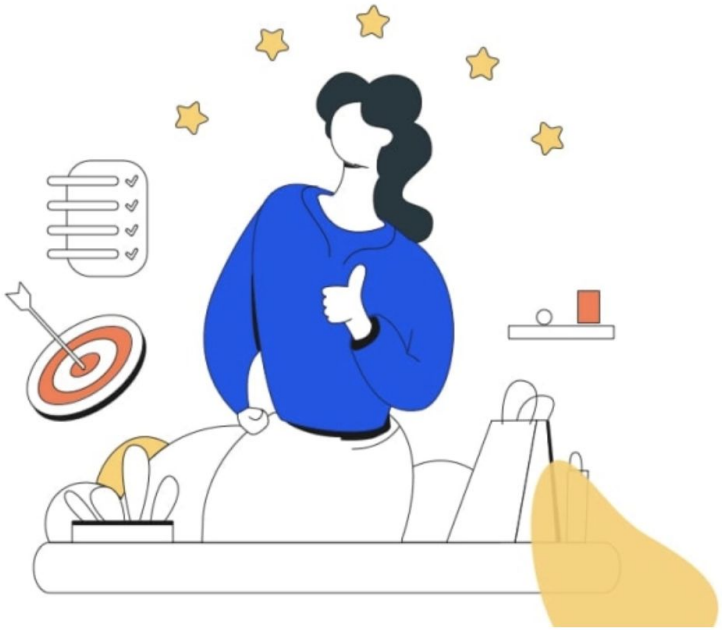
PONTIFICIA
UNIVERSIDAD
CATÓLICA
DE CHILE

IIC3103

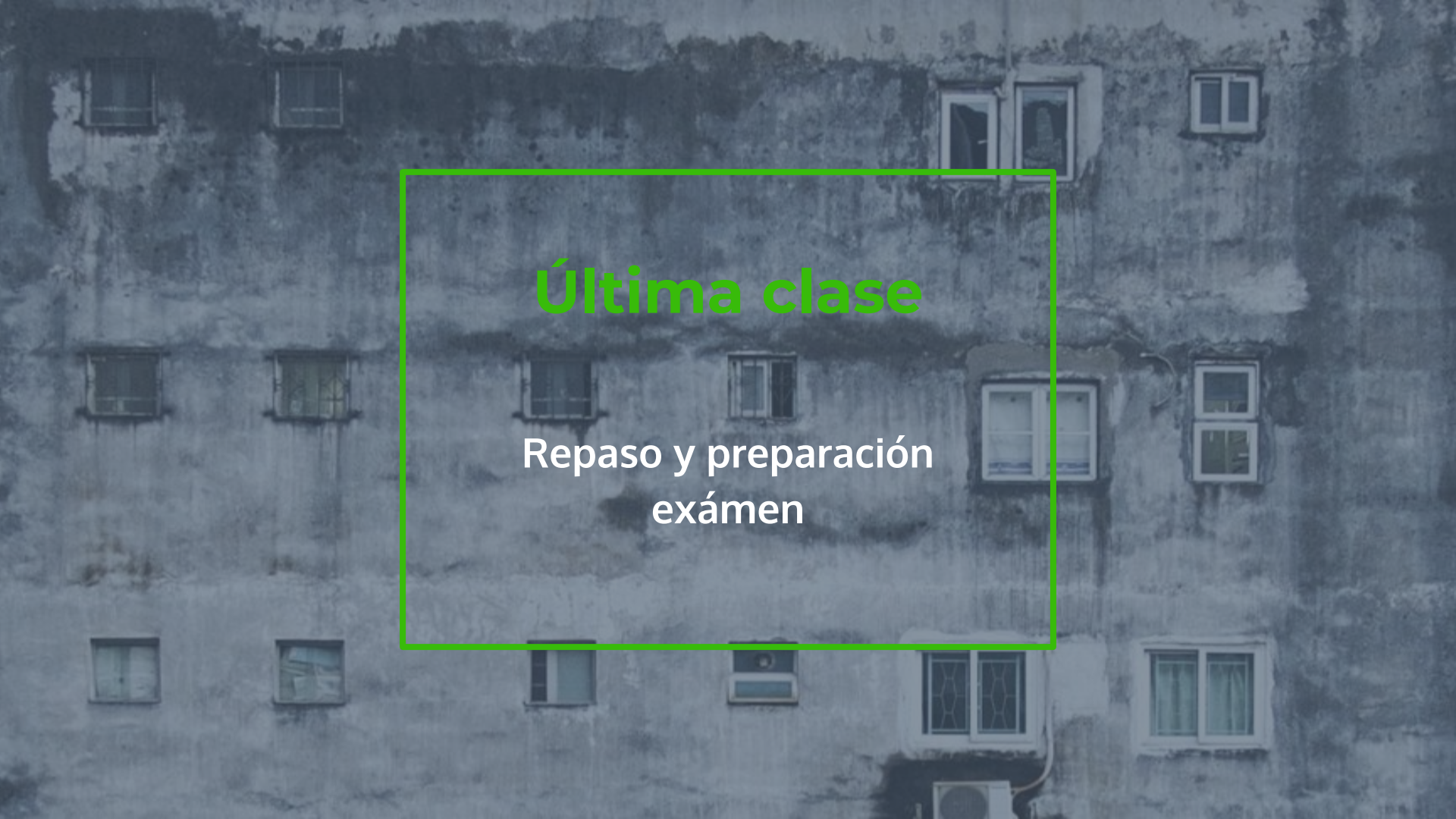
Taller de Integración

Profesores

Arturo Tagle / Daniel Darritchon



Dudas ENTREGA 3


Última clase
Repaso y preparación
exámen

Examen

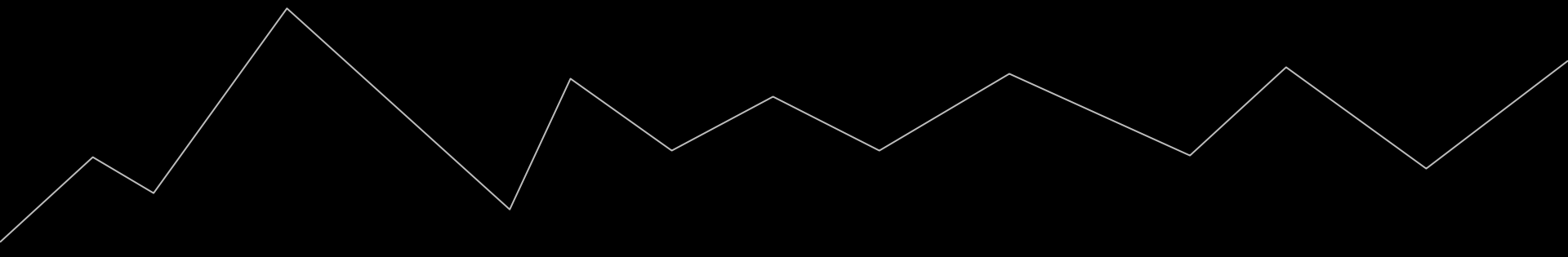
Fecha

6 de diciembre, 17:20 (fecha asignada por DIPRE).

Modalidad

Presencial

Arquitecturas orientadas a eventos



Características de los eventos

1. Bajo acoplamiento.

2. La comunicación es entre múltiples sistemas.

3. La comunicación es iniciada por el emisor del evento.

4. La comunicación es asíncrona.

5. La comunicación sólo tiene un mensaje: el evento.

6. Suscriptores deben registrarse para recibir eventos.

Concentradores de eventos

También conocidos como brokers, middlewares o bus de eventos.

Son utilizados para facilitar, mediar y enriquecer la interacción de emisores y suscriptores en sistemas orientados a eventos.

Las arquitecturas orientadas a eventos son, normalmente, arquitecturas con intermediarios. Los concentradores, son parte fundamental de estas arquitecturas.

Servicios vs Eventos

Sobre los datos	SERVICIOS	EVENTOS
Entidad	Objeto	Hecho o cambio de estado
Esquema	Estricto, cambia poco	Flexible, distintos hechos pueden generar eventos diferentes
