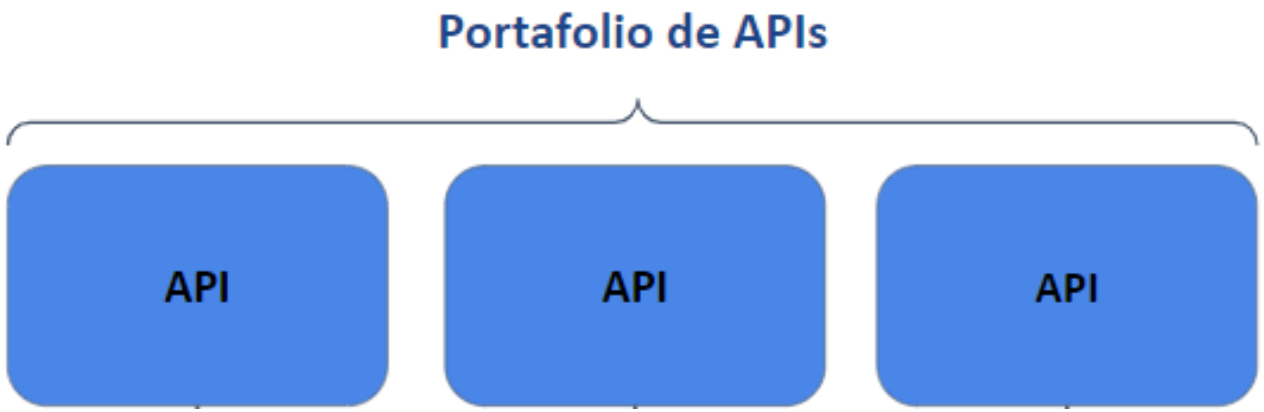
Plataforma de gestión - Tecnologías



[1] Imagen tomada de: https://github.com/davidsugianto/elastic-logs-kibana

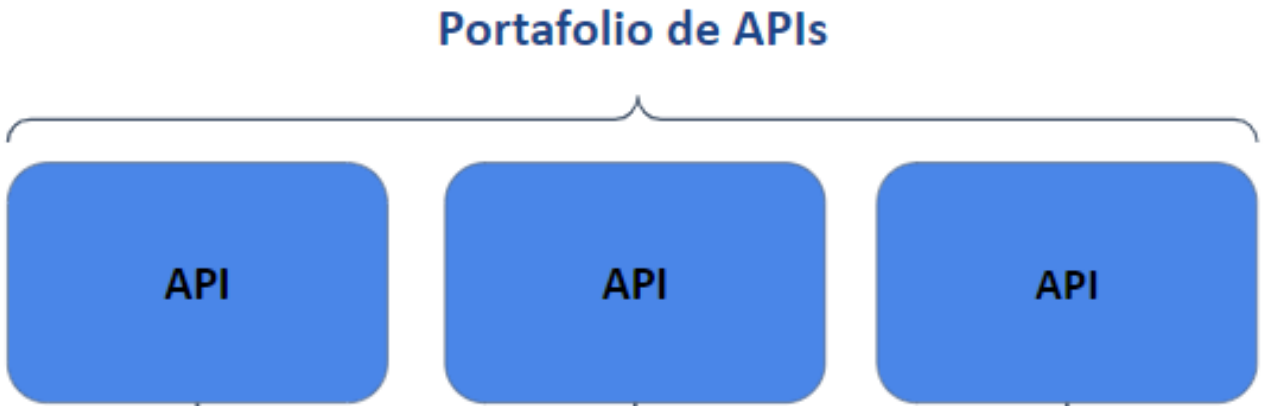
2. Portafolio de APIs

Portafolio de APIs



Un portafolio de APIs debe ser diseñado de forma que todas las APIs del portafolio sean consistentes unas con otras, reutilizables, descubribles y personalizables.

Portafolio de APIs



El portafolio debe cumplir con requisitos funcionales y no funcionales:

- Consistencia
- Reuso
- Personalización
- Descubrimiento
- Longevidad

Portafolio de APIs

Consistencia:

Un cliente o una solución seguramente usarán varias APIs del portafolio. Es deseable que la salida de un API pueda ser usada como entrada en otra API.

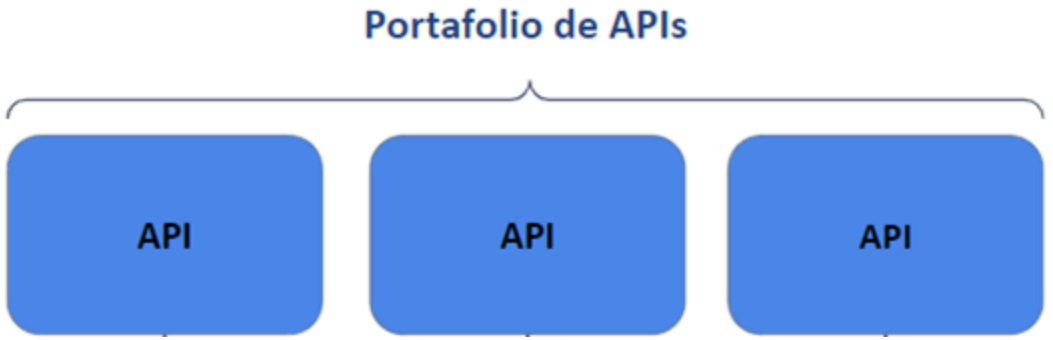
- La URI de las APIs debe seguir una estructura común e intuitiva.
- Nombres de campos y formatos deben ser consistentes a lo largo del portafolio.
- Validaciones a las entradas deben hacerse de forma similar en todo el portafolio.
- Los mecanismos de seguridad deben ser los mismos en todo el portafolio.

Portafolio de APIs

Reuso:

De ser posible las APIs no debe ser construidas para un cliente específico. Se espera que su diseño e implementación sirvan a múltiples consumidores, al menos parcialmente.

- Reuso de algunas características de las APIS
- Reuso de APIs completas
- Reuso de APIs propias
- Reuso de APIs de terceros



Portafolio de APIs

Personalizable:

Si bien las APIs no deberían ser diseñadas para un cliente específicamente, si deben poder ser configurables y personalizables para la necesidad particular un cliente o grupo de clientes.

Aspectos a personalizar:

- Formateo de datos
- Entrega de datos
- Agrupamiento de datos

Portafolio de APIs

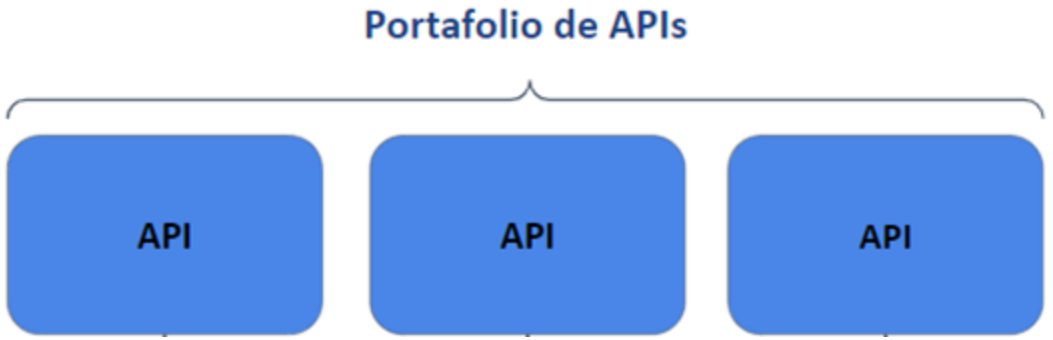
Longevidad:

Se debe buscar que el API sea funcional por la mayor cantidad de tiempo posible.

Desde el punto de vista técnico, se espera que las APIs evolucionen constantemente, pero las firmas de las operaciones deben permanecer estables tanto como sea posible.

Manejo del cambio

- Compatibilidad hacia atrás
- Compatibilidad hacia adelante



Portafolio de APIs

Gobernable:

Se debe buscar equilibrio:

- Desarrollo de APIs ágil que permita probar fácilmente nuevas iniciativas de negocio
- Reglas y políticas firmes para evitar el desorden en el portafolio de APIs

Puede manejar:

- Portafolio laboratorio: medidas más suaves que faciliten experimentar.
- Portafolio Producción: APIs controladas

3. Patrones de arquitectura

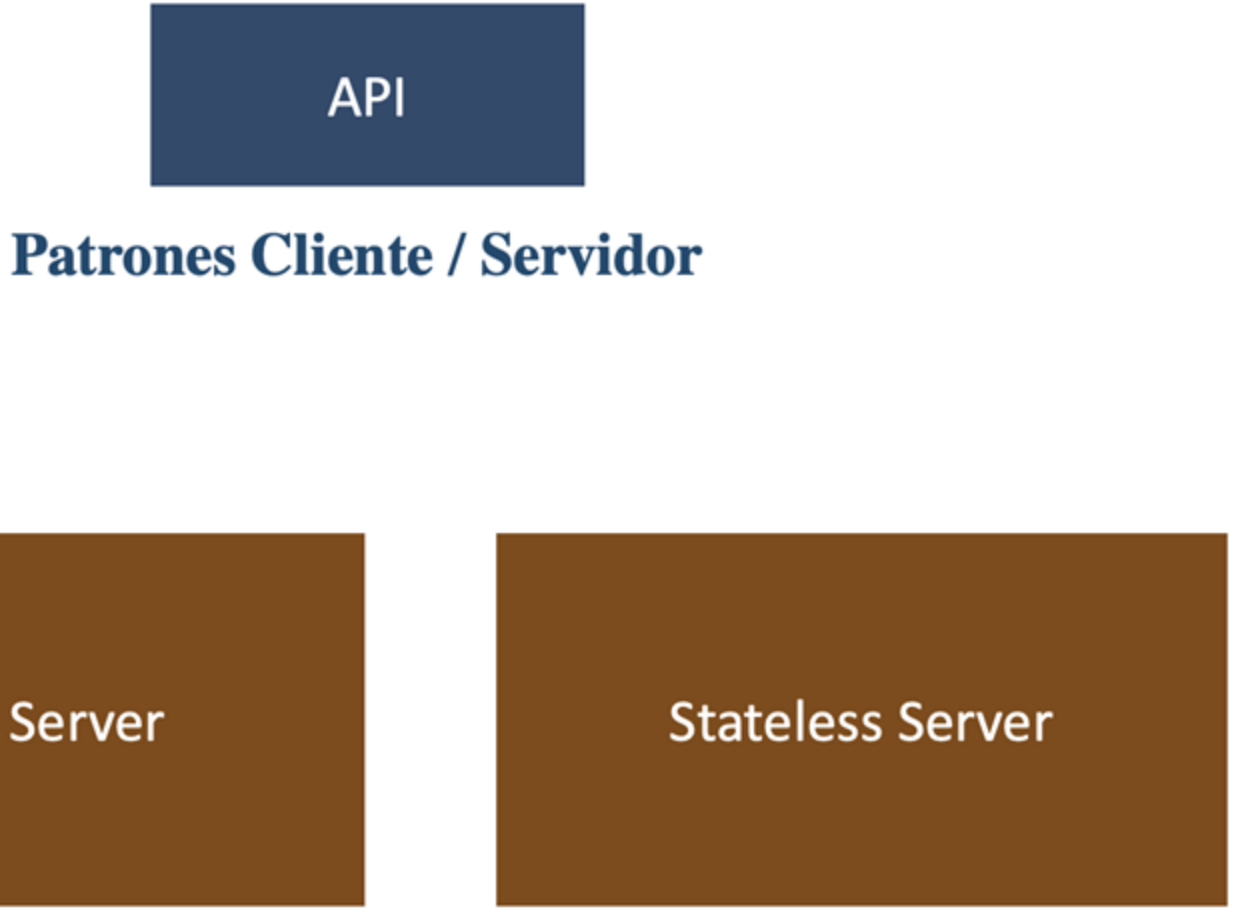
Responsabilidades principales de las APIs

- Agrupación / Recolección de datos
- Estructuración y formateo de datos
- Entrega de datos
- Proveer protección y seguridad a los datos

Responsabilidades deseables de las APIs

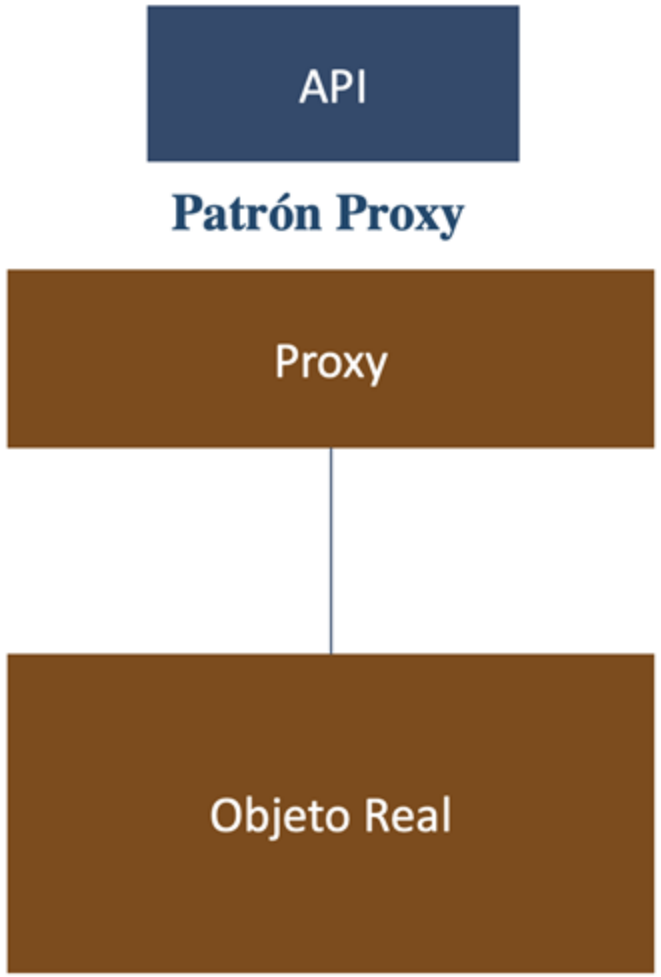
- Centrada en los consumidores
- Simples
- Auto-explicables, Intuitivas y predecibles
- Explorables y descubribles
- Bien documentadas
- Atómicas
- Buen manejo del error
- Con buen desempeño, escalables y disponibles
- Interoperables
- Reusables
- Compatibles hacia atrás

