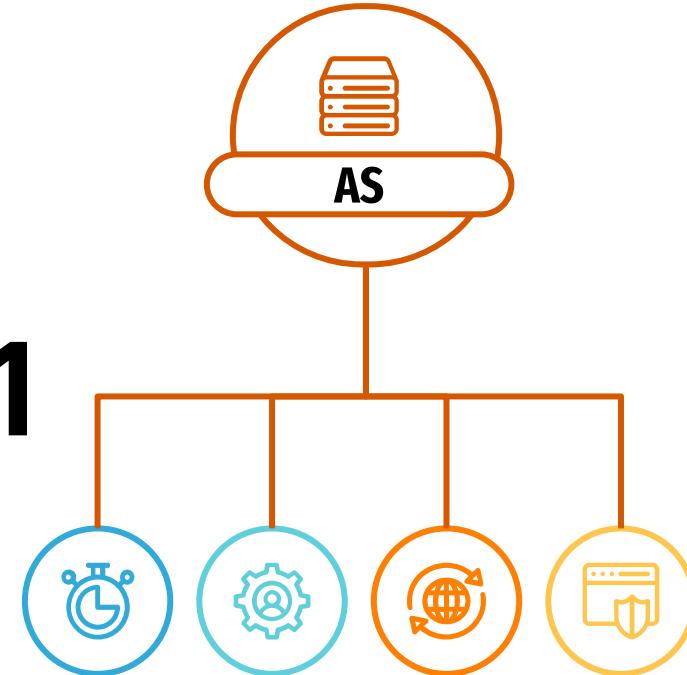




# Estilos Arquitectónicos 1

MSc. Marco Túlio Aldana Prillwitz

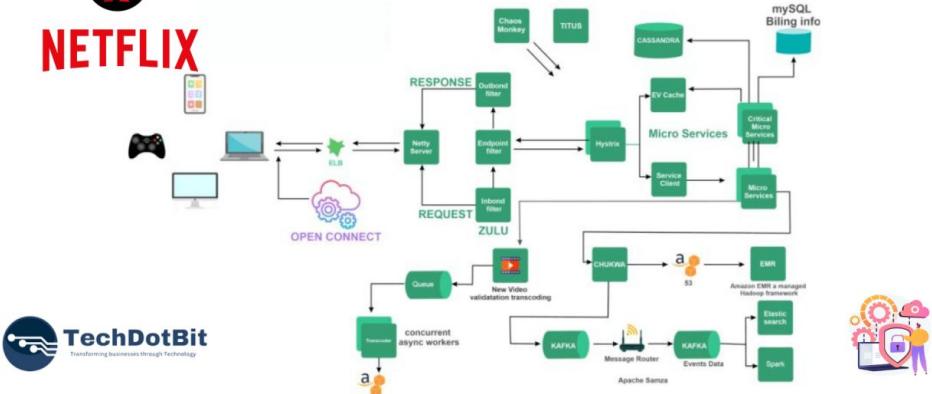




# Arquitectura de Software

- La arquitectura de software se basa en la toma de decisiones de diseño cruciales para organizar el software y asegurar que cumpla con los atributos de calidad deseados.

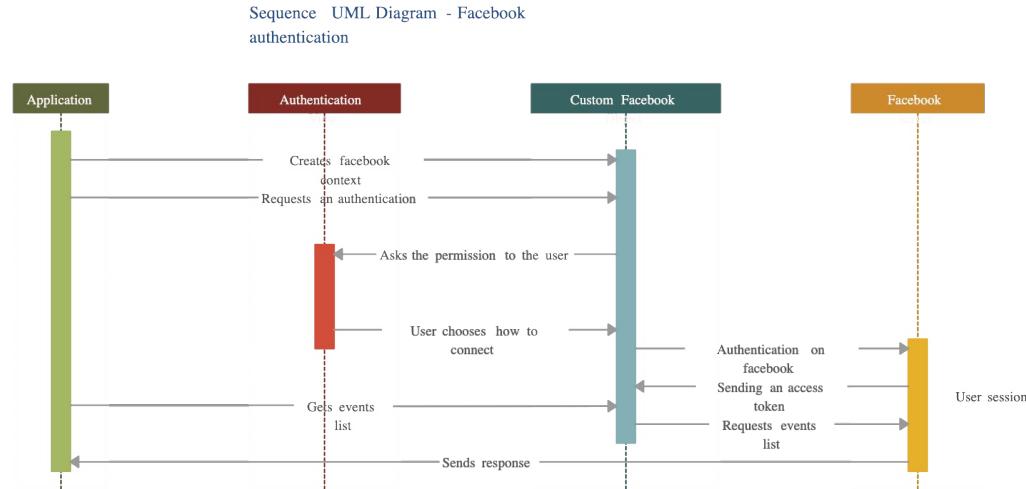
## Inside Netflix: A Deep Dive Into Its Cutting-Edge System Architecture





# ¿Qué es arquitectura de software?

- Se refiere a la estructura y diseño de un sistema de software.
- Es una representación de alto nivel que define cómo los componentes del software interactúan entre sí, cómo se organizan y cómo cumplen con los requisitos funcionales y no funcionales del sistema.