Qué es	Una foto del objeto en ese momento del tiempo	Un hecho o cambio de estado en el momento que se produce
Qué describe	Describe el presente	Describe una tendencia en el tiempo
Cómo se actualiza el objeto	Se actualiza, no es necesario mantener historia	Se agrega. Es necesario recorrer toda la historia para ver estado actual.
Complejidad	O(n) Lineal según cantidad de objetos	O(n*k) Según cantidad de objetos (n) y cambios notificados (k)

Sobre las características	**Servicios**	**Eventos**
Inicio de la comunicación	Cliente (pull)	Emisor del evento (push)
Participantes	Cliente y servidor	Un emisor, múltiples suscriptores
Sincronía	Comunicación síncrona	Comunicación asíncrona
Uso	Basta con conocer contrato de comunicación para realizar un request	Registro previo del suscriptor, quién notifica eventos en la medida que suceden
Tipo de datos	Estado actual de objetos	Estado en un cierto momento
Arquitectura	Cliente - servidor	Emisor - Event broker - Suscriptores

Suponga que se han instalado 100 medidores inteligentes de temperatura en diferentes salas del campus. Para una mayor eficiencia, los medidores pueden ser cambiados de sala ya sea por fallas o según patrones de uso. Cada medidor reporta la temperatura de la sala donde se encuentra ubicado cada 10 segundos.

