ESTILOS Y PATRONES DE ARQUITECTURA – SOA, MAP-REDUCE, MULTI-TIER

ELSA ESTEVEZ
UNIVERSIDAD NACIONAL DEL SUR
DEPARTAMENTO DE CIENCIAS E INGENIERIA DE LA COMPUTACION

CONTENIDO

1	ARQUITECTURA ORIENTADA A SERVICIOS (SOA)
	CONCEPTO Y COMPONENTES
	PROPIEDADES Y BENEFICIOS
	CONEXIÓN E INTERMEDIARIOS
2	MAP-REDUCE
3	MULTI-TIER

ARQUITECTURA ORIENTADA A SERVICIOS – CONCEPTO

DEFINICION 1

SOA es una arquitectura de software donde todos los servicios y procesos implementados en software están diseñados como servicios a ser consumidos a través de una red.

DEFINICION 2

La funcionalidad es realizada por un conjunto de componentes cooperando que proveen o consumen servicios sobre una red. La funcionalidad es, en algunos casos, descripta a través de un lenguaje de workflows.

Palabra clave: SERVICIO

SOA – DESCRIPCION

- SOA describe una colección de componentes distribuidos que proveen y/o consumen servicios.
- Los servicios son, en gran medida, instalados independientemente.
- Los servicios pertenecen, frecuentemente, a diferentes sistemas y organizaciones.

SOA – ENFOQUE

El foco del diseño es la interface del servicio.

Un servicio:

- tiene una interface bien definida
- puede ser potencialmente invocado en una red
- puede ser reusado en múltiples contextos de negocio

Una aplicación:

- es integrada a nivel de interface y no a nivel de implementación
- es construida para trabajar con cualquier implementación de un contrato, resultando en un sistema menos acoplado y mas flexible

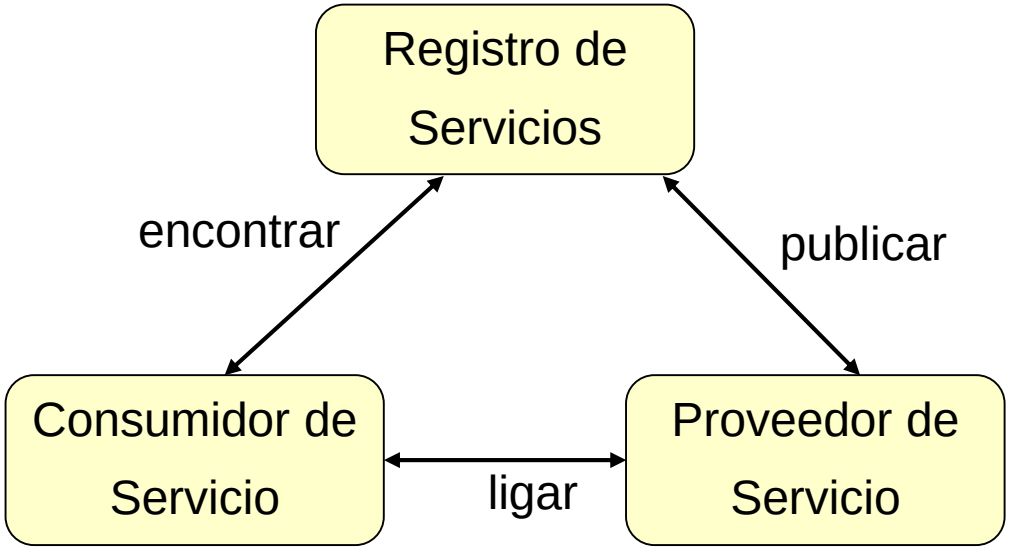
SOA – COMPONENTES

ROLES

- proveedor de servicio
- consumidor de servicio
- registro de servicio

OPERACIONES

- publicar
- encontrar
- ligar



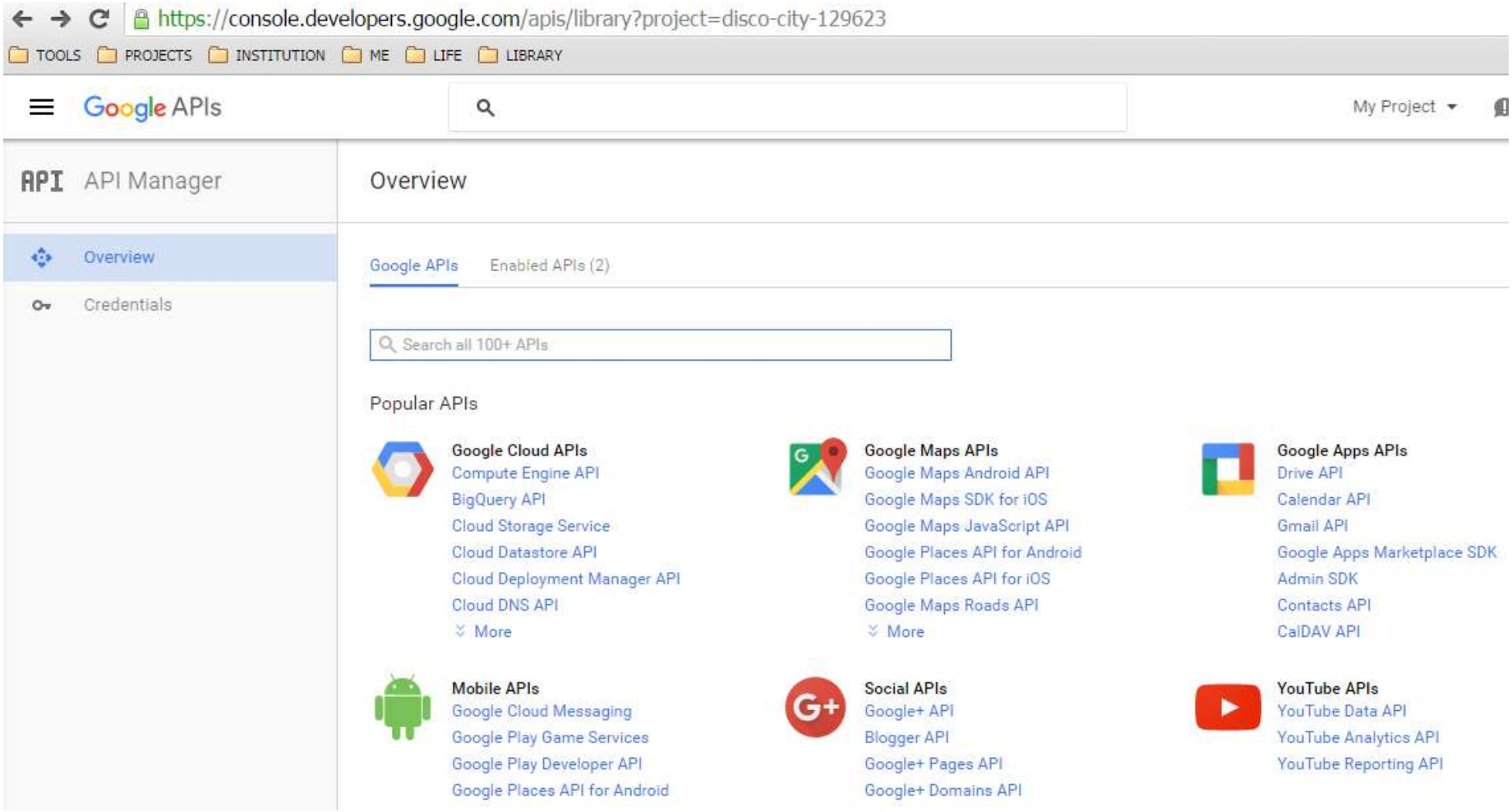
ROL – PROVEEDOR DE SERVICIO

Que hace un proveedor de servicio?

- crea una descripción de servicio
- despliega el servicio en un entorno de ejecución para hacerlo disponible a otras entidades sobre la red
- publica la descripción del servicio en uno o mas registros de servicios
- recibe mensajes invocando el servicio por parte de los consumidores del servicio

Cualquier entidad que aloja a un servicio web disponible a través de la red es un proveedor de servicios

PROVEEDOR DE SERVICIO – EJEMPLO



ROL – CONSUMIDOR DE SERVICIO

Que hace un consumidor de servicio?

- encuentra la descripción de un servicio publicada en un registro de servicios
- aplica la descripción del servicio para ligar e invocar al servicio web alojado en el proveedor del servicio

Un consumidor de servicio puede ser cualquier “entidad” que requiera de un servicio disponible en la red.