# Ventajas de la arquitectura de software

- Permite planificar el desarrollo del proyecto y elegir las mejores herramientas para cada etapa.
  - Organización eficiente
  - Reutilización de componentes : Financieramente la más importante
  - Mantenibilidad
  - Escalabilidad
  - Rendimiento
  - Seguridad
  - Colaboración
  - Adaptabilidad
  - Documentación y comunicación
  - Resolución de problemas



# Elementos de la arquitectura de software

- Se conforma mediante la combinación de varios elementos y conceptos. Estos se combinan para definir la estructura y el diseño del sistema de software :
  - Componentes
  - Conexiones
  - Patrones de diseño
  - Estilos arquitectonicos
  - Requisitos : Funcionales, no funcionales, estrategicos, etc.
  - Tecnología y herramientas
  - Documentación
  - Evolución y mantenimiento
  - Plan de comunicación



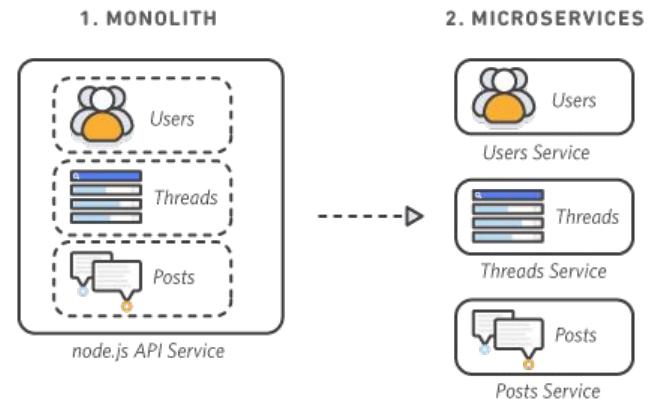
# Estilos Arquitectónicos

- Son soluciones probadas para problemas comunes en el diseño y desarrollo de aplicaciones. Se aprovecha su uso para reutilizarlas y ganar eficiencia en tiempo de implementación como en infraestructura de mantenimiento.
  - Arquitectura monolítica
  - Arquitectura orientada a servicios (SOA)
  - Modelo Vista Controlador (MVC)
  - Capas
  - Arquitectura de Microservicios
  - Arquitectura Hexagonal
  - Arquitectura de Capas de Servicios
  - Modelo Vista Vista Modelo (MVVM)



# Arquitectura Monolítica

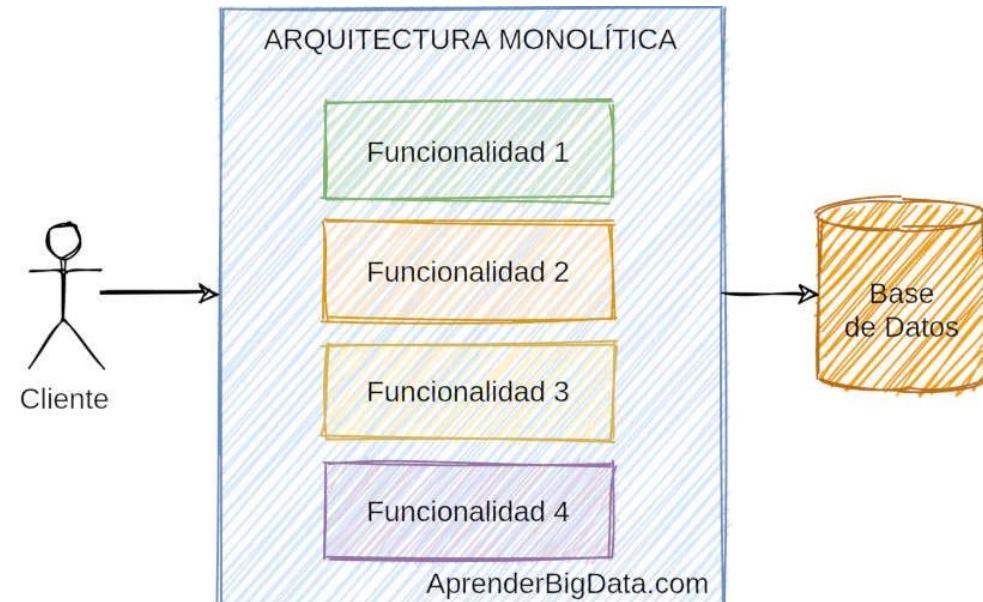
- Es un enfoque de diseño de software en el cual todas las funciones y componentes del sistema están agrupados en una única unidad o aplicación.
- Todo el software se desarrolla, implementa y despliega como una sola entidad. Este enfoque contrasta con las arquitecturas distribuidas o basadas en microservicios, donde el sistema se divide en módulos más pequeños e independientes que interactúan entre sí.
- Ciclo de vida
  - Planificación
  - Proceso de desarrollo
  - Implementación
  - Depuración
  - Modificaciones
  - Escalado