Pregunta

Si se siguen las mejores prácticas para el diseño de un mensaje de un evento, además del identificador del medidor y la temperatura registrada, ¿qué otro(s) campo(s) debería enviar cada medidor de temperatura en cada mensaje?

I. Fecha y hora que se gatilla el evento

II. Temperatura anterior registrada

III. Identificador de la sala donde se encuentra ubicado

IV. Temperatura mínimas y máximas del día

a) Sólo I

b) Sólo III

c) Sólo I y III

d) Sólo II y IV

e) I, II, III y IV

Pregunta

Cuál es la mínima cantidad de instancias de procesamiento trabajando en paralelo debería tener un suscriptor para poder procesar todos los mensajes antes de recibir una nueva tanda de mediciones? Asuma que cada instancia procesa de a un mensaje por vez y cada mensaje demora en promedio 0.5 segundos en procesar.

a) 5 instancias

b) 10 instancias

c) 20 instancias

d) 50 instancias

e) 100 instancias

Procesamiento de eventos

Consumo de eventos: ¿Push o pull?

PUSH

El middleware determina cuando se debe enviar un evento al suscriptor.

El suscriptor no tiene que tener ninguna información del middleware porque recibe todos los datos.

La forma más común es que el evento se envía como un request a través de un webhook.

PULL

El procesador determina cuándo ir a buscar un evento.

El suscriptor debe mantener la información para ir a consultar la cola o middleware que tiene los eventos.

Pregunta

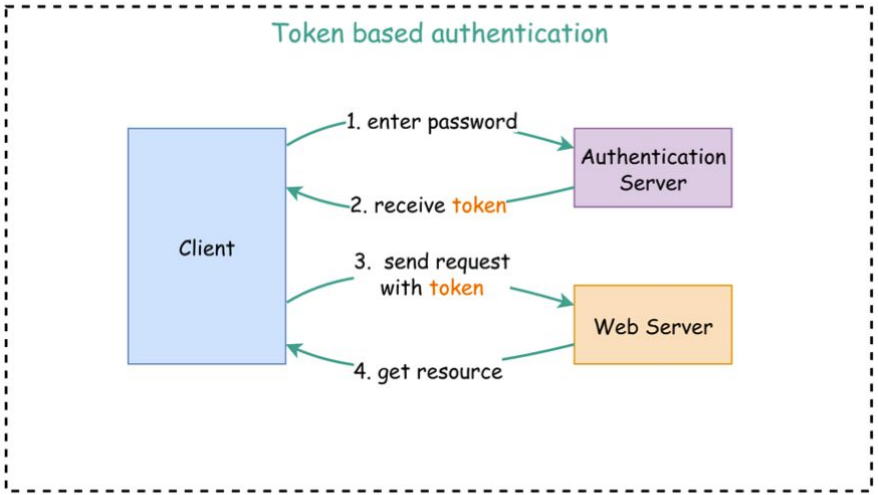
En un sistema de mensajes de eventos, los suscriptores pueden ir a buscar mensajes pendientes de lectura (sistemas tipo “pull”) o pueden recibir notificaciones cuando se generan nuevos mensajes (sistemas tipo “push”).

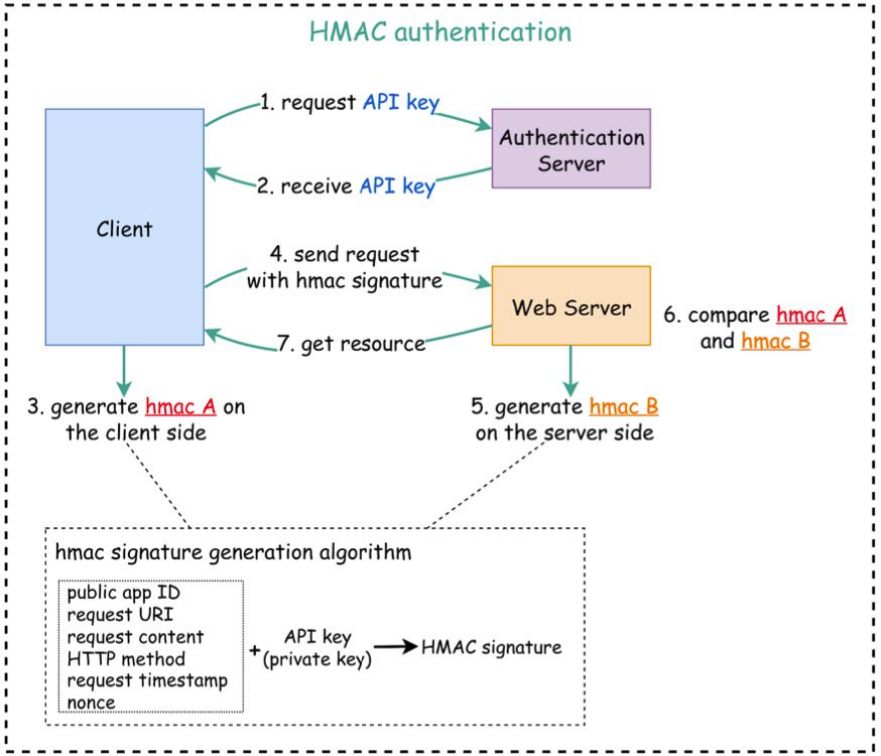
De las siguientes afirmaciones, ¿cuáles son verdaderas?