CONSUMIDOR DE SERVICIO – EJEMPLO

```
<!DOCTYPE html>
<html>
<head>
<script
src="http://maps.googleapis.com/maps/api/js">
</script>

<script>
function initialize() {
  var mapProp = {
    center:new google.maps.LatLng(-38.716666666667,-62.266666666667),
    zoom:5,
    mapTypeId:google.maps.MapTypeId.ROADMAP
  };
  var map=new google.maps.Map(document.getElementById("googleMap"), mapProp);
}
google.maps.event.addDomListener(window, 'load', initialize);
</script>
</head>

<body>
<div id="googleMap" style="width:500px;height:380px;"></div>

</body>
</html>
```

ROL – REGISTRO DE SERVICIO

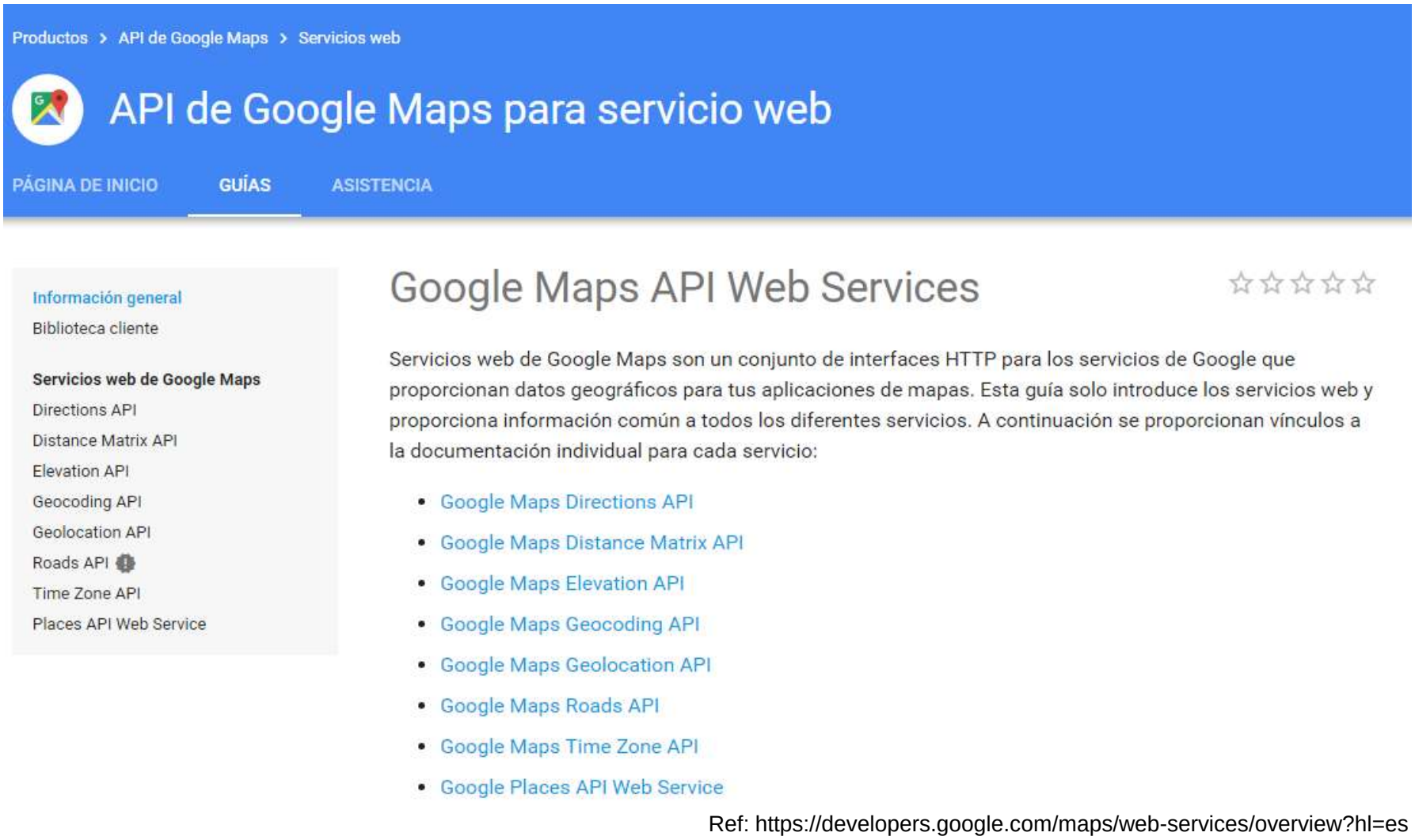
Que hace un registro de servicio (intermediario)?

- acepta pedidos de los proveedores de servicios para publicar y difundir las descripciones de los servicios web
- permite que los consumidores de servicios busquen en la colección de descripciones de servicios contenida dentro del registro

El rol del registro de servicios es permitir ligar a los proveedores de servicios con los consumidores.

Una vez que se han ligado, las interacciones se realizan directamente entre el proveedor y el consumidor del servicio.

REGISTRO DE SERVICIO – EJEMPLO 1



Ref: https://developers.google.com/maps/web-services/overview?hl=es

REGISTRO DE SERVICIO – EJEMPLO 2

Estructura de documento WSDL

```
<definitions>

<types>
  data type definitions........
</types>

<message>
  definition of the data being communicated....
</message>

<portType>
  set of operations......
</portType>

<binding>
  protocol and data format specification....
</binding>

</definitions>
```

Ejemplo de Operación “Request-Response”

```
<message name="getTermRequest">
  <part name="term" type="xs:string"/>
</message>

<message name="getTermResponse">
  <part name="value" type="xs:string"/>
</message>

<portType name="glossaryTerms">
  <operation name="getTerm">
    <input message="getTermRequest"/>
    <output message="getTermResponse"/>
  </operation>
</portType>
```

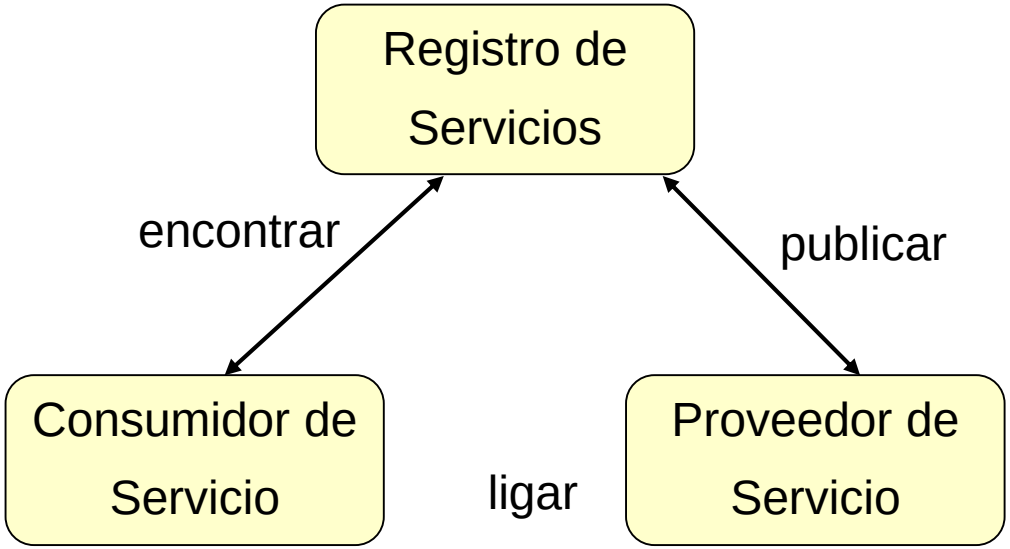
SOA – COMPONENTES

ROLES

- proveedor de servicio
- solicitante de servicio
- registro de servicio

OPERACIONES

- publicar
- encontrar
- ligar

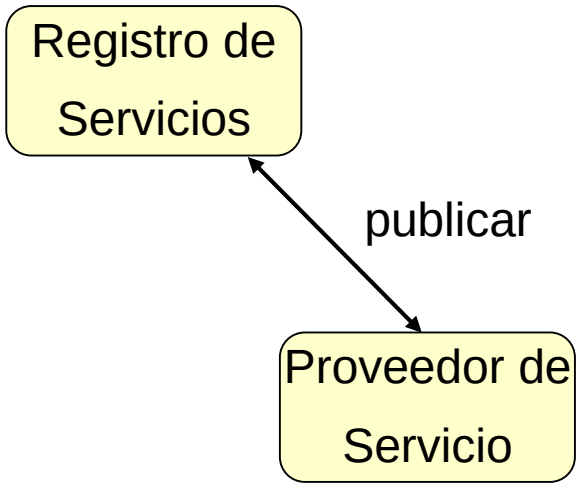


OPERACIÓN – PUBLICAR

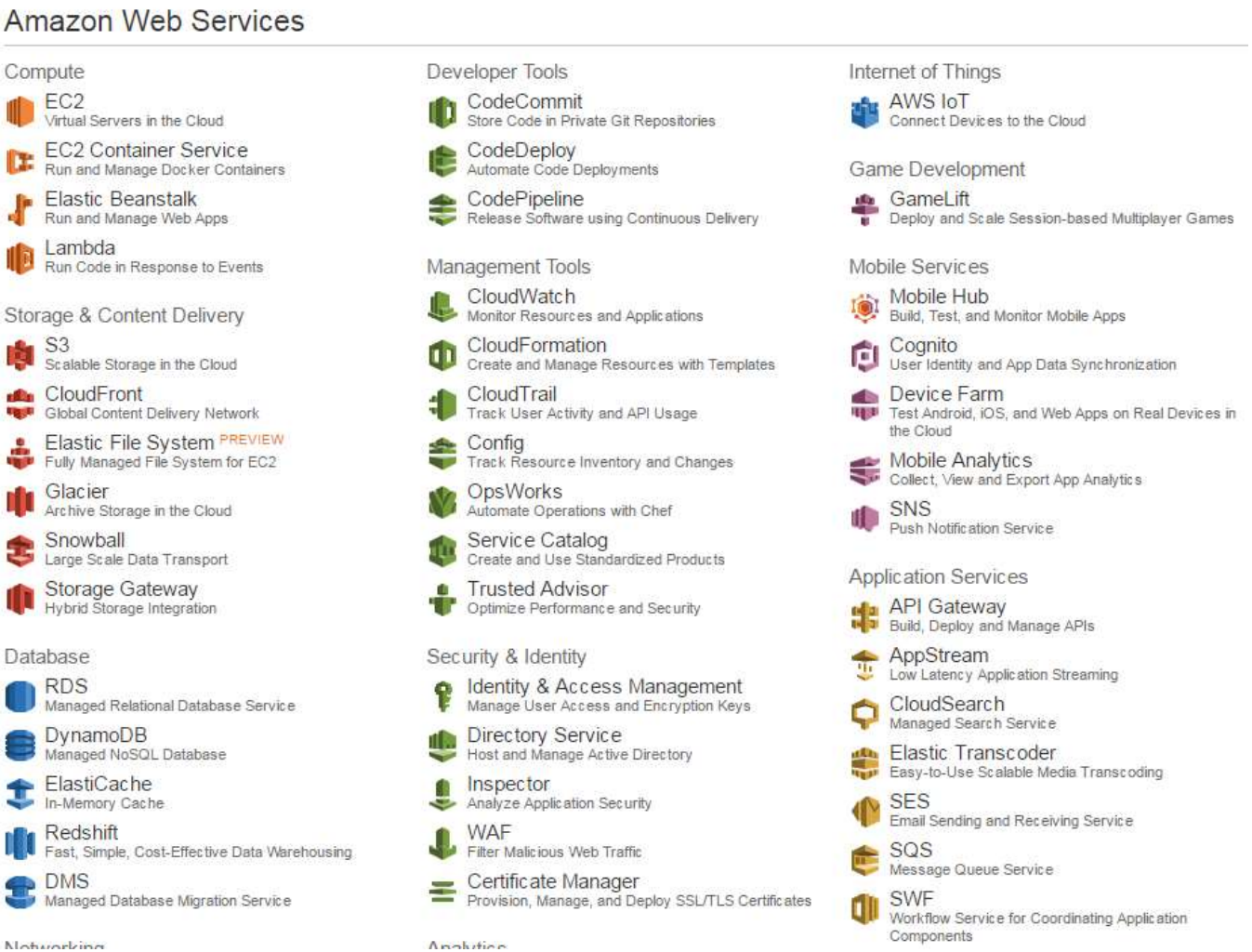
La operación de publicar es un acto de registración o anuncio del servicio.