# Ventajas de la Arquitectura Monolítica

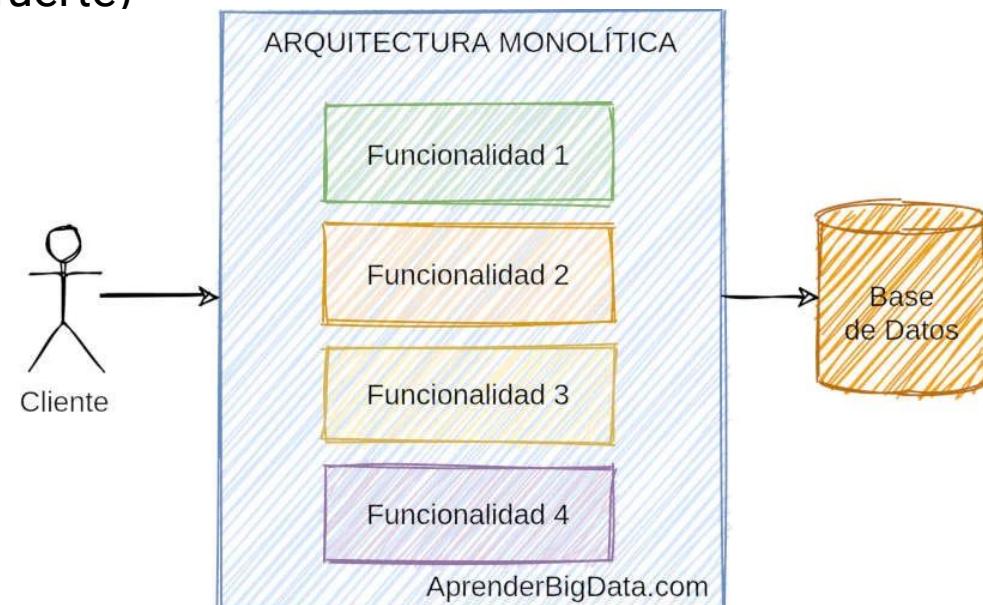
- Facilidad de despliegue
- Fácil de someter a pruebas y depuración
- Menor latencia
- Agilidad





# Desventajas de la Arquitectura Monolítica

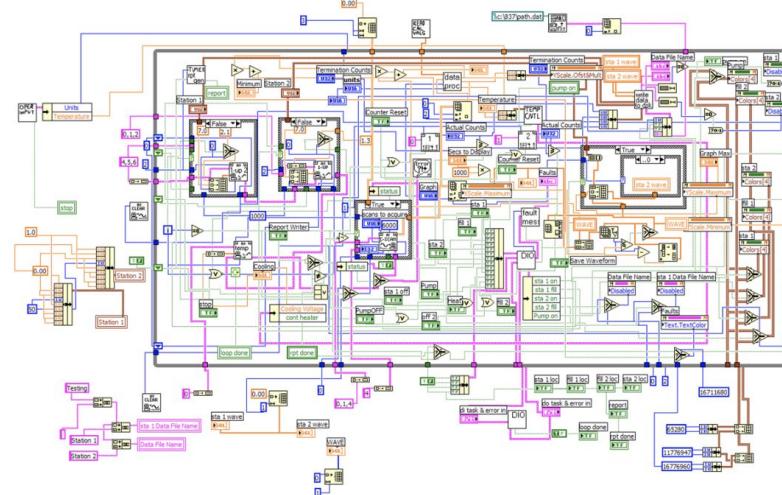
- Compleja de mantener
- Escalabilidad
- Atada a la tecnología (acoplamiento fuerte)
- Punto único de fallo





# Arquitectura Monolítica y Spaghetti Code

- Spaghetti Code es un estilo de programación caracterizado por la falta de estructura organizada y la presencia de conexiones complejas y entrelazadas entre las partes del código fuente. Este tipo de código es difícil de entender y dar mantenimiento debido a su complejidad y la falta de claridad en la estructura
  - Flujo de control no lineal
  - Dependencias globales
  - Falta de modularidad
  - Anidamiento excesivo
  - Escaso uso de comentarios



# Arquitectura Orientada a Servicios (SOA)

Según Manuel Jesús Morales Lara autor del libro "SOA : la tecnología al servicio del negocio", es una estrategia empresarial en tecnología de información que establece un marco normativo de interoperabilidad en toda la organización, de obligado conocimiento y cumplimiento por parte de todos los responsables y participantes en proyectos de tecnología, gestión de cambios, responsables funcionales, proveedores externos, área de sistemas, etc.

