Microservicios – descripción general

Carlos Lombardi

Universidad Nacional de Quilmes – Argentina

24 de noviembre de 2017

Microservicios – primer acercamiento

De qué se trata

Conjunto de pautas sobre cómo organizar los entregables de una aplicación.

(entregable = "lo que se deploya")

Microservicios – primer acercamiento

De qué se trata

Conjunto de pautas sobre cómo organizar los entregables y el desarrollo
de una aplicación.

(entregable = "lo que se deploya")

Microservicios – primer acercamiento

De qué se trata

Conjunto de pautas sobre cómo organizar los entregables y el desarrollo
de una aplicación.

(entregable = "lo que se deploya")

Motivación

Problemáticas que surgen en aplicaciones de gran porte.

Veamos un ejemplo.

El ejemplo

Empresa que vende productos.

El ejemplo

Empresa que vende productos.

Desarrollo de aplicación administrativa.

El ejemplo

Empresa que vende productos.

Desarrollo de aplicación administrativa.

Equipo de cuatro personas.

Ana	gestión **a**dministrativa (facturación y pagos).
Víctor	**v**entas, promociones, acciones de venta con los clientes.
Darío	**d**epósito, stock, productos.
Irma	**i**nterfaz de usuario, pantallas complejas.

Aplicación administrativa – decisiones iniciales

- Desarrollo en Java usando POO.
- Una BD relacional que integra todos los datos.
- Bases de código separadas.

Aplicación administrativa – interacciones (1)

Todos necesitan
búsqueda de clientes y dirección de cada cliente.

Aplicación administrativa – interacciones (1)

Todos necesitan
búsqueda de clientes y dirección de cada cliente.

Se encarga Víctor

Aplicación administrativa – interacciones (1)

Todos necesitan
búsqueda de clientes y dirección de cada cliente.

Se encarga Víctor

- Interacción: clases `ClienteHome`, `Cliente`, `Direccion`.

Aplicación administrativa – interacciones (1)

Todos necesitan
búsqueda de clientes y dirección de cada cliente.

Se encarga Víctor

- Interacción: clases `ClienteHome`, `Cliente`, `Direccion`.
- Víctor implementa las clases, y pasa los fuentes necesarios al resto.
- También pasa datos de prueba.

Aplicación administrativa – interacciones (1)

Todos necesitan
búsqueda de clientes y dirección de cada cliente.

Se encarga Víctor

- Interacción: clases `ClienteHome`, `Cliente`, `Direccion`.
- Víctor implementa las clases, y pasa los fuentes necesarios al resto.
- También pasa datos de prueba.

Se define que la clase `Cliente` quede acotada.

Aplicación administrativa – interacciones (2)

Víctor necesita de Darío
información sobre productos con mucho stock
(para armar promociones)

Aplicación administrativa – interacciones (2)

Víctor necesita de Darío
información sobre productos con mucho stock
(para armar promociones)

Darío no tiene definido su modelo, entonces

Aplicación administrativa – interacciones (2)

Víctor necesita de Darío
información sobre productos con mucho stock
(para armar promociones)

Darío no tiene definido su modelo, entonces

- Interacción: consultas específicas que devuelven estructuras de datos simples
  (arrays con solamente Strings y números).

Aplicación administrativa – interacciones (2)

Víctor necesita de Darío
información sobre productos con mucho stock
(para armar promociones)

Darío no tiene definido su modelo, entonces

- Interacción: consultas específicas que devuelven estructuras de datos simples
  (arrays con solamente Strings y números).

Aplicación administrativa – interacciones (2)

Víctor necesita de Darío
información sobre productos con mucho stock
(para armar promociones)

Darío no tiene definido su modelo, entonces

- Interacción: consultas específicas, interfaz de servicios que devuelven estructuras de datos simples
(arrays con solamente Strings y números).

Aplicación administrativa – interacciones (2)

Víctor necesita de Darío
información sobre productos con mucho stock
(para armar promociones)

Darío no tiene definido su modelo, entonces

- Interacción: consultas específicas, interfaz de servicios que devuelven estructuras de datos simples
  (arrays con solamente Strings y números).
- Implementación inicial para no trabar a Víctor.
- Después se va ajustando sin afectar la interfaz definida.

Aplicación administrativa – interacciones (3)

Víctor necesita de Ana
información sobre los pagos con tarjeta
(para decidir condiciones de pago para cada cliente)

Aplicación administrativa – interacciones (3)

Víctor necesita de Ana
información sobre los pagos con tarjeta
(para decidir condiciones de pago para cada cliente)

Ana está complicada con los tiempos, entonces

Aplicación administrativa – interacciones (3)

Víctor necesita de Ana
información sobre los pagos con tarjeta
(para decidir condiciones de pago para cada cliente)

Ana está complicada con los tiempos, entonces

- Interacción: Ana comparte con Victor las clases `Tarjeta`, `TarjetaHome` y `Pago`.