Patrón Cliente / Servidor



Patrón fachada

Patrón Proxy



4. Tipos de comunicación

Request-Reply

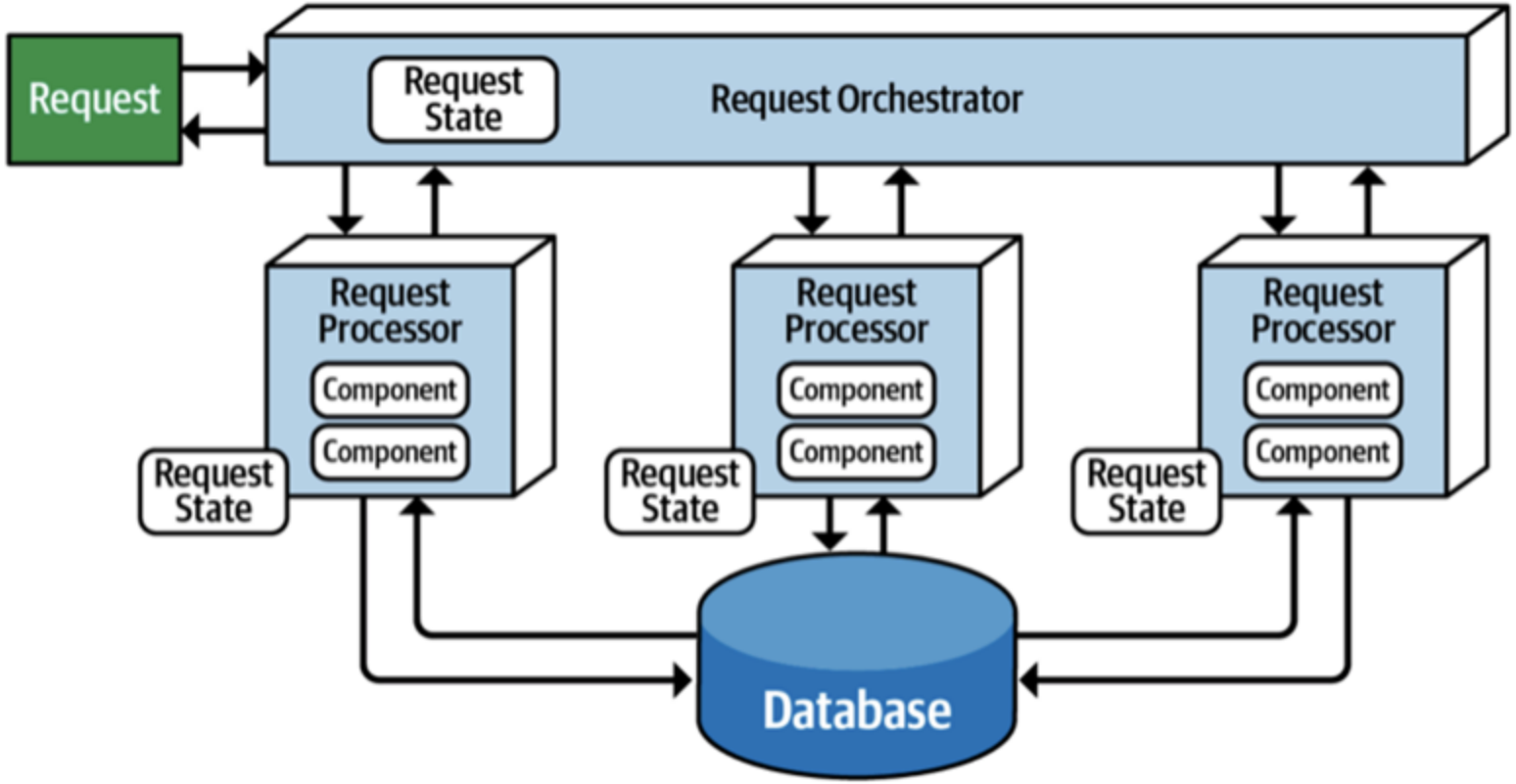
Ventajas:

- Velocidad de respuesta
- Facilidad de pruebas
- Facilidad en el diseño

Desventajas:

- Acoplamiento
- Disponibilidad

Request-Reply



[2] Imagen tomada de: Fundamentals of software Architecture. Neil Ford y Mark Richards. O'Reilly, 2020

Comunicación RPC

- Acciones que pueden cambiar el estado del sistema
- Típicamente se ejecutan de forma sincrónica
- Operaciones que requieren ser ejecutadas y retornar una respuesta
- Se sugiere ejecutar dentro de un bounded context

Comunicación basada en eventos

Eventos:

- Son hechos y notificaciones.
- Representan algo que sucedió pero no necesariamente implican una acción futura.
- Unidireccionales y no requieren una respuesta.
- Mecanismo de comunicación bajo en acoplamiento.

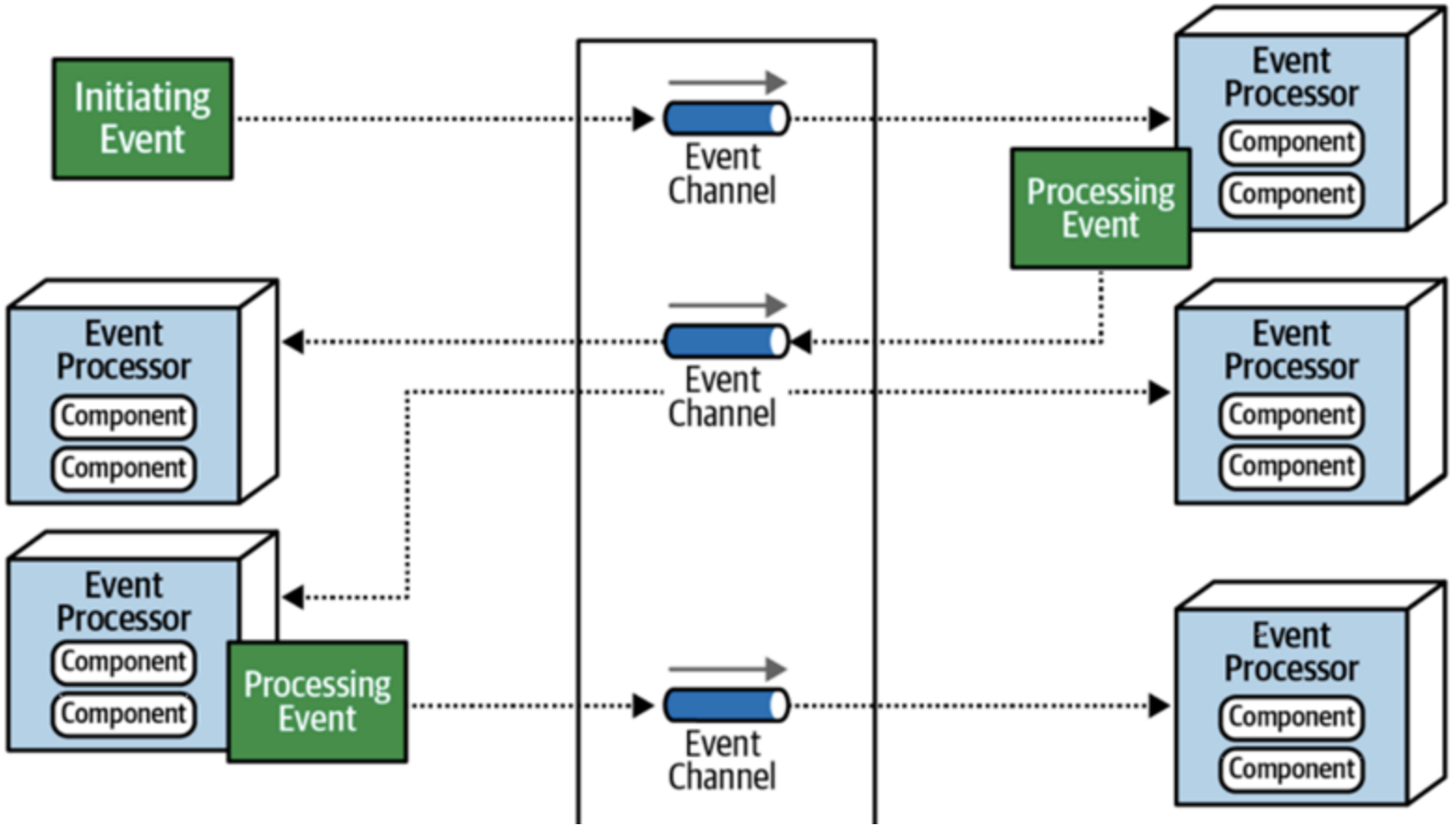
Comunicación basada en eventos

Queries:

- Consultas que se caracterizan por estar libres de efectos de borde, no alteran el estado del sistema de ningún modo.
- Debe tenerse en cuenta el volumen de datos esperado como respuesta.
- Comunicación entre servicios.

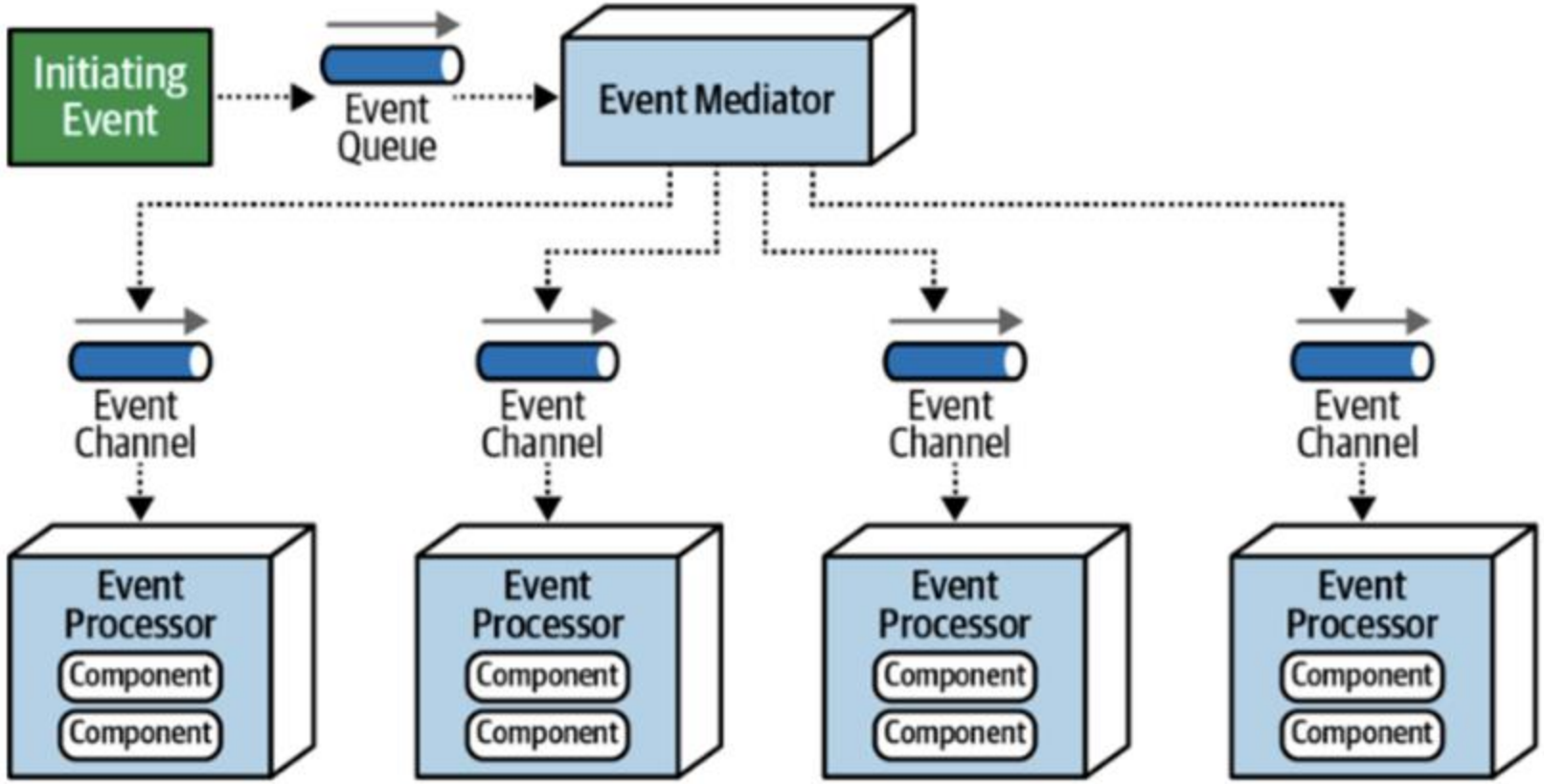
5. Topologías EDA

Broker



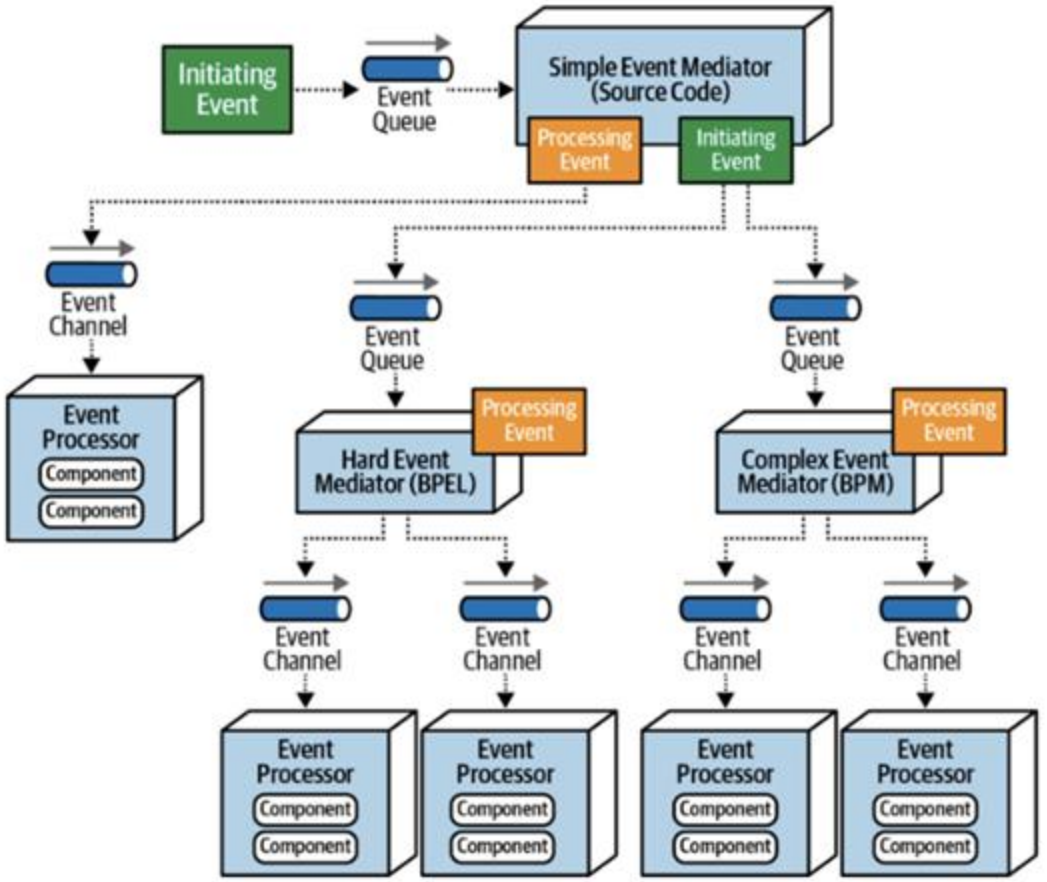
[2] Imagen tomada de: Fundamentals of software Architecture. Neil Ford y Mark Richards. O'Reilly, 2020

Mediator



[2] Imagen tomada de: Fundamentals of software Architecture. Neil Ford y Mark Richards. O'Reilly, 2020

