MicroServicios

Techiepoint #2

Sergio Maurenzi

Arquitectura e Innovación – IT
Telecom Argentina

30-Abr-2015

AGENDA

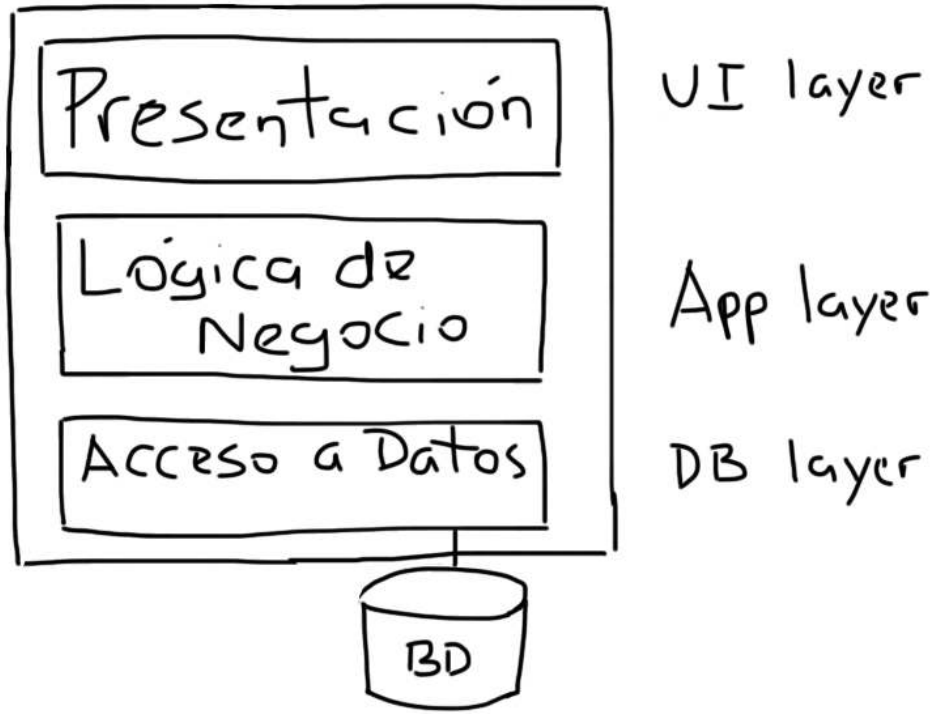
- Arquitectura Monolítica vs. SOA vs. Microservicios
- Descomposición de Servicios
- Ventajas / Desventajas
- Características de los Microservicios
- Primeras experiencias
- Conclusión

AGENDA

- **Arquitectura Monolítica vs. SOA vs. Microservicios**
- Descomposición de Servicios
- Ventajas / Desventajas
- Características de los Microservicios
- Primeras experiencias
- Conclusión

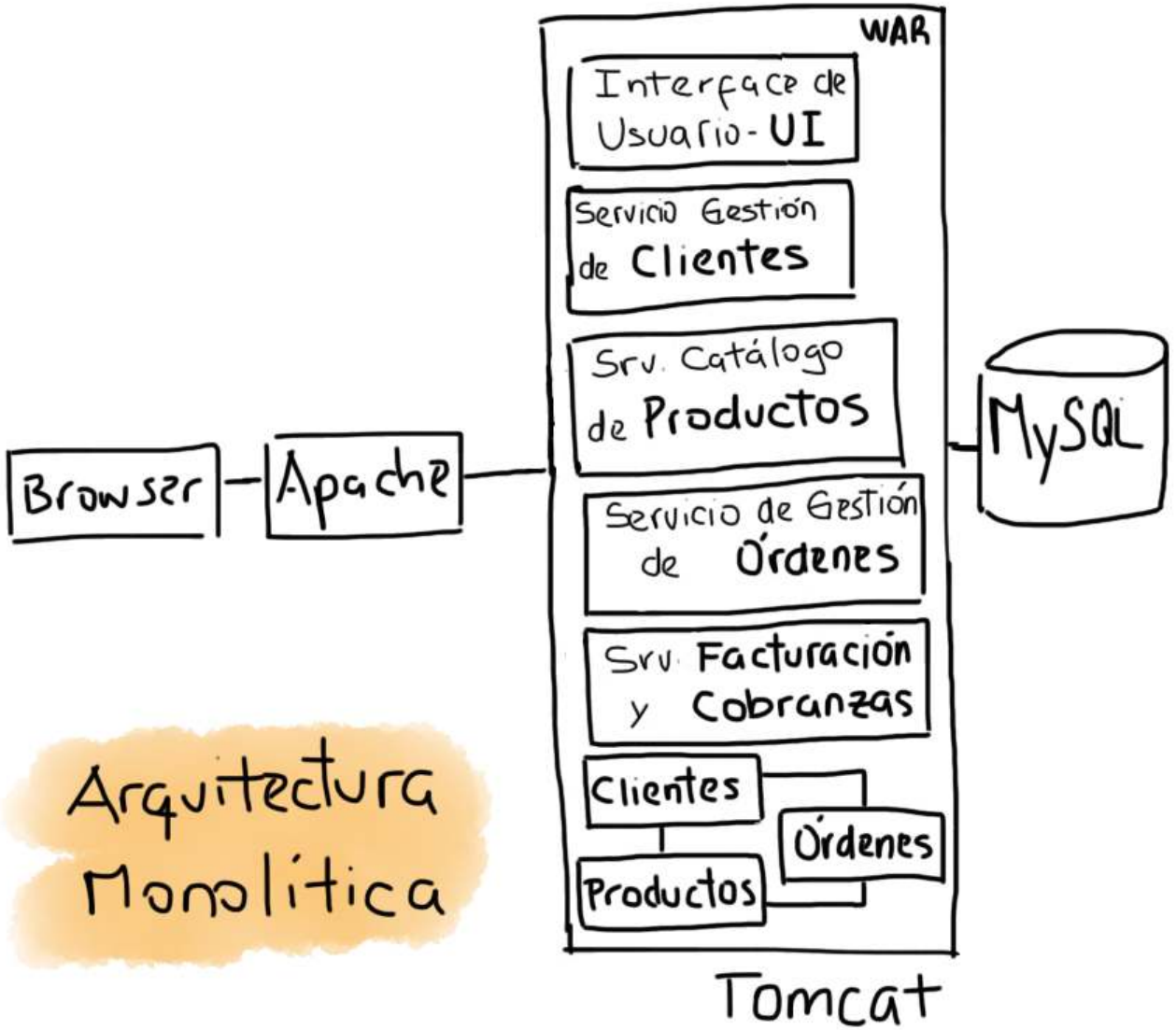

Arquitectura Monolítica

Aplicación simple – tres capas



Arquitectura de 3 capas

Arquitectura Monolítica - Simple CRM



Arquitectura Monolítica – Pros y Contras

Para aplicaciones simples …

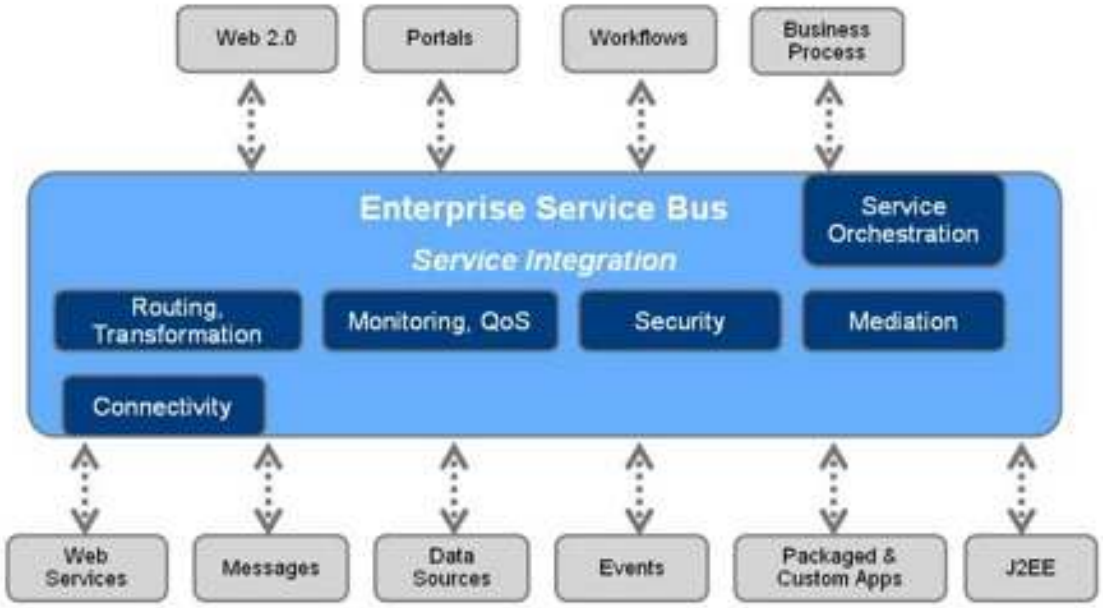
- **Simple de desarrollar** - único ciclo de vida; IDEs preparadas para una única aplicación
- **Simples de Probar (testing)** - pruebas sobre una única aplicación
- **Simples de desplegar** - copiar un archivo o un directorio, a un equipo (servidor)
- **Simple de escalar** - se pueden ejecutar múltiples copias de la aplicación detrás de un balanceador de carga.

Para aplicaciones complejas …

- **Complejo manejar Cambios** – por cada cambio, → redesplegar toda la aplicación. +Riesgos → +miedos → +tests → +tiempos → +coordinación y +comunicación
- **Mayores tiempos** – el negocio debe esperar
- **Calidad no garantizada** – por dependencias en el código.
- **Escalabilidad y Resilencia disminuida** – si un módulo requiere escalar lo debemos hacer para toda la aplicación. Si algo falla, todo deja de funcionar.

SOA (Service Oriented Architecture)

La **Arquitectura Orientada a Servicios** (SOA) es un paradigma de arquitectura para diseñar y desarrollar sistemas distribuidos. Las soluciones SOA han sido creadas para satisfacer los objetivos de negocio los cuales incluyen facilidad y flexibilidad de integración con sistemas legados, alineación directa a los procesos de negocio reduciendo costos de implementación, innovación de servicios a clientes y una adaptación ágil ante cambios incluyendo reacción temprana ante la competitividad.(*)



(*) Wikipedia
http://es.wikipedia.org/wiki/Arquitectura_orientada_a_servicios

Qué son los Microservicios?

Microservicios*: es un estilo de arquitectura, una forma de desarrollar una aplicación, basado en un conjunto de pequeños servicios, cada uno corriendo en sus propios procesos y comunicándose mediante mecanismos livianos, generalmente un recurso API de HTTP.

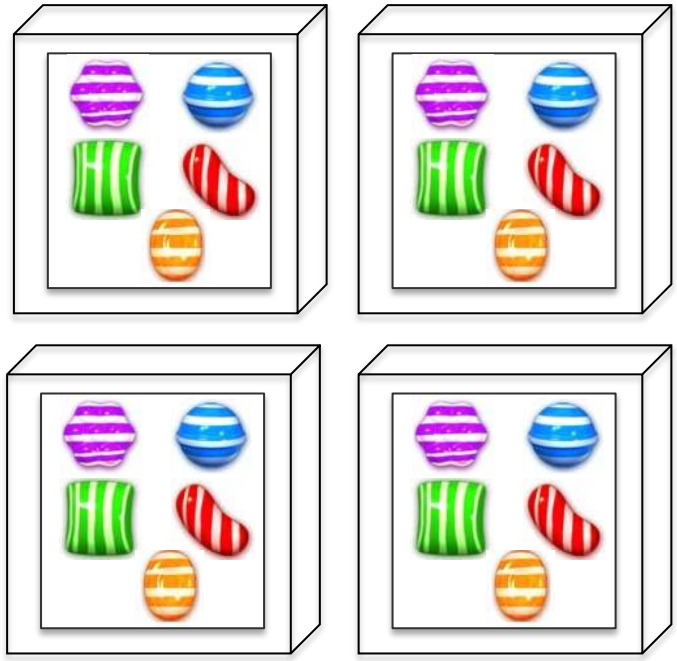
Estos servicios son altamente desacoplados, construidos alrededor de funcionalidades de negocio y desplegados de manera independiente a través de una maquinaria de despliegue completamente automatizada.