Cuando un proveedor de servicio publica su servicio en la web, está anunciando el servicio a toda la comunidad de potenciales solicitantes del servicio.

Los detalles de la operación de publicación depende de como el registro de servicios esté implementado.



OPERACIÓN PUBLICAR – EJEMPLO

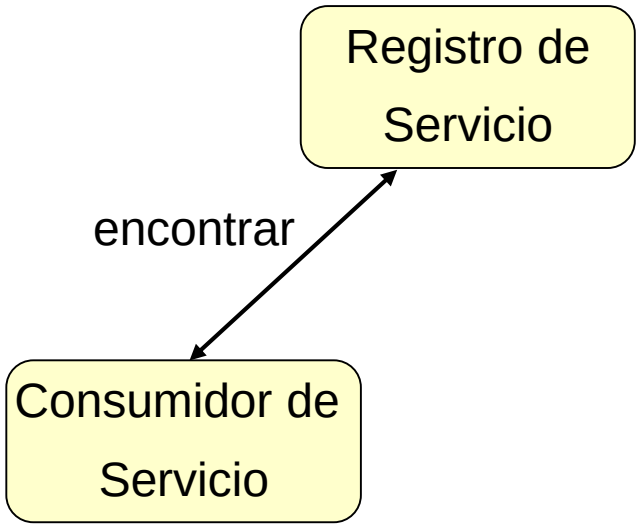


Ref: https://us-west-2.console.aws.amazon.com/console/...

OPERACIÓN – ENCONTRAR

La operación de encontrar es el acto de buscar por un servicio que satisfaga determinadas condiciones:

- El solicitante del servicio define el criterio de búsqueda, por ejemplo: tipo de servicio, calidad, etc.
- El registro de servicio busca de acuerdo al criterio definido en las descripciones de servicio que tiene publicadas.



El resultado es una lista de descripciones de servicio que satisfacen el criterio de búsqueda.

Los detalles de la operación depende de la implementación del registro de servicios.

OPERACIÓN ENCONTRAR – EJEMPLO

Browse the world's largest API repository

Search Over 14,993 APIs **SEARCH APIS**

Filter APIs

By Category | By Protocols/Formats | Include Deprecated APIs

API Name	Description	Category	Updated
Google Maps	The Google Maps API allow for the embedding of Google Maps onto web pages of outside developers, using a simple JavaScript interface or a Flash interface. It is designed to work on both mobile...	Mapping	12.05.2005
Twitter	The Twitter micro-blogging service includes two RESTful APIs. The Twitter REST API methods allow developers to access core Twitter data. This includes update timelines, status data, and user...	Social	12.08.2006
YouTube	The Data API allows users to integrate their program with YouTube and allow it to perform many of the operations available on the website. It provides the capability to search for videos, retrieve...	Video	02.08.2006
Flickr	The Flickr API can be used to retrieve photos from the Flickr photo sharing service using a variety of feeds - public photos and videos, favorites, friends, group pools, discussions, and more. The...	Photos	09.04.2005

Ref: http://www.programmableweb.com/apis/directory

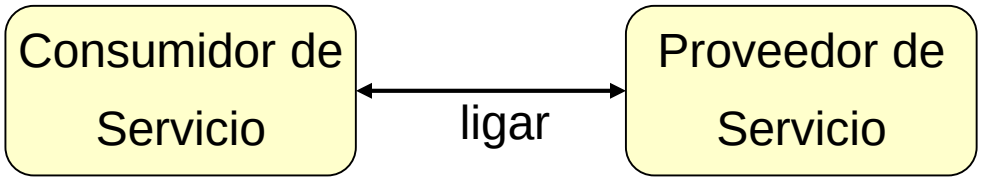
OPERACIÓN – LIGAR

La operación de ligar crea la relación cliente-servidor entre el proveedor y el consumidor del servicio.

La operación puede ser:

- dinámica – se crea un proxy cliente-servidor “en el aire” en base a la descripción del servicio para invocar al servicio web

- estática – el desarrollador incluye en el código la forma en que el cliente invoca el servicio web



OPERACIÓN LIGAR – EJEMPLO

Ejemplo de invocación dinámica

```
public class Main {

  public static void main(String[] args) throws Exception {

    // Setup the global JAXM message factory
    System.setProperty("javax.xml.soap.MessageFactory",
      "weblogic.webservice.core.soap.MessageFactoryImpl");
    // Setup the global JAX-RPC service factory
    System.setProperty( "javax.xml.rpc.ServiceFactory",
      "weblogic.webservice.core.rpc.ServiceFactoryImpl");

    // create service factory
    ServiceFactory factory = ServiceFactory.newInstance();

    // define qnames
    String targetNamespace =
      "http://www.themindelectric.com/"
      + "wsdl/net.xmethods.services.stockquote.StockQuote/";

    QName serviceName =
      new QName(targetNamespace,
                "net.xmethods.services.stockquote.StockQuoteService");

    QName portName =
      new QName(targetNamespace,
                "net.xmethods.services.stockquote.StockQuotePort");

    QName operationName = new QName("urn:xmethods-delayed-quotes",
                                    "getQuote");

    URL wsdlLocation =
      new URL("http://services.xmethods.net/soap/urn:xmethods-delayed-quotes.wsdl");

    // create service
    Service service = factory.createService(wsdlLocation, serviceName);

    // create call
    Call call = service.createCall(portName, operationName);

    // invoke the remote web service
    Float result = (Float) call.invoke(new Object[] {
      "BEAS"
    });
```

SOA – PROPIEDADES 1

SOA en una clase de arquitectura para sistemas distribuidos

PROPIEDADES

Visión lógica	Los servicios son una abstracción de lo que los programas, bases de datos y procesos de negocio son capaces de hacer.
Abstracción	SOA esconde los detalles subyacentes de implementación – e.g. de los lenguajes, procesos, estructuras de base de datos, etc.
Relevancia de mensajes	Un servicio es definido en término de los mensajes que se intercambian entre el agente proveedor y el consumidor, no en términos de las propiedades de los agentes en si mismos.

SOA – PROPIEDADES 2

SOA en una clase de arquitectura para sistemas distribuidos

PROPIEDADES

Metadatos	Un servicio es descripto por meta-datos procesables automáticamente.
Operaciones	Un servicio tiende a depender de un número pequeño de operaciones con mensajes relativamente largos y complejos.
Red	Los servicios están definidos para ser usados sobre una red.
Independencia de plataforma	Los mensajes son envidos en un formato estandarizado definido, usualmente XML, enviados a través de interfaces.