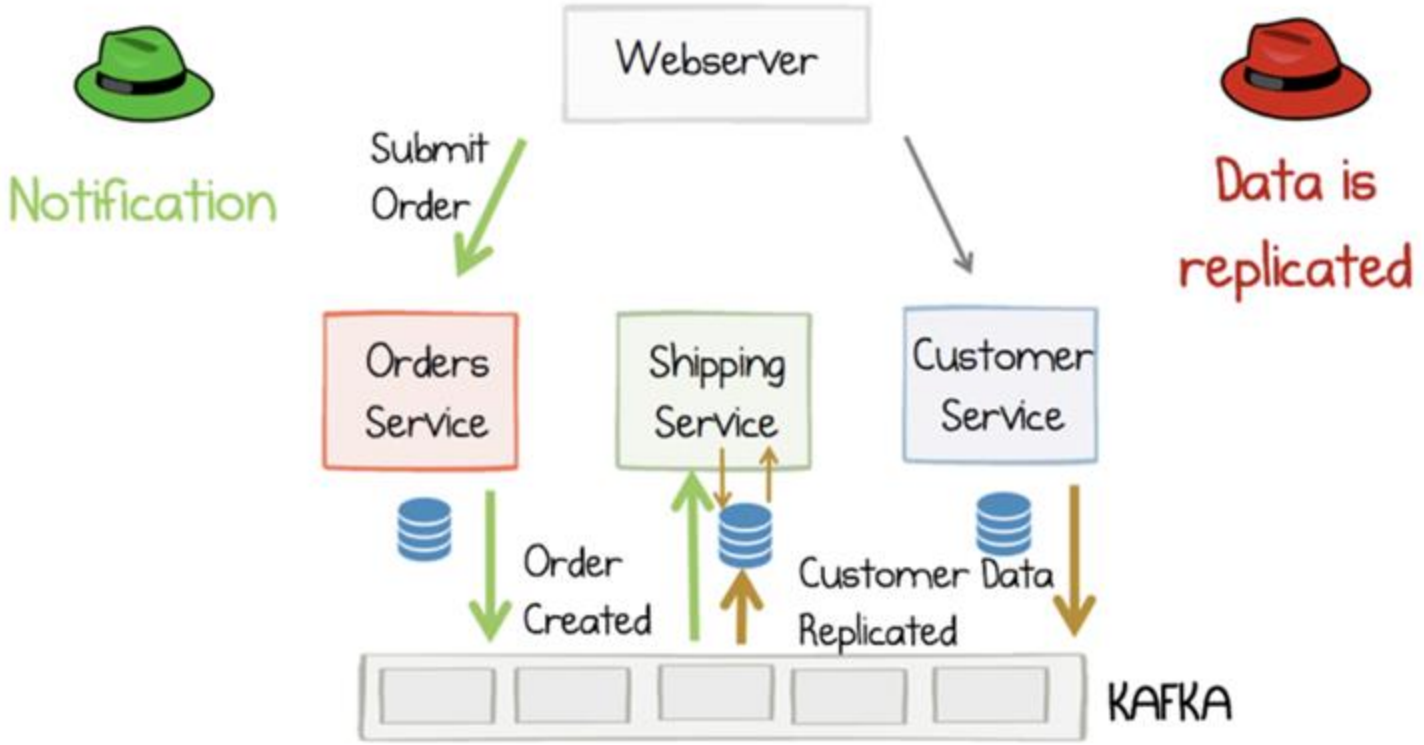
Mediator



[2] Imagen tomada de: Fundamentals of software Architecture. Neil Ford y Mark Richards. O'Reilly, 2020

Transferencia de estado

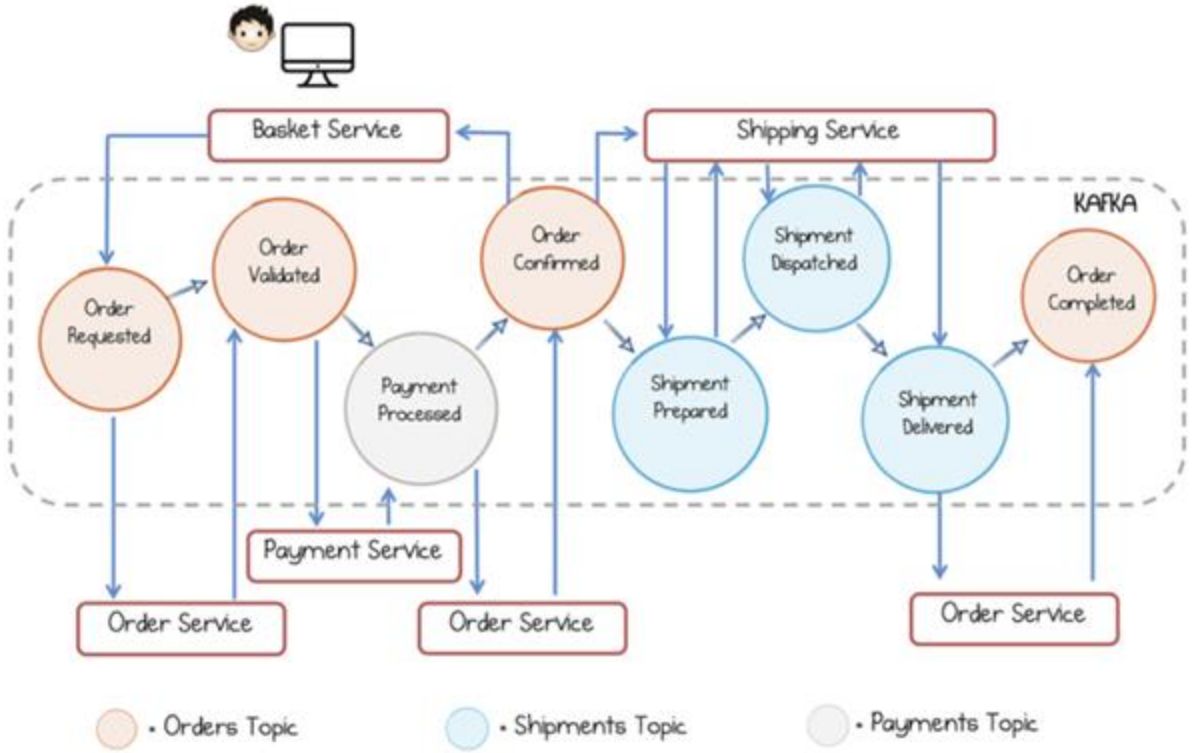
En lugar de realizar consultas a otro servicio, se replican los datos y se realiza la consulta localmente



[2] Imagen tomada de: "Designing Event-Driven Systems". Ben Stopforrd. 2018. O'Reilly Media Inc.

Colaboración por eventos

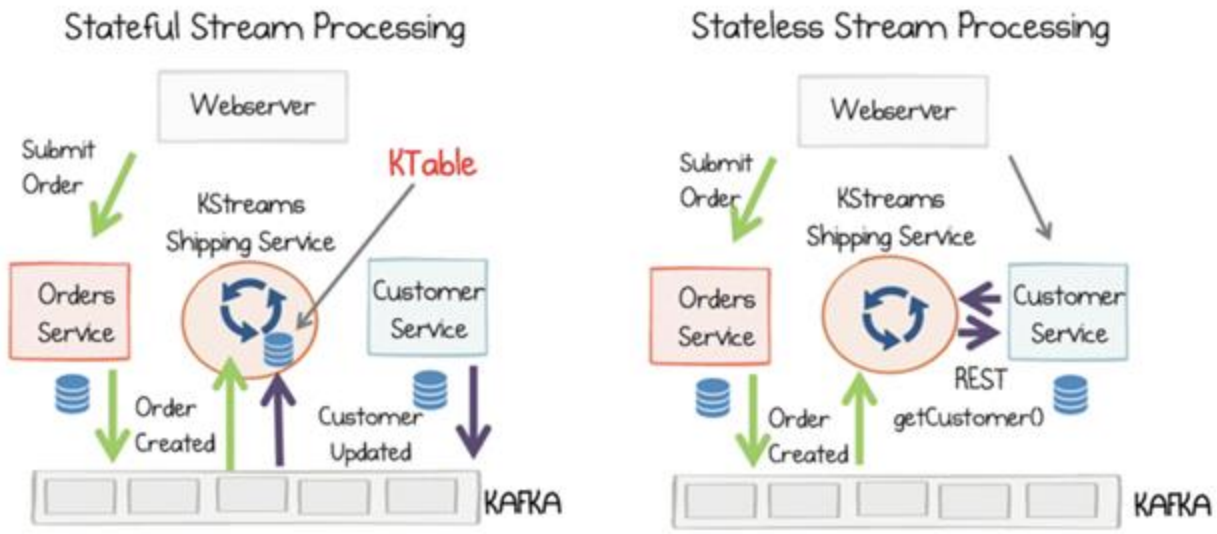
Ningún servicio tiene el control completo del proceso. Todos los procesos se comunican mediante eventos. Ningún servicio sabe de la existencia de otros servicios. Todo el proceso se basa en coreografía de servicios.



[2] Imagen tomada de: "Designing Event-Driven Systems". Ben Stopforrd. 2018. O'Reilly Media Inc.

Colaboración por eventos

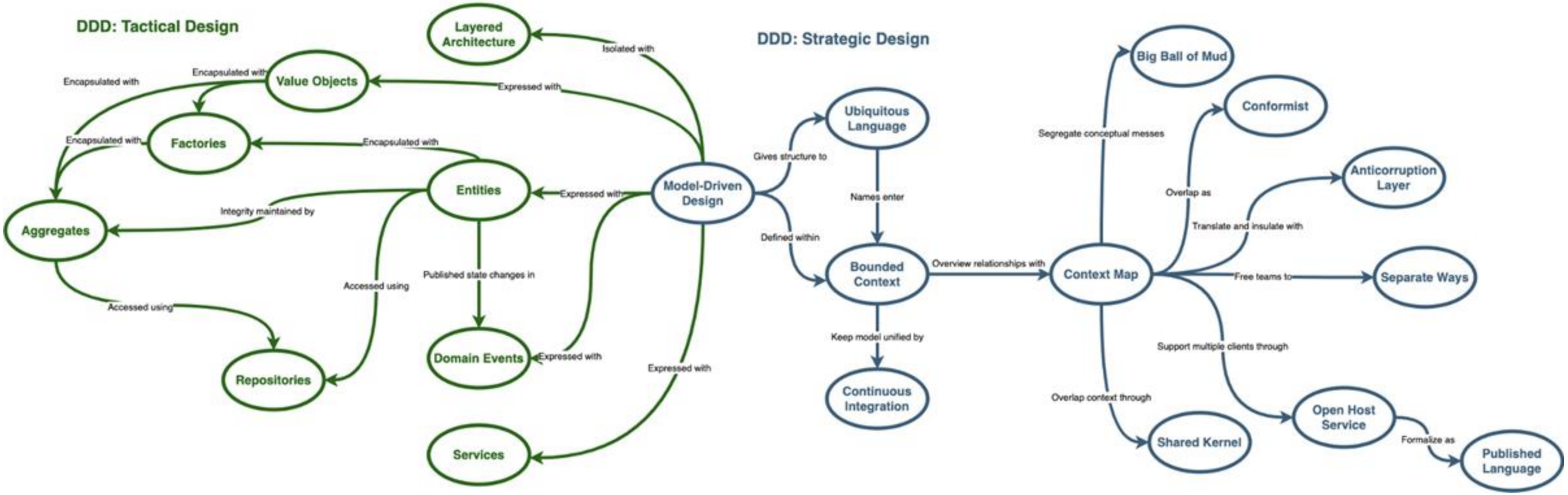
Se puede pasar de usar la plataforma de mensajería con eventos discretos a un enfoque de streaming. Los streams trabajan típicamente sobre colecciones de mensajes.



[2] Imagen tomada de: "Designing Event-Driven Systems". Ben Stopforrd. 2018. O'Reilly Media Inc.