Servicio : Es una función bien definida, autocontenido e independiente del contexto o estado de otros servicios que agrega valor a la organización.



# Arquitectura Orientada a Servicio (SOA)

“SOA es una arquitectura de acoplamiento flexible, diseñada para conocer las necesidades del negocio de una organización”. Microsoft

“SOA” es un modelo de componentes que interrelaciona las diferentes unidades funcionales de las aplicaciones, denominadas servicios, a través de interfaces y contratos bien definidos entre esos servicios. La interfaz se define de forma neutral y debería ser independiente de la plataforma hardware, del sistema operativo y del lenguaje de programación utilizado. Esto permite a los servicios, construidos sobre sistemas heterogéneos, interactuar entre ellos de una manera uniforme y universal”. IBM





# Orientación a Servicios

- Computación orientada a servicios : un contenedor de servicios sobre una plataforma de computación distribuida donde se incluyen definiciones como diseño, patrones, lenguajes, tecnologías y marcos de trabajo.
  - Interoperabilidad : Compatibilidad con elementos de su misma naturaleza
  - Federación : Gobernanza individual y evolución necesaria
  - Plataforma neutral : No alineamiento a una arquitectura específica
  - Alineamiento de negocio y tecnología
  - Retorno de Inversión (ROI)
  - Reducción de la carga tecnológica

Servicio -> Orientación a Servicios -> Arquitectura Orientada a servicios

Arquitectura Orientada a Servicios: un enfoque arquitectónico centrado en un determinado negocio y basado en principios orientados a servicios

Orientación a Servicios: una forma de organizar y relacionar las tareas y sus respectivas salidas, con un enfoque funcional

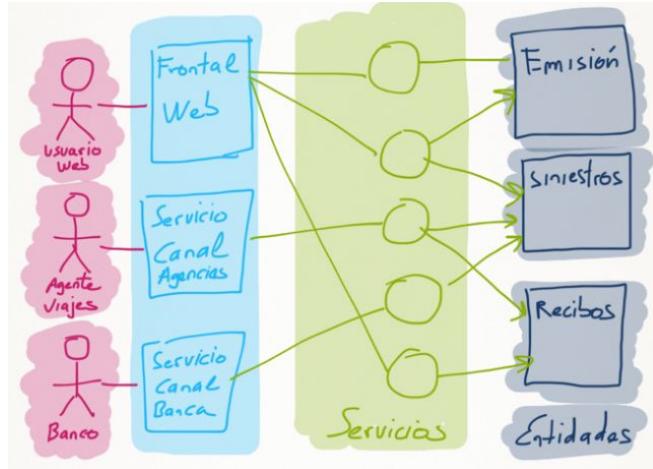
Servicio: funcionalidad consumida repetidamente





# Orientación a Servicios

- Computación orientada a servicios : un contenedor de servicios sobre una plataforma de computación distribuida donde se incluyen definiciones como diseño, patrones, lenguajes, tecnologías y marcos de trabajo.
- Reusabilidad : Puede consumirse por uno o varios canales.





# Servicios



Los servicios son las unidades lógicas que permiten la composición de soluciones en el paradigma de la orientación a servicios. Su diseño e implementación como una pieza independiente de software con características específicas obedecen al alineamiento con las estrategias de negocio en la arquitectura SOA

A cada servicio se le asigna su propia funcionalidad distintiva y es comprometido con un conjunto de capacidades relacionadas con un contexto específico, un servicio puede ser considerado como un contenedor de capacidades asociadas con un propósito común





# El modelo de un servicio

Clasificación que permite identificar a qué tipo base predefinido pertenece un servicio según ciertos criterios de utilización

- Servicio tarea : Es un servicio con un propósito único y específico asociado a la lógica de un proceso de negocio padre. Revisar NIT.
- Servicio entidad : Es un servicio reutilizable con funcionalidad agnóstica al contexto de ejecución y se asocia por lo general a una entidad de negocio, Facturar, Clientes, etc.
- Servicio utilitario : Servicio reutilizable y explícitamente separado de las especificaciones resultado del negocio y de los modelos derivados. ejemplo notificación, correo, bitácoras, etc.