I. Los sistemas tipo “push” permiten generar experiencias de integración en tiempo real o cercanas a tiempo real
II. Los sistemas tipo “pull” otorgan mayor resiliencia frente a fallas ya que los mensajes son almacenados hasta que sean leídos.
III. Ambas formas de integración pueden ser consideradas como una forma de comunicación síncrona.

a) Sólo I
b) Sólo II
c) Sólo III
d) I y II
e) Todas son verdaderas

Diseñando API's seguras





Diseñando API’s seguras

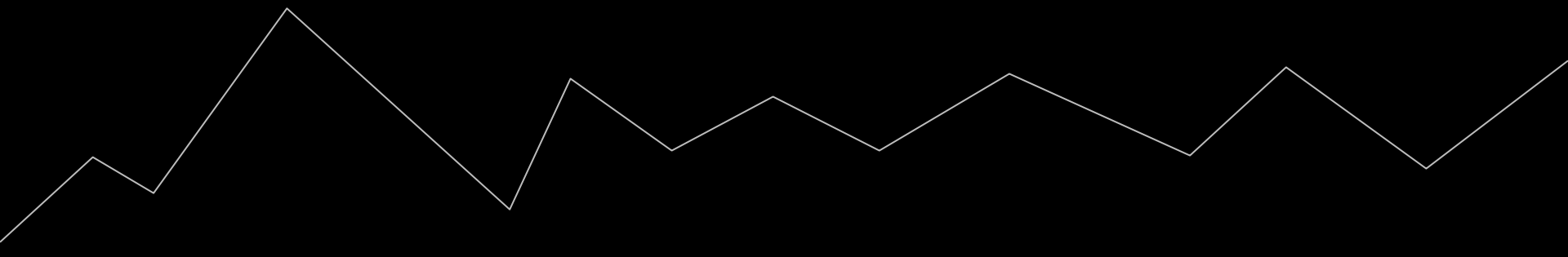
Token based

1. The user enters their password into the client, and the client sends the password to the Authentication Server.
2. The Authentication Server authenticates the credentials and generates a token with an expiry time.
3. Now the client can send requests to access server resources with the token in the HTTP header. This access is valid until the token expires.

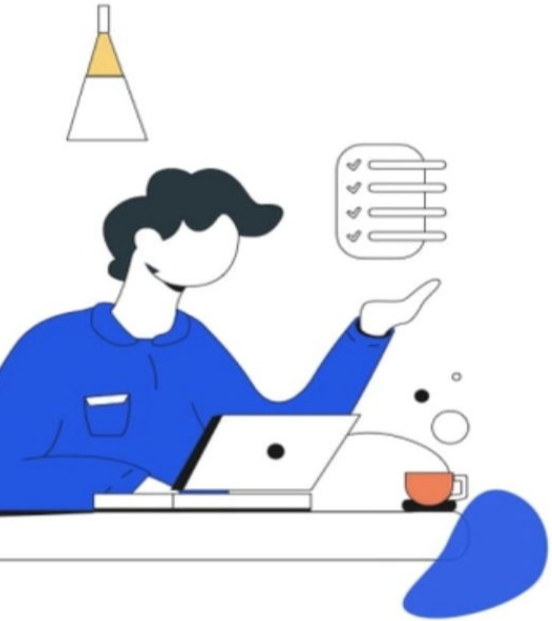
HMAC based

1. Steps 1 and 2 - the server generates two keys, one is Public APP ID (public key) and the other one is API Key (private key).
2. Step 3 - we now generate a HMAC signature on the client side (hmac A). This signature is generated with a set of attributes listed in the diagram.
3. Step 4 - the client sends requests to access server resources with hmac A in the HTTP header.
4. Step 5 - the server receives the request which contains the request data and the authentication header. It extracts the necessary attributes from the request and uses the API key that’s stored on the server side to generate a signature (hmac B.)
5. Steps 6 and 7 - the server compares hmac A (generated on the client side) and hmac B (generated on the server side). If they are matched, the requested resource will be returned to the client.