SOA – BENEFICIOS 1

SOA permite que los agentes que participan en el intercambio de mensajes estén débilmente acoplados; esto permite una mayor flexibilidad:

- un cliente sólo se acopla a un servicio, no a un servidor - la integración del servidor se lleva a cabo fuera del ámbito de la aplicación cliente
- las componentes funcionales y sus interfaces están separadas - nuevas interfaces pueden ser fácilmente añadidas
- funcionalidad vieja y la nueva se puede encapsular como componentes de software que proporcionan y solicitan servicios

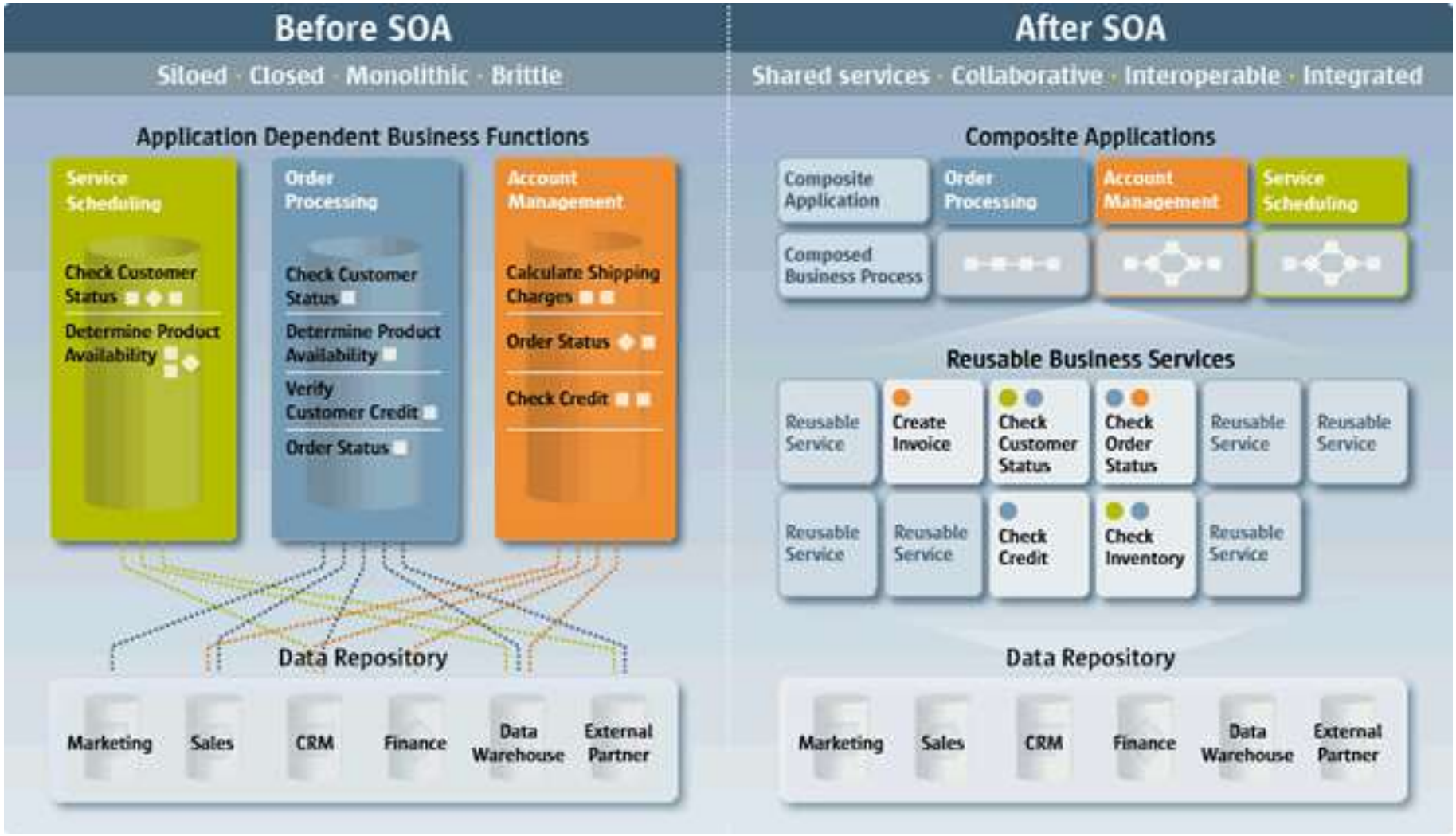
SOA – BENEFICIOS 2

- el control de los procesos de negocio puede ser aislado:
  - ✓ un motor de procesos de negocio (“workflow engine”) puede controlar el flujo de trabajo de un proceso de negocio
  - ✓ dependiendo del estado, el motor invoca diferentes servicios
- los servicios se pueden incorporar de forma dinámica en tiempo de ejecución
- los enlaces de servicio se especifican mediante los archivos de configuración y se pueden adaptar fácilmente para satisfacer las nuevas necesidades

SOA – POTENCIALES PROBLEMAS

- complejidad en el diseño e implementación, debido al binding dinámico y el uso de meta-datos
- overhead en la performance del middleware (intermediario) que se interpone entre servicios y clientes
- carencia de garantía de performance, ya que los servicios se comparten y no están bajo el control del cliente

MIGRACION A SOA



SOA – DESCRIPCION DE SERVICIOS

El elemento clave de SOA es la descripción del servicio:

- es publicada por el proveedor de servicios en el registro de servicios
- es devuelta al consumidor del servicio como consecuencia de la operación de búsqueda
- especifica al consumidor del servicio:
  - ✓ cómo enlazar e invocar el servicio web
  - ✓ qué información se devuelve como resultado de la invocación

CONTENIDO

1	SERVICE-ORIENTED ARQUITECTURE (SOA)
	CONCEPTO Y COMPONENTES
	PROPIEDADES Y BENEFICIOS
	CONEXIÓN E INTERMEDIARIOS
2	MAP-REDUCE
3	MULTI-TIER

CONEXIONES

Tres posibles tipos de conexión:

1) Simple Object Access Protocol – SOAP

2) Representational State Transfer – REST

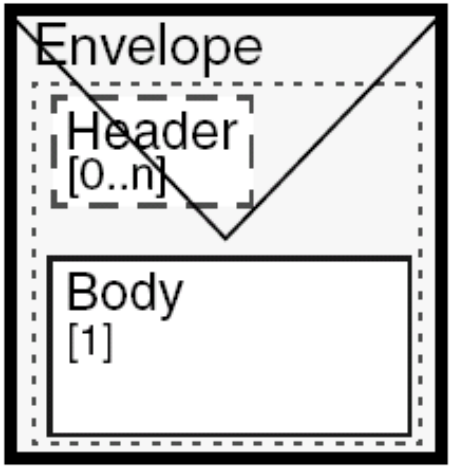
3) Mensajes asincrónicos

CONEXIONES – SOAP

- es el protocolo estándar de Web Services
- los proveedores y consumidores interactúan intercambiando mensajes XML del tipo requerimiento/respuesta sobre HTTP
- Está acompañado por un conjunto de estándares (WS*):
  - ✓ WSDL (Web Services Description Language, W3C https://www.w3.org/TR/wsdl
  - ✓ UDDI (Universal, Description, Discovery and Integration, UDDI) http://uddi.xml.org/
  - ✓ WS-Transaction, OASIS https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ws-tx
  - ✓ WS-Reliability, OASIS http://docs.oasis-open.org/wsrm/ws-reliability/v1.1/wsrm-ws_reliability-1.1-spec-os.pdf

SOAP – EJEMPLO

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
  <soap:Body>
    <x:TransferFunds xmlns:x="urn:examples-org:banking">
      <x:from>983-23456</x:from>
      <x:to>672-24806</x:to>
      <x:amount>1000.00</x:amount>
    </x:TransferFunds>
  </soap:Body>
</soap:Envelope>
```



[Cortesía de IBM]

CONEXIONES – REST

- Funciona como una arquitectura cliente-servidor, con una interface bien definida y separa claramente las responsabilidades entre cliente y servidor
- El consumidor envía requerimientos HTTP no bloqueantes
- Los servicios no tiene estado asociado al cliente
- Todos los servicios usan la misma forma de invocación, utilizando 4 métodos básicos: POST (crear), GET (recuperar), PUT (actualizar), DELETE (eliminar)

REST – EJEMPLO

REST