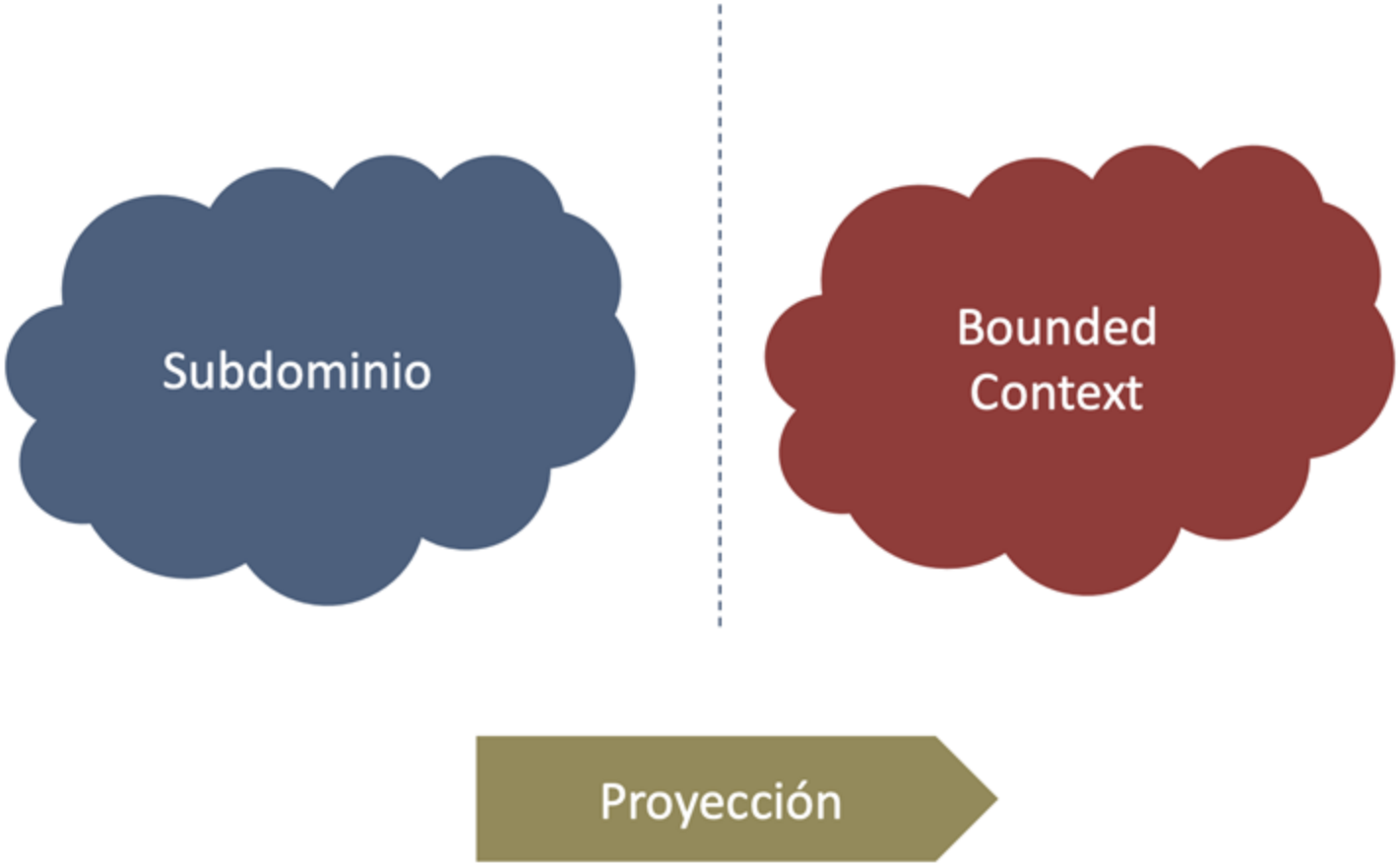
6. Domain Driven Design

DDD: Táctico y Estratégico

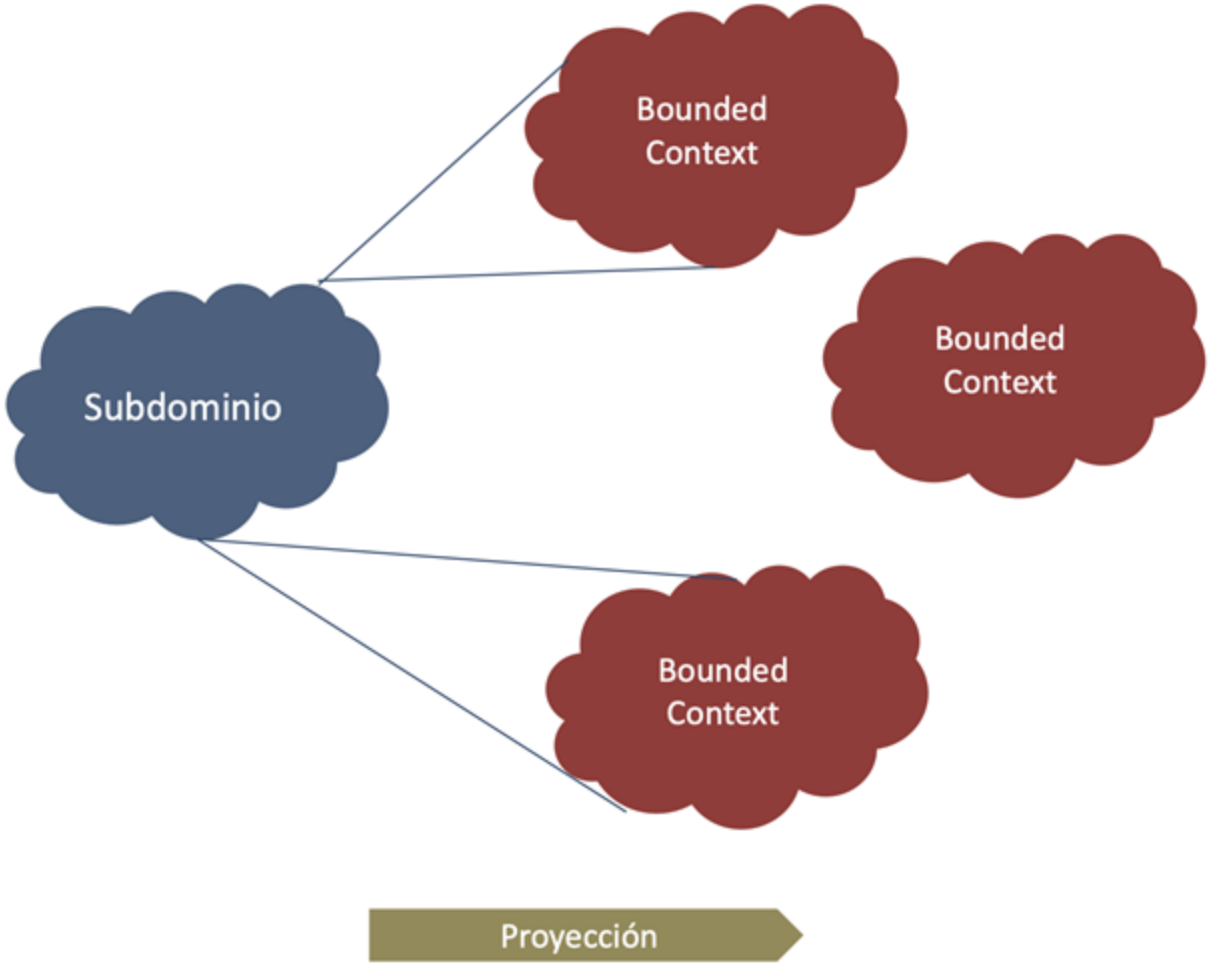


[3] Imagen tomada de: "Eric Evans DDD."

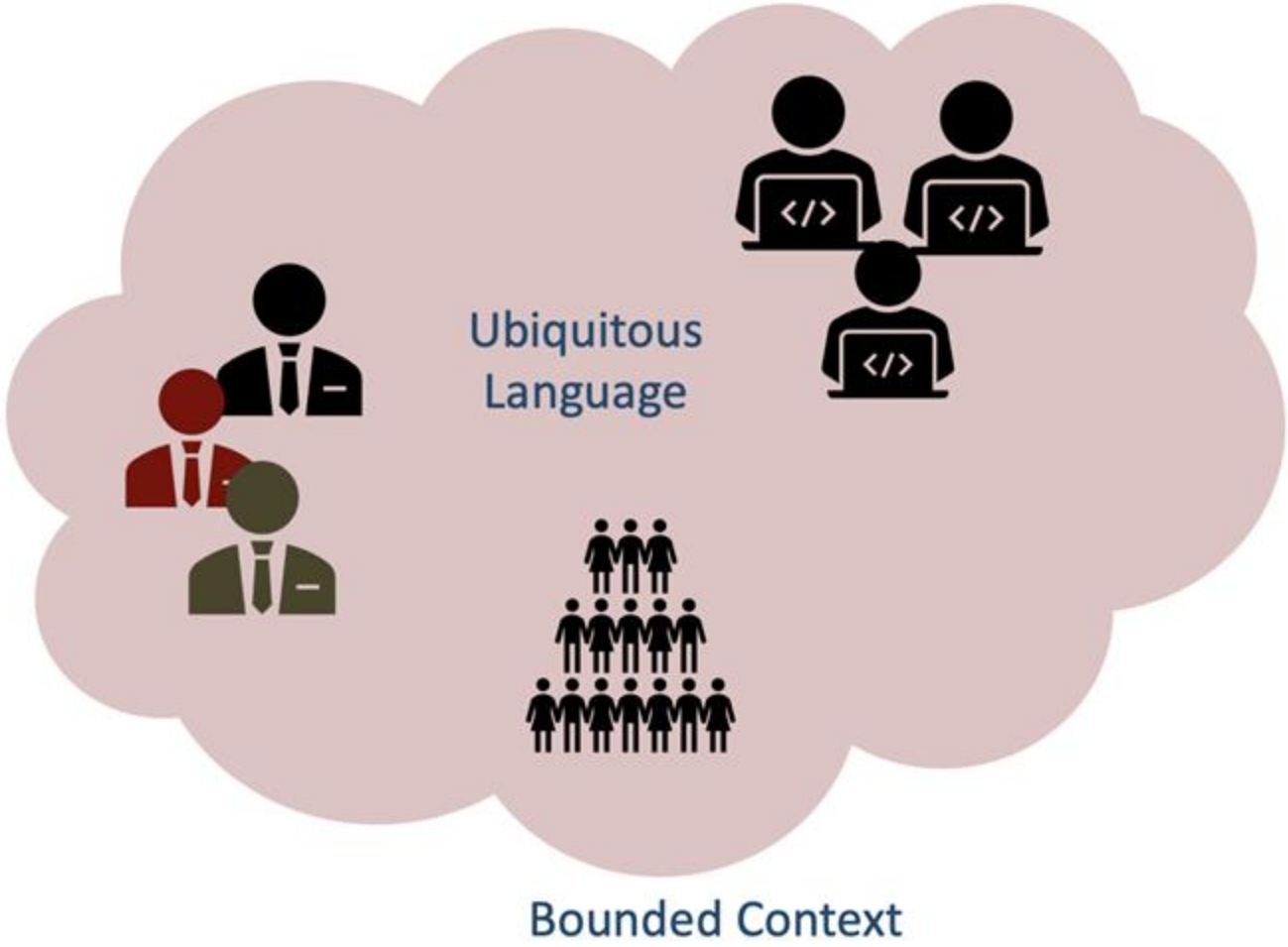
Dominio del problema y de la solución



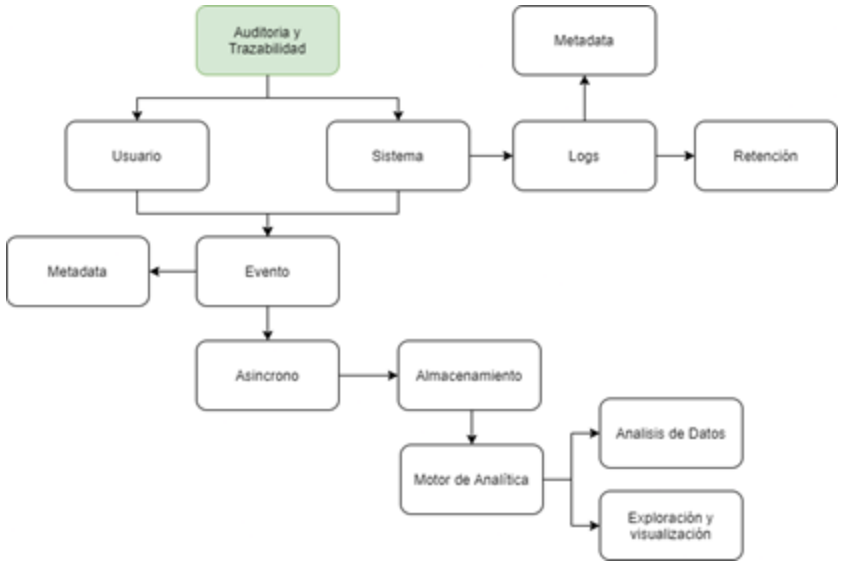
Dominio del problema y de la solución



Dominio de la solución



Dominio de la solución

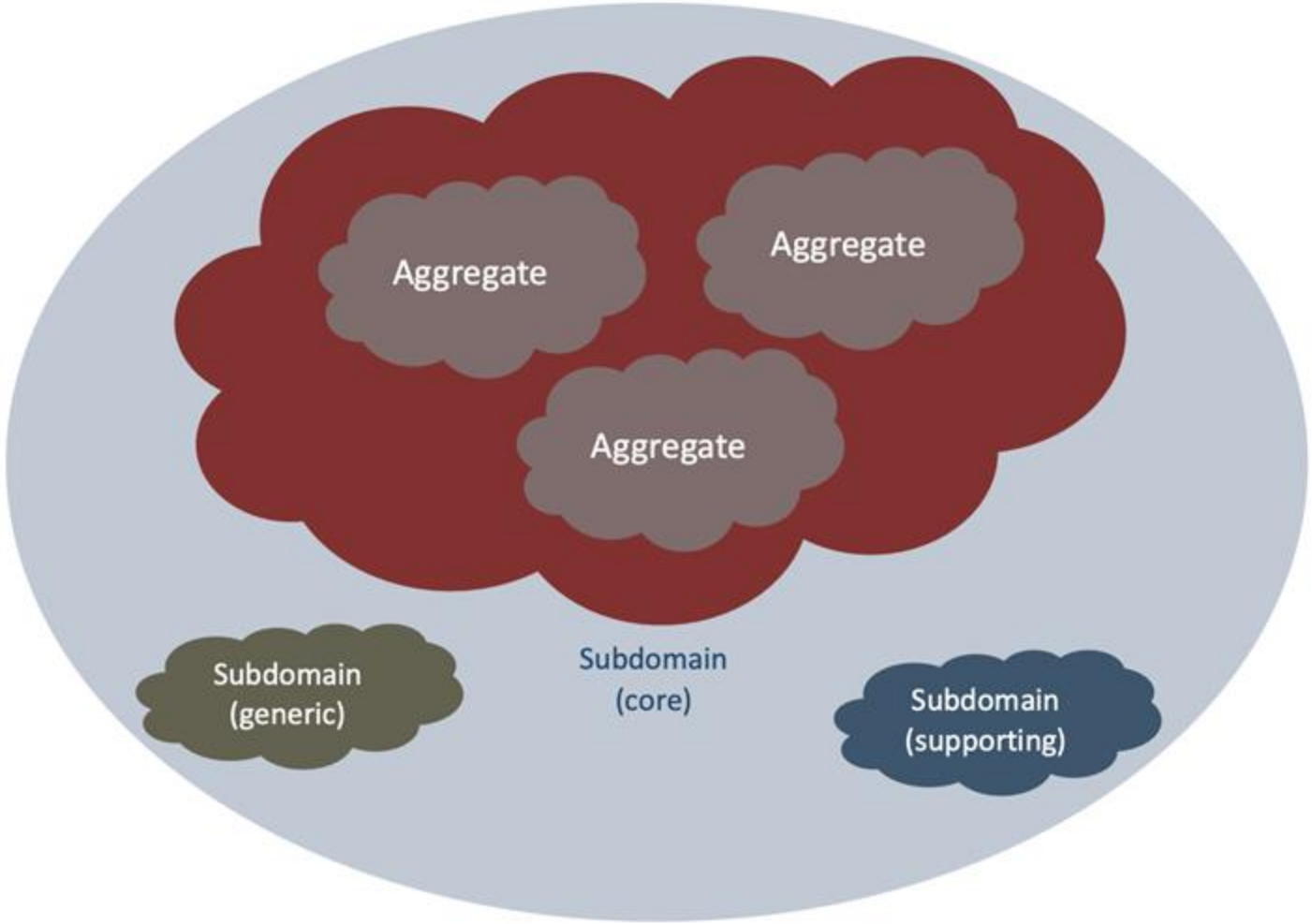
Nombre Dominio - Subdominio	Bounded Context
Auditoría y Trazabilidad	
Clasificación estratégica	
Genérico	
Descripción del Dominio / Subdominio / BC	
Sistema para auditoria y trazabilidad de la solución, que permite llevar control, registro, y exploración de la data en cada evento.	

Lenguaje ubicuo

Término de dominio	Descripción
Usuario	Persona que utilizará la plataforma de Procesos, suelen ser trabajadores de la organización o proveedores (terceros).
Sistema	Rol que se le da a las tareas automáticas, que no se ejecutan de forma manual por una persona.
Logs	Bitacora de sistema utilizado para escribir las acciones, errores, y cualquier pista de seguimiento que permita hacer diagnósticos.
Retención	Periodo de tiempo en el cual se debe conservar la información.
Metadata	Data extra asociada a eventos, como la hora, persona que realiza la acción, dirección IP, o cualquier otra información que no hace parte del evento en sí pero es relevante.
Evento	Acción o suceso dentro del sistema, ejecutado de forma automática o manual, que sea susceptible de ser almacenado como fuente para analítica o probatoria.
Asincronía	Que no tiene lugar en completa correspondencia temporal con el evento en sí, esto da lugar a que la analítica y gestión de trazabilidad no afecte el desempeño del sistema, tengan sistemas resilientes y menor probabilidad de perdida de información.
Almacenamiento	Base de datos de alta transaccionalidad donde quedan almacenados los eventos y su metadata.
Motor Analítica	Motor que permite la explotación y consulta de los datos, suele ir de la mano con el almacenamiento.
Visualización de datos	Elemento que permite explorar de forma gráfica la información contenida en este sistema.
Análisis de datos	Interacción y uso del motor de analítica para ir más allá de la exploración y pasar a la explotación de los datos.