Arquitecturas de integración de datos



Transferencia de archivos

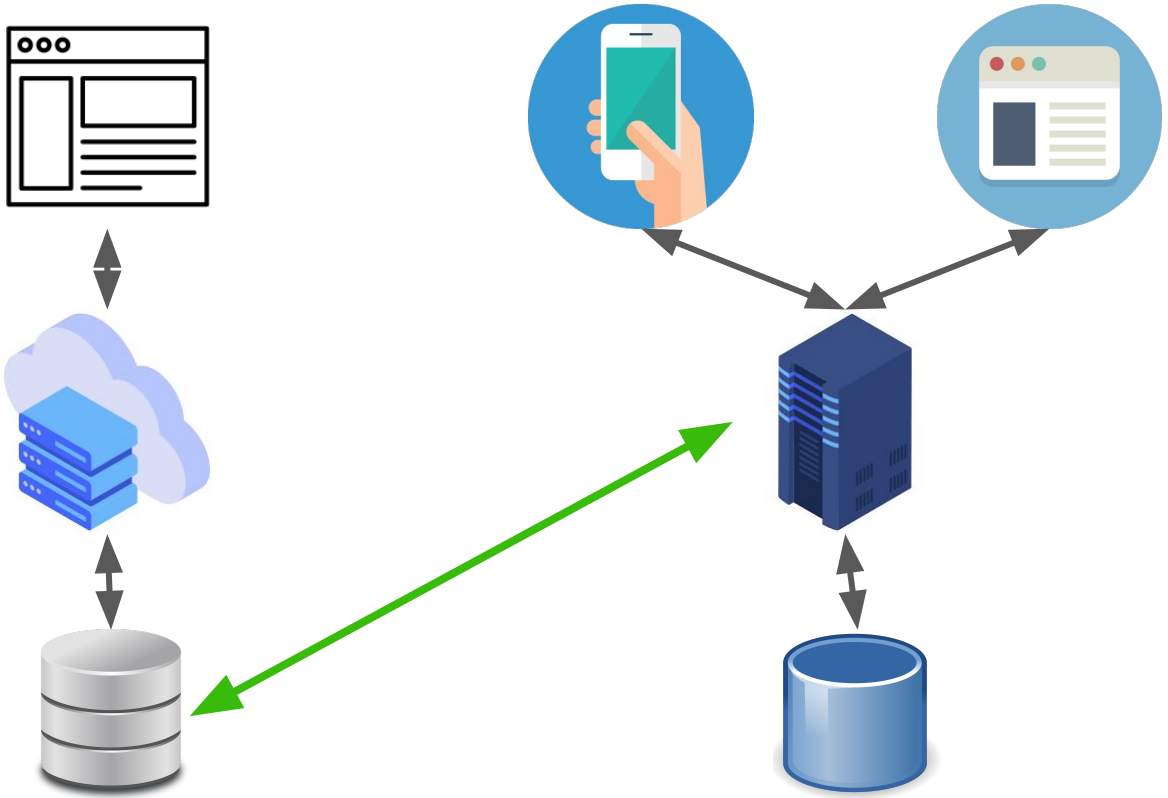


Se reduce a procesamiento y traspaso manual de datos mediante un archivo.

La principal razón de este caso es por falta de desarrollo o limitaciones tecnológicas en todo el stack de la aplicación.