Aplicación administrativa – interacciones (3)

Víctor necesita de Ana
información sobre los pagos con tarjeta
(para decidir condiciones de pago para cada cliente)

Ana está complicada con los tiempos, entonces

- Interacción: Ana comparte con Victor las clases `Tarjeta`, `TarjetaHome` y `Pago`.
- Le explica cómo obtener la información que necesita.
- Si Ana hace cambios, debe enviar las nuevas versiones.

Aplicación administrativa – interacciones (4)

Irma necesita de todos
información para armar un "tablero de control" de un cliente
(que incluye ventas, pagos y envíos de mercadería)

Aplicación administrativa – interacciones (4)

Irma necesita de todos
información para armar un "tablero de control" de un cliente
(que incluye ventas, pagos y envíos de mercadería)

Estamos al filo de la entrega, entonces

Aplicación administrativa – interacciones (4)

Irma necesita de todos
información para armar un "tablero de control" de un cliente
(que incluye ventas, pagos y envíos de mercadería)

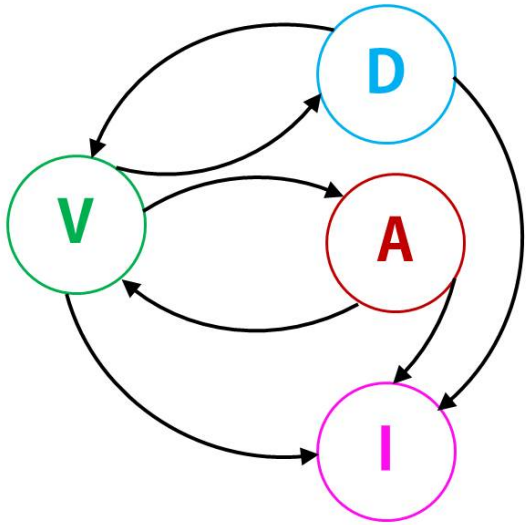
Estamos al filo de la entrega, entonces

- Interacción: Todos exponen su modelo para que Irma use lo que necesita.

Aplicación administrativa – interacciones – resumen

Distintas formas de interacción

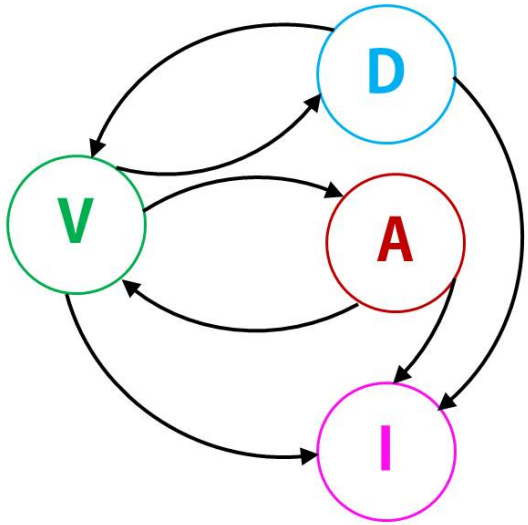
- Clases diseñadas para ser compartidas (`Cliente`).
- Servicios con interfaces de datos primitivos (productos con mucho stock).
- Exponer (parte d)el modelo de negocio (`Tarjeta`, tablero de control).



Aplicación administrativa – interacciones – resumen

Distintas formas de interacción

- Clases diseñadas para ser compartidas (`Cliente`).
- Servicios con interfaces de datos primitivos (productos con mucho stock).
- Exponer (parte d)el modelo de negocio (`Tarjeta`, tablero de control).



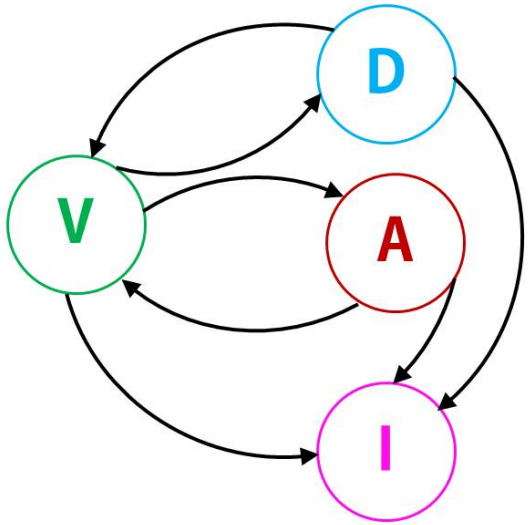
Observación

- Interacciones definidas por circunstancias del desarrollo.

