Maestría en Ingeniería

Curso de Arquitectura de Software

Sesión 11: Tecnologías para SOA

Fernando Barraza A.
fbarraza@javerianacali.edu.co

Sesion 11

- Objetivo: Presentar al estudiante las tecnologías más representativas en la implementación de SOA
- Temas:
  - BPEL
  - REST
  - RESTful Web Services

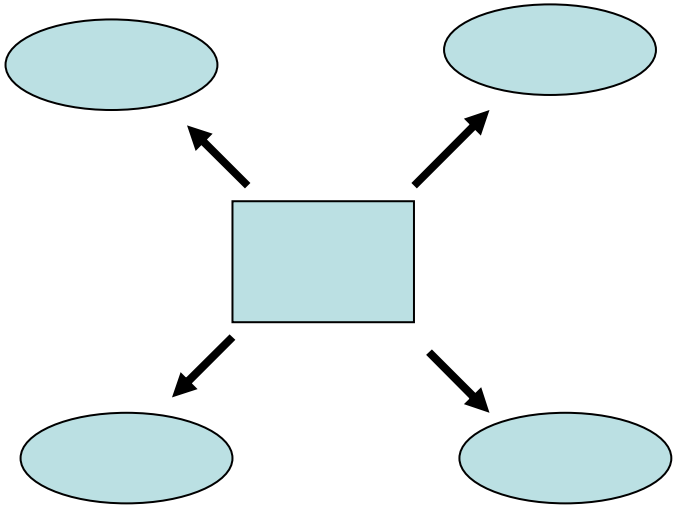
BPEL

- ***(Web Services) Business Process Execution Language***, WS-BPEL (Antes BPEL4WS)
- Desarrollado por OASIS
- Sirve para la composición de Servicios Web
- Tiene como predecesores a WSFL y XLANG

Que es BPEL?

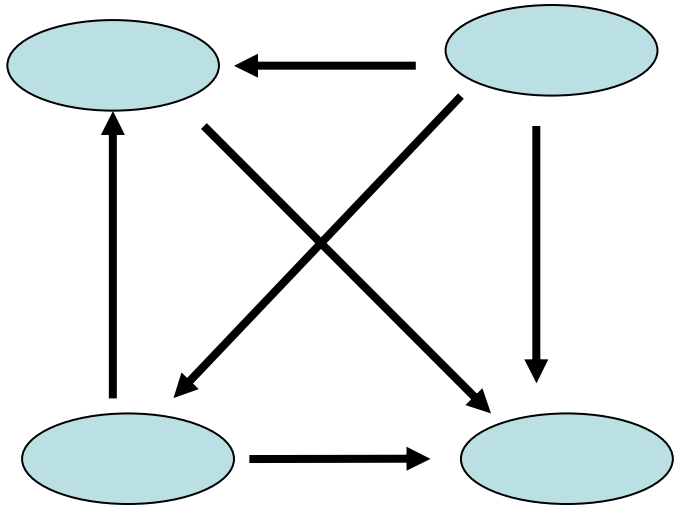
- Es un lenguaje basado en XML
- Diseñado para “orquestar” y “controlar” invocación de diferentes servicios Web, añadiendo lógica de negocio a un alto nivel
- No es un lenguaje para “Coreografía” de servicios Web (tal como lo intentaba ser WS-CDL)

Orquestación vs. Coreografía



ORQUESTACIÓN:

Tiene un punto unico de control.



COREOGRAFIA

Modelo que abarca todas las interacciones deseadas con un control ditribuido

Que provee BPEL

- Correlación de mensajes basado en propiedades.
- Variables de tipo XML y WSDL.
- Lenguaje extensible para expresiones y consultas con otros lenguajes como Xpath
- Estructuras de programación como if-then-elseif-else, while, sequence, flow (ejecución paralela)
- Contextos de programación con variables locales, excepciones a errores, eventos.

Beneficios al usar BPEL

- Estandarización al definir los procesos de negocio que interactúan con entidades externas mediante operaciones de un servicio Web definidas usando WSDL 1.1 y que se manifiestan a sí mismas como *servicios Web*.
- Reusar servicios Web como modelo para la descomposición y ensamblaje de procesos.
- Aplicaciones de alto nivel modulares y extensibles basadas en SOA.