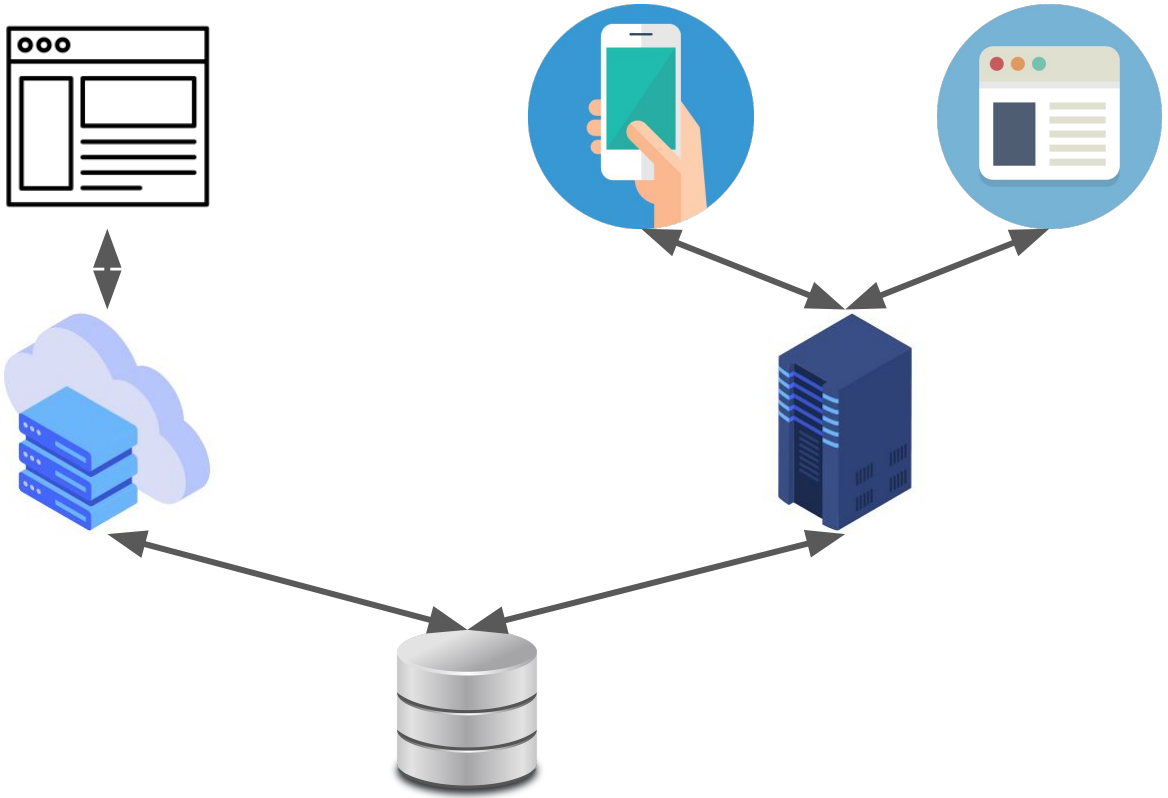
Arquitecturas de integración de datos

A nivel de capa de datos - Lectura cruzada



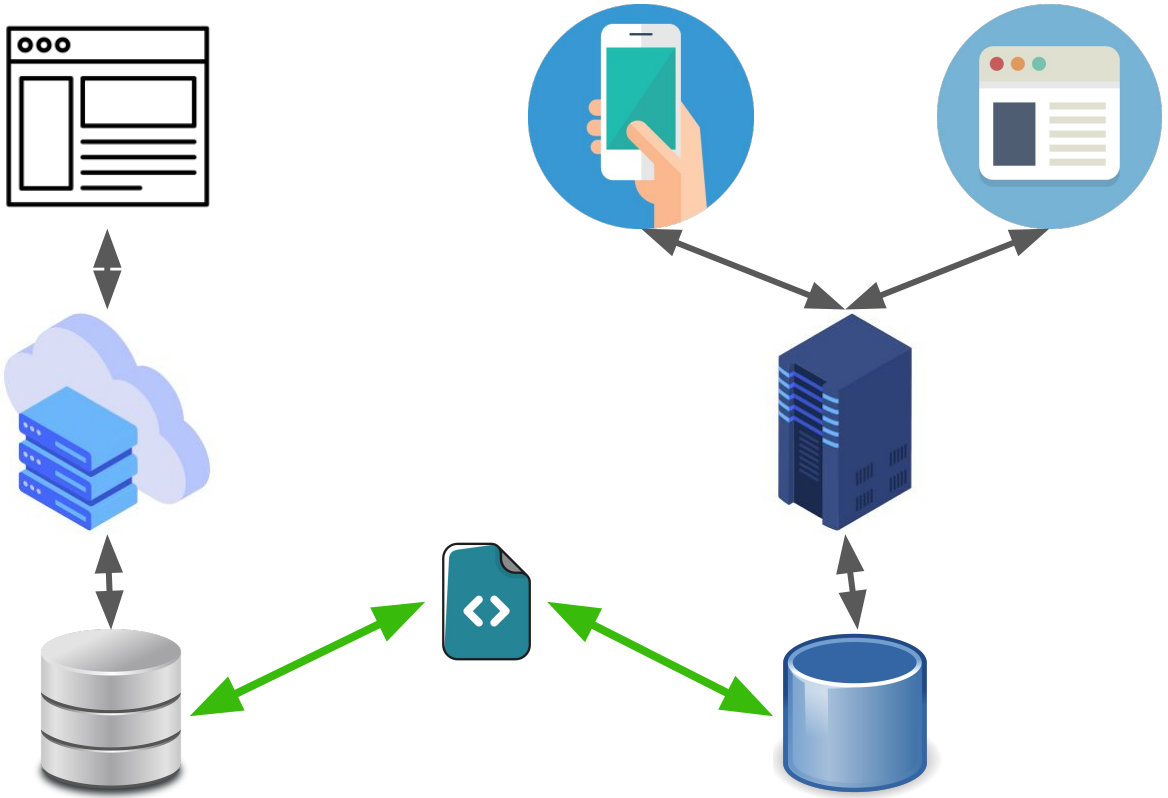
Arquitecturas de integración de datos

A nivel de capa de datos - BD compartida



Arquitecturas de integración de datos

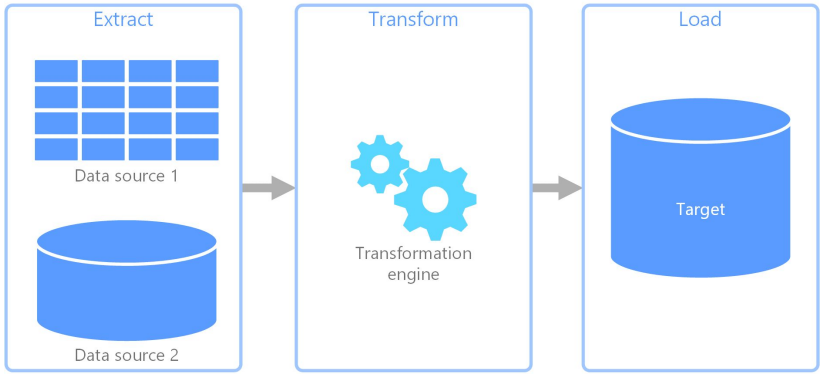
A nivel de capa de datos - Procesos de datos