Diseño estratégico y diseño táctico

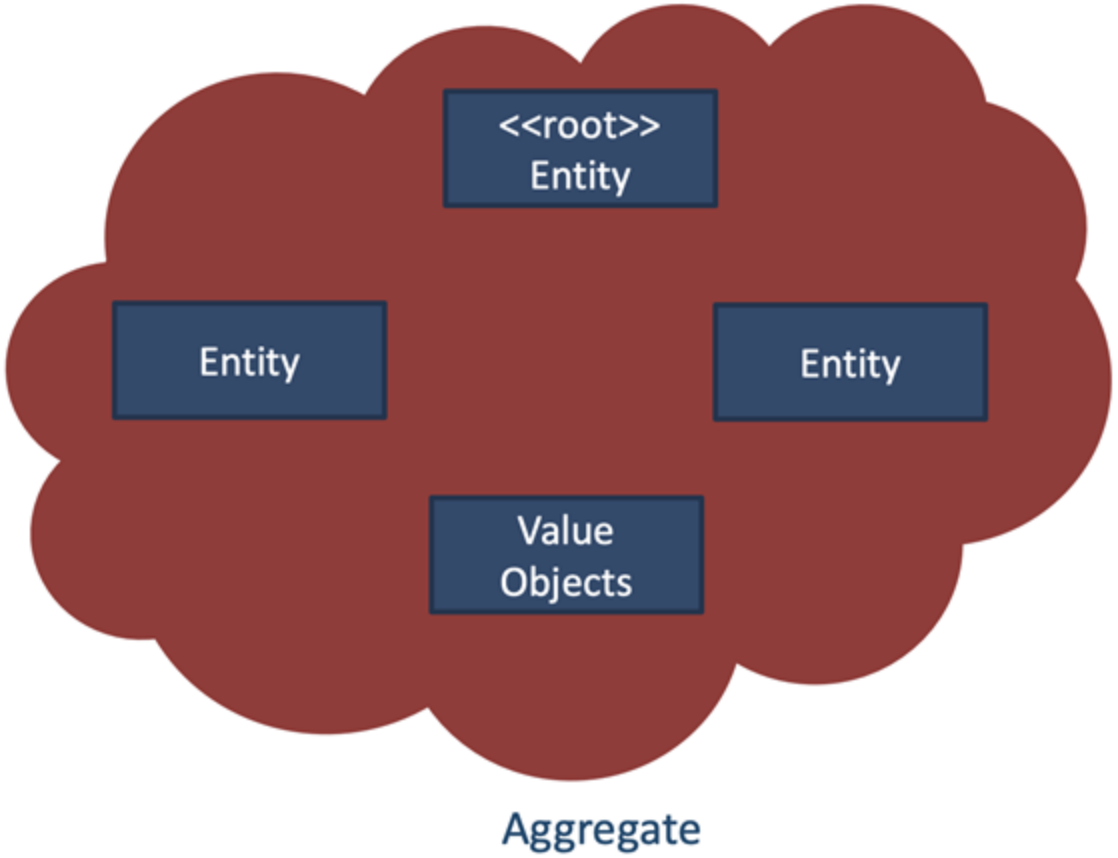
Diseño estratégico y diseño táctico



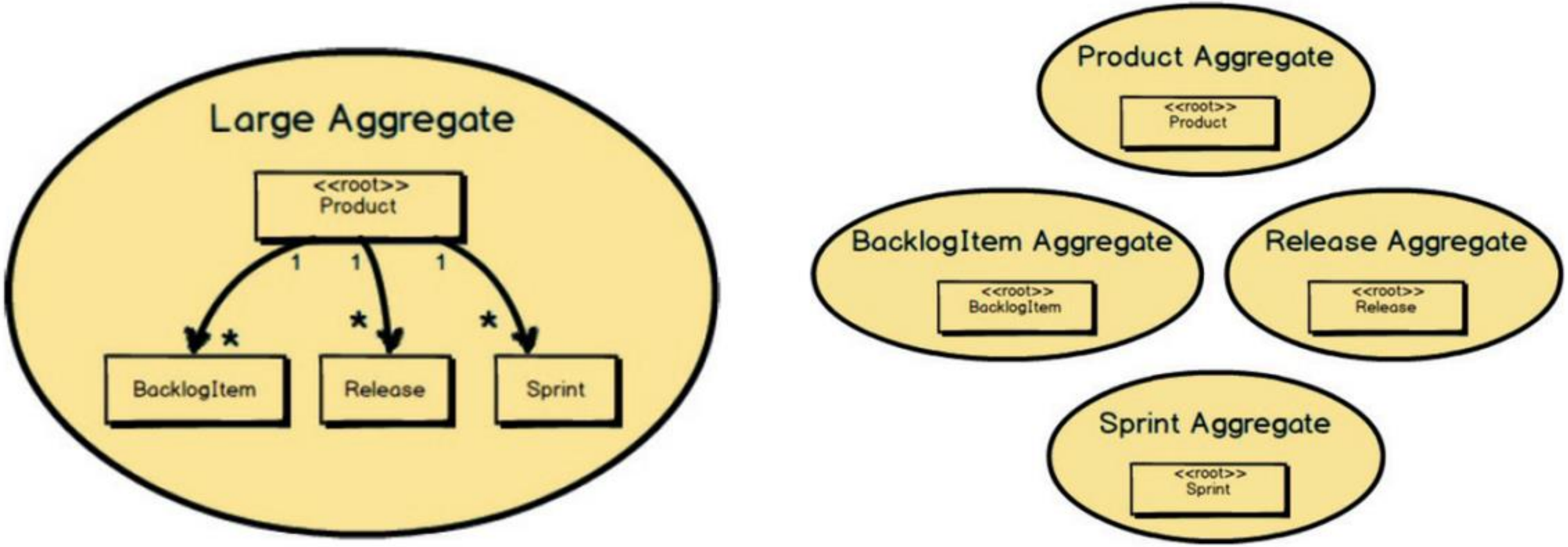
Diseño táctico

Cada Aggregate, forma una frontera transaccional y al interior todo las partes son consistentes

1. *Propender por diseñar Aggregates pequeños*
2. *Solo se referencian otros aggregates por su identificador*
3. *Los Aggregates se actualizan mediante la consistencia eventual*

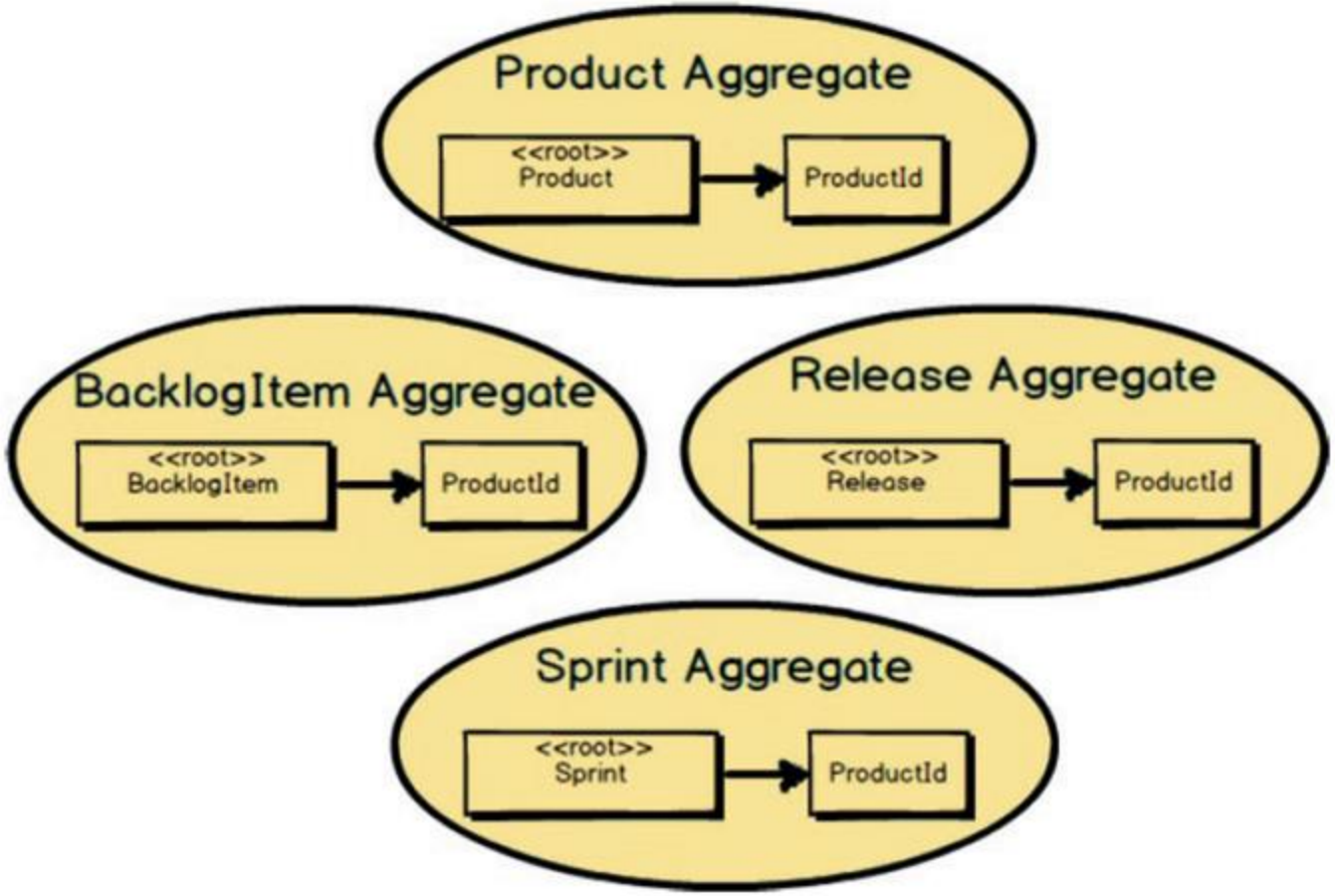


Propender por diseñar Aggregates pequeños



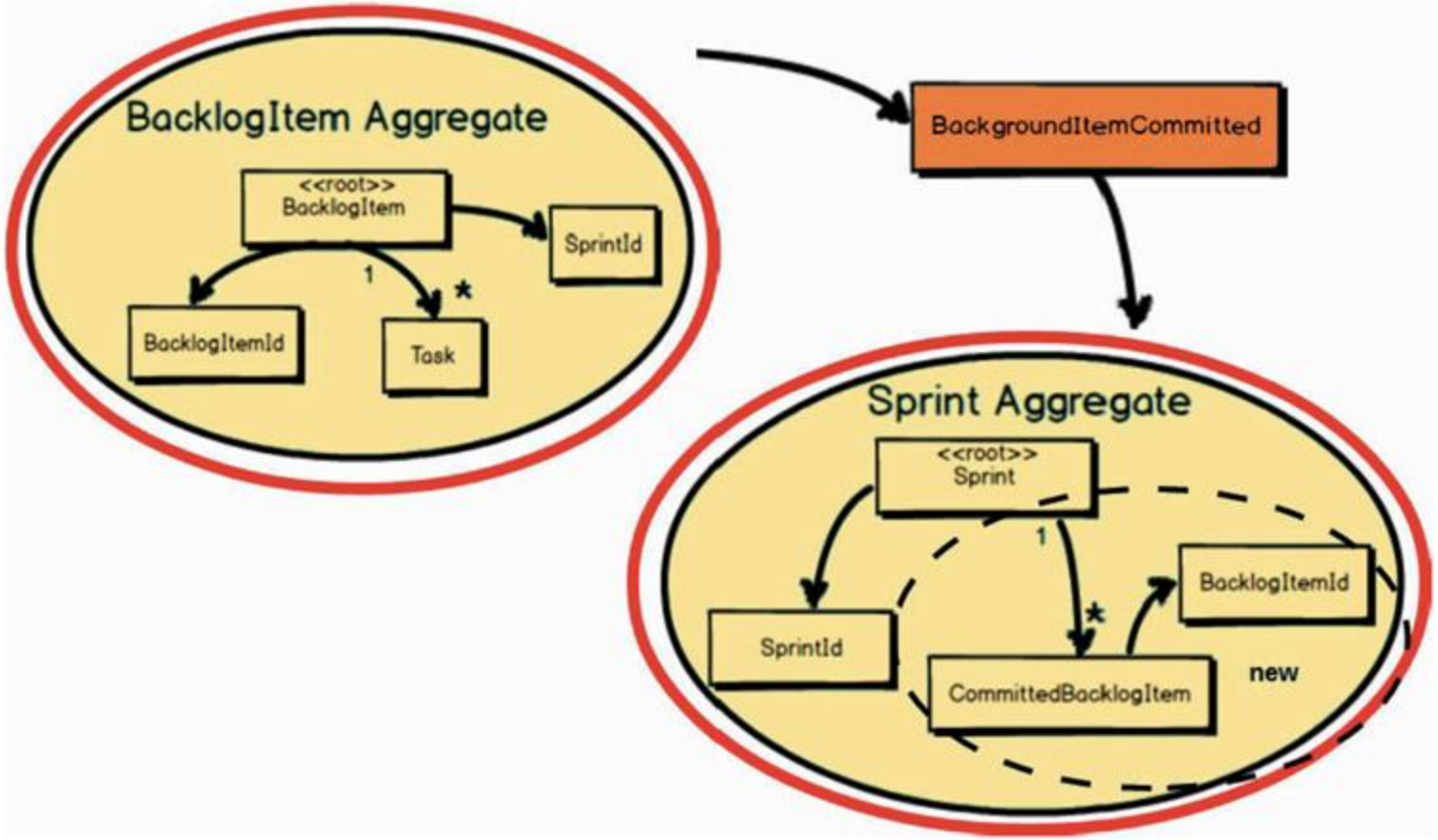
[4] Imagen tomada de: Domain-Driven Design Distilled. Vaughn Vernon. Addison Wesley

Solo se referencian otros aggregates por su identificador



[4] Imagen tomada de: Domain-Driven Design Distilled. Vaughn Vernon. Addison Wesley