# Portafolio e Inventory de Servicios

Inventario de Servicios : Colección independiente estandarizada y gobernada de servicios complementarios dentro de los límites que representa a una empresa o un segmento de la misma. Una organización puede contar con múltiples inventarios de servicios

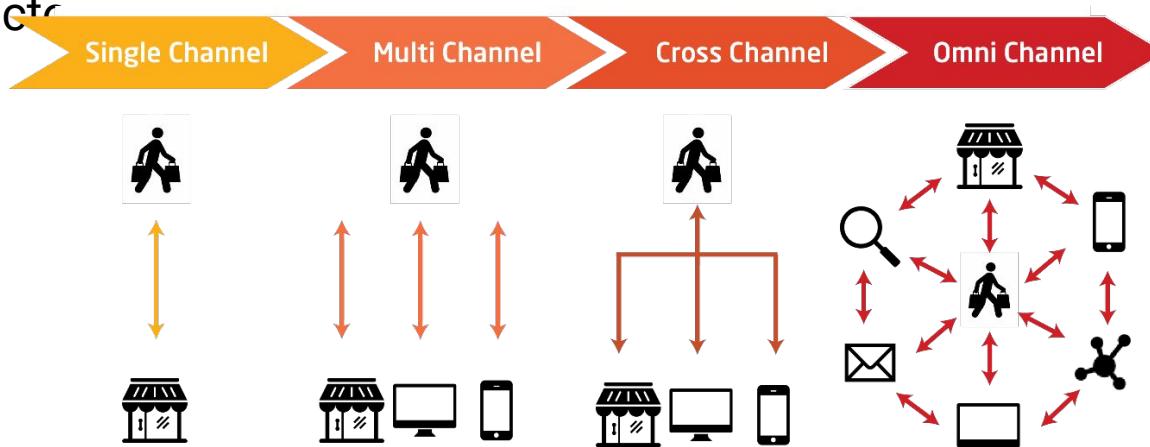
Portafolio de Servicios : Catálogo de servicios para representar un conjunto de servicios dentro de la infraestructura de tecnologías de información. Un portafolio puede contener a uno o más inventarios de servicios de una o más organizaciones





# Servicios y Canales

- La estrategia cross channel se basa en la idea de que un cliente pueda realizar una acción saltando de un canal a otro. Por ejemplo, un cliente puede encontrar un producto que le llama la atención en un post de redes sociales y de ahí, saltar a un landing page para poder ver con profundidad los detalles del producto.



# Ejemplos prácticos de SOA

1. Banca en línea : Servicios que se pueden desarrollar para permitir que los clientes puedan acceder a las cuentas, realizar transferencias, pagar facturas, solicitar préstamos, etc.
2. Comercio electrónico : Un sitio web puede implementar SOA mediante la creación de servicios para la gestión de catálogos de productos, procesamiento de pedidos, gestión de inventario, gestión de pagos, entre otros.
3. Viajes y turismo : Una agencia de viajes en línea puede utilizar SOA para desarrollar servicios que ofrezcan funcionalidades como la búsqueda de vuelos, reserva de hoteles, alquileres de automóviles, planificación de itinerarios.
4. Integración empresarial : En entornos empresariales, puede ser utilizado para facilitar la integración entre diferentes sistemas y aplicaciones.
5. Gobierno electrónico : los gobiernos pueden implementar SOA para ofrecer a los ciudadanos solicitud o renovación de documentos de identificación, presentación de impuestos, obtención de permisos, entre otros. Cada servicio se puede desarrollar de manera independiente y luego combinarse para formar un portal de servicios.



# Principios de Arquitectura SOA

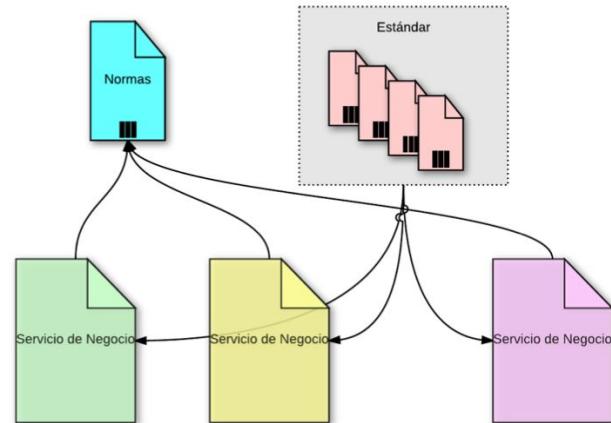
- Contratos de servicio estandarizados
- Bajo acoplamiento de servicios
- Abstracción de los servicios
- Reutilización de los servicios
- Autonomía de los servicios
- Descubrimiento de los servicios
- Servicios sin estado
- Composición de los servicios



# Principios de Arquitectura Orientada a Servicio (SOA)

## Contratos de servicios estandarizados

Todos los contratos de servicios del catálogo de servicios SOA deben cumplir los mismos estándares y normas de diseño

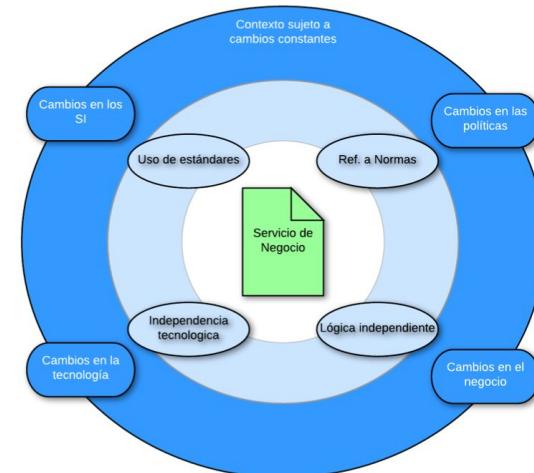


La estandarización de los Contratos de Servicios pasa por implementar el estándar y referenciar la normativa

# Principios de Arquitectura Orientada a Servicio (SOA)

## Bajo acoplamiento de servicios

Los contratos de servicios imponen requerimientos de bajo acoplamiento a los sistemas que los usan, y están ellos mismos desacoplados a su entorno.