```
http://www.empresa.com/directorio/ClientDetails/1407
```

SOAP

```
<?xml version="1.0"?>
 <soap:Envelope
   xmlns:
   soap="http://www.w3.org/2001/12/soap-envelope"
   soap:encodingStyle=
       "http://www.w3.org/2001/12/soap-encoding">
   <soap:body em="http://www.empresa.com/directorio">
      <em:GetClientDetails>
            <em:ClientID>1407</em:ClientID>
      </em:GetClientDetails>
</soap:Body>
</soap:Envelope
```

MENSAJES ASINCRONICOS

- Intercambio a través de eventos que se comunican a través de mensajes
- Los participantes no tienen que esperar un reconocimiento de la recepción del mensaje
- Se asume que la infraestructura entregará los mensajes exitosamente
- Pueden ser del tipo point-to-point o publish-subscriber
- Implementaciones
  - ✓ IBM's WebSphere MQ
  - ✓ Microsoft MSMQ
  - ✓ Apache ActiveMQ

DIFERENCIAS ENTRE SOAP Y REST 1

	SOAP	REST
Origen	Modelo de llamadas entre aplicaciones del tipo Llamada a Procedimiento Remoto (RPC)	Modelo cliente-servidor sobre HTTP
Transmisión de Mensajes	HTTP y RPC Otros protocolos son posibles.	HTTP
Restricciones	Muchos aspectos a resolver: ○ seguridad ○ transacciones ○ interpretación de mensaje ○ esquema de direccionamiento	Pocas. ○ único esquema de direccionamiento (URI) ○ operaciones limitadas (CRUD)
Sistema de tipos	Simple, comparable al de muchos lenguajes de programación	No tiene

DIFERENCIAS ENTRE SOAP Y REST 1

	SOAP	REST
Semántica	Las aplicaciones que interactúan necesitan acordar cómo interpretar los mensajes	Si bien la sintaxis es bastante auto descriptiva, no está garantizada
Tradeoffs	○ completitud ○ estandarización de la interoperabilidad ○ mensajes estructurados	○ simplicidad ○ mínimo overhead ○ performance
Calidad de Servicio	La familia de tecnologías de Servicios Web tiene soporte para seguridad, disponibilidad, manejo de transacciones, etc.	No brinda soporte intrínseco. Apropiado para operaciones read-only

INTERMEDIARIOS

Tres ejemplos comúnmente usados son:

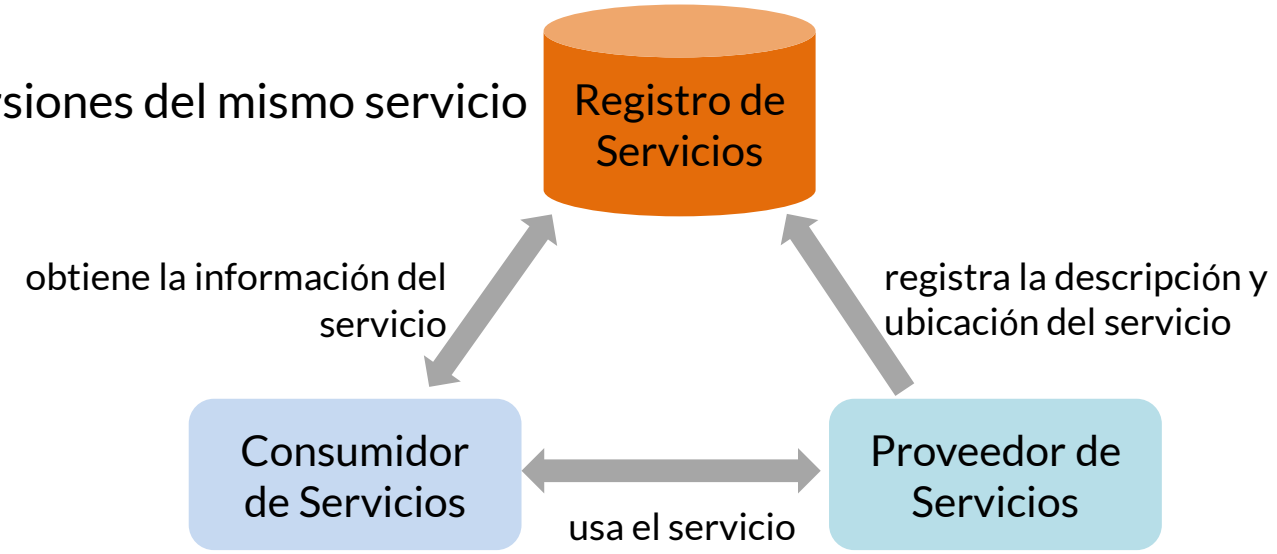
1) Registro de Servicios

2) Enterprise Service Bus (ESB)

3) Orquestación de Servicios

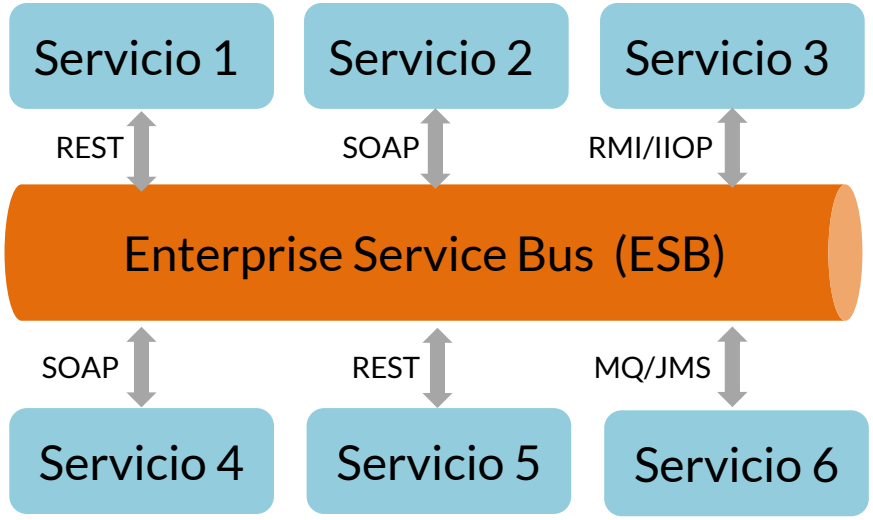
REGISTRO DE SERVICIOS

- es un componente que permite que los servicios se registren en tiempo de ejecución, mejorando la independencia de los proveedores de servicios
- permite el descubrimiento de los servicios en tiempo de ejecución
- promueve la facilidad de modificar y ocultar la identidad y ubicación de los proveedores de servicios
- permite múltiples versiones del mismo servicio



ENTERPRISE SERVICE BUS