(*) Según Martin Fowler y James Lewis - Thoughtworks
http://martinfowler.com/articles/microservices.html

Arquitectura Monolítica vs. Microservicios

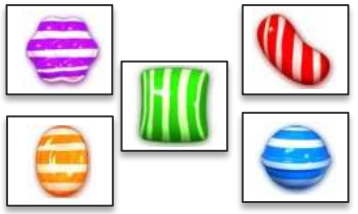
Una aplicación monolítica pone todas sus funcionalidades en un único proceso…

…y escala replicando lo monolítico en múltiples servidores

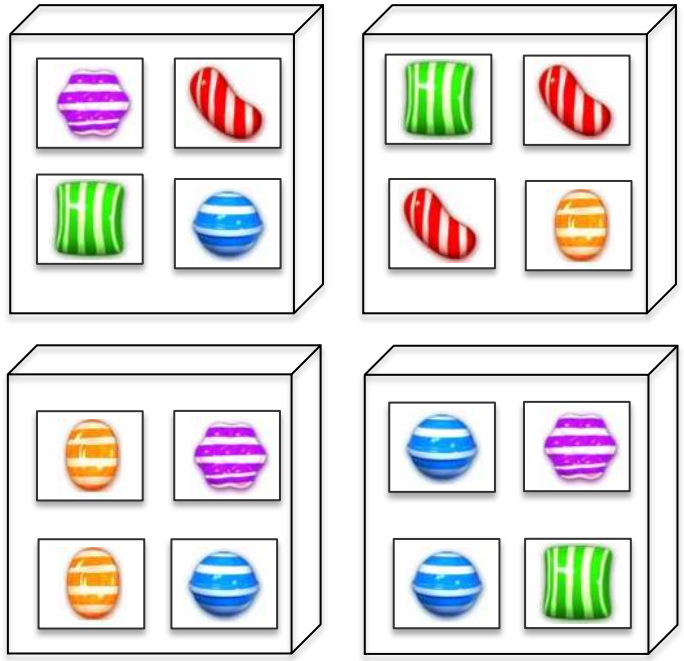


Arquitectura Monolitica

Una arquitectura de microservicios pone cada elemento de funcionalidad en un servicio separado…



…y escala distribuyendo dichos servicios a través de los servidores, según se requiera

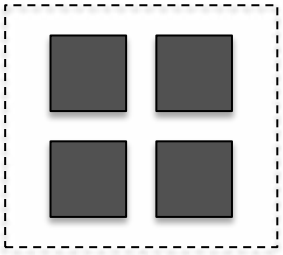


Arquitectura Microservicios

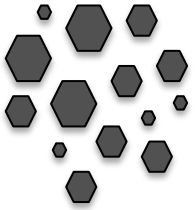
Monolítico vs. SOA vs. Microservicios

Monolítico

SOA



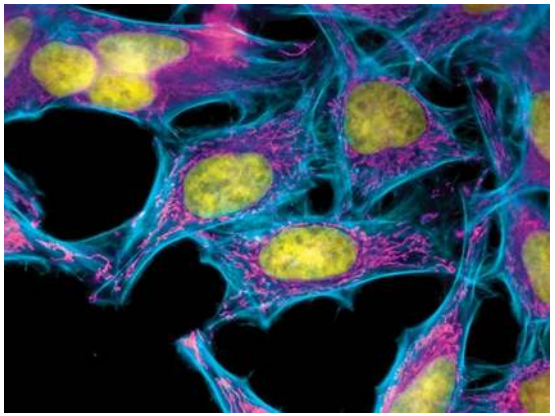
Microservicios



1990´s **2000´s** **2010´s**

Acoplamiento fuerte

Acoplamiento débil



Desacoplado

Personal Arnet GRUPO TELECOM

Ej. evolución arq. Monolítica a Microservicios

Monolítico C++ → Perl / C++ → Java / Scala → microservicios

Monolítico Perl → Monolítico C++ → Java → microservicios

Monolítico Rails → JS / Rails / Scala → microservicios

Monolítico Java → Java / Scala → microservicios

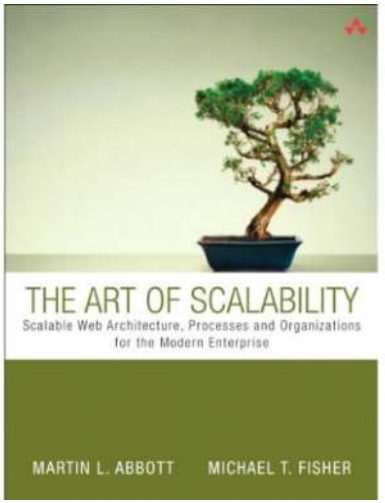
Drivers comunes

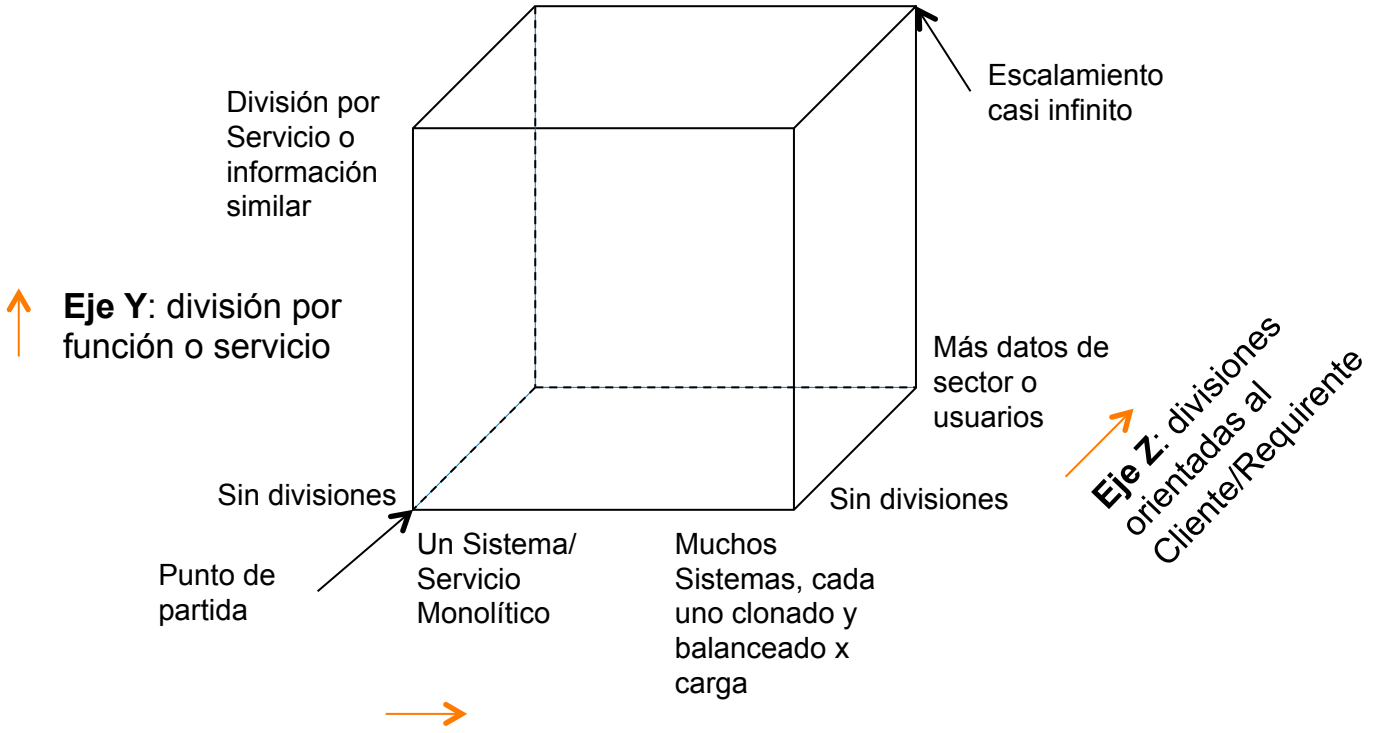
1) Rápido **Crecimiento**
2) Control de **Costos**
3) Aumentar la **Velocidad**

AGENDA

- Arquitectura Monolítica vs. Microservicios vs. SOA
- **Descomposición de Servicios**
- Ventajas / Desventajas
- Características de los Microservicios
- Primeras experiencias
- Conclusión

El cubo de escalamiento aplicativo

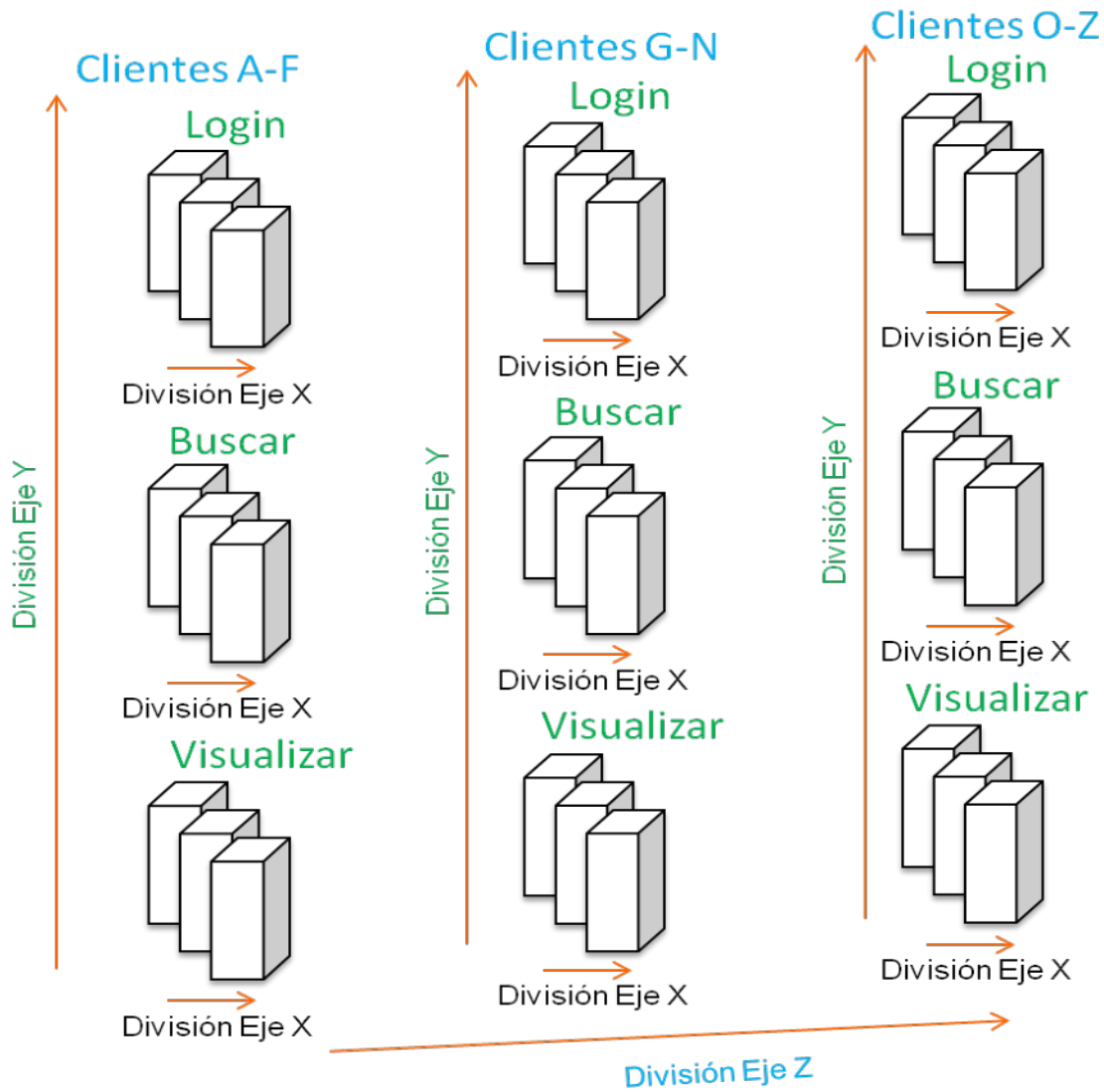




Cubo de Escalamiento Aplicativo

Personal Arnet GRUPO TELECOM

Ej. escalamiento aplicativo en los 3 ejes



Escalabilidad en 3 ejes

Single Responsibility Principle – Principio Responsabilidad Individual

"There should never be more than one reason for a class to change"

Se deben reunir aquellas cosas que cambian por la misma razón y separar aquellas cosas que cambian por diferentes razones

Demasiadas responsabilidades en una sola cosa pueden causar problemas

Ej. simple: diseño de una clase que llama a un número de móvil específico, con previa validación del mismo

SRP violado

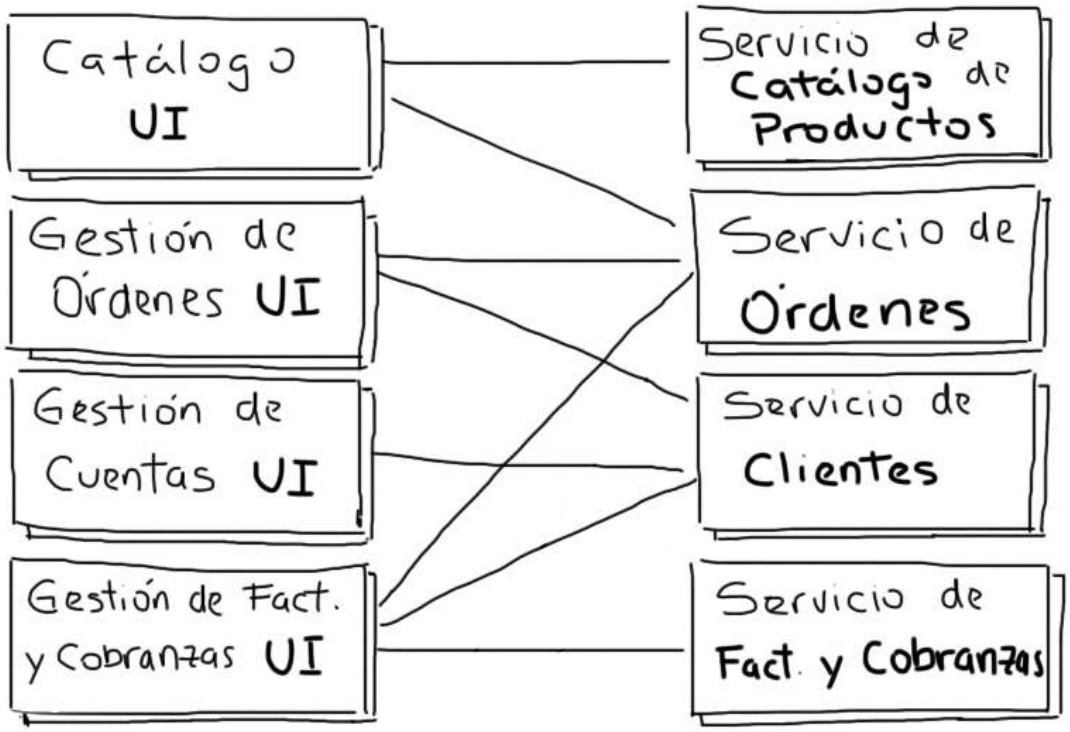
```
1  public class MakeCallToMobile
2  {
3    public void callMobile(MobileNo mobileNo)
4    {
5      if(validateMobileNumber(mobileNo))
6      {
7         //code to make a call on the mobile number
8      }
9    }
10
11   public boolean validateMobileNumber(MobileNo mobileNo)
12   {
13     //Verify if the mobile no is valid.
14   }
15 }
```