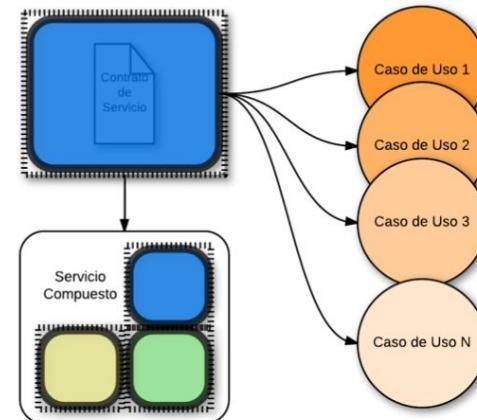
El acoplamiento tiene varias caras, y debe ser minimizado en todas para lograr servicios desacoplados, no condicionados por los cambios que se suceden en su contexto



# Principios de Arquitectura Orientada a Servicio (SOA)

## Abstracción de los servicios

Los contratos de servicios únicamente contienen información esencial para su diseño, y la información disponible sobre los servicios se limita a lo que está publicado en los contratos de servicios

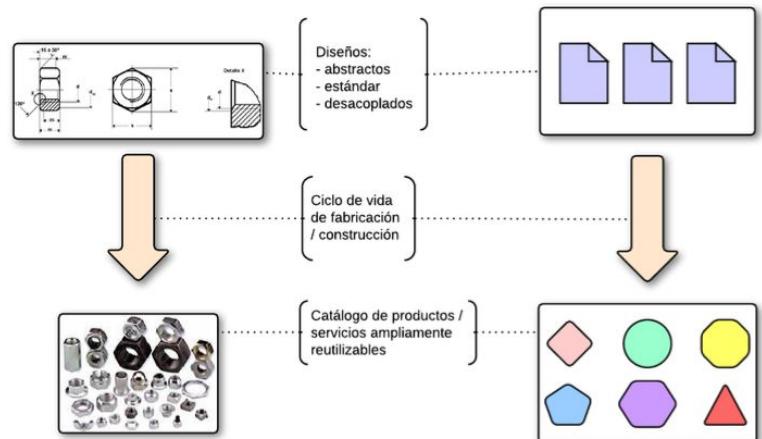


*La abstracción potencia la reutilización en distintos casos de uso y en composición de servicios*

# Principios de Arquitectura Orientada a Servicio (SOA)

## Reusabilidad de los servicios

Los servicios contienen y expresan lógica independiente y deben poder ofrecerse como recursos empresariales reutilizables



Los principios de diseño y fabricación industrial aplicados a las TIC fomentan la reutilización

# Principios de Arquitectura Orientada a Servicio (SOA)

## Descubrimiento de los servicios

Los servicios se suministran con metadatos informativos, que permiten que sean descubiertos e interpretados

## Autonomía de los servicios

Los servicios tienen un alto control sobre sus entornos en tiempo de ejecución

## Servicios sin estado

Los servicios minimizan el uso de recursos, evitando el mantenimiento de información de estado siempre que sea posible