Aplicación administrativa – hay que deployar (1)

Riesgos

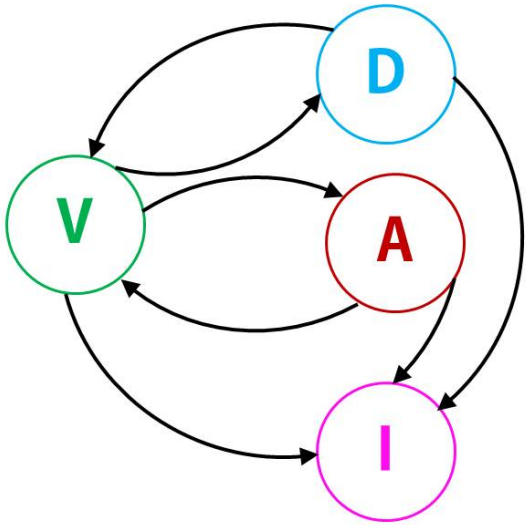
- Uso de versiones no actualizadas de componentes compartidos.
- Duplicación de nombres, en clases/interfaces o en el esquema de BD.
- Uso de distintas versiones del mismo paquete.
- Armado de línea de base en los repositorios de código.



Aplicación administrativa – hay que deployar (2)

Generación de entregables – cómo se resuelve.

- Se juntan los cuatro en un mismo lugar, y estabilizan hasta que llegan a una versión funcionando.



Cambiemos la escala

Cuatro desarrolladores en la misma ciudad.

Cambiemos la escala

~~Cuatro desarrolladores en la misma ciudad.~~

120 desarrolladores repartidos en 4 ciudades.

Cambiemos la escala

~~Cuatro desarrolladores en la misma ciudad.~~

120 desarrolladores repartidos en 4 ciudades.

Explosión combinatoria en

- Riesgos.
- Esfuerzo de estabilización.

Hora de hablar de . . .

Hora de hablar de . . .

Microservicios

Microservicios – de qué estamos hablando

Estilo arquitectural – pautas acerca de

- productos de software, y la
- forma de desarrollarlos

Microservicios – la pauta inicial

Un gran entregable.

Microservicios – la pauta inicial

~~Un gran entregable.~~

Deploy distribuido

Varios entregables
que interactúan mediante servicios con interfaces sencillas.

Microservicios – la pauta inicial

~~Un gran entregable.~~ ← Monolito

Deploy distribuido

Varios entregables
que interactúan mediante servicios con interfaces sencillas.

Microservicios – repasamos patrones de interacción

Exponer modelo (tarjeta, pago)

Clases pensadas para interacción (cliente, dirección)

Servicios con interfaces simples (stock)

Microservicios – repasamos patrones de interacción

- ✗ Exponer modelo (tarjeta, pago)
- Clases pensadas para interacción (cliente, dirección)
- Servicios con interfaces simples (stock)

Microservicios – repasamos patrones de interacción

- ✗ Exponer modelo (tarjeta, pago)
- ✗ Clases pensadas para interacción (cliente, dirección)
- Servicios con interfaces simples (stock)

Microservicios – repasamos patrones de interacción

- ✗ Exponer modelo (tarjeta, pago)
- ✗ Clases pensadas para interacción (cliente, dirección)
- ✓ Servicios con interfaces simples (stock)

Microservicios – alcance (1)

Pautas en aspectos técnicos

- Manejo de entregables.
- Manejo de la base de código.
- Diseño de software.
- Ciclo de desarrollo.

Microservicios – alcance (2)

Pautas en aspectos sociales

- Ciclo de desarrollo.
- Conformación de equipos de trabajo.
- Coordinación y autonomía.
- Standards.

Microservicios – motivaciones

Iniciales – manejo de deploy

- Dificultad para generar entregable monolito.
- Escalar un componente aislado.

Microservicios – motivaciones

Iniciales – manejo de deploy

- Dificultad para generar entregable monolito.
- Escalar un componente aislado.

Profundas – desarrollo a gran escala

- Comunicación entre equipos.
- Motivación de desarrolladores.
- Dirección del proyecto.
- Posibilitar desarrollo evolutivo.

Microservicios – motivaciones

Iniciales – manejo de deploy

- Dificultad para generar entregable monolito.
- Escalar un componente aislado.

Profundas – desarrollo a gran escala

- Comunicación entre equipos.
- Motivación de desarrolladores.
- Dirección del proyecto.
- Posibilitar desarrollo evolutivo.