Los Aggregates se actualizan mediante la consistencia eventual



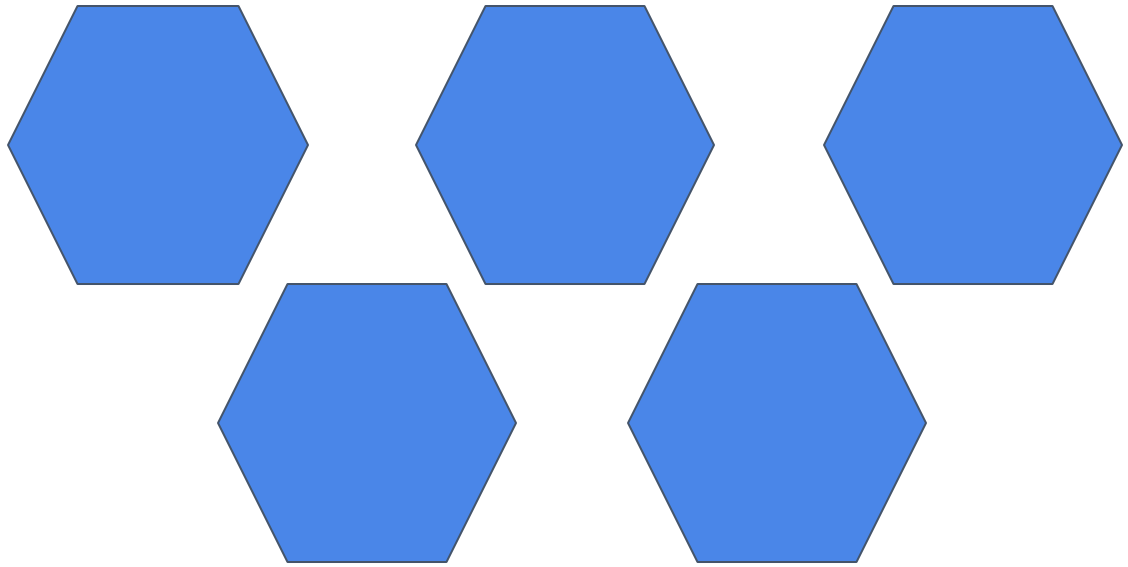
[4] Imagen tomada de: Domain-Driven Design Distilled. Vaughn Vernon. Addison Wesley

7. Microservicios

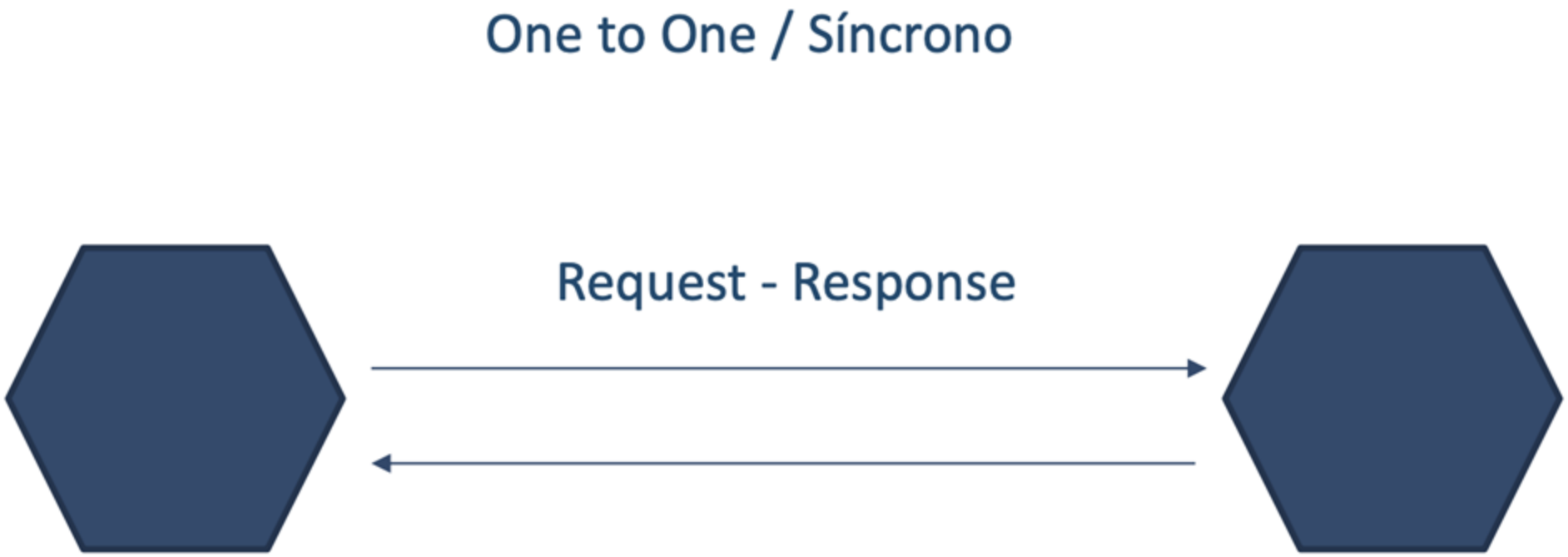
Sistemas distribuidos

"Colección de computadores independientes que a la vista de los usuarios se ven como un solo sistema coherente"