# Arquitectura Orientada a Servicio (SOA)

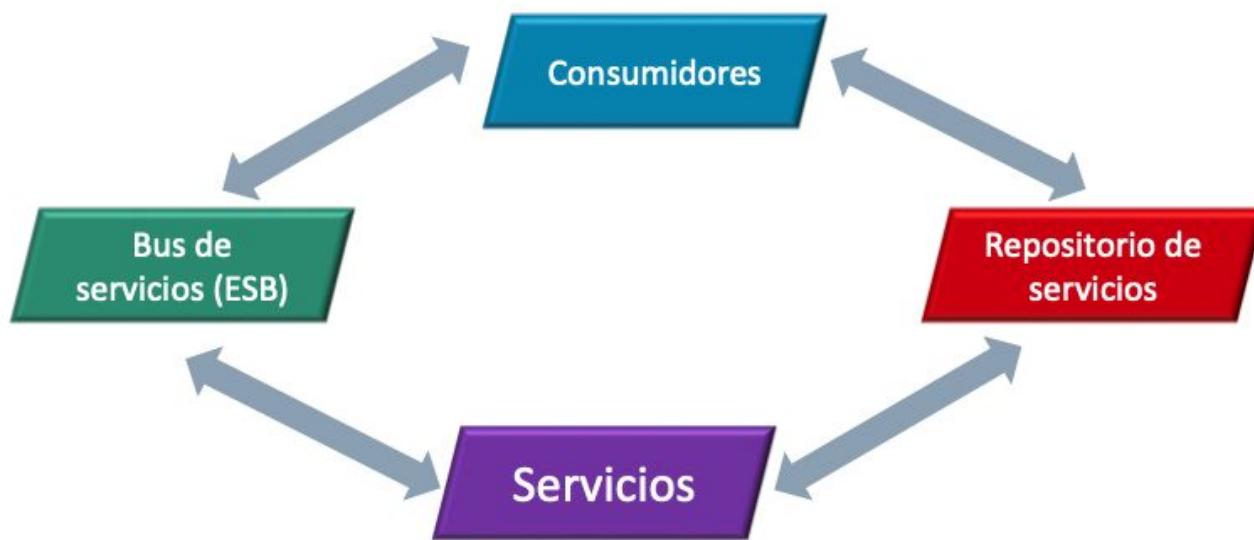
- Es una colección de servicios comunicados entre sí
- La conexión puede involucrar el intercambio de un dato o la coordinación de una actividad completa

En SOA la funcionalidad deseada se descompone en unidades (servicios) que puede ser distribuidos en diferentes nodos conectados a través de una red y de igual forma son combinados para alcanzar un resultado deseado



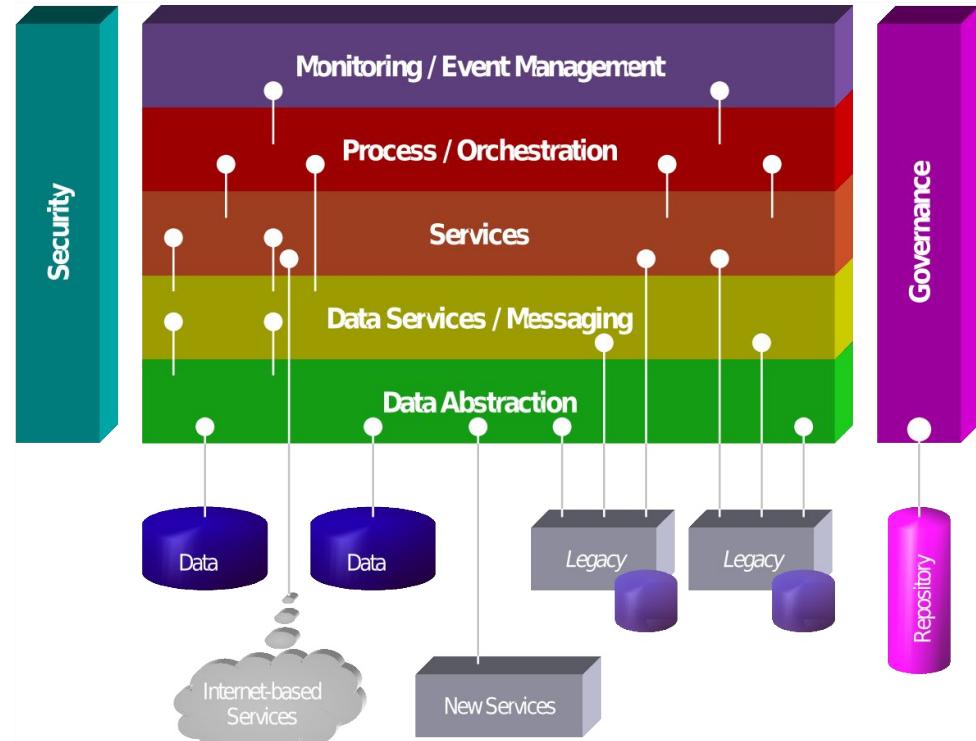
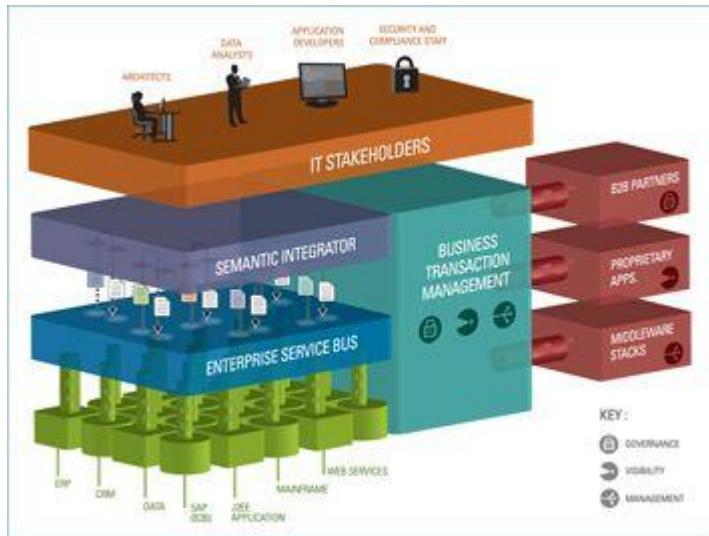
# Arquitectura Orientada a Servicio (SOA)