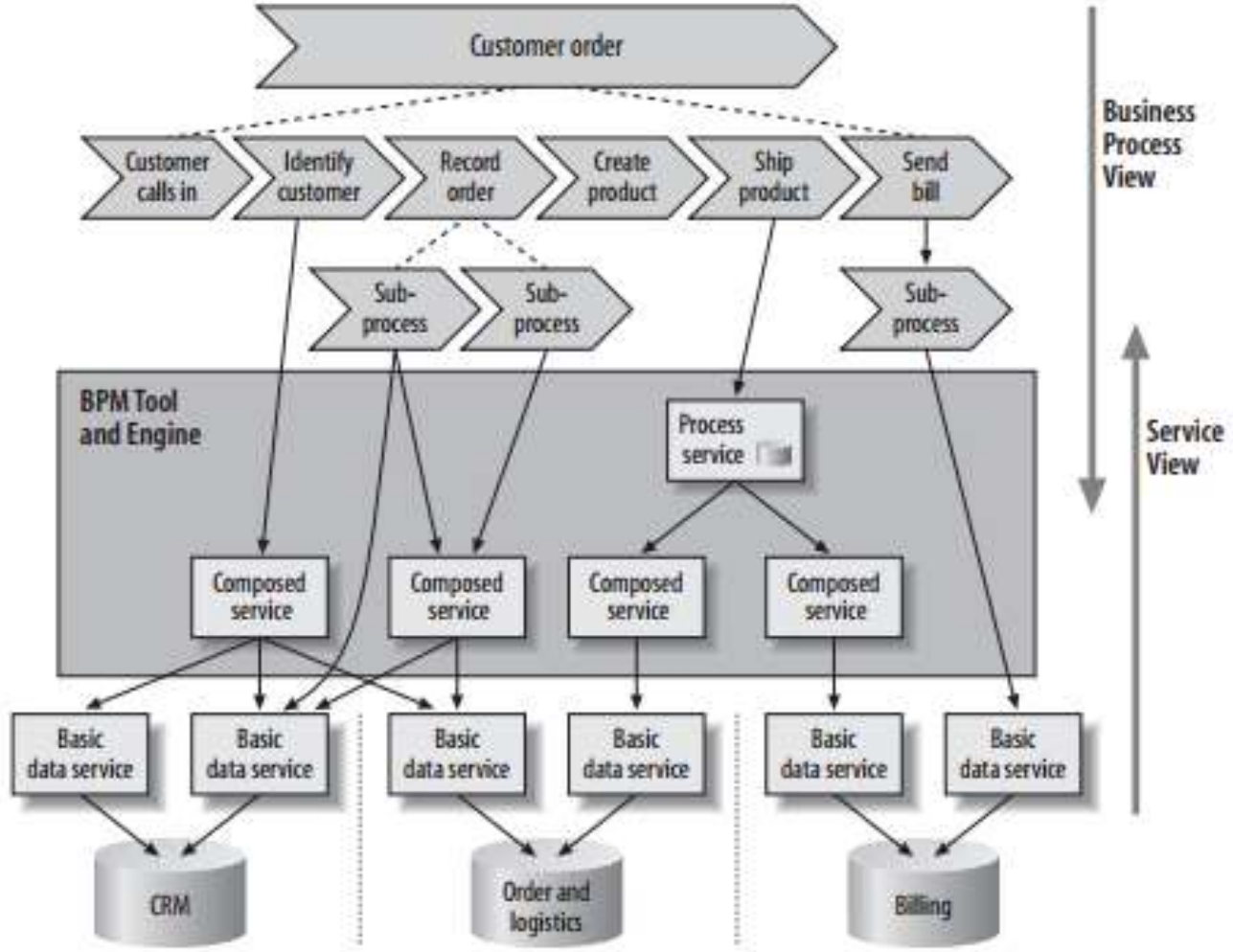
- direcciona mensajes entre proveedores y consumidores
- convierte mensajes de un protocolo (o tecnología) a otro
- realiza transformaciones sobre los datos
- realizar chequeos de seguridad
- maneja transacciones
- promueve interoperabilidad, seguridad y modificabilidad



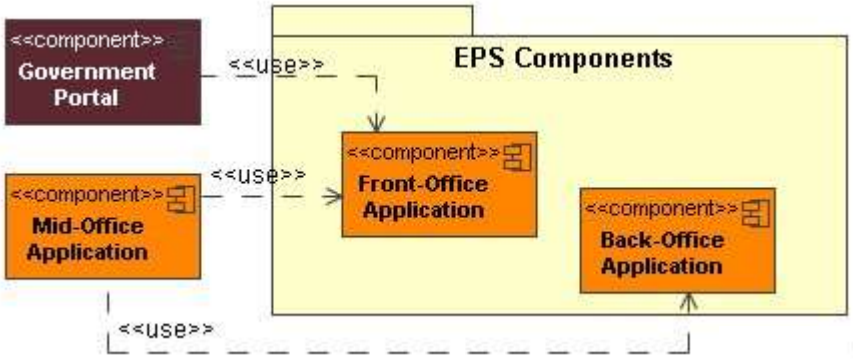
ORQUESTACION

- orquesta la interacción entre varios consumidores y proveedores de servicios
- ejecuta scripts ante la ocurrencia de un evento específico (BPEL, BPMN, etc.)
- permite crear nuevos servicios a partir de otros más granulares
- útil para aplicaciones con procesos de negocio bien definidos que involucran interacciones con componentes o sistemas distribuidos
- favorecen la modificabilidad, interoperabilidad y confiabilidad
- Ejemplos: jBPM, Mule Enterprise, BizTalk, Oracle BPM, Orchestra

ORQUESTACION – EN UNA IMAGEN



SOA – APLICACION

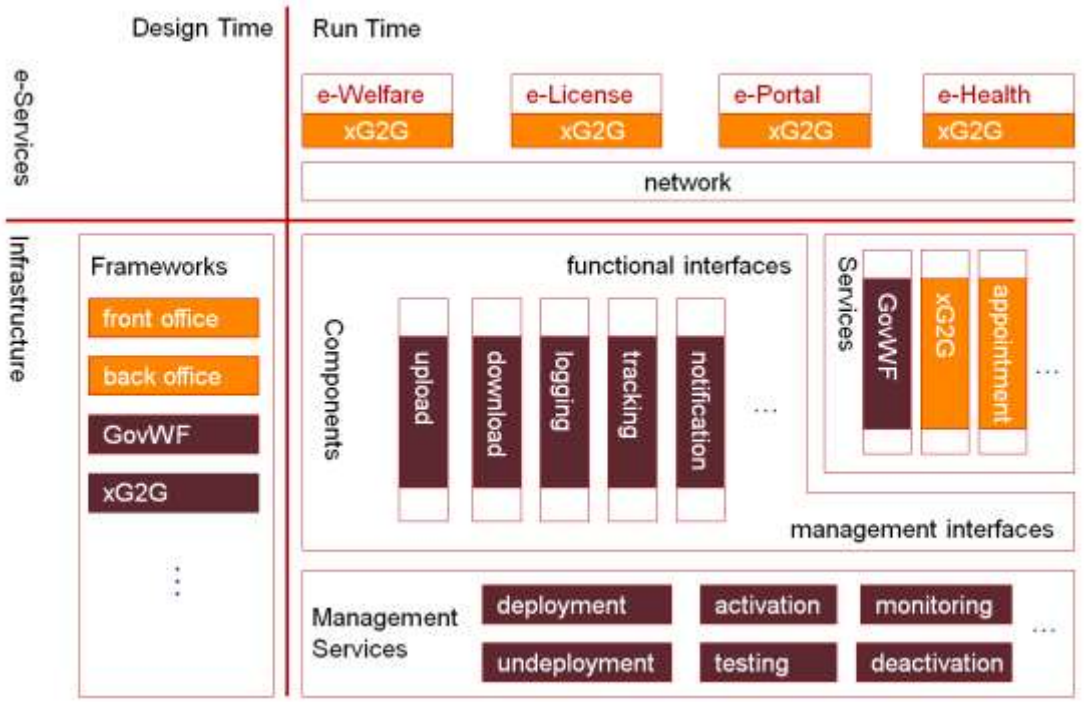


MAIN SOFTWARE COMPONENTS

1) front-office framework
2) back-office framework

SOFTWARE INFRASTRUCTURE

1) design time frameworks
   - front-office
   - back-office
2) run time services:
   - messaging
   - appointment
   - notification
   - ...



CONTENIDO

1 ARQUITECTURA ORIENTADA A SERVICIOS (SOA)

CONCEPTO Y COMPONENTES

PROPIEDADES Y BENEFICIOS

CONEXIÓN E INTERMEDIARIOS

2 MAP-REDUCE

3 MULTI-TIER

MAP-REDUCE – MOTIVACION

- en la actualidad, las organizaciones necesitan analizar rápidamente grandes volúmenes de datos que generan en un orden de petabyte (10^{15} bytes)
  - interacciones en redes sociales
  - repositorios de datos o documentos masivos
  - pares <origen, destino> para un motor de búsqueda.
- se debe asegurar eficiencia y resiliencia con respecto a fallas en el hardware

MAP-REDUCE – DESCRIPCION

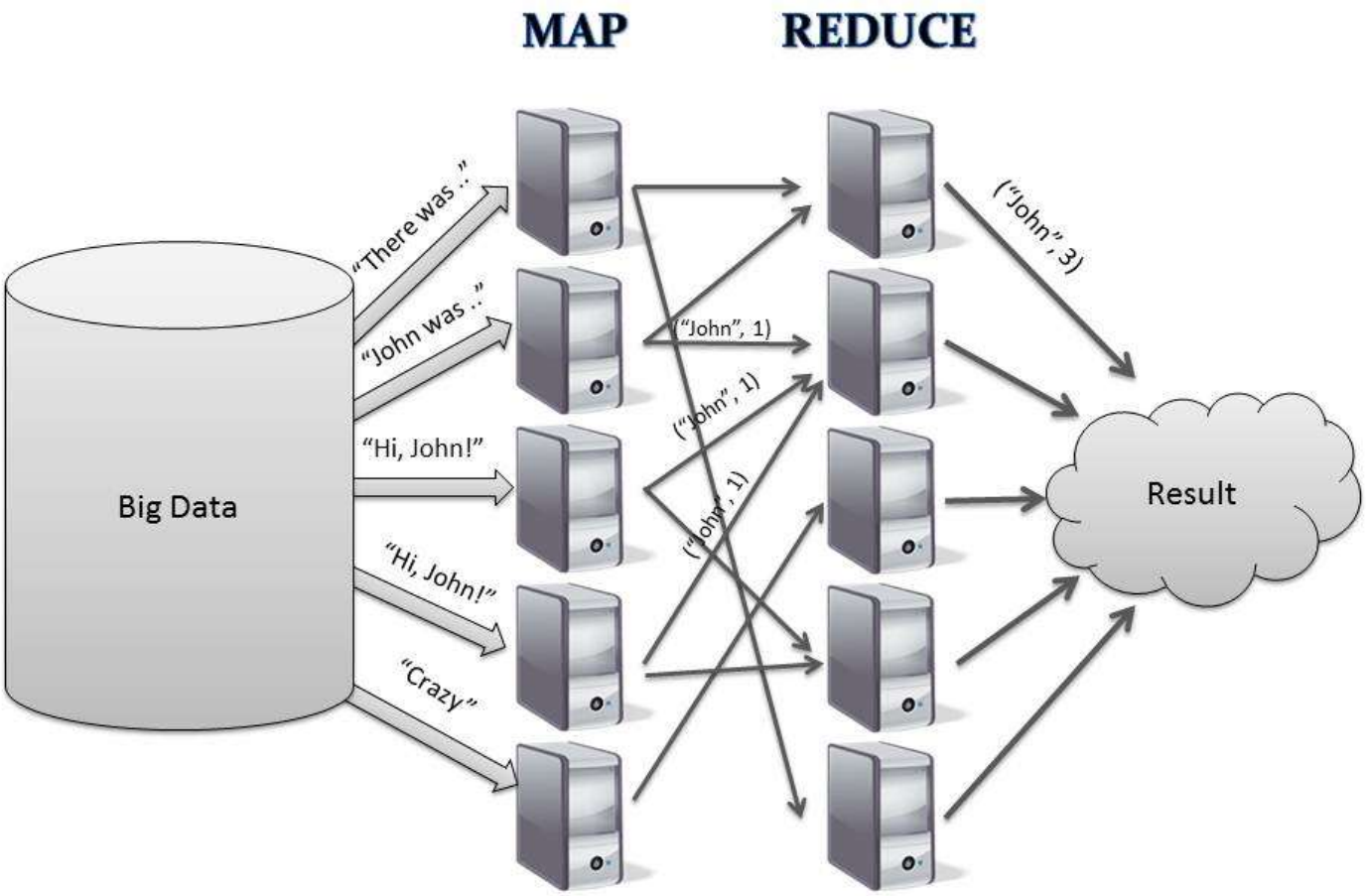
- Map-Reduce provee un marco de trabajo para analizar un conjunto de datos grande y distribuido, que se ejecuta en paralelo sobre un conjunto de procesadores.
- el paralelismo permite baja latencia y alta disponibilidad

Requerimientos:

Una infraestructura especializada que se encargue de:

- alocar los componentes de software a los nodos de hardware en un ambiente de computación masivamente paralelo, asegurando disponibilidad y recuperación ante caídas
- manejar el ordenamiento de los datos como se necesite
- dos funciones: map y reduce - definidas en términos de t-uplas

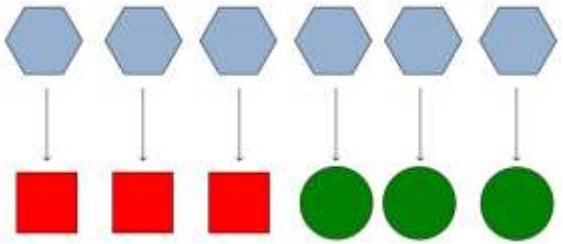
MAP REDUCE – EN UNA IMAGEN



Ref: dme.rwth-aachen.de

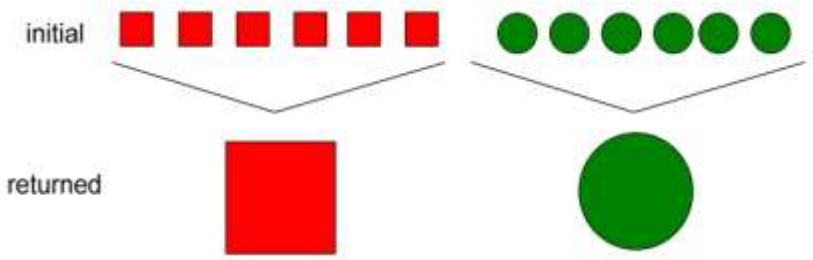
FUNCIONAMIENTO – FUNCION MAP