SRP válido

```
1  public class MakeCallToMobile
2  {
3    public void callMobile(MobileNo mobileNo)
4    {
5      if(ValidationeService.validateMobileNumber(mobileNo))
6      {
7         //code to make a call on the mobile number
8      }
9    }
10 }
11
12 public class ValidationeService
13 {
14   public static boolean validateMobileNumber(MobileNo mobileNo)
15   {
16     //Verify if the mobile no is valid.
17   }
18 }
```

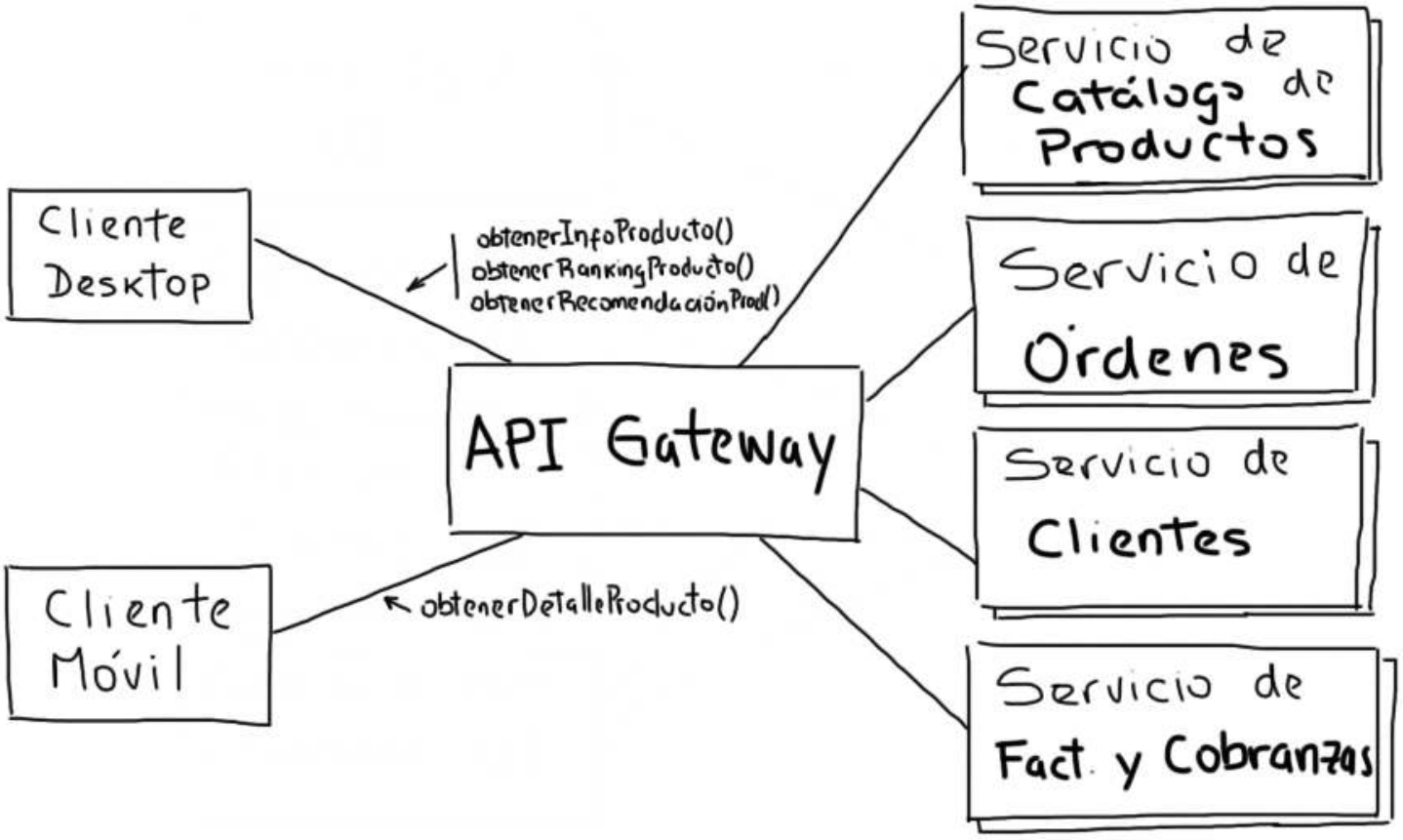
(*) SRP - Principio de diseño de Programación Orientada a Objetos - Robert C. Martin
http://www.objectmentor.com/resources/articles/srp.pdf

Arquitectura Microservicios – Ej. CRM



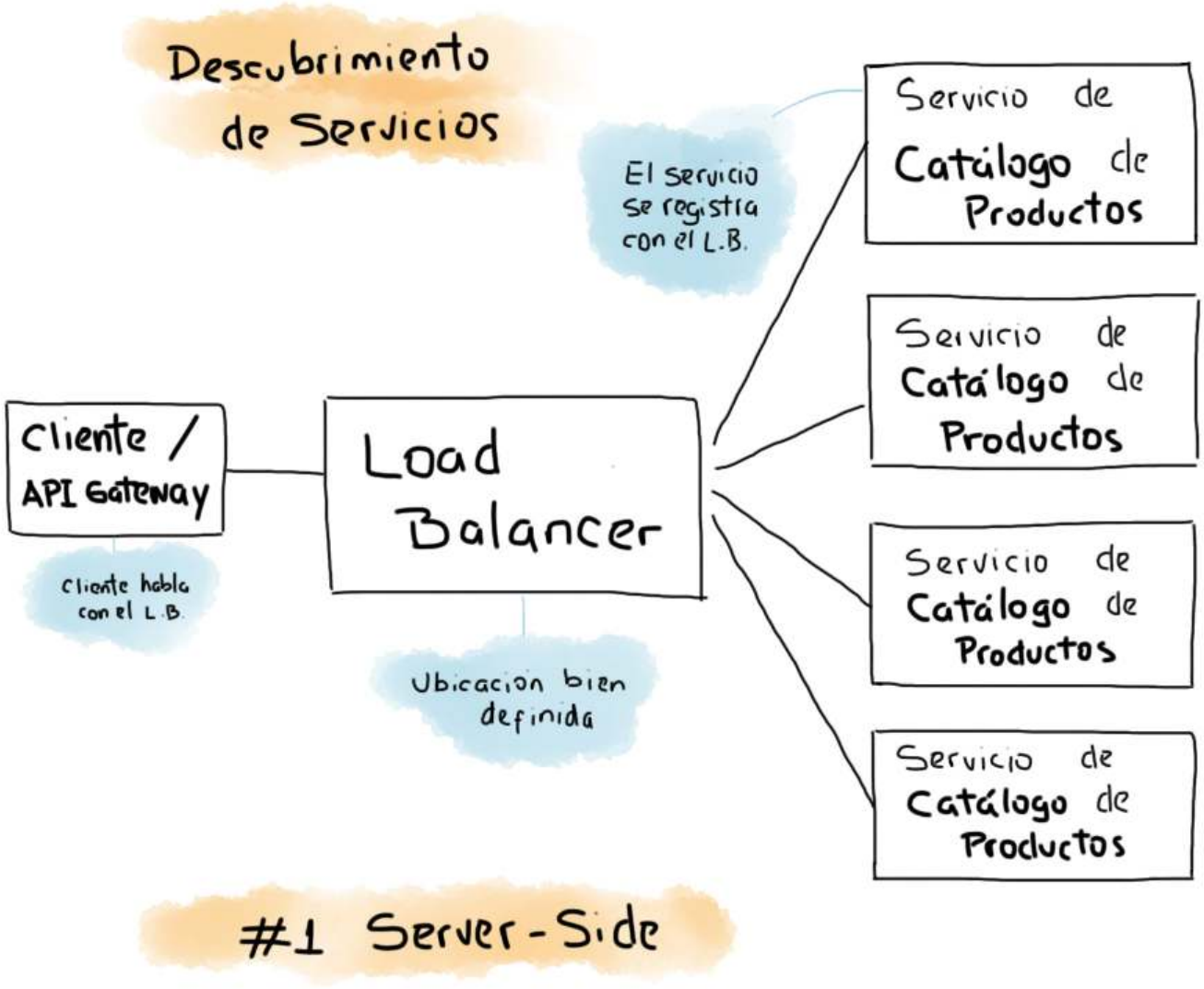
Arquitectura de Microservicios

Arquitectura Microservicios – Cómo interactúan los clientes?