REST

- Relational State Transfer Protocol
- Creado en el 2000
- Permite acceder a recursos a través de URL’s
- Mayormente implementado sobre HTTP
- Se diferencia de SOAP en que permite aprovechar aspectos como la autenticación, caching, negociación de contenidos entre otros.

Como funciona REST

- REST propone un vocabulario de acciones fijo que permite obtener recursos del servidor, que una vez interpretados por el cliente, hacen que este cambie de estado, haciendo uso de las funciones propias de HTTP (PU, GET, POST, DELETE, etc.)

Principios de REST

- Cliente y Servidor plenamente identificados y separados
- No hay almacenamiento de estado en las transiciones (Stateless)
- Respuestas “cacheables” (Para mejorar las interacciones y el performance)
- Capas: Pueden haber varias capas de servidores intermedios
- Código por demanda: La funcionalidad de un cliente se puede extender con código trasmitido por el servidor (por ejemplo con JavaScript). Este es el único requerimiento opcional.
- Interfaz uniforme (del protocolo)

Objetivos REST

- Escalabilidad de componentes
- Generalidad de Interfaces
- Distribución Independiente de componentes
- Existencia de componentes intermediaros que reducen latencia, aumentan la seguridad y encapsulan sistemas de legado.

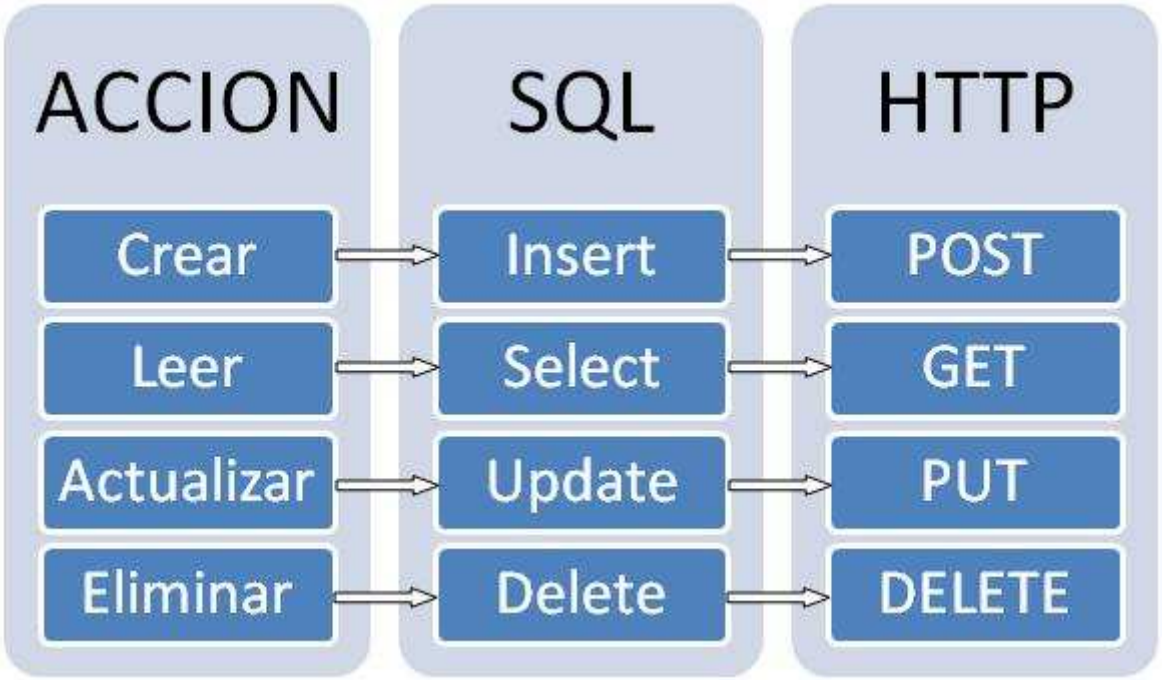
Recursos en REST

- Se definen como fuentes de información específica y son muy importantes en REST.
- Ejemplo: Una aplicación empresarial con información de recursos humanos:
  - La lista de empleados es un recurso del servidor y para acceder a ellos, se requiere un identificador único (por ejemplo una URI), que es usado por el cliente para referenciarlo y una interfaz de comunicación estandarizada (por ejemplo HTTP).
  - Se debe tener clara la acción a ejecutar sobre ese objeto.
- Una aplicación interactúa con los recursos sabiendo su identificador único y la acción a ejecutar sobre este. (Ej: Recurso: Empleado con Id:3520. Acción: Borrar.)
- Los recursos generalmente son transmitidos entre el servidor y el cliente usando formatos como HTML, XML o JSON.
- Los recursos también pueden ser imágenes, texto plano o cualquier otro formato.