Moraleja

Relevancia de aspectos sociales.

Microservicios – ¿se usa?

Parece que sí

Microservicios – ¿se usa?

Parece que sí

- Netflix
- Spotify
- Amazon
- Ebay
- Uber
- Groupon

Microservicios – ¿todo o nada?

No

Microservicios – ¿todo o nada?

No

Pautas que pueden ser útiles en cualquier desarrollo.

Microservicios – ¿todo o nada?

No

Pautas que pueden ser útiles en cualquier desarrollo.

Pero

Microservicios – ¿todo o nada?

No

Pautas que pueden ser útiles en cualquier desarrollo.

Pero

Las pautas forman un conjunto coherente.
Si se aplican algunas, conviene evaluar el resto.

Microservicios – Resumen hasta acá

De qué se trata

Pautas que abarcan todos los aspectos del desarrollo.

Pautas iniciales

Separar en varios entregables.
Interacción: servicios con interfaces sencillas.

Motivación

Inicial: manejo de entregables.
Profunda: complejidades técnicas y sociales de proyectos grandes.

Uso

Extendido (Netflix, Spotify, Amazon, Ebay, etc.)

Pautas

Pautas

Fortalezas

Pautas

Fortalezas

Debilidades/riesgos

Pautas

Fortalezas

Debilidades/riesgos

Consecuencias

Pautas – separación entre componentes

De qué se trata

Separar la aplicación en componentes, que se comunican solamente mediante servicios.

Fortalezas

Pautas – separación entre componentes

De qué se trata

Separar la aplicación en componentes, que se comunican solamente mediante servicios.

Fortalezas

- Componentes más manejables.
- Interfaces bien definidas entre componentes.
- Habilita descentralización del equipo de trabajo.

Debilidades/riesgos

Pautas – separación entre componentes

De qué se trata

Separar la aplicación en componentes, que se comunican solamente mediante servicios.

Fortalezas

- Componentes más manejables.
- Interfaces bien definidas entre componentes.
- Habilita descentralización del equipo de trabajo.

Debilidades/riesgos

- Cómo "cortar" una aplicación en componentes.
- Esfuerzo adicional para definir y desarrollar servicios.

Pautas – deploys ágiles, descentralizados, automáticos

De qué se trata

Empaquetado e instalación independiente de cada componente.
Hacer del deploy una operación rutinaria y automatizada.

Fortalezas

Pautas – deploys ágiles, descentralizados, automáticos

De qué se trata

Empaquetado e instalación independiente de cada componente.
Hacer del deploy una operación rutinaria y automatizada.

Fortalezas

- Se evita empaquetar monolitos.
- Permite implementar cambios más rápido.
- Facilita evolución independiente.

Debilidades/riesgos

Pautas – deploys ágiles, descentralizados, automáticos

De qué se trata

Empaquetado e instalación independiente de cada componente.
Hacer del deploy una operación rutinaria y automatizada.

Fortalezas

- Se evita empaquetar monolitos.
- Permite implementar cambios más rápido.
- Facilita evolución independiente.

Debilidades/riesgos

- Costo en performance por comunicación via servicios.
- No hay una "última versión" de cada componente.

Pautas – interfaces livianas

De qué se trata

Interfaces de servicios mediante texto estructurado (XML, JSON, YAML).
Se contrapone a serialización de objetos.

Fortalezas

Pautas – interfaces livianas

De qué se trata

Interfaces de servicios mediante texto estructurado (XML, JSON, YAML).
Se contrapone a serialización de objetos.

Fortalezas

- Fuerza separación entre componentes.
- Simplifica simulaciones para desarrollo y test.

Debilidades/riesgos

Pautas – interfaces livianas

De qué se trata

Interfaces de servicios mediante texto estructurado (XML, JSON, YAML).
Se contrapone a serialización de objetos.

Fortalezas

- Fuerza separación entre componentes.
- Simplifica simulaciones para desarrollo y test.

Debilidades/riesgos

- Potencial duplicación de lógica.

Pautas – organización del equipo por funcionalidad

De qué se trata

Equipos full-stack.
Cada equipo resuelve una parte relevante de la aplicación, de punta a punta.

Fortalezas

Pautas – organización del equipo por funcionalidad

De qué se trata

Equipos full-stack.
Cada equipo resuelve una parte relevante de la aplicación, de punta a punta.

Fortalezas

- Reducción de burocracia para desarrollar cambios.
- Cercanía con el dominio y los usuarios/clientes/stakeholders (evita YAGNI).