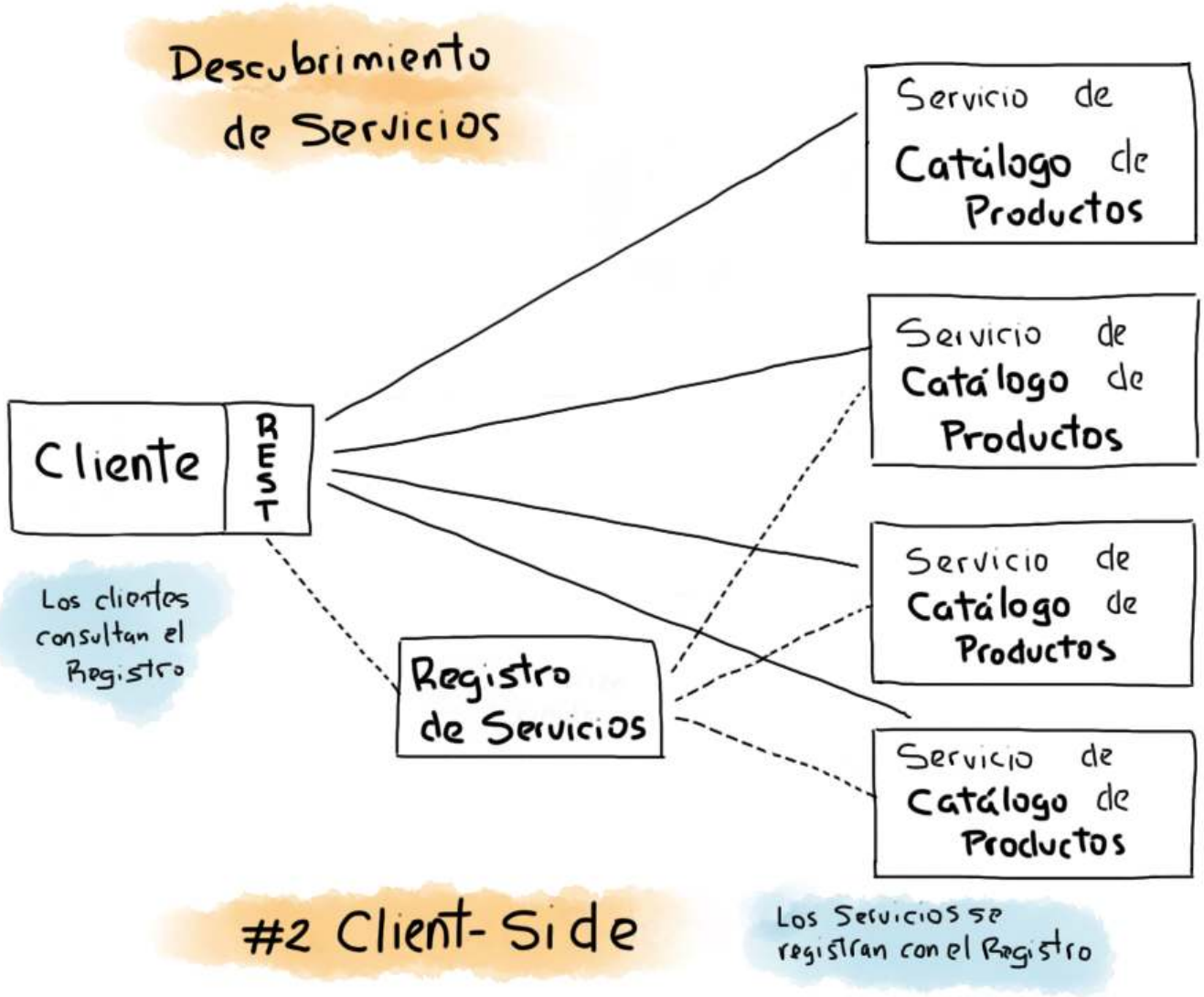


Servicios a través de API Gateway

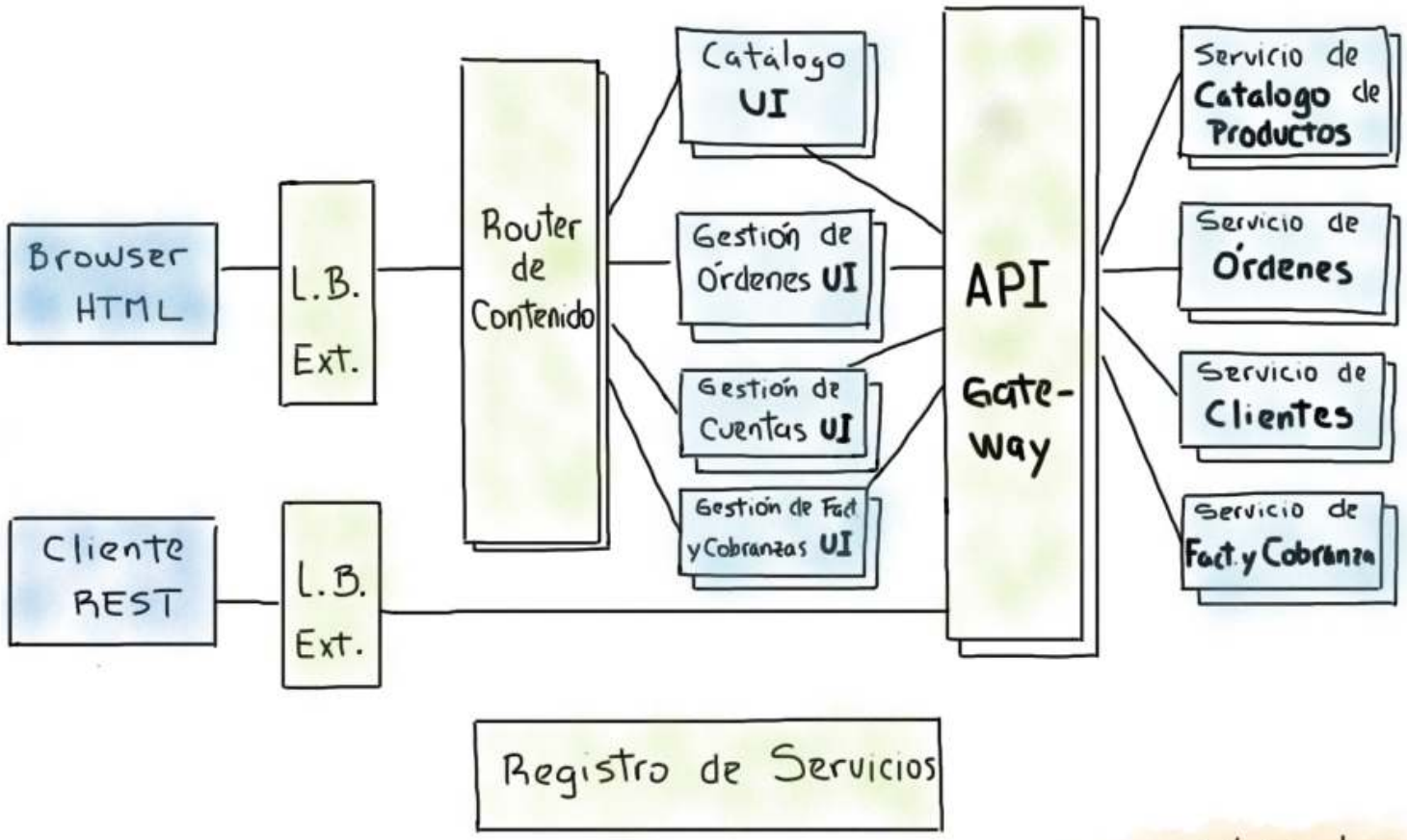
Arquitectura Microservicios – Cómo interactúan los clientes?



Arquitectura Microservicios – Cómo interactúan los clientes?



Arquitectura Microservicios – Ej. CRM



Arq de Microservicios ⇒ múltiples partes móviles

AGENDA

- Arquitectura Monolítica vs. Microservicios vs. SOA
- Descomposición de Servicios
- **Ventajas / Desventajas**
- Características de los Microservicios
- Primeras experiencias
- Conclusión

Microservicios

Ventajas / Desventajas

Arquitectura Microservicios – Pros y Contras

- **Servicios pequeños** – mejor entendimiento de los desarrolladores
- **Despliegue independiente** – autonomía y velocidad en la cadena de despliegue
- **Desarrollo escalable** - organiza el esfuerzo de desarrollo en varios equipos
- **Aislamiento de fallas** – se afecta sólo el servicio donde se produce la falla
- **Libre selección de tecnología** – libera compromiso tecnológico y habilita probar nuevas tecnologías más fácilmente

- **Mayor complejidad** – desarrollo y testing mas complejo por sistema distribuido
- **Transacciones distribuidas por múltiples servicios** – manejo más complejo
- **Complejidad en la implementación** – complejidad operativa de despliegue, monitoreo y gestión

AGENDA

- Arquitectura Monolítica vs. SOA vs. Microservicios
- Descomposición de Servicios
- Ventajas / Desventajas
- **Características de los Microservicios**
- Primeras experiencias
- Conclusión

#1 Componentización a través de servicios

Componentes = Servicios, No Bibliotecas

#2 Organizado en torno a capacidades de negocio

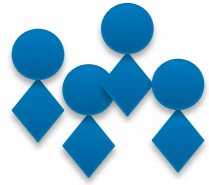
Ley de Conway: *"Any organization that designs a system (defined broadly) will produce a design whose structure is copy of the organization's communication structure."*

-- Melvyn Conway, 1967

Organización Tradicional

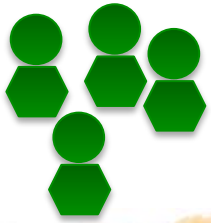
Organización

Especialistas UI



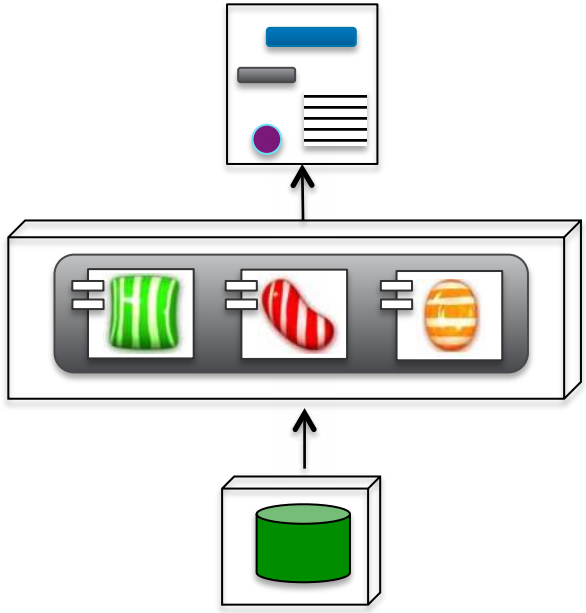
Especialistas Middeware

DBAs



Equipos funcionales organizados en silos ...

Arquitectura Aplicativa

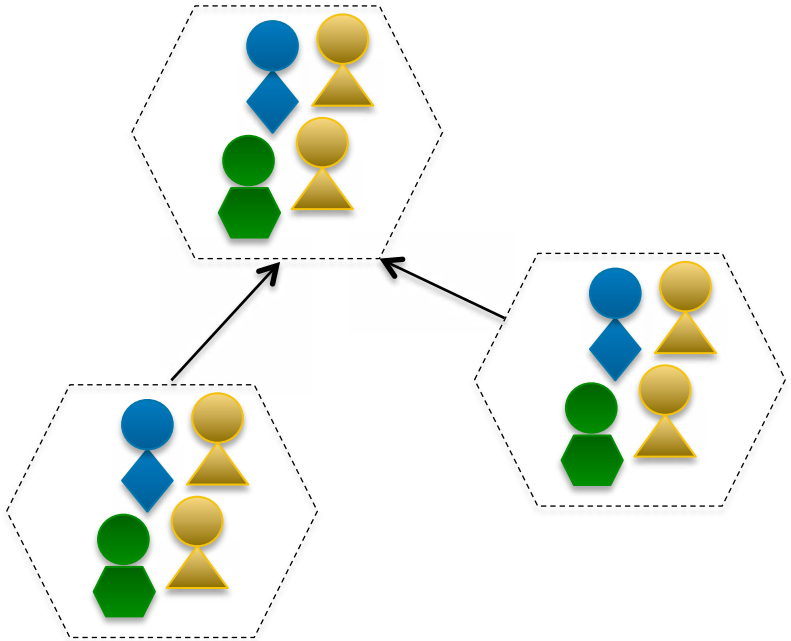


... llevan a arquitecturas aplicativas en silos

#2 Organizado en torno a capacidades de negocio

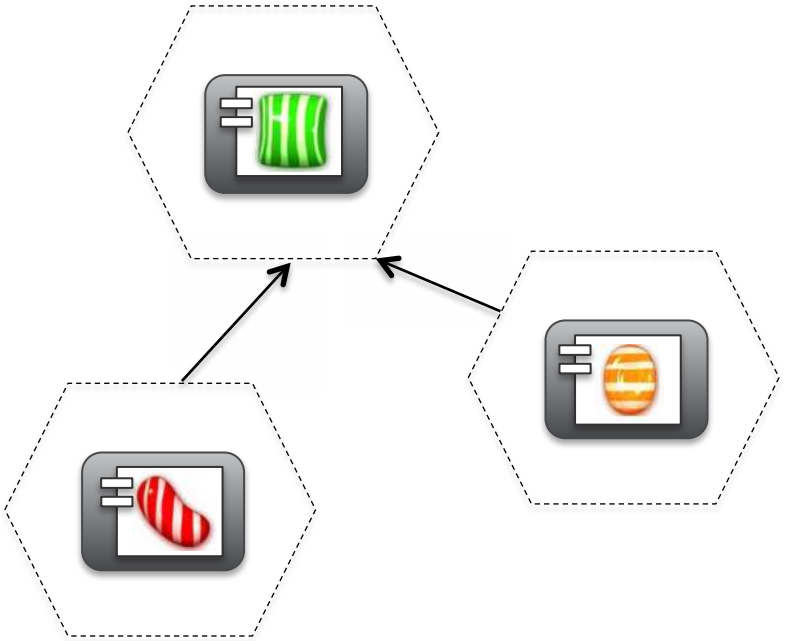
Organización Microservicios

Organización



Equipos Cross-funcionales ...

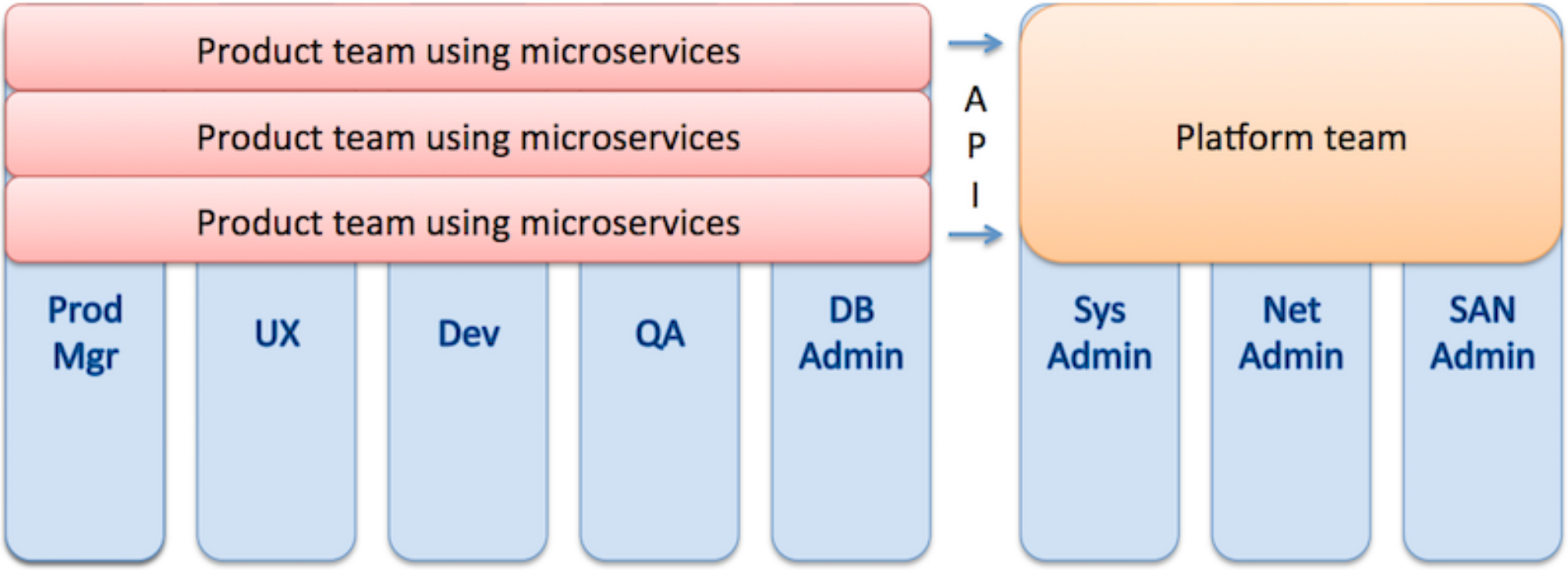
Arquitectura Aplicativa



... Organizados en torno a las capacidades ... por la ley de Conway

#2 Organizado en torno a capacidades de negocio

Organización en Netflix



Fuente: Adrian Cockcroft

#3 Productos, no Proyectos

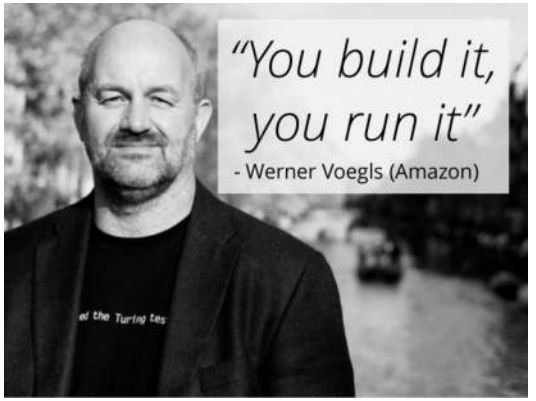
Products not Projects

"*Delivery teams run software products - not projects – that run from inception to retirement*"