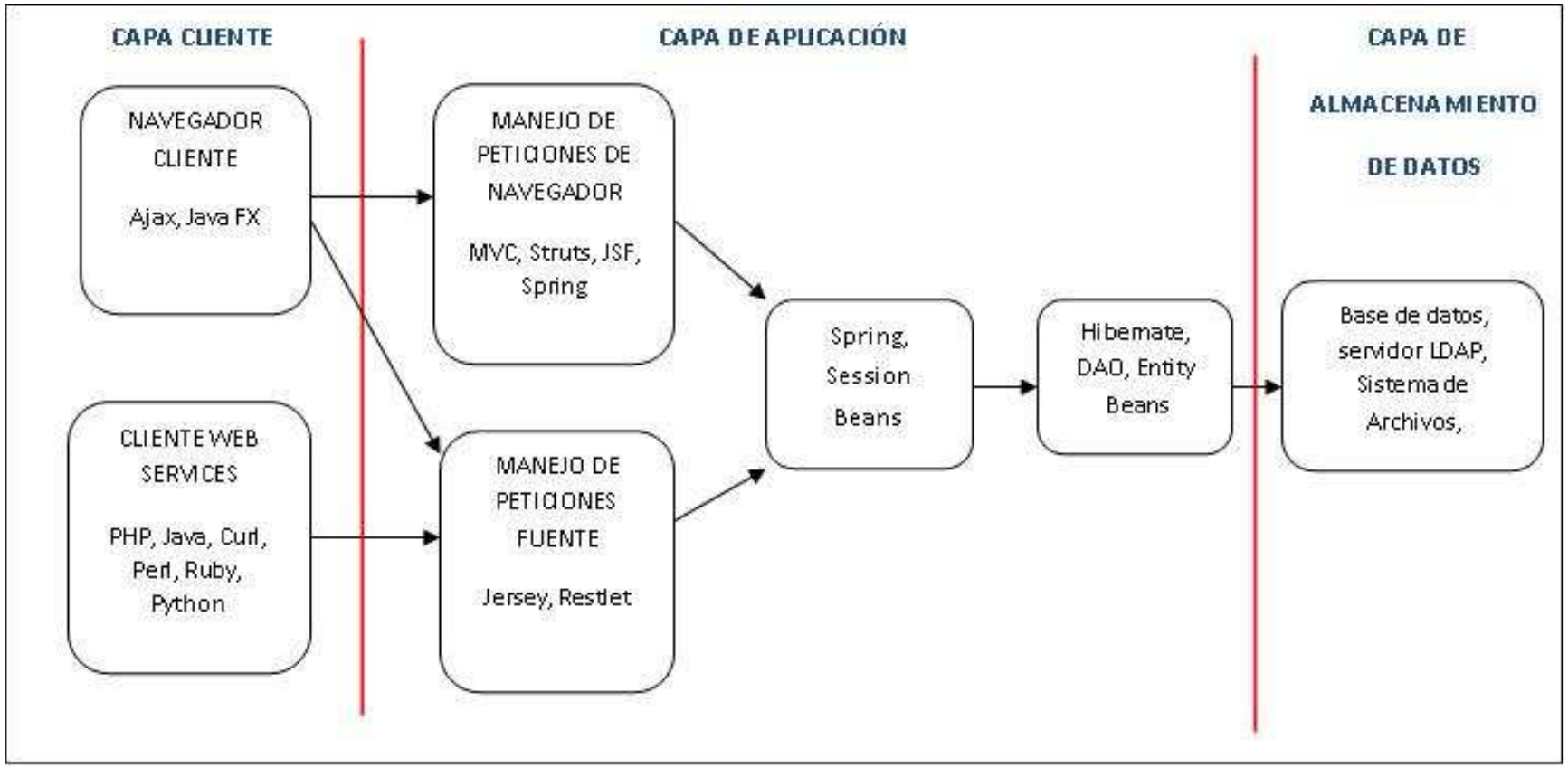
RESTful Web Services

- Son servicios WEB implementados usando HTTP y los principios de REST, con la siguiente información:
  - Una URI base para el nombre del servicio: http://example.com/miservicio/
  - El tipo MIME de los datos soportados por el servicio
- En REST sobre HTTP, lo más normal es tener estas operaciones
    - POST: Crea nuevos recursos. Como retorno, se ofrece el ID automáticamente creado
    - GET: Lista un recurso
    - PUT: Reemplaza un recurso con otro (Útil para el update)
    - DELETE: Elimina un recurso
- EJEMPLO: Si se quiere eliminar el empleado con ID 3520 se haría un request con DELETE a la siguiente URL desde el cliente: http://example.com/miservicio/empleados/3520