```
Map(k1,v1) -> list(k2,v2)
```



- toma una lista de pares en un dominio de datos y devuelve una lista de pares en un dominio diferente (datos intermedios)
- su propósito es filtrar el data set, determinando si un registro va a estar involucrado en el procesamiento posterior
- k2 es importante ya que será usada para ordenar (agrupar) los datos de salida de esta función
- es aplicada en paralelo a cada ítem (o lote de ítems) de la entrada de datos
- el resultado de esta función es tomado por la infraestructura que junta los pares con la misma clave y los agrupa, creando un grupo por c/u de las claves.

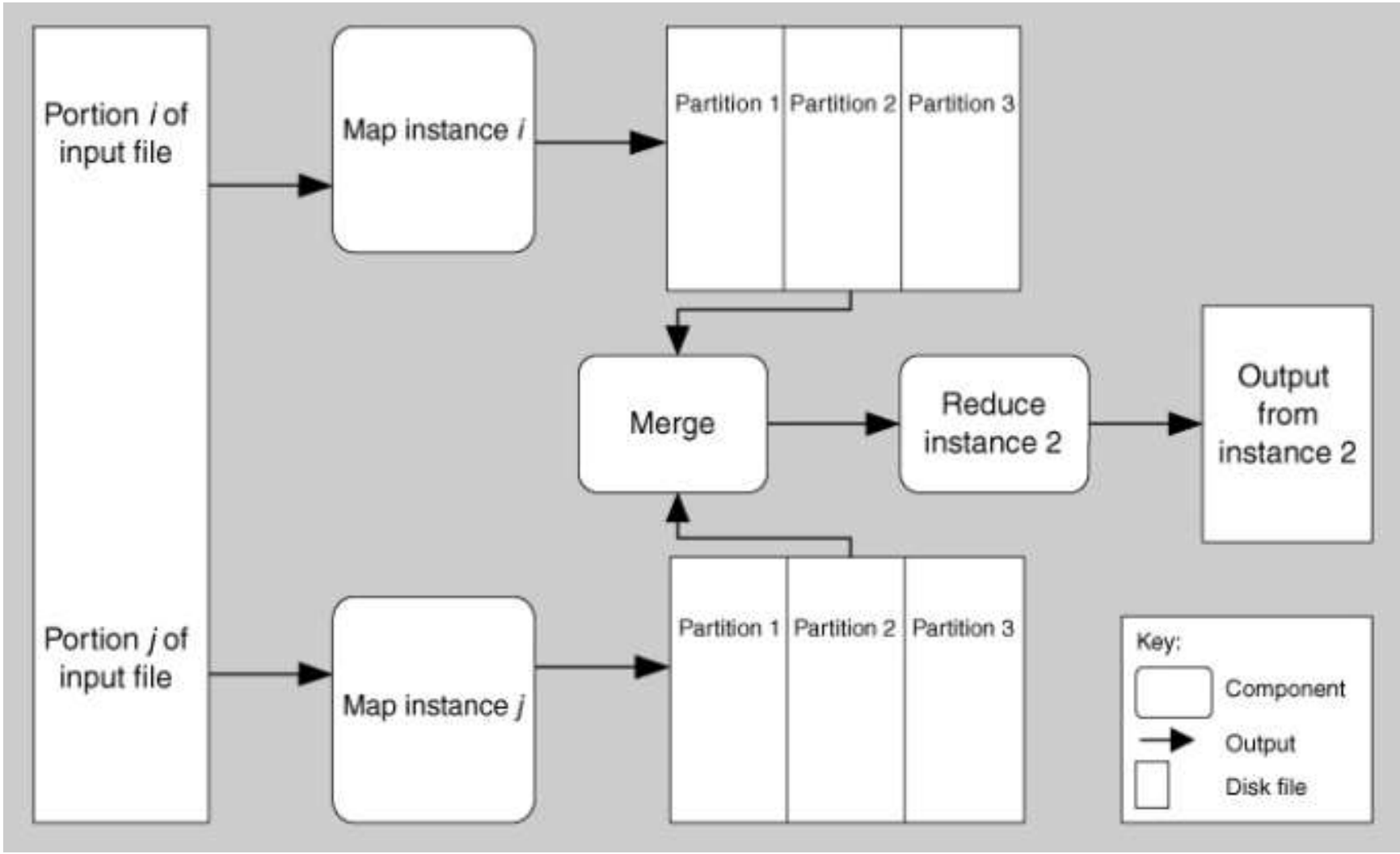
FUNCIONAMIENTO – FUNCION REDUCE