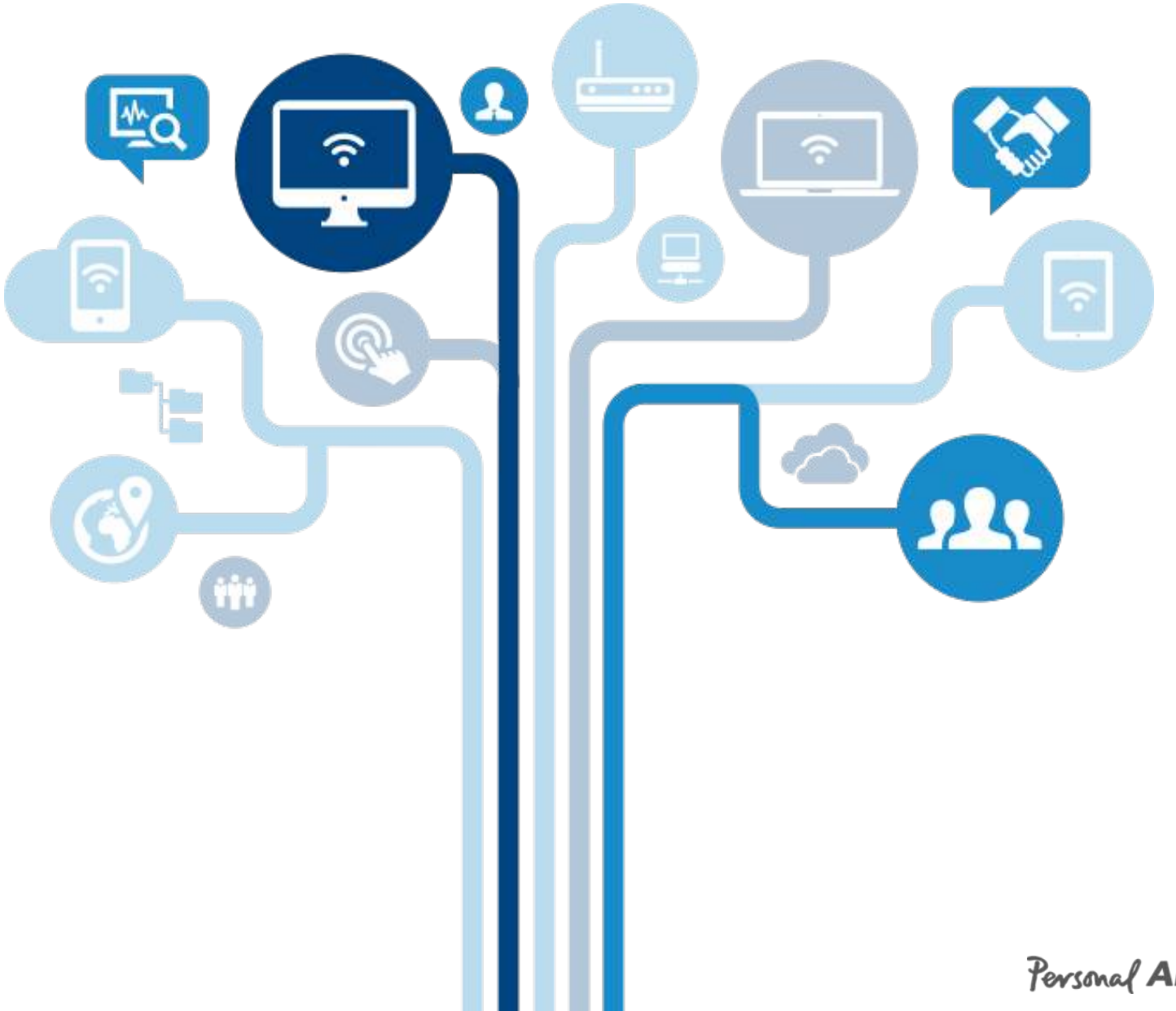
– *Jez Humble, (Thoughtworks)*

https://sites.google.com/a/jezhumble.net/devops-manifesto/

DevOpsGuys



#4 Extremos inteligentes, tuberías bobas



Personal Arnet GRUPO TELECOM

#4 Extremos inteligentes, tuberías bobas

Orquestación

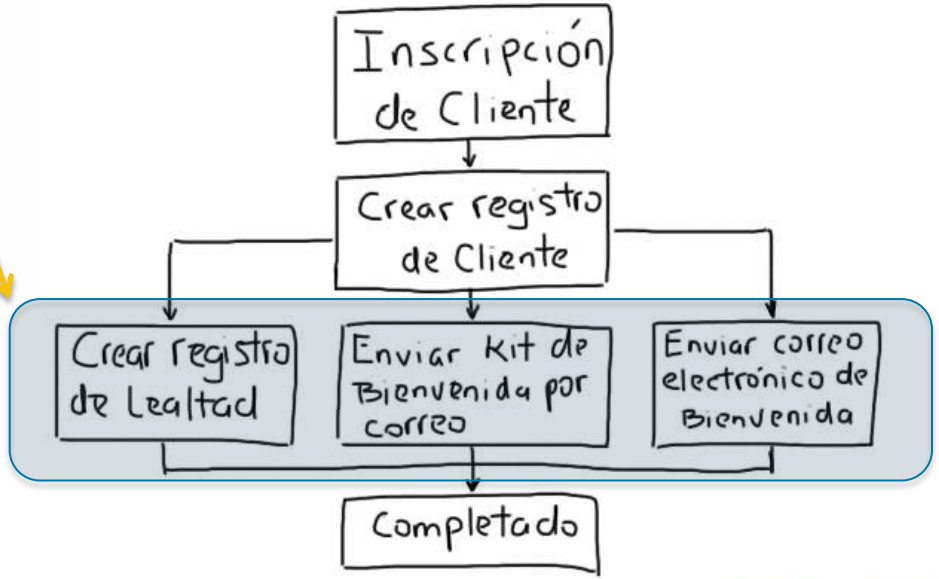
Vs.

Coreografía

Ej. Orquestación vs. Coreografía

Servicio CRM, pasos posteriores al alta de un nuevo cliente

1. Crear registro de fidelidad/lealtad y acreditarle una cantidad de puntos de bienvenida
2. Enviarle un kit de bienvenida, a través del correo postal
3. Enviarle un correo electrónico de bienvenida al cliente

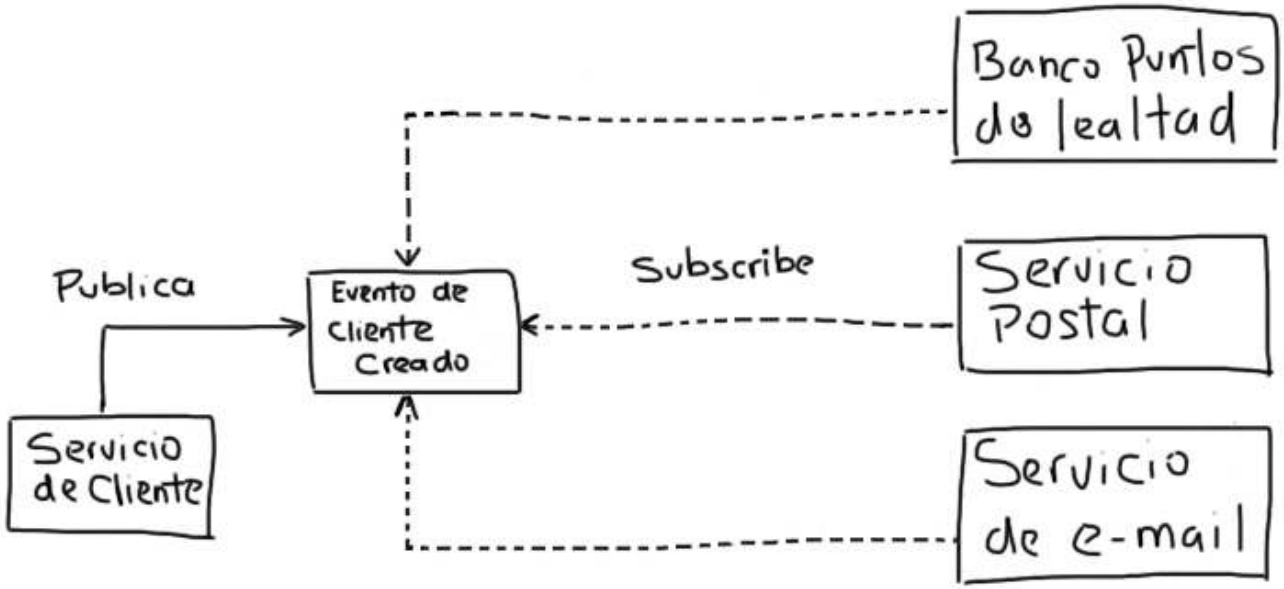


Proceso para crear un nuevo cliente

Ej. Orquestación vs. Coreografía, cont.

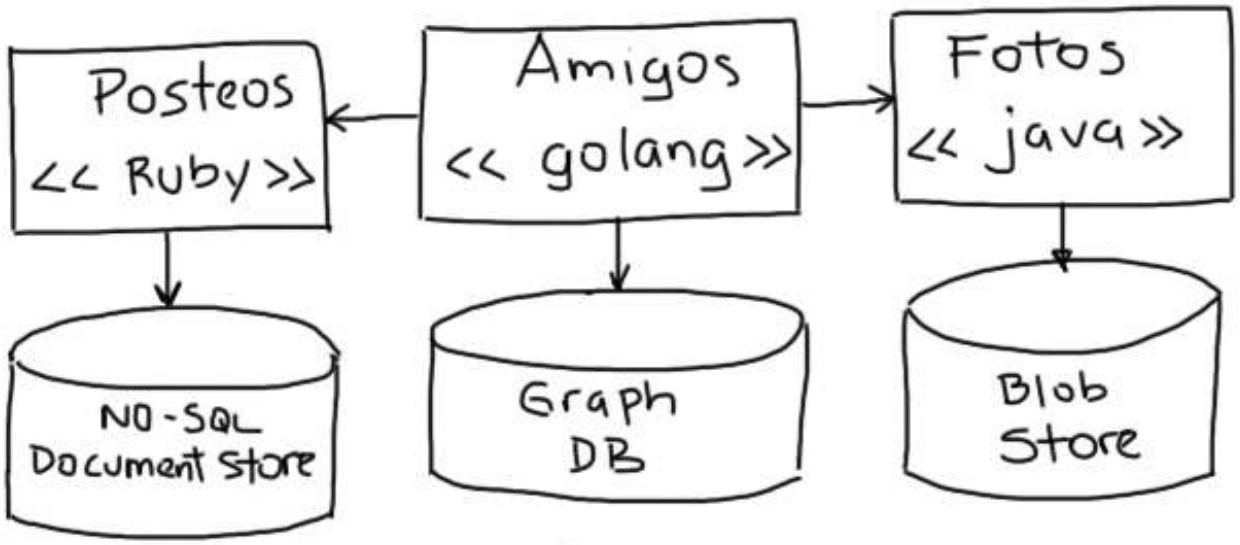
Creación de Cliente vía Orquestación

Ej. Orquestación vs. Coreografía, cont.