Procesamiento de datos: ETL

Un ETL es un proceso de extracción, transformación y carga de datos a una nueva fuente de datos

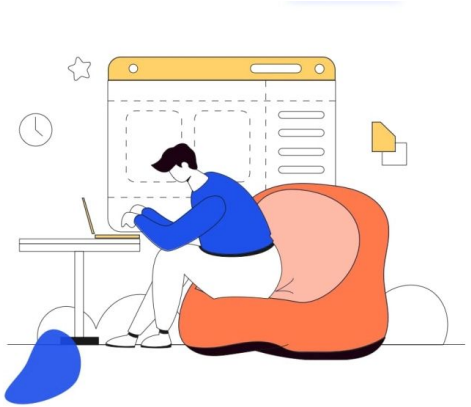
Normalmente se relacionan con procesos de carga de información en Datawarehouses



Procesamiento de datos: ETL

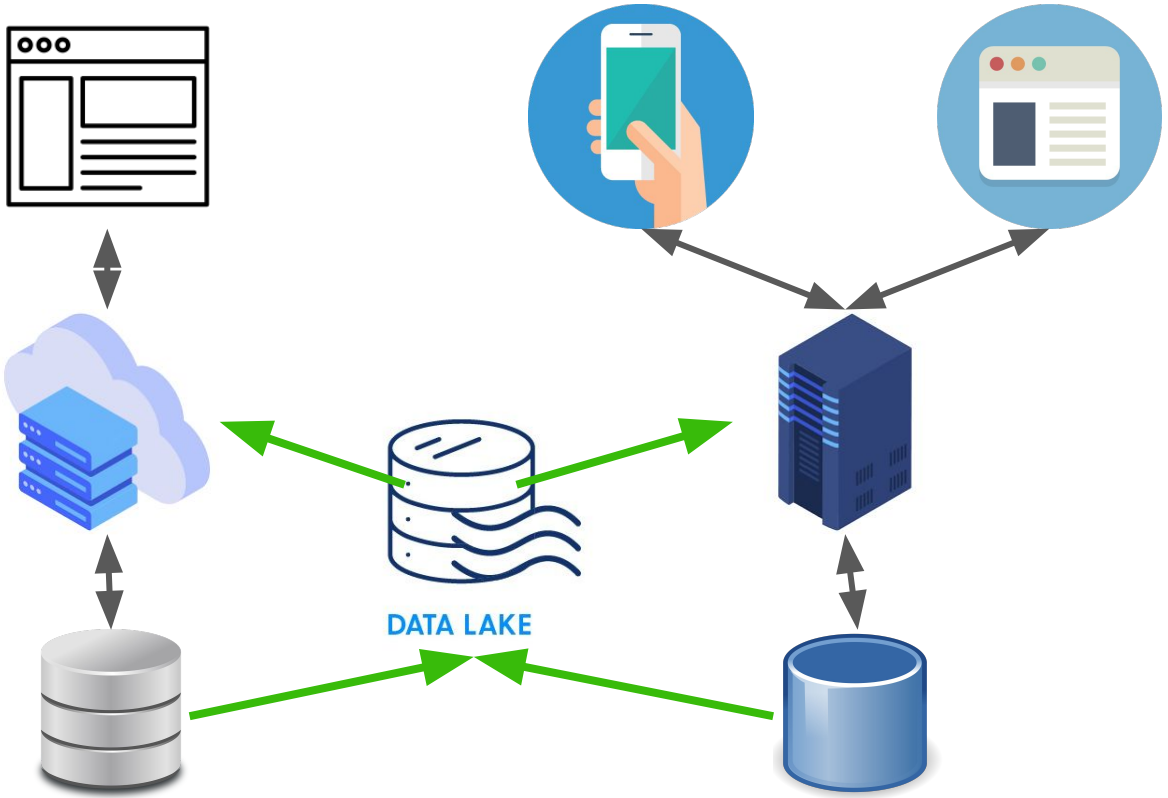
Casos de usos comunes:

- Extraer información de sistemas transaccionales a bases de analítica
- Migración de sistemas
- Integración de datos entre sistemas



Arquitecturas de integración de datos

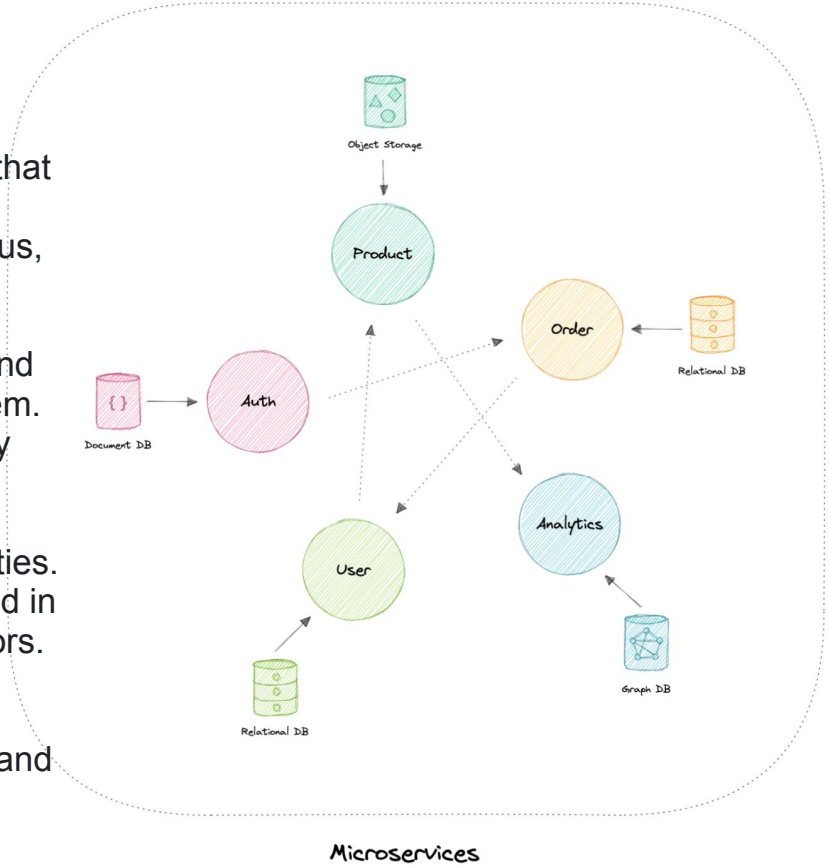
A nivel de capa de datos - Datos centralizados



Ejemplo: Arquitecturas de microservicios

The microservices architecture style has the following characteristics:

- Loosely coupled: Services should be loosely coupled so that they can be independently deployed and scaled. This will lead to the decentralization of development teams and thus, enabling them to develop and deploy faster with minimal constraints and operational dependencies.
- Small but focused: It's about scope and responsibilities and not size, a service should be focused on a specific problem. Basically, *"It does one thing and does it well"*. Ideally, they can be independent of the underlying architecture.
- Built for businesses: The microservices architecture is usually organized around business capabilities and priorities.
- Resilience & Fault tolerance: Services should be designed in such a way that they still function in case of failure or errors. In environments with independently deployable services, failure tolerance is of the highest importance.
- Highly maintainable: Service should be easy to maintain and test because services that cannot be maintained will be rewritten.



Otras referencias

1. System Design: Netflix - DEV Community
    a. ¿Cómo harías que la API descrita en el artículo sea una API REST?
2. AlgoDaily - Pub-Sub and Event Driven Architecture (EDA) - Introduction
    a. Responder preguntas en sitio

PONTIFICIA
UNIVERSIDAD
CATÓLICA
DE CHILE

IIC3103

Taller de Integración

Profesores

Arturo Tagle / Daniel Darritchon