```
Reduce(k2,list (v2)) -> list(v3))
```

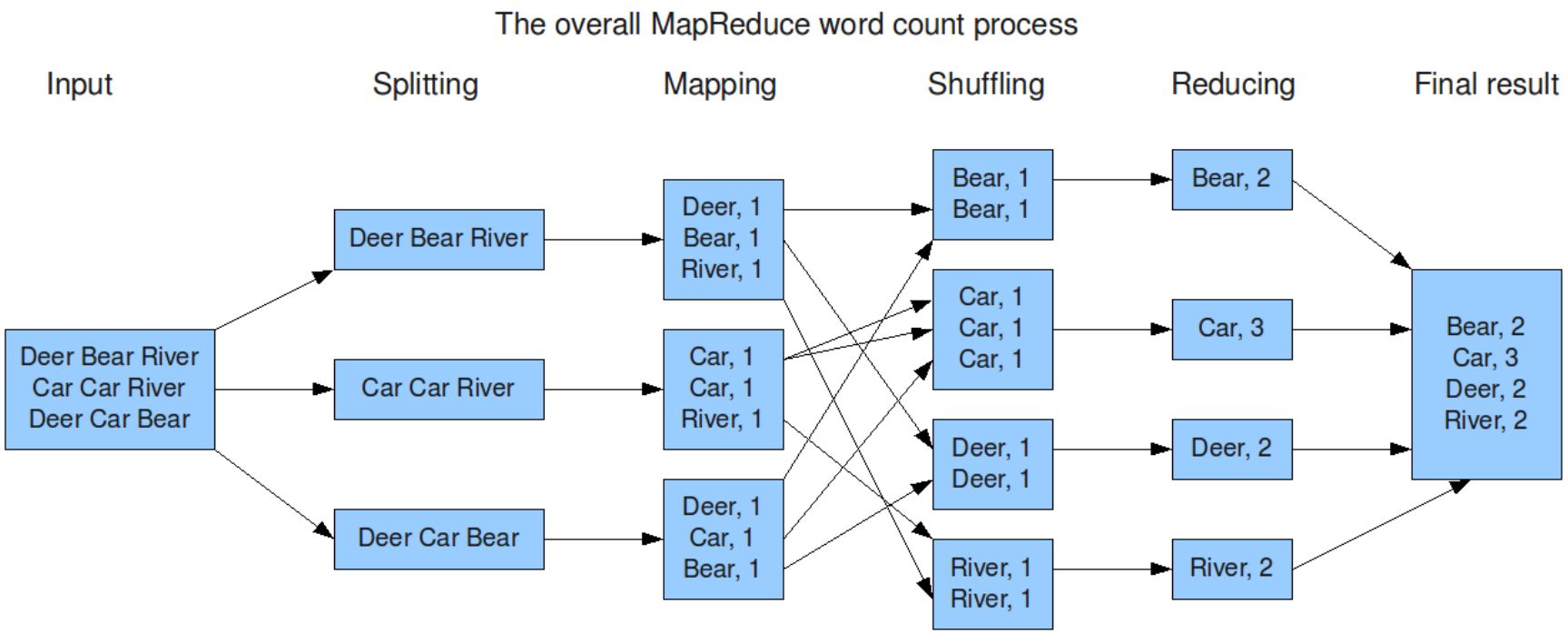
- es aplicada en paralelo a cada grupo, produciendo una colección de valores para cada dominio
- el conjunto de datos de salida es siempre mucho más chico que el de entrada

FUNCIONAMIENTO – EN UNA IMAGEN



MAP REDUCE – EJEMPLO

Objetivo: Contar las ocurrencias de cada palabra en un conjunto de documentos



MAP REDUCE – ORGANIZACIÓN

- existe un clúster de máquinas que oficiarán de “workers” dentro de la solución Map-Reduce
- generalmente la cantidad de instancias a utilizar es configurable
- uno de los “workers” toma el rol de master, encargándose de:
  - ✓ Recopilar los workers en reposo (sin tarea asignada)
  - ✓ Asignar tareas a los workers, ya sea de map() o de reduce()
- Cada worker puede tener 3 estados: reposo, trabajando, completo

MAP REDUCE – A TENER EN CUENTA

¿Cuándo NO es apropiado este patrón?

- si no existen grandes volúmenes de datos – ya que el overhead de Map-Reduce no se justifica
- si no se puede dividir el conjunto de datos inicial en subconjuntos de tamaño uniforme – en este caso, las ventajas del paralelismo se diluyen
- si se tienen operaciones que requieren múltiples “reduce” - aunque es posible, es más difícil de orquestar.

MAP REDUCE – IMPLEMENTACIONES

- Apache Hadoop, The Apache Software Foundation
- Couchdb, The Apache Software Foundation
- Infinispan, redhat
- MongoDB, MongoDB Inc.

CONTENIDO

1 ARQUITECTURA ORIENTADA A SERVICIOS (SOA)

CONCEPTO Y COMPONENTES

PROPIEDADES Y BENEFICIOS

CONEXIÓN E INTERMEDIARIOS

2 MAP-REDUCE

3 MULTI-TIER

MULTI-TIER – MOTIVACION

En un sistema distribuido, frecuentemente existe la necesidad de distribuir la infraestructura del mismo en subconjuntos distintos.

Esto puede ser por razones operativas o de negocio:

- Operativas – asegurar determinados atributos de calidad, e.g. confiabilidad, eficiencia
- Negocio – delegar la toma de decisiones, necesidad de procesamiento específico local

MULTI-TIER - CONCEPTO

- Las estructuras de ejecución de muchos sistemas están organizadas como agrupaciones lógicas de componentes - a cada agrupación se la denomina “tier”

- Cada capa puede estar basado en distintos criterios:
  - ✓ tipo de componente
  - ✓ compartir el mismo ambiente de ejecución
  - ✓ tener el mismo propósito en ejecución

- Las capas introducen restricciones topológicas que indican qué componentes pueden comunicarse con otros
  - ✓ conexión entre componentes dentro de un mismo tier
  - ✓ conexión entre componente de tiers adyacentes; e.g. call-return en una dirección, pero notificaciones basadas en eventos en la otra

MULTI-TIER – FORTALEZAS Y DEBILIDADES

FORTALEZAS

- Seguridad
- Disponibilidad
- Modificabilidad

COMPONENTES Y ALOCACION

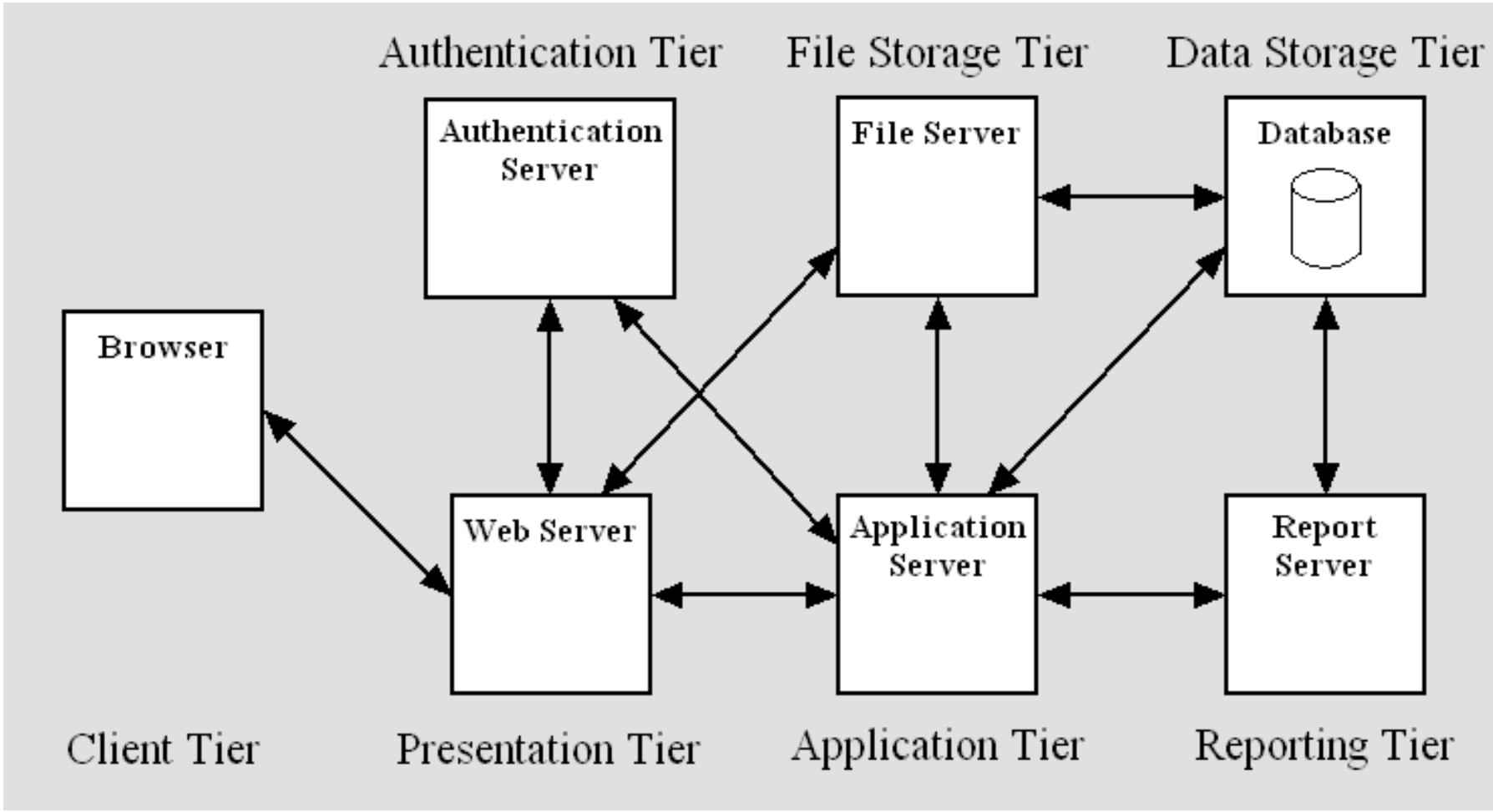
Relevancia del criterio para definir las capas:

- si las capas agrupan componentes de similar funcionalidad – componentes y conectores
- la capa cliente en un sistema empresarial no correrá en el mismo host que la base de datos – tema de alocación

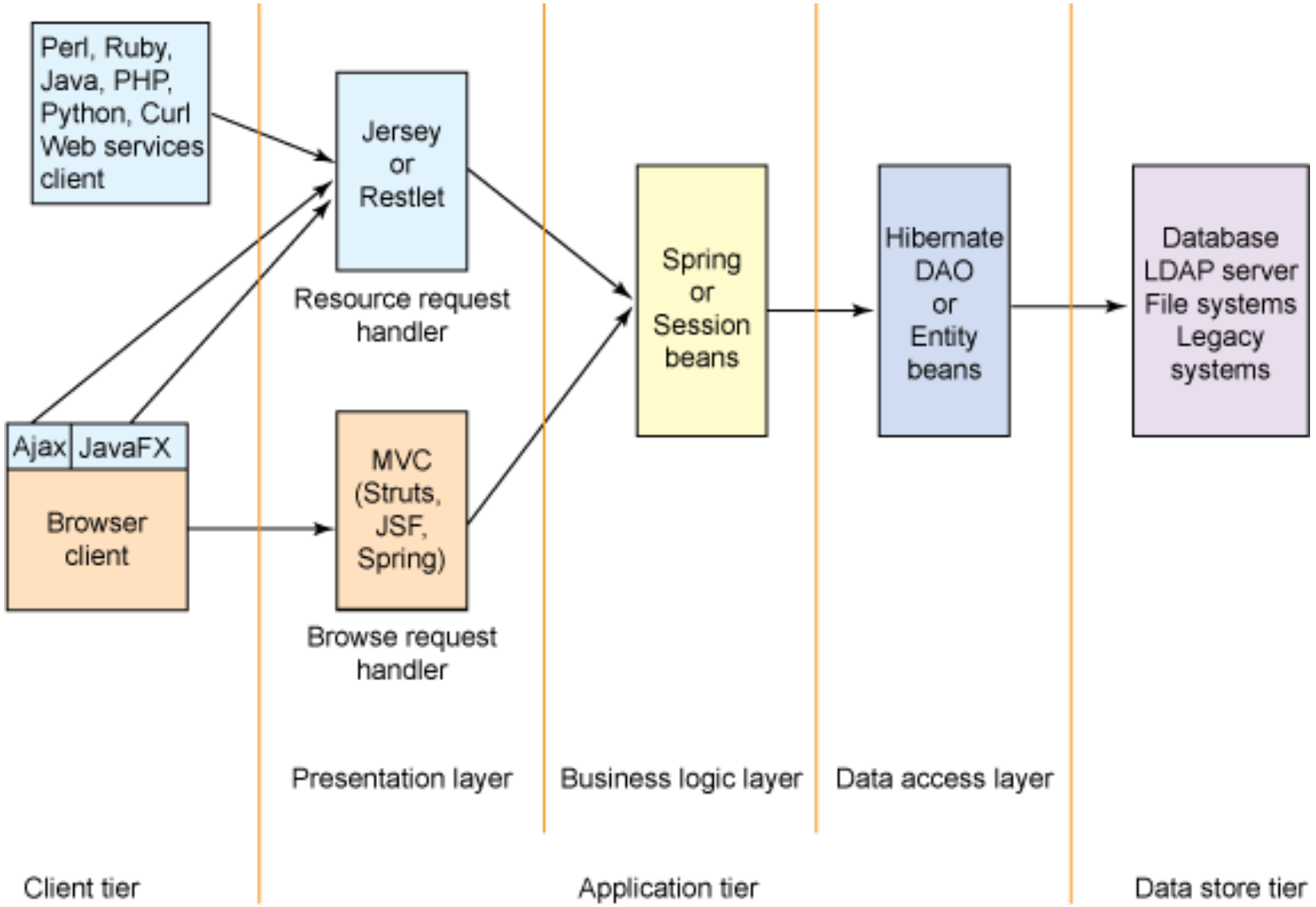
DEBILIDADES

- Costo y complejidad
  - ✓ Para sistemas simples, los beneficios pueden no justificar sus costos iniciales y de mantenimiento
  - ✓ Requerimientos de hardware, software
  - ✓ Complejidad de diseño e implementación

MULTI-TIER – EJEMPLO GENERICO



MULTI-TIER – EJEMPLO GENERICO EN APLICACIONES JEE



RESUMEN- PATRONES DE ARQUITECTURA

- ESTADO-LOGICA-PRESENTACIóN
- MODEL-VIEW-CONTROLLER
- SENSE-COMPUTE-CONTROL
- BROKER
- SOA
- MAP-REDUCE
- MULTI-TIER

Elsa Estevez
ece@cs.uns.edu.ar