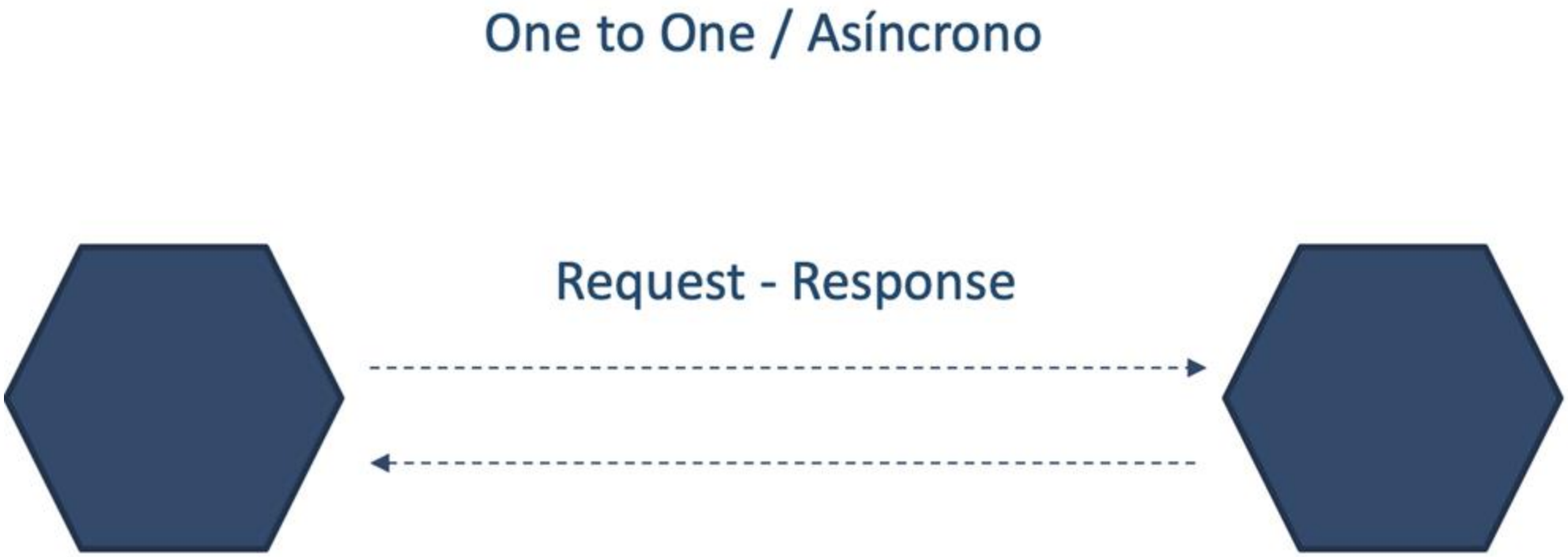
Tanebaum 2007



Mecanismos de comunicación

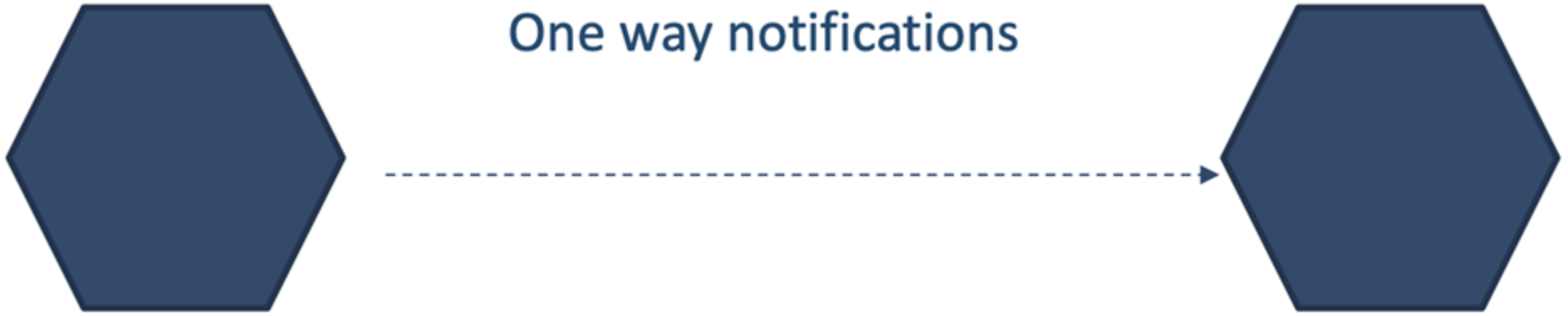


Mecanismos de comunicación

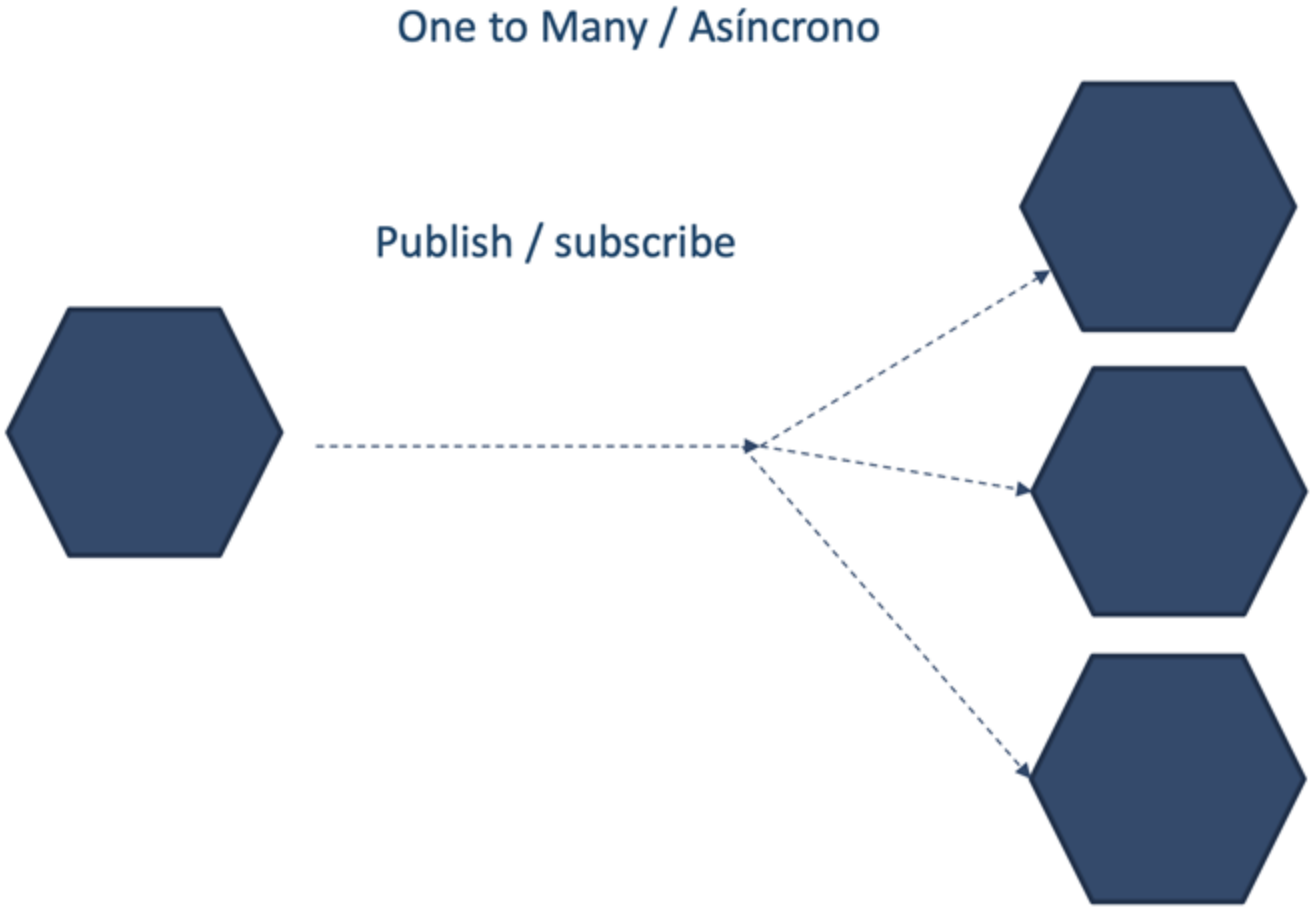


Mecanismos de comunicación

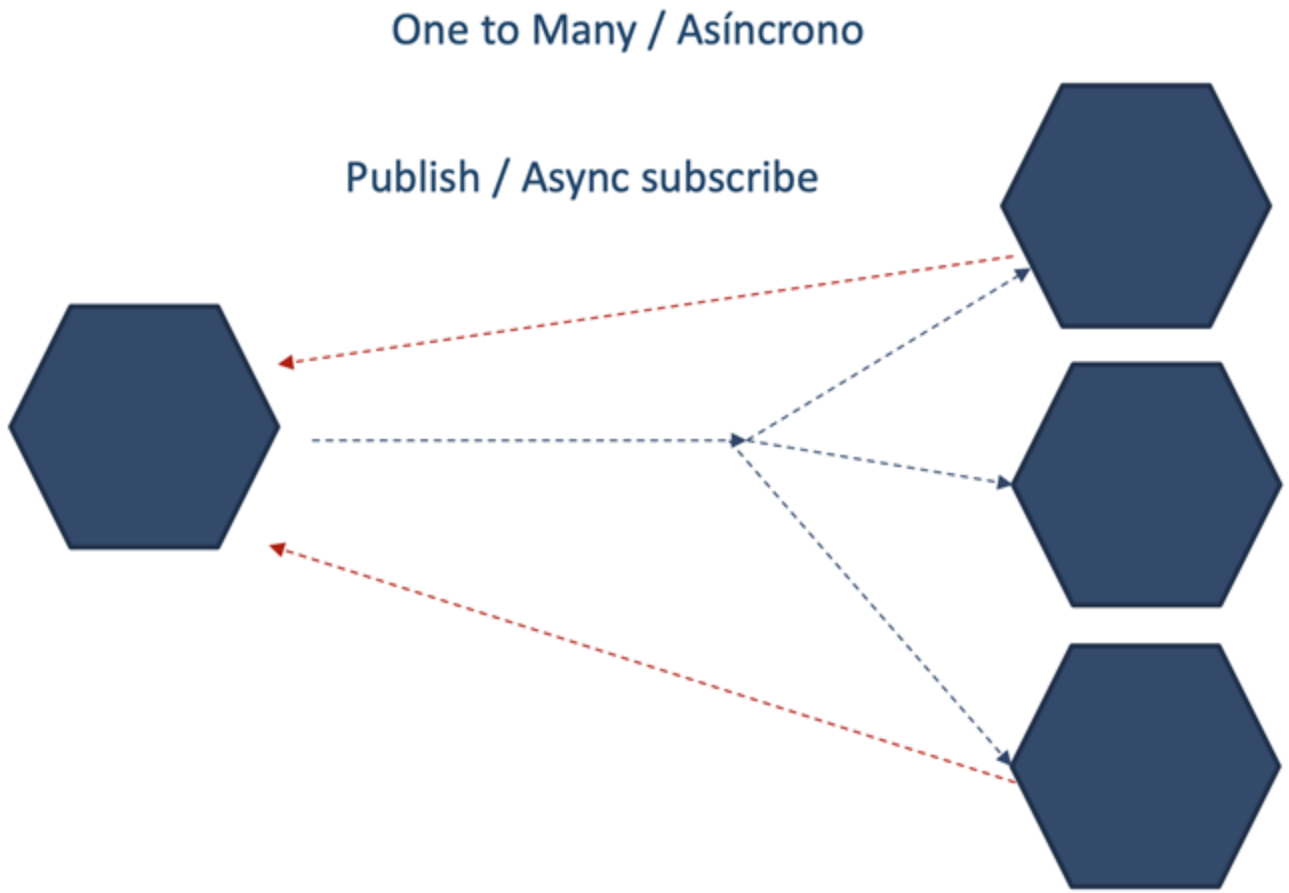
One to One / Asíncrono



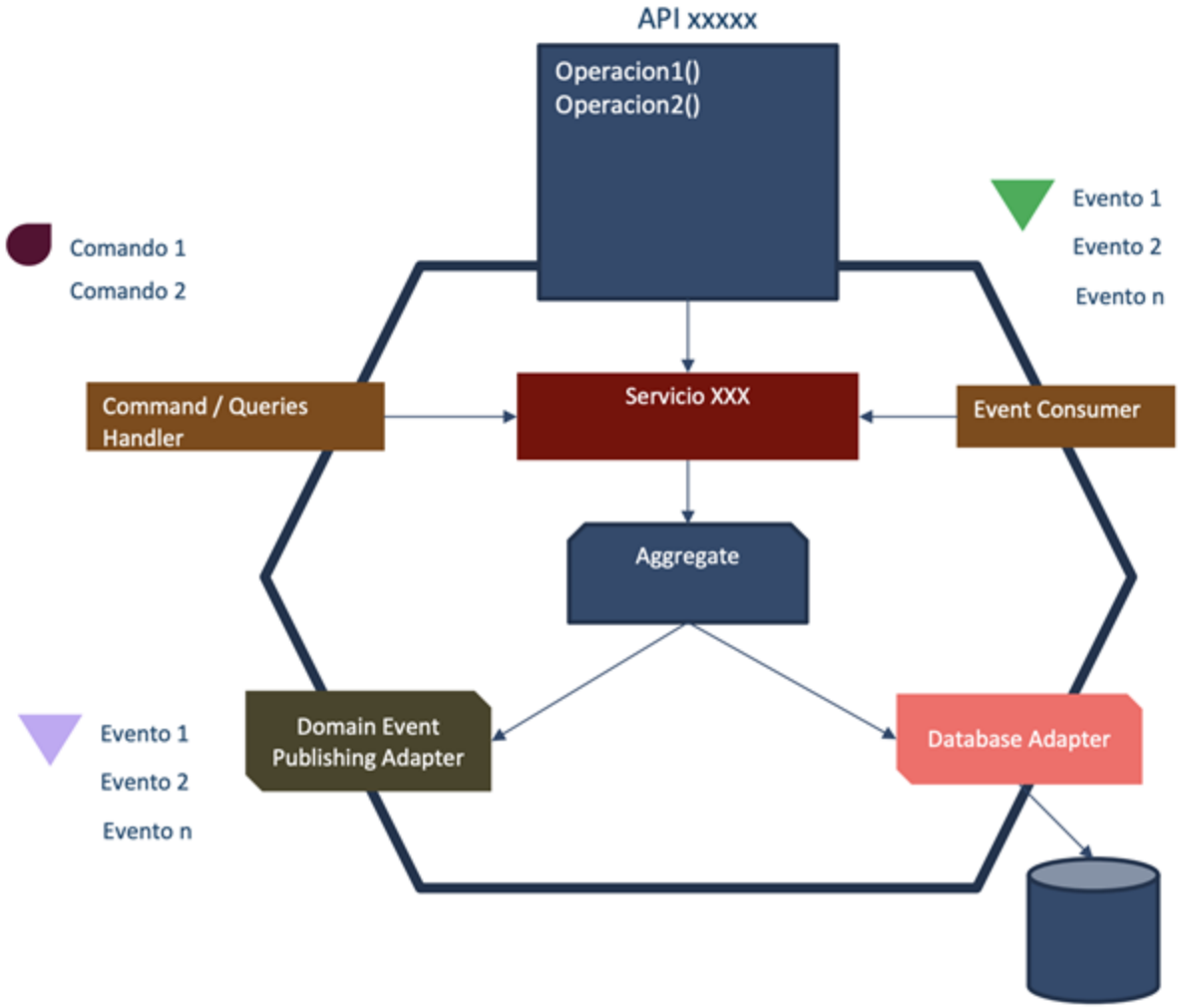
Mecanismos de comunicación



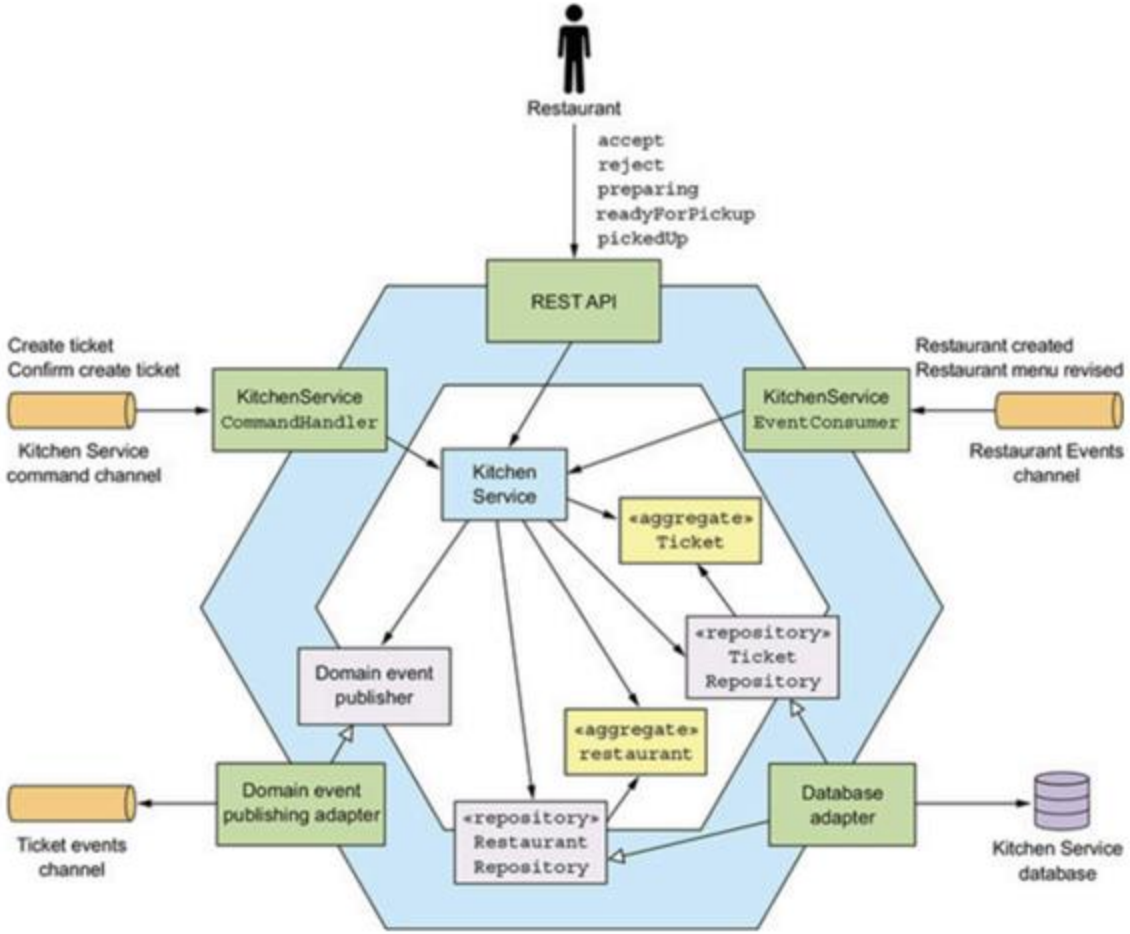
Mecanismos de comunicación



Arquitectura hexagonal



Arquitectura hexagonal



[5] Imagen tomada de: Microservices Patterns. Chris Richardson. Manning

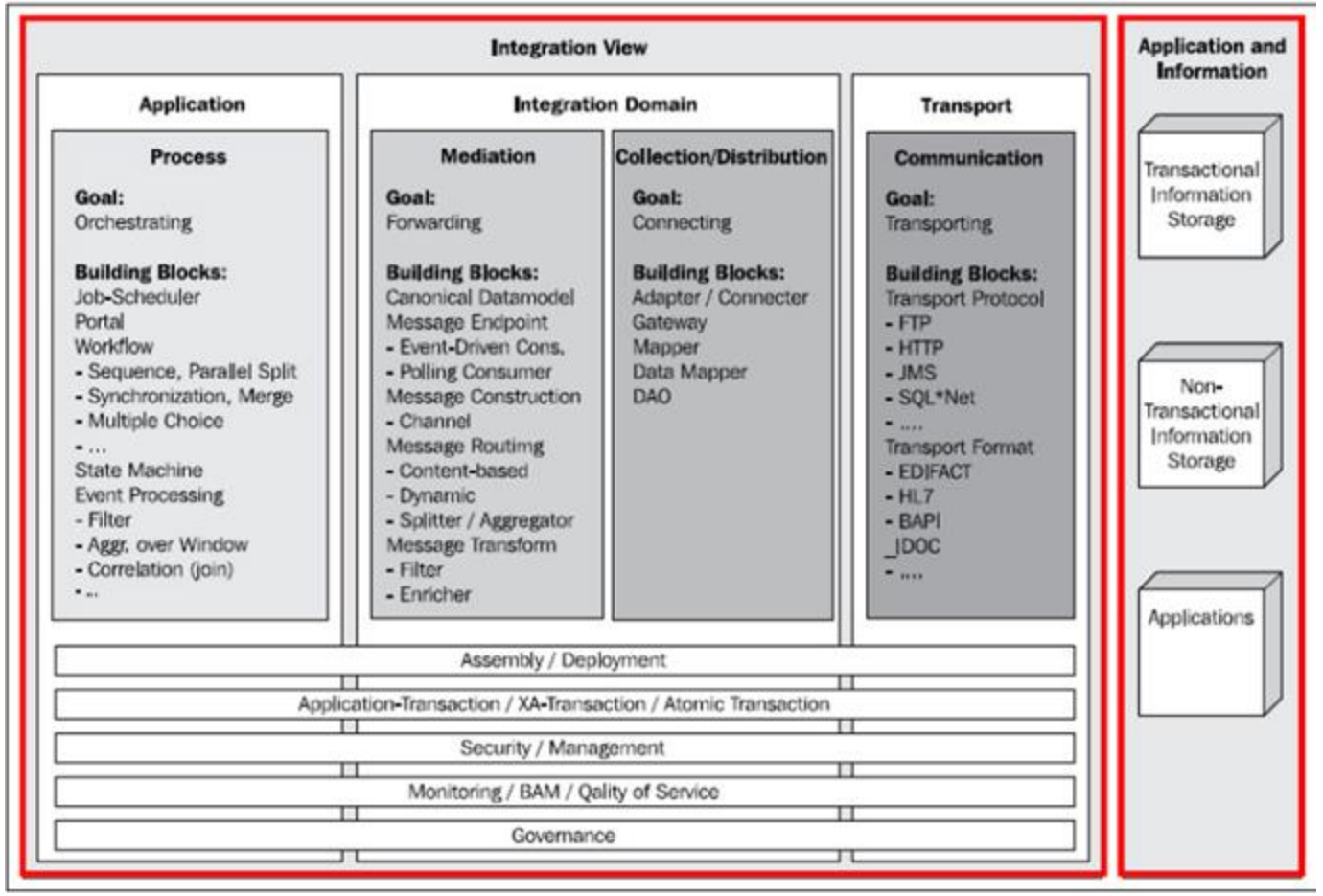
Objetivo de negocio

Objetivo de Negocio	ON_003
Detalle	
Se requiere por parte del área de seguridad de la información del cliente, que todas las acciones del sistema sean rastreables en el tiempo, determinando fecha completa, IP, usuario que realiza la acción, número de solicitud, y otra información relevante	
Qué función cumplirá	El API deberá auditar con precisión y detalle todas las acciones realizadas por usuarios y por el propio sistema, debe dar información relevante para el área de seguridad y debe poderse comprobar la autenticidad de esta.
Que restricciones se tienen	El almacenamiento de la trazabilidad debe realizarse mediante Amazon Elasticsearch Service
Que atributos de calidad se requieren	Seguridad: • Responsabilidad

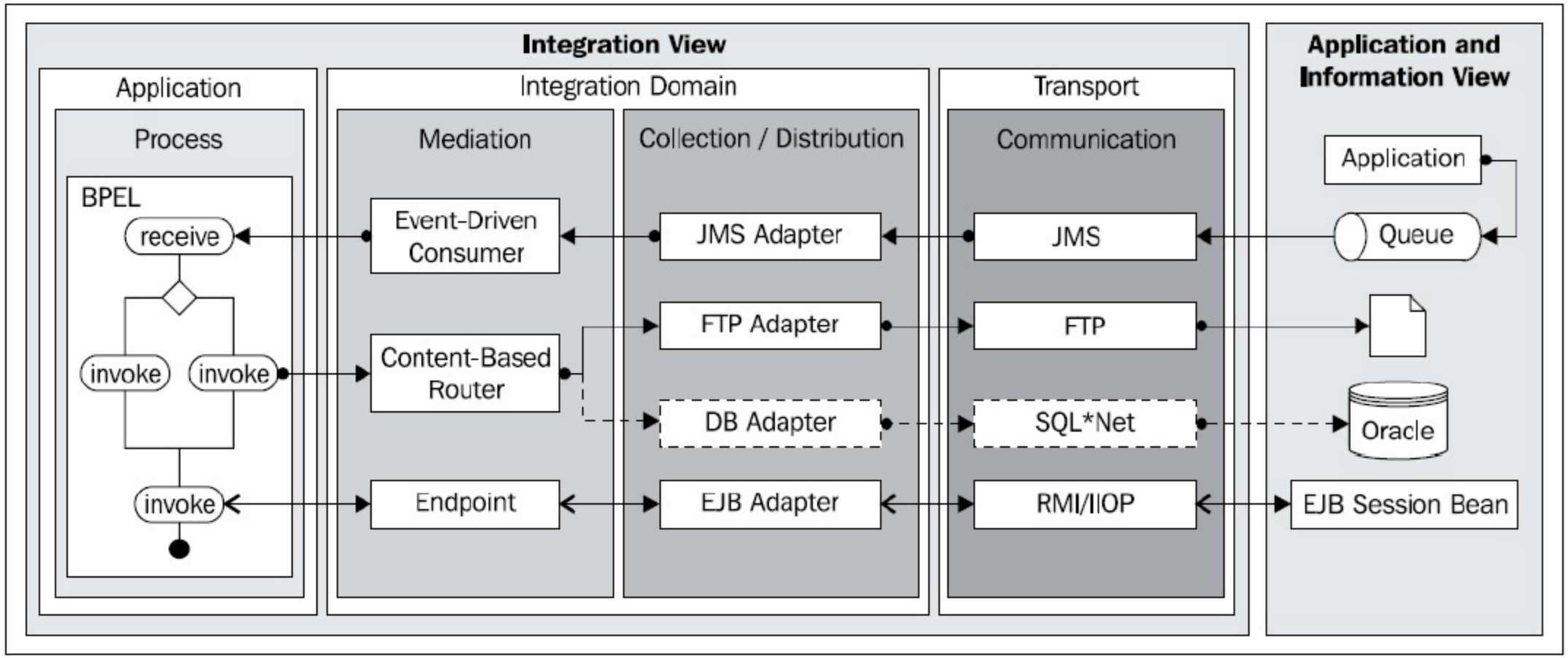
Escenario operacional

ID	AT_F001	Versión	1.0	
Fuente	Estímulo	Artefacto	Especificación	Ambiente
Sistema	Se consulta un cliente	CRM	Tolerancia a fallos	Operación normal
Respuesta		Medida de la respuesta		
El API continúa atendiendo peticiones retornando la información de clientes		El 95% de las veces las peticiones son procesadas exitosamente		