Creación de Cliente vía Coreografía

#5 Gobierno descentralizado

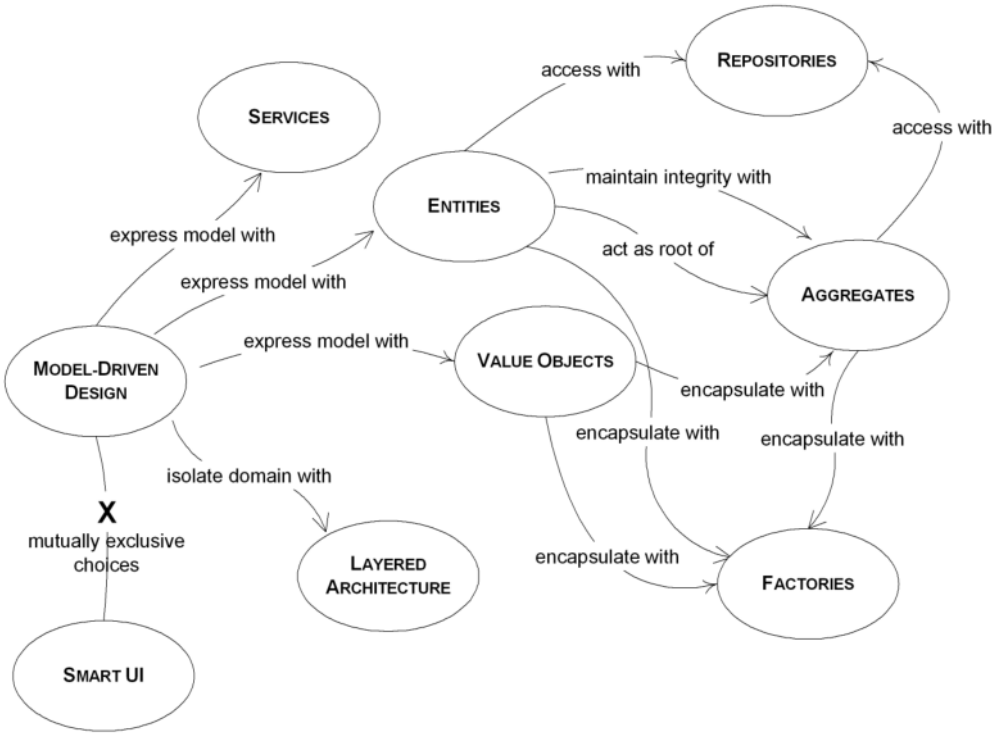


Heterogeneidad Tecnológica

#6 Gestión de datos descentralizada

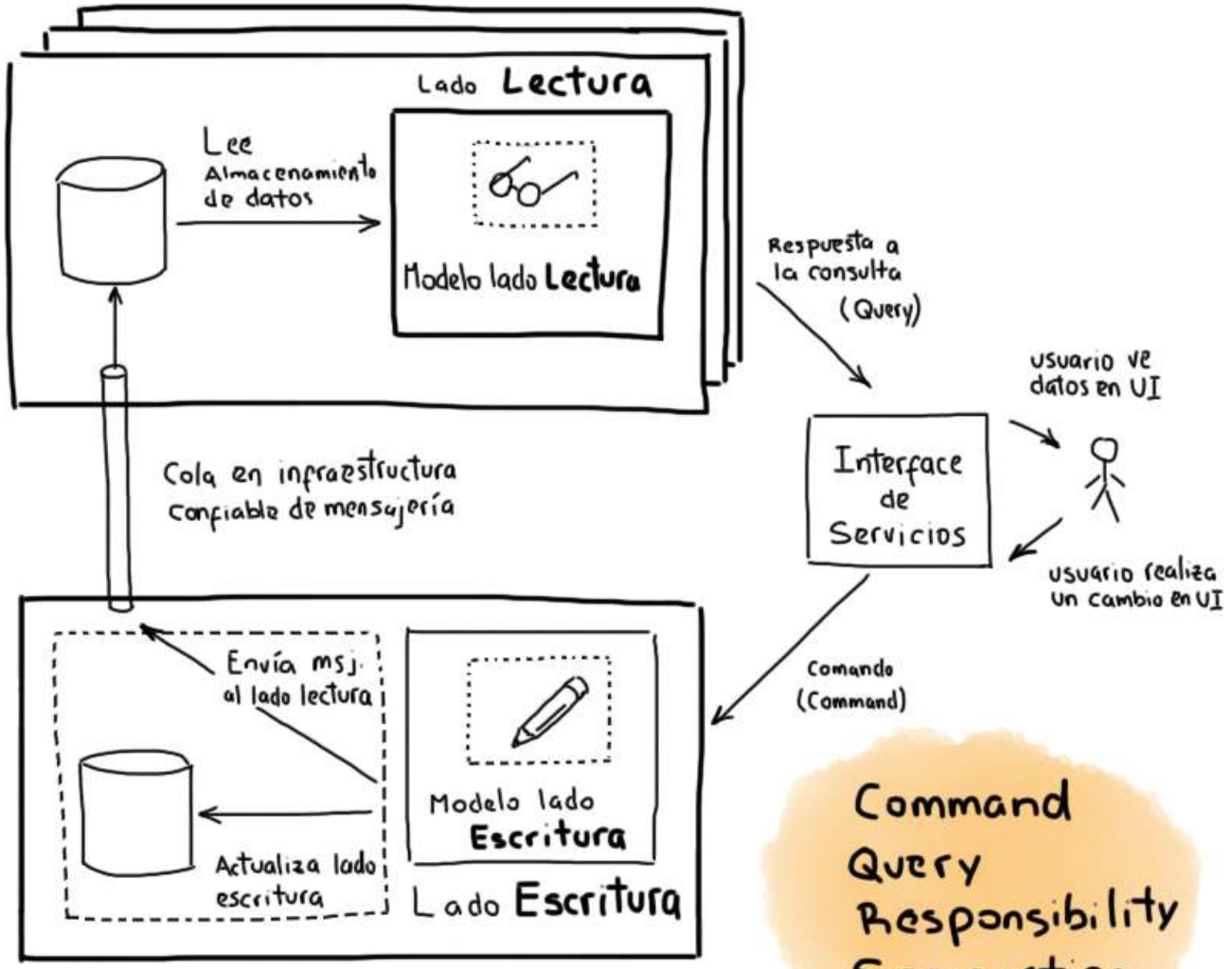
Domain-Driven Design

- DDD es un patrón de diseño
- Dominio: contexto del problema real de negocio
- Modelo: abstracción del dominio
- Software design ≠ Code Design
- Metodologías: Waterfalls vs. Agile
- Colaboración entre expertos de dominio y especialistas de software
- Idioma común = Ubiquitous Language



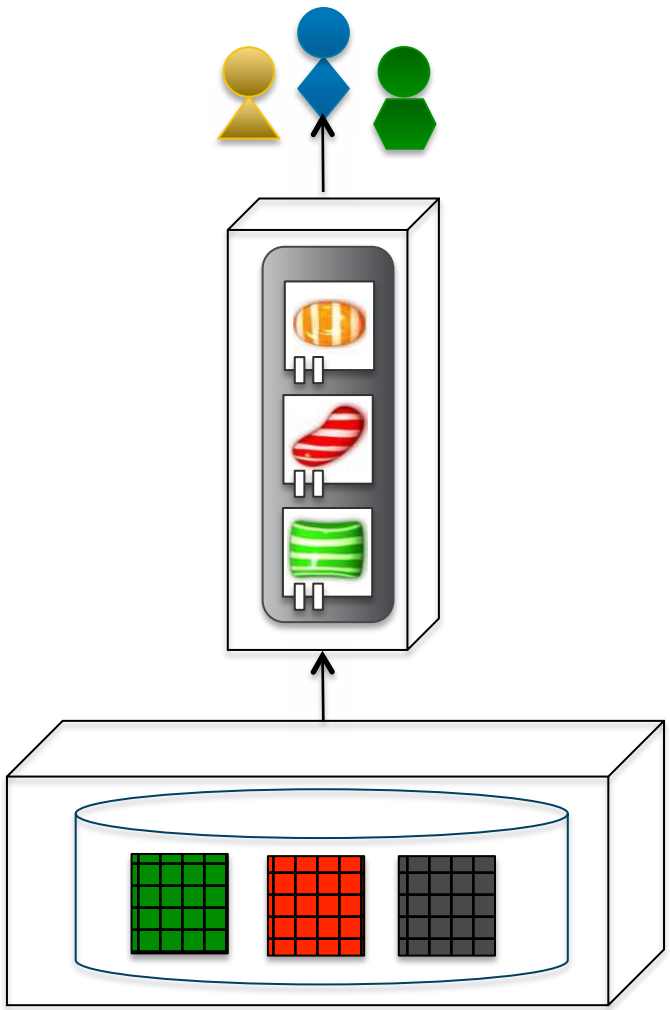
#6 Gestión de datos descentralizada

CQRS

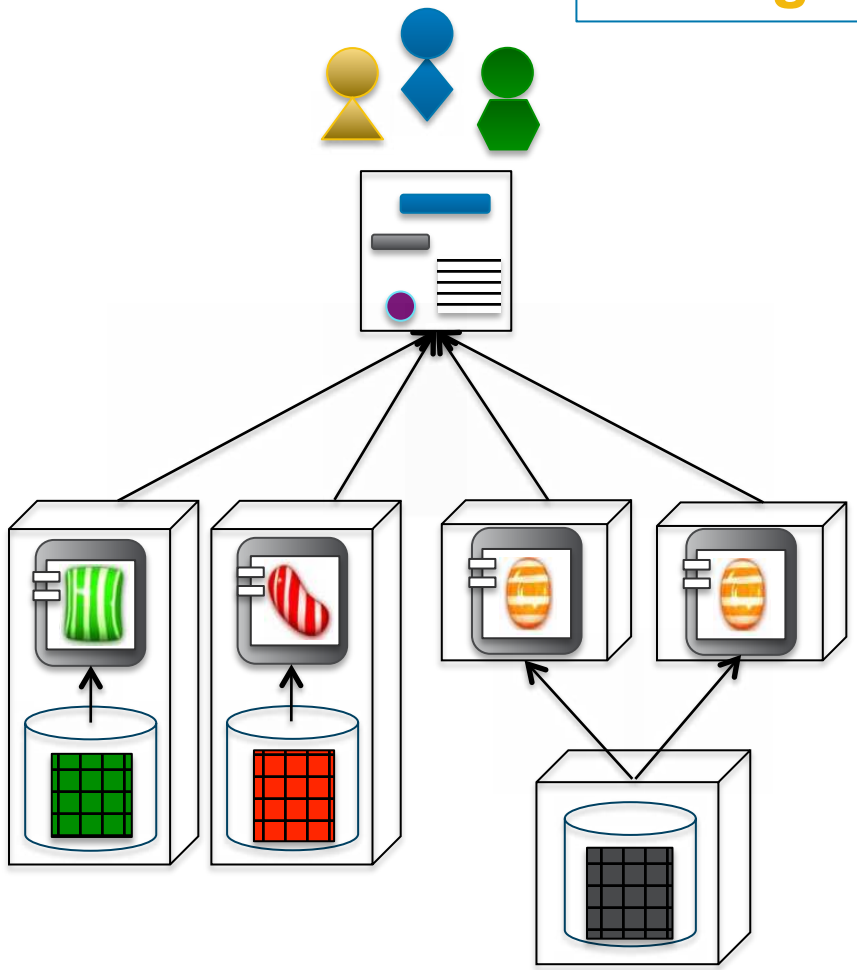


#6 Gestión de datos descentralizada

Persistencia Políglota

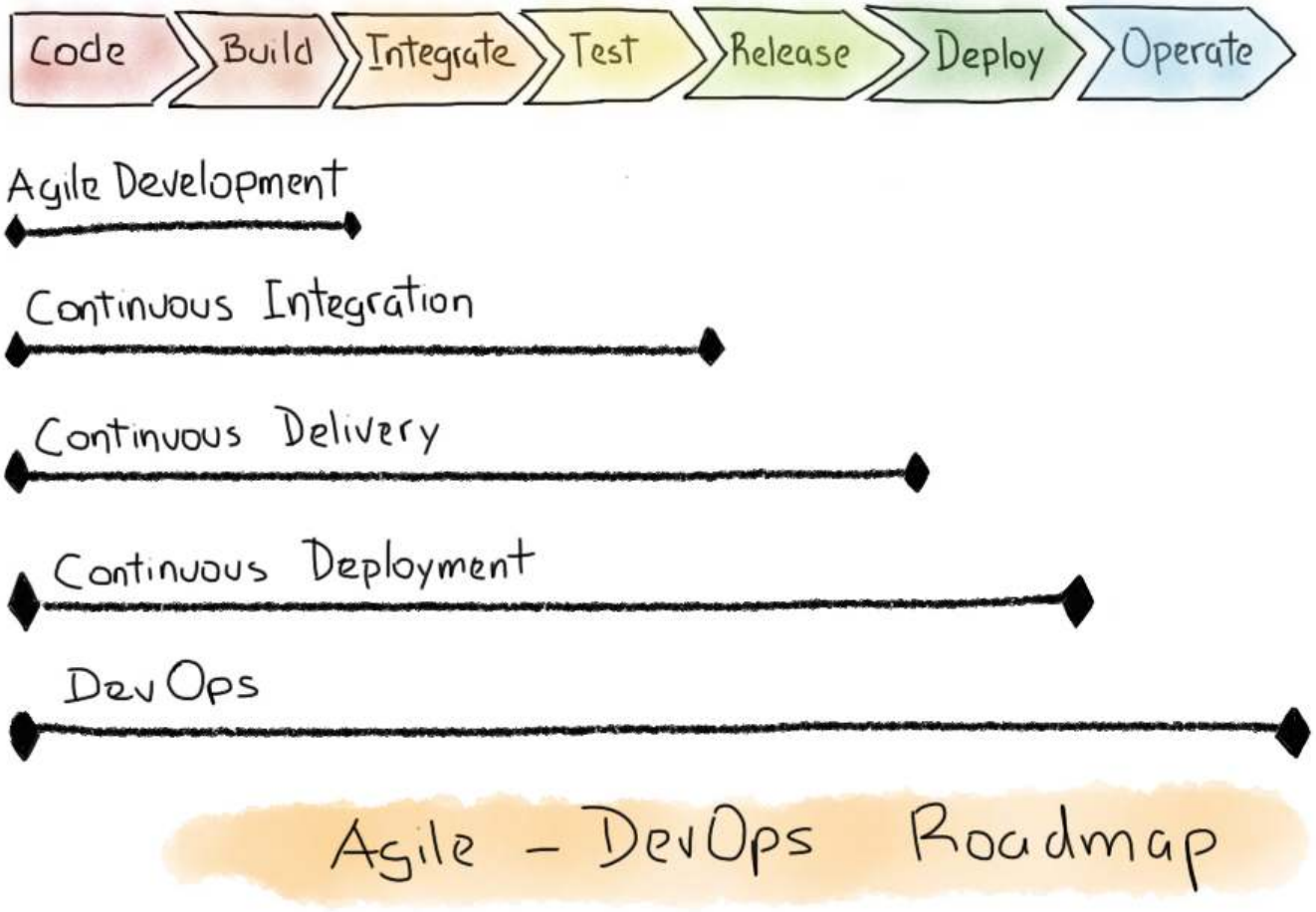


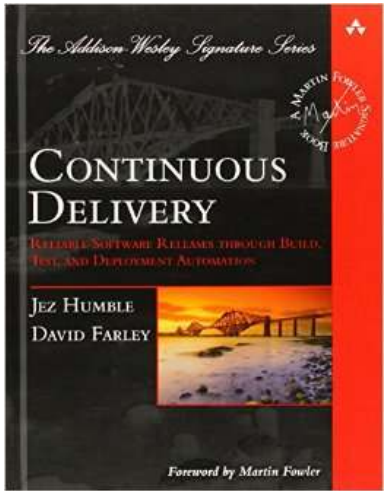
Monolítico – Única Base de Datos



Microservicios – Bases de Datos Aplicativas

#7 Automatización de la infraestructura





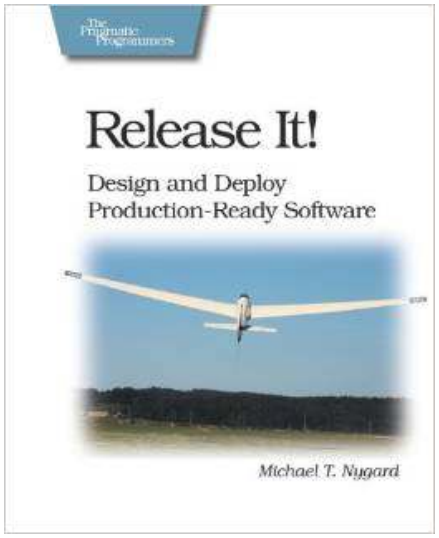
Clave:

1. Automatización
2. Control de versiones y gestión de las configuraciones
3. Arquitectura

#8 Diseño pensado en las fallas

timeouts

circuit breakers



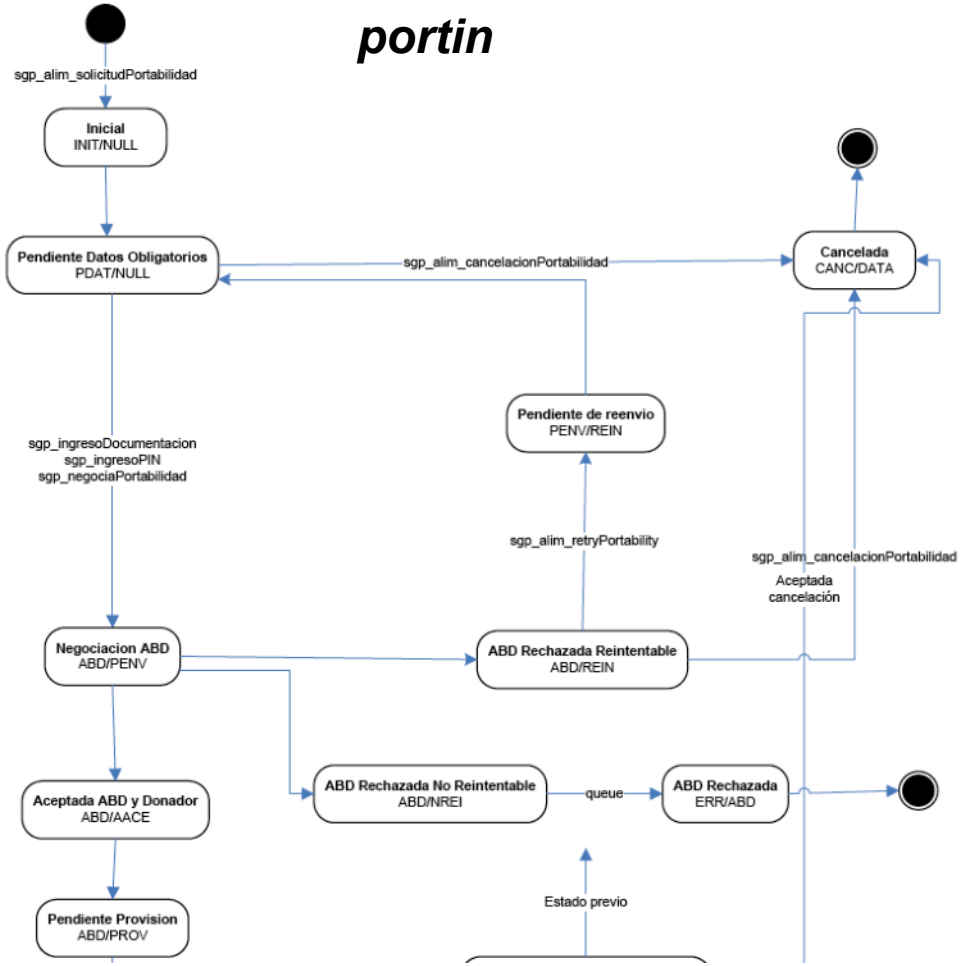
bulkheads

#8 Diseño pensado en las fallas

Ej. Portabilidad Numérica Móvil: Mensajes SOAP

timeouts

portin



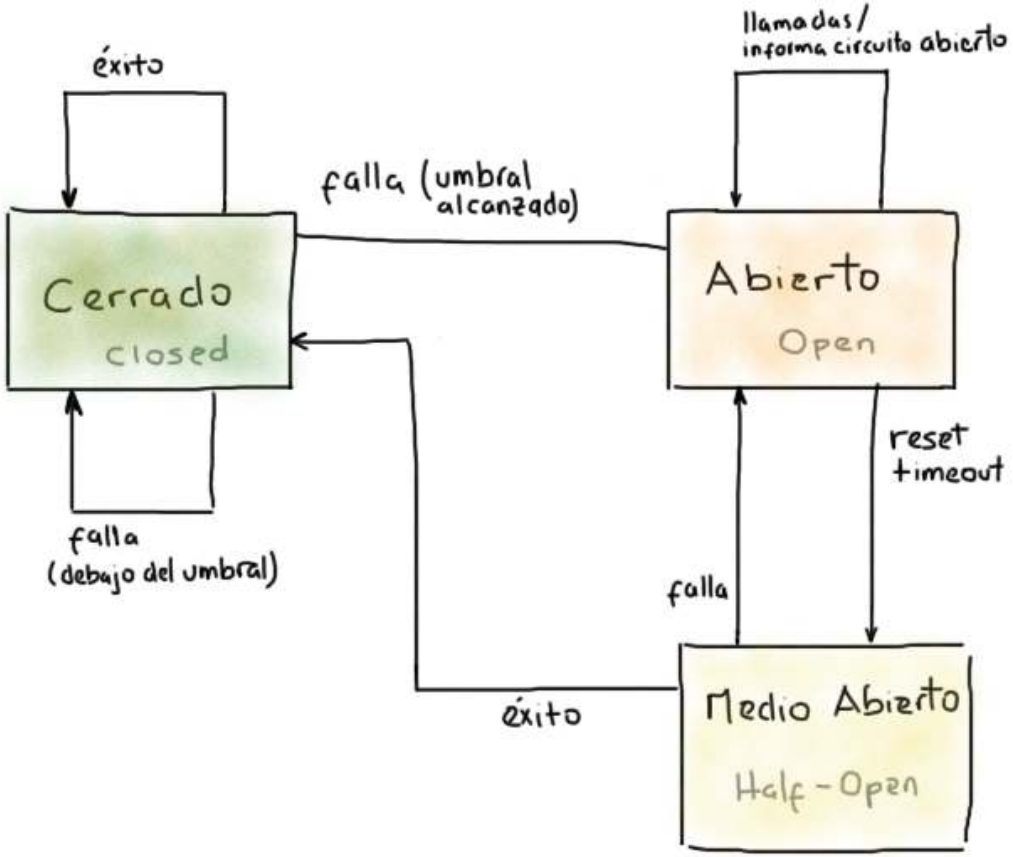
Element Name	XML Schema Definition	Mand./ Opt.	Descripción
TransactionID	TransactionIDType	M	El ID que se utiliza para identificar esta transacción.
Location	Len200Str	M	Ubicación y nombre del archivo (o el primero de los archivos si son mas de uno) que contiene los datos de Sincronización La cantidad de archivos se indicará en el nombre del archivo con n_n Ejemplos: /ftp/555/bulksync/10120100304011436131 9_1_1.txt o /ftp/555/bulksync/10120100304011436131 9_1_4.txt
Comments	CommentsType	O	Notas o comentarios del ABD

Table 46. **RejectMsgType**

Mensaje 9999

El Mensaje 9999 se utiliza cuando hay un problema técnico con un mensaje entrante que debe transmitirse al Prestador, pero que no termina el proceso. Por ejemplo, si se envía un mensaje antes o después de ser permitido, se enviará un Mensaje 9999 con el código de motivo "mensaje fuera de secuencia". Se informa al prestador y se continúa el proceso.

#8 Diseño pensado en las fallas



Estados de un Disyuntor

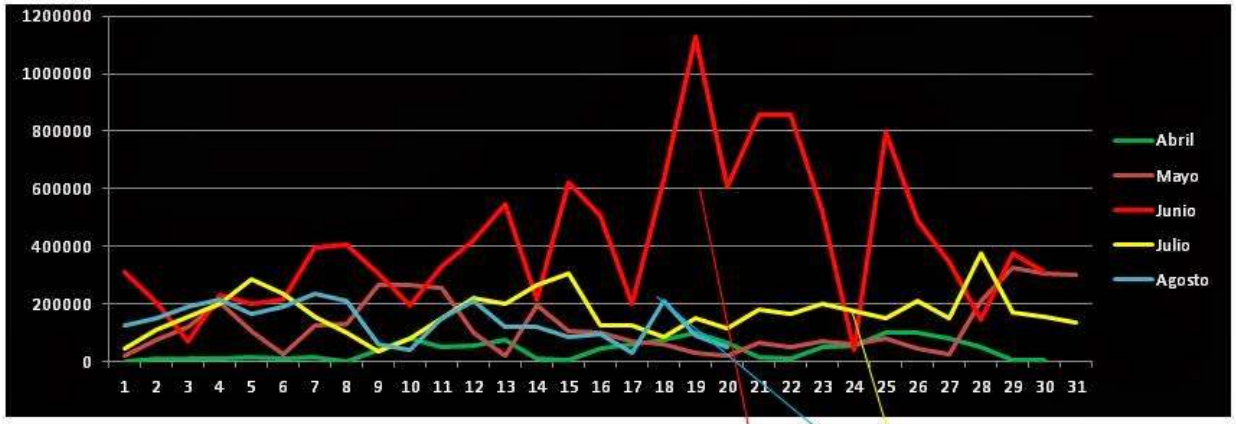
circuit breakers

#8 Diseño pensado en las fallas

circuit breakers

Ej. Problemas con robot “validar IMSI”

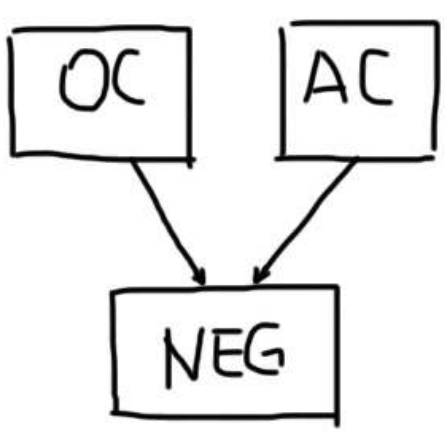
Evolución de Validar IMSI (En MDW)
Transacciones atendidas por MDW solicitadas desde Autogestión

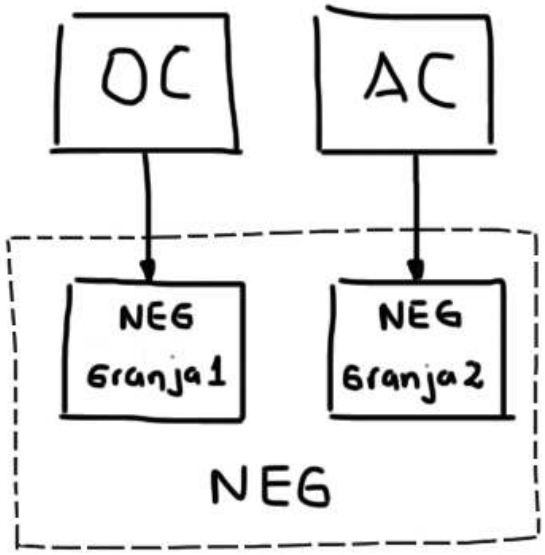


Impacto al Negocio – Novedades de Customer .