Debilidades/riesgos

Pautas – organización del equipo por funcionalidad

De qué se trata

Equipos full-stack.
Cada equipo resuelve una parte relevante de la aplicación, de punta a punta.

Fortalezas

- Reducción de burocracia para desarrollar cambios.
- Cercanía con el dominio y los usuarios/clientes/stakeholders (evita YAGNI).

Debilidades/riesgos

- ¿Quién detecta, y quién implementa, componentes core?.

Pautas – gestión descentralizada (1/2)

De qué se trata

Autonomía de cada equipo para elegir:

- tecnologías
- herramientas
- forma de trabajo

Se contrapone a:
equipos centrales de arquitectura y procesos que define standards.

Pautas – gestión descentralizada (2/2)

Autonomía para elegir tecnologías, herramientas, forma de trabajo.

Fortalezas

Pautas – gestión descentralizada (2/2)

Autonomía para elegir tecnologías, herramientas, forma de trabajo.

Fortalezas

- Facilita experimentación.
- Permite usar tecnologías específicas.
- Simplifica la incorporación de nuevos desarrolladores.

Debilidades/riesgos

Pautas – gestión descentralizada (2/2)

Autonomía para elegir tecnologías, herramientas, forma de trabajo.

Fortalezas

- Facilita experimentación.
- Permite usar tecnologías específicas.
- Simplifica la incorporación de nuevos desarrolladores.

Debilidades/riesgos

- Hace más complejo el mantenimiento.
- Dificulta la introducción de mejoras globales.
- Menor facilidad para los cambios de equipo.

Pautas – datos descentralizados

De qué se trata

Se alienta a que la persistencia de cada componente se haga en una BD separada.

Fortalezas

Pautas – datos descentralizados

De qué se trata

Se alienta a que la persistencia de cada componente se haga en una BD separada.

Fortalezas

Se evitan cuellos de botella.

Debilidades/riesgos

Pautas – datos descentralizados

De qué se trata

Se alienta a que la persistencia de cada componente se haga en una BD separada.

Fortalezas

Se evitan cuellos de botella.

Debilidades/riesgos

Consistencia eventual.

Pautas – Design for failure

De qué se trata

Programación defensiva entre componentes.

Fortalezas

Pautas – Design for failure

De qué se trata

Programación defensiva entre componentes.

Fortalezas

Facilita desacoplamiento entre equipos.

Aplicaciones más robustas.

Debilidades/riesgos

Pautas – Design for failure

De qué se trata

Programación defensiva entre componentes.

Fortalezas

Facilita desacoplamiento entre equipos.

Aplicaciones más robustas.

Debilidades/riesgos

Se agrega complejidad al desarrollo.

Pautas – resumen

Separación entre componentes.

Deploys ágiles, descentralizados, automáticos.

Interfaces livianas.

Organización del equipo por funcionalidad.

Gestión descentralizada.

Datos descentralizados.

Design for failure.

Vamos cerrando – algunas características

Se lleva bien con

- Desarrollo ágil.
- Equipos autónomos.
- Reacción rápida.
- Evolución.

Tener especial cuidado con

- Consistencia de datos.
- Rotación entre grupos.
- Overhead por comunicación mediante servicios.
- Monitoreo.

Vamos cerrando – algunas características

Foco en escalar

- recursos asignados a deploy.
- equipo de trabajo.
- funcionalidad.

Foco en las personas

- relevancia de motivación.
- autonomía / emponderamiento.
- codificación de prácticas observadas en grupos de desarrolladores.

Vamos cerrando – hablemos un minuto sobre

Vamos cerrando – hablemos un minuto sobre

Tests

Vamos cerrando – hablemos un minuto sobre

- Tests
- Standards

Vamos cerrando – hablemos un minuto sobre

- Tests
- Standards
- Operaciones

Vamos cerrando – hablemos un minuto sobre

Tests

Standards

Operaciones

Documentación

Microservicios - surgen de la práctica

Microservices (. . .) weren't invented or described before the fact; they emerged as a trend, or a pattern, from real-world use.

Sam Newman, Building Microservices – designing fine-grained systems. O'Reilly, 2015.

Teams building microservices prefer a different approach to standards (. . .).

James Lewis y Martin Fowler, Microservices - a definition of this new architectural term. `martinfowler.com`, 2014.

Vamos cerrando – riesgos, ¿de dónde vienen?

- Duplicación.
- Problemas de consistencia en datos.
- Falta o debilidad en standards.
- Dificultad para rotar.

¿Provienen de

adoptar enfoque basado en microservicios

o de

la escala del desarrollo?

Finalmente – de qué estamos hablando

Finalmente – de qué estamos hablando

Complejidad