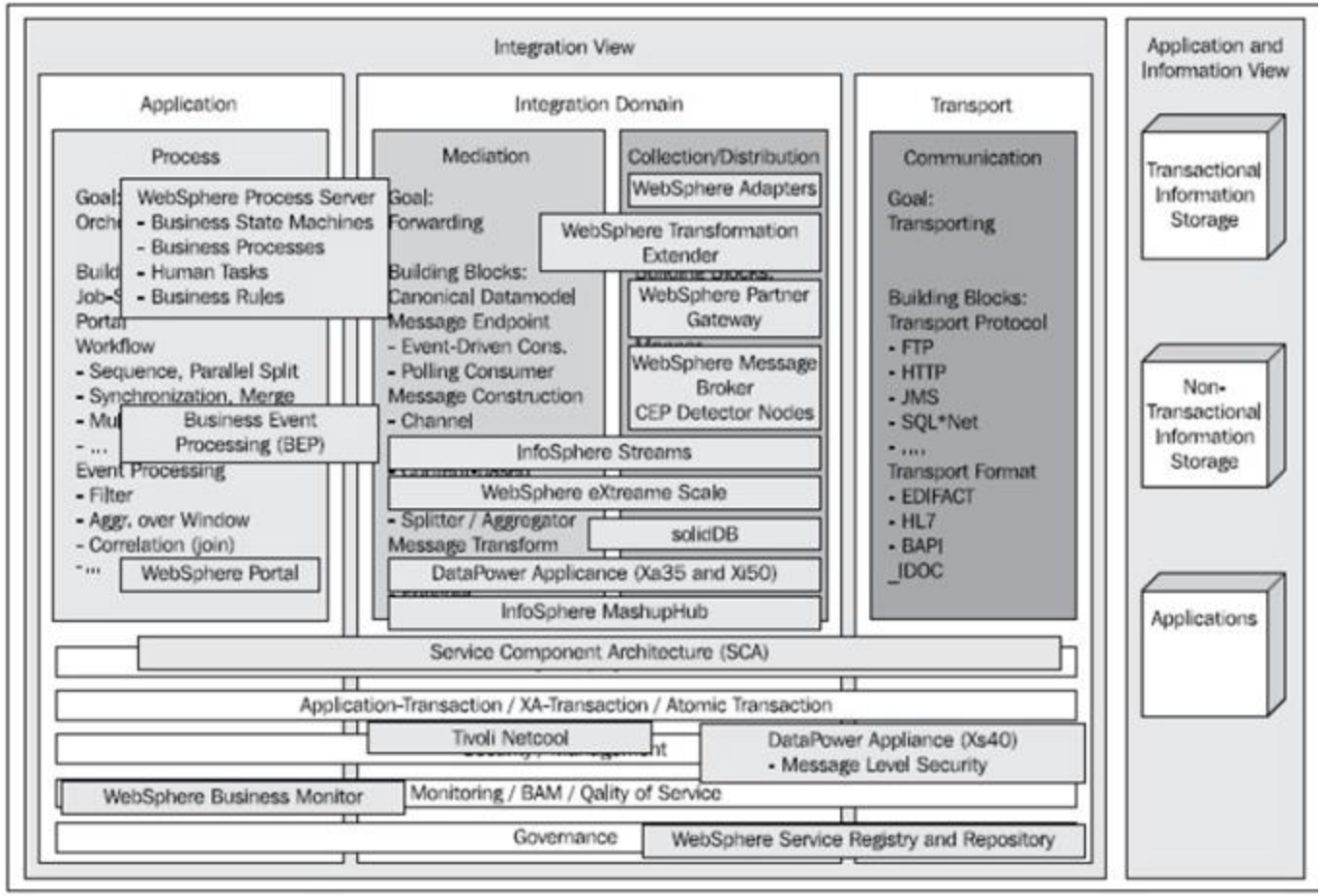
Trivadis Integration Architecture Blueprint



Trivadis Integration Architecture Blueprint



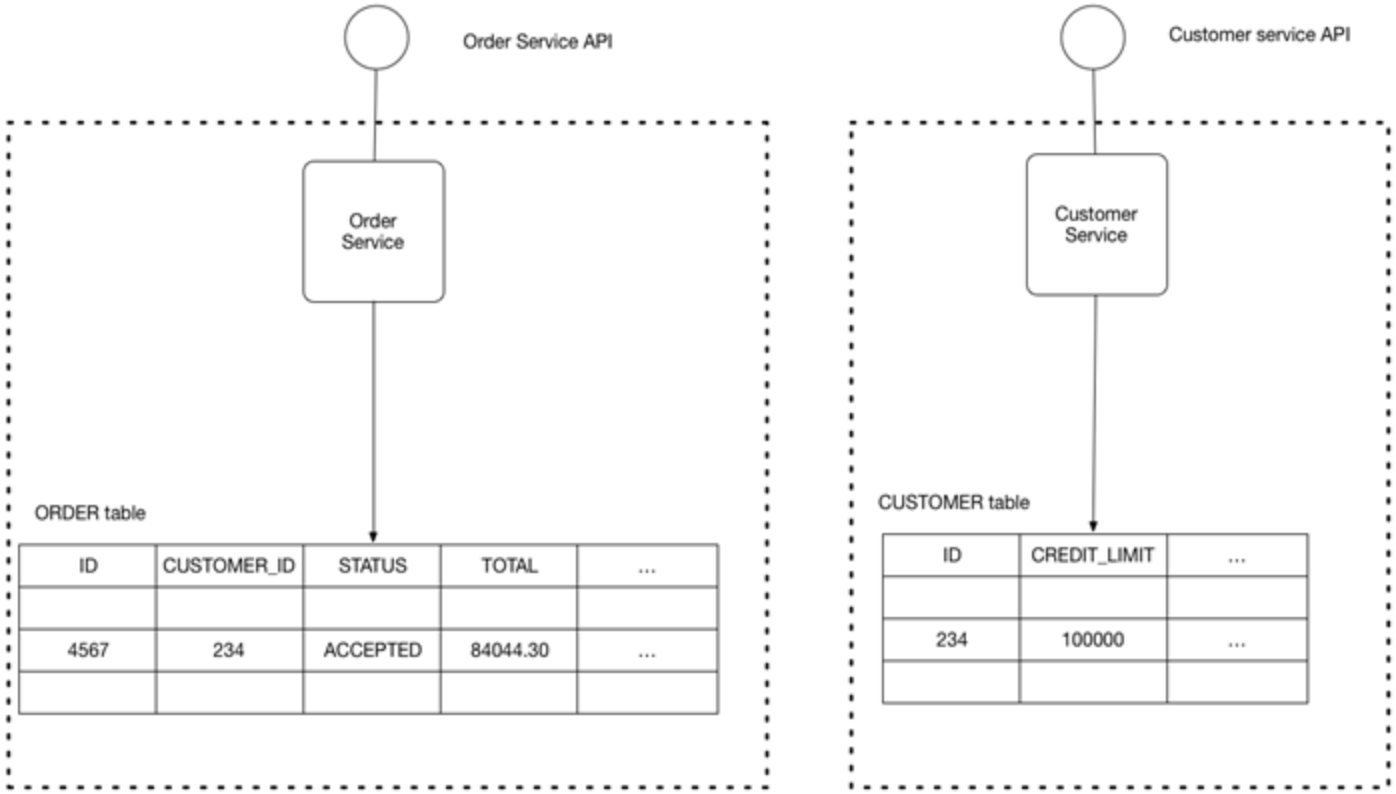
Trivadis Integration Architecture Footprint



8. Patrones

Database per service

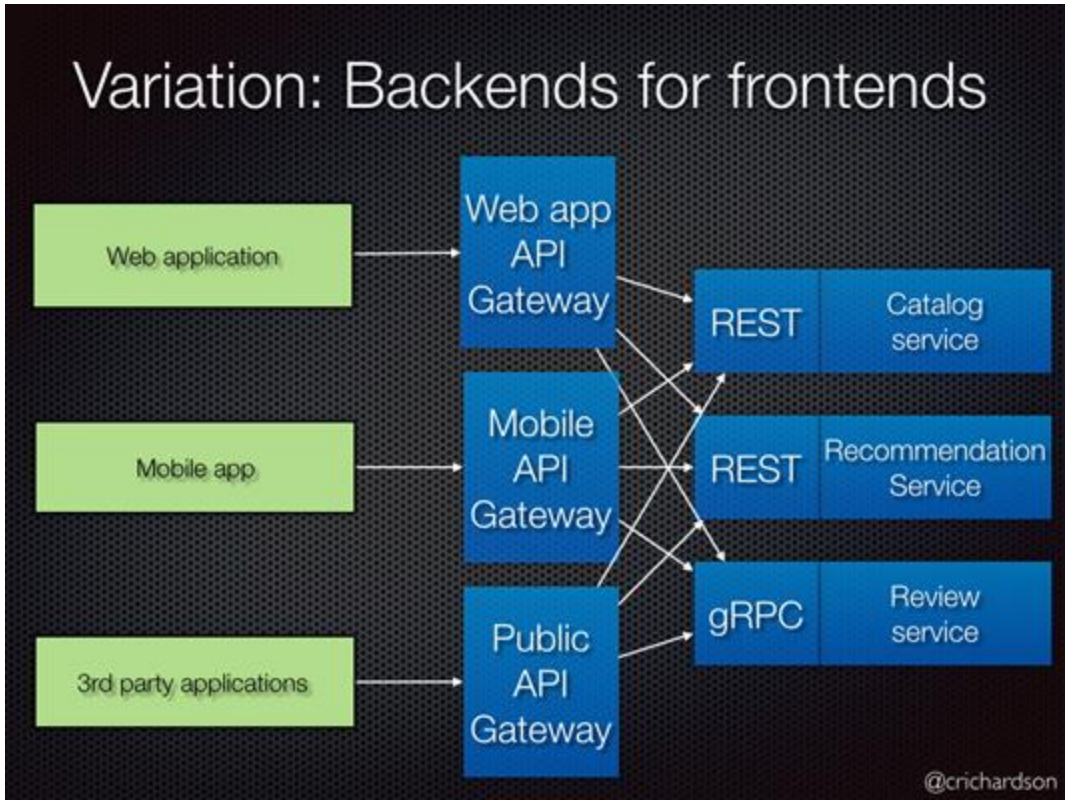
Mantenga los datos persistentes de cada microservicio privados para ese servicio y accesibles sólo a través de su API. Las transacciones de un servicio solo involucran su base de datos.



[6] Imagen tomada de: https://microservices.io/patterns/data/database-per-service.html

API Gateway – Backend for frontends

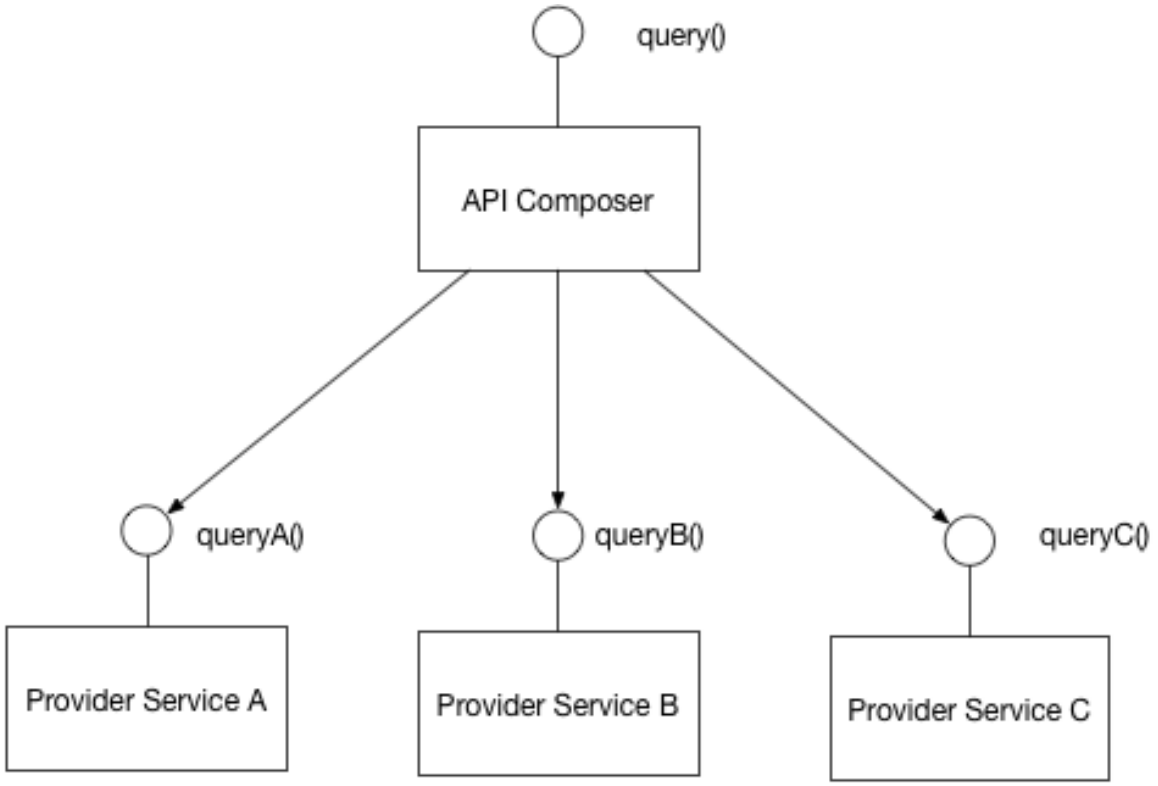
El patrón Backends for frontends es una variación del patrón API Gateway. Define un API Gateway separado para cada tipo de cliente.



[7] Imagen tomada de :https://microservices.io/patterns/apigateway.html

API Composition

Implemente una consulta definiendo un API Composer, que invoca los servicios que poseen los datos y realiza una unión de los resultados en memoria.



[8] Imagen tomada de :https://microservices.io/patterns/data/api-composition.html

Preguntas