Componentes



# Arquitectura Orientada a Servicio (SOA)

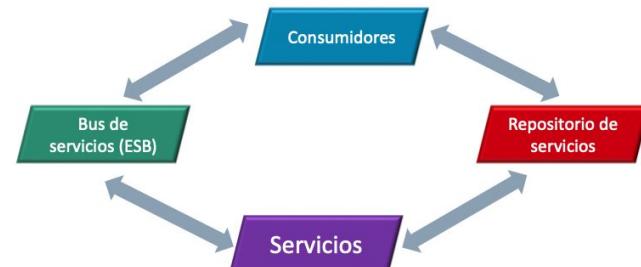
## Arquitectura



# Arquitectura Orientada a Servicio (SOA)

## Consumidores

- Puede ser una aplicación, módulo u otro servicio
- Demanda la funcionalidad que el servicio proporciona
- Ejecuta en una interfaz definida
- Estima que se le ofrezca el mismo SLA que a los demás
  - SLA : Service Level Agreement, deben cumplir los objetivos SMART
  - SMART :
    - Specific : Específico
    - Measurable : Medible
    - Achievable : Alcanzable
    - Relevant : Relevante
    - Timely : Dentro de un marco de tiempo



# Arquitectura Orientada a Servicio (SOA)

## Servicios

- Componentes reutilizable de software



# Arquitectura Orientada a Servicio (SOA)

## Repositorio de servicio

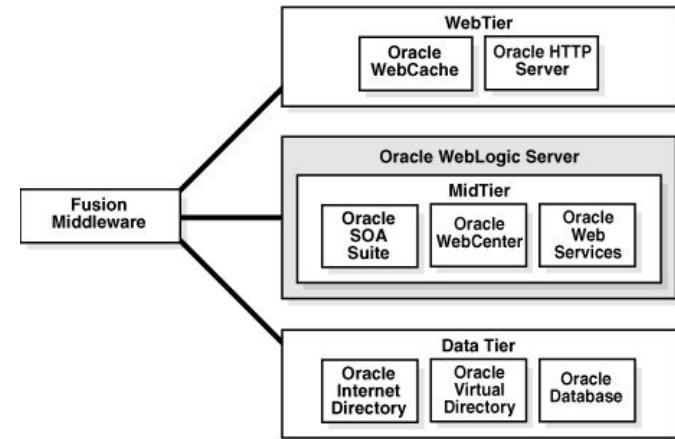
- Facilita la búsqueda de servicios
- Permite la adquisición de la información necesaria para uso de servicios
- Fuera del tiempo y función del proyecto para el que se crearon



# Arquitectura Orientada a Servicio (SOA)

## Bus de servicios

- Software (Middleware) que conecta los servicios con sus consumidores y proporciona:
  - Conectividad
  - Soporte a la heterogeneidad de tecnologías
  - Soporte a la heterogeneidad de paradigmas de comunicación



# Mitos de SOA

Mito	Realidad
SOA es una tecnología	SOA es una filosofía independiente del producto, la tecnología o la industria. No es posible ofrecer a una organización
SOA requiere servicios Web	SOA puede ser realizado a través de Servicios Web, pero los servicios Web no necesariamente requieren la implementación de SOA.
SOA es nuevo y revolucionario	Existen tecnologías para el desarrollo de componentes de software distribuido como DCOM y arquitecturas como CORBA desde 1991, que facilita el desarrollo de aplicaciones distribuidas en entornos heterogéneos.
SOA asegura la alineación de la tecnología al negocio	SOA no es una metodología
Necesitamos construir un SOA	SOA es un medio, no un fin.





# Bibliografía

- "Software Architecture in Practice" (Arquitectura de Software en la Práctica), Len Bass, Paul Clements, Rick Kazman
- "Pattern-Oriented Software Architecture: A System of Patterns" (Arquitectura de Software Orientada a Patrones: Un Sistema de Patrones), Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, Michael Stal
- "Domain-Driven Design: Tackling Complexity in the Heart of Software" (Diseño Dirigido por Dominios: Abordando la Complejidad en el Corazón del Software), Eric Evans
- "Clean Architecture: A Craftsman's Guide to Software Structure and Design" (Arquitectura Limpia: Guía del Artesano para la Estructura y Diseño de Software), Robert C. Martin (Uncle Bob)
- "Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions" (Patrones de Integración Empresarial: Diseño, Construcción e Implementación de Soluciones de Mensajería, Gregor Hohpe, Bobby Woolf)
- "Microservices Patterns: With Examples in Java" (Patrones de Microservicios: Con Ejemplos en Java), Chris Richardson