#8 Diseño pensado en las fallas

Ej. Separación ventas internas OC / externas AC

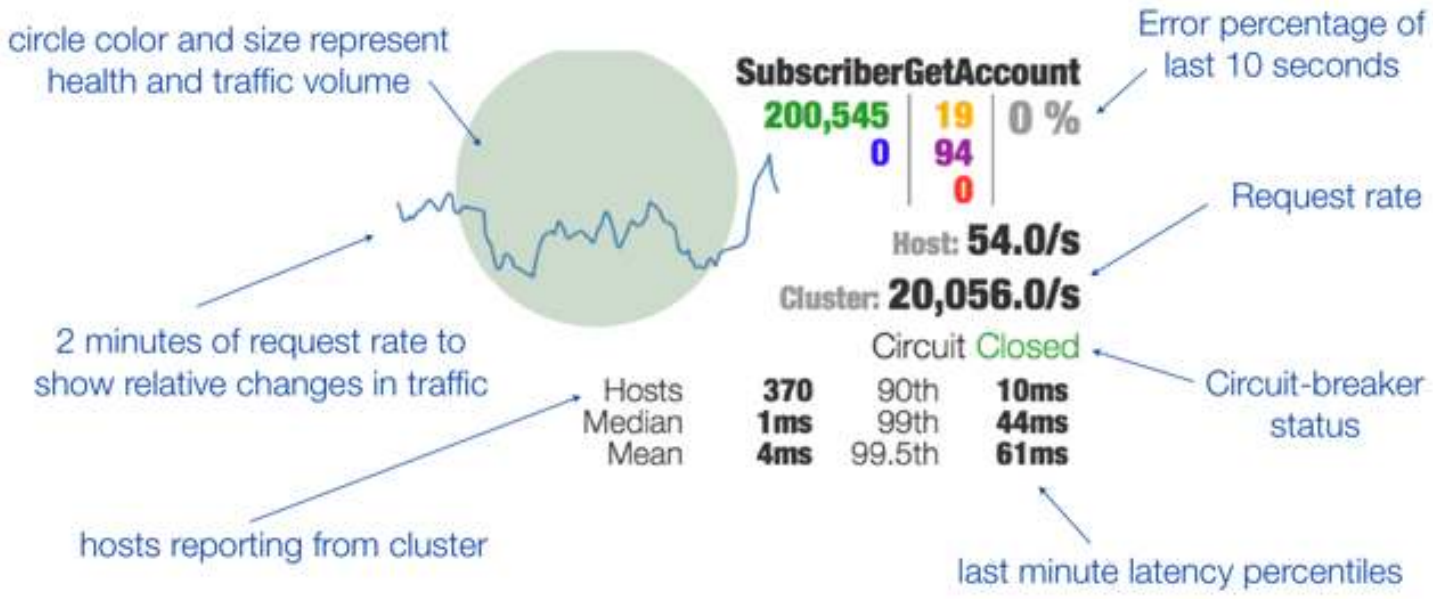




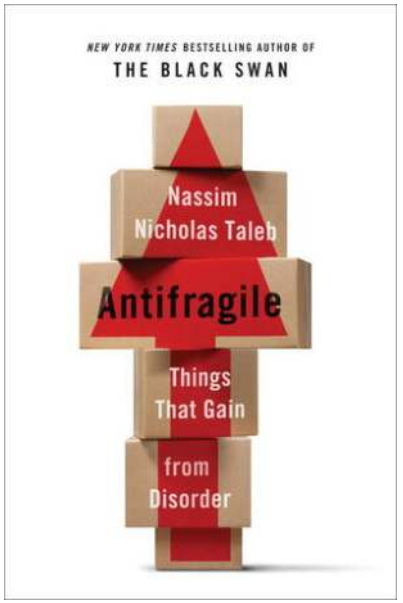
bulkheads

Vinculos Ocultos Vs. Sistema Particionado

#8 Diseño pensado en las fallas



NETFLIX

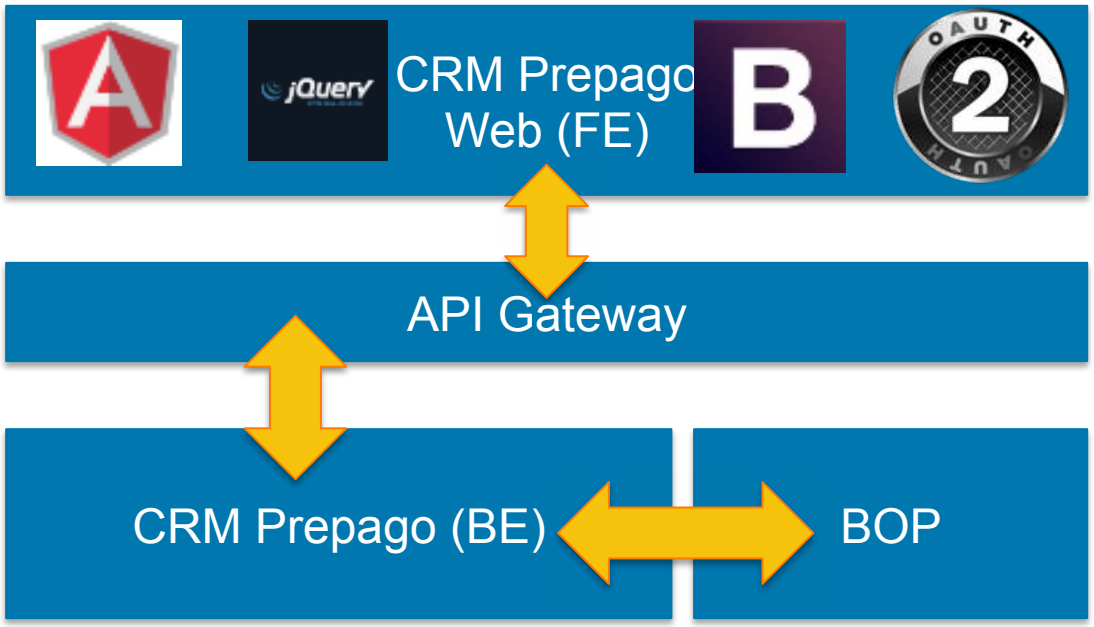


AGENDA

- Arquitectura Monolítica vs. SOA vs. Microservicios
- Descomposición de Servicios
- Ventajas / Desventajas
- Características de los Microservicios
- **Primeras experiencias**
- Conclusión


Hecho en casa

Ej. CRM Prepago



- Solución Front-End ejecutada en el browser utilizando JavaScript, Angular, Jquery, Bootstrap.
- Basada en Código Abierto
- Tecnologías y prácticas de uso en el mercado
- Soporte comunitario de las herramientas
- Construcción de servicios (FE / BE) por equipos diferentes de desarrollo.

Ej. CRM Prepago Web (FE)

Utiliza consulta completa de línea para el estado

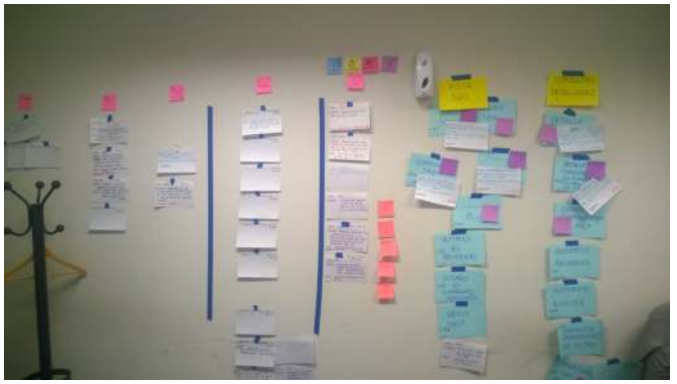
Consulta de linea

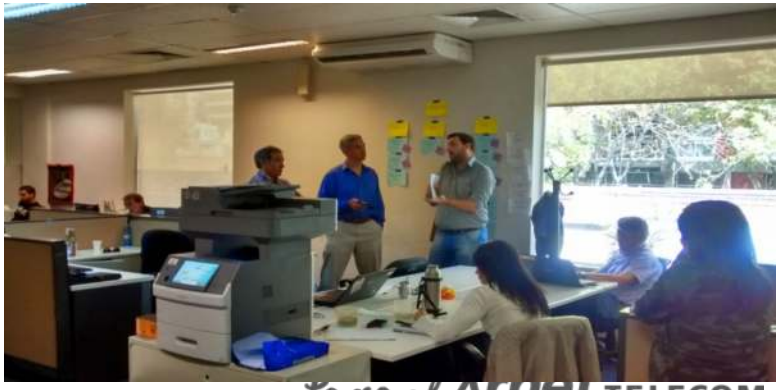
Consulta Lite de saldo

Consulta histórico de recargas

Ej. CRM Prepago – Servicios Back End (BE)

Ej. CRM Prepago – Equipo





AGENDA

- Arquitectura Monolítica vs. SOA vs. Microservicios
- Descomposición de Servicios
- Ventajas / Desventajas
- Características de los Microservicios
- Primeras experiencias
- **Conclusión**

Recap. Microservicios

- ✓ Servicios pequeños
- ✓ Contexto acotado, hacen una cosa bien
- ✓ Organizados en torno a capacidades de negocio
- ✓ Extremos inteligentes, tuberías bobas
- ✓ Productos, no proyectos
- ✓ Tecnología heterogénea
- ✓ Gestión de datos descentralizada
- ✓ Automatización → Continuous Delivery, DevOps
- ✓ Diseño contemplando fallas → Organización Antifrágil

Conclusión

Los microservicios representan,
un nuevo estilo de diseño de software,
una nueva manera de optimizar en velocidad,
de reemplazar los silos por equipos de alto desempeño,
de generar una cultura de gran autonomía y
responsabilidad, en resumen…
de generar valor al negocio.

Muchas Gracias!


sergio.maurenzi@personal.com.ar


@sergiomaurenzi


sergiomaurenzi.blogspot.com.ar


ar.linkedin.com/in/sergiomaurenzi

Referencias

[1] Microservices - Martin Fowler / James Lewis
http://martinfowler.com/articles/microservices.html

[2] eBay Architecture - Randy Shoup & Dan Pritchett / Tony Ng
http://www.slideshare.net/RandyShoup/the-ebay-architecture-striking-a-balance-between-site-stability-feature-velocity-performance-and-cost
http://www.slideshare.net/tcng3716/ebay-architecture?next_slideshow=1

[3] Episode 216 - Modern Cloud-based platform - Adrian Cockcroft
http://www.se-radio.net/2014/12/episode-216-adrian-cockcroft-on-the-modern-cloud-based-platform/

[4] The Art of Scalability - Martin Abbott y Michael Fisher
http://theartofscalability.com

[5] Single Responsibility Principle - Robert C. Martin
http://programmer.97things.oreilly.com/wiki/index.php/The_Single_Responsibility_Principle
http://www.objectmentor.com/resources/articles/srp.pdf

[6] A pattern language for microservices - Chris Richardson
http://microservices.io/patterns/index.html

[7] The Conway's Law - The original paper - Melvin Conway
http://www.melconway.com/Home/Committees_Paper.html

[8] A conversation with Werner Vogels (Amazon CTO)
https://queue.acm.org/detail.cfm?id=1142065

[9] Domain-Driven Design
http://dddcommunity.org/learning-ddd/what_is_ddd/
http://www.infoq.com/resource/minibooks/domain-driven-design-quickly/en/pdf/DomainDrivenDesignQuicklyOnline.pdf

[10] Episode 218 - CQRS (Command Query Responsibility Segregation) - Udi Dahan
http://www.se-radio.net/2015/01/episode-218-udi-dahan-on-cqrs-command-query-responsibility-segregation/

Referencias, cont.

[11] Episode 221 - Continuous Delivery - Jez Humble
http://www.se-radio.net/2015/02/episode-221-jez-humble-on-continuous-delivery/

[12] The Facebook Release Process - Chuck Rossi, Presentation & video
http://www.infoq.com/presentations/Facebook-Release-Process

[13] Release It! - Design and Deploy Production-Ready Software - Michael T. Nygard, Excerpt and Review of the book
http://www.infoq.com/articles/nygard-release-it/

[14] Fault Tolerance in a High-volume Distributed System - The Netflix Tech Blog
http://techblog.netflix.com/2012/02/fault-tolerance-in-high-volume.html

[15] Introducing Hystrix for Resilience Engineering - The Netflix Tech Blog
http://techblog.netflix.com/2012/11/hystrix.html

[16] Antifragile - Nassim Nicholas Taleb
http://www.tematika.com/libros/ciencias_de_la_salud__naturales_y_divulgacion_cientifica--7/divulgacion_cientifica--1/en_general--1/antifragil--565390.htm

[17] Chaos Monkey Released into the wild - The Netflix Tech Blog
http://techblog.netflix.com/2012/07/chaos-monkey-released-into-wild.html

[18] Tha Netflix Simian Army - The Netflix Tech Blog
http://techblog.netflix.com/search/label/chaos%20monkey

[19] Building Microservices, Designing Fine-grained Systems – Sam Newman
http://shop.oreilly.com/product/0636920033158.do

[20] Microservices: It's not (only) the size that matters, it's (also) how you use them - Tigerteam
https://www.tigerteam.dk/2014/microservices-its-not-only-the-size-that-matters-its-also-how-you-use-them-part-4/

Agradecimientos,

flickr

creative commons

1. Honeycomb: https://flic.kr/p/2UHtEd
2. Stonehenge: https://flic.kr/p/5x4KLZ
3. Belts and pulleys: https://flic.kr/p/Be77A
4. The EKC Studio Scale. Eastman Kodak Company: https://flic.kr/p/9vxKRp
5. Carnegie Mellon Philharmonic: https://flic.kr/p/aQE9vi
6. Cirque Du Soleil: Ovo - Gran Carpa Santa Fe: https://flic.kr/p/aB1Gh3
7. Stopwatch: https://flic.kr/p/81m4L5
8. Circuit breaker: https://flic.kr/p/Ce71k
9. Home Made Gingerbread: https://flic.kr/p/bkq5Qt
10. Buque Containers Maersk: http://blogs.lainformacion.com/futuretech/files/2013/09/maersk2.jpg

Dibujos a mano alzada realizados con "*Tayasui Sketches for Ipad*":
https://itunes.apple.com/us/app/tayasui-sketches-draw-paint/id641900855?mt=8

Personal Arnet GRUPO TELECOM