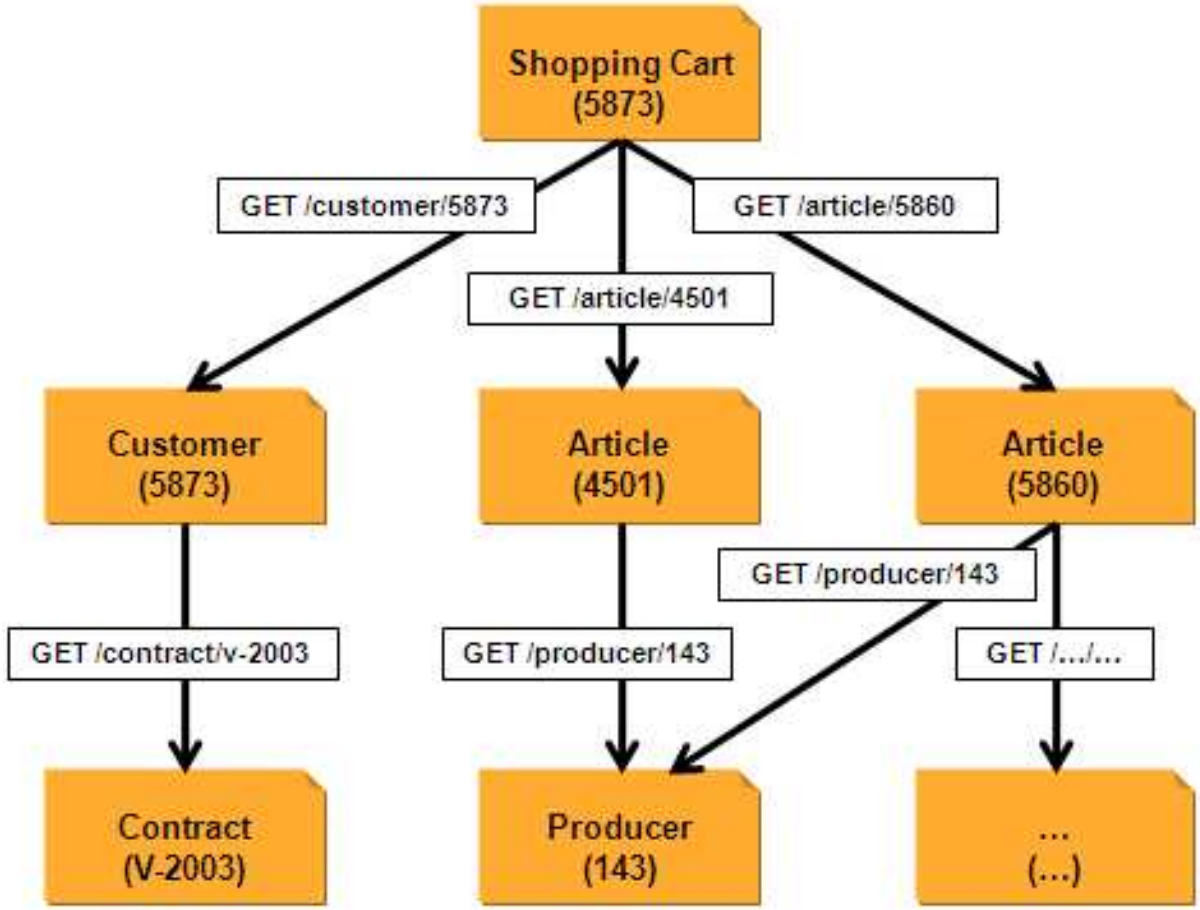
Relación entre acciones



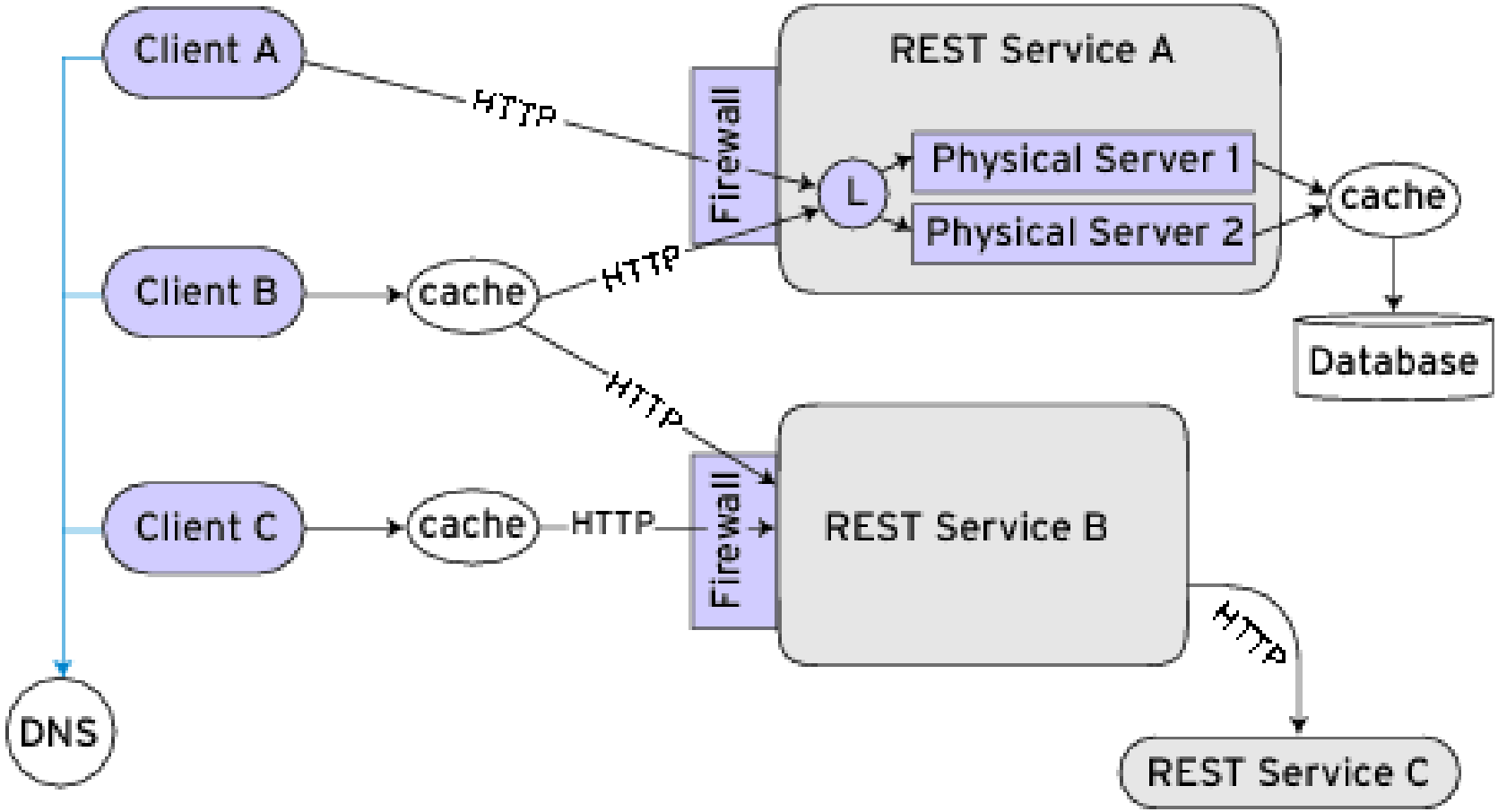
Estructura de una app web con RESTful web service



Ejemplo


Shopping Cart
(5873)
GET /customer/5873
GET /article/4501
GET /article/5860
Customer
(5873)
Article
(4501)
Article
(5860)
GET /producer/143
GET /contract/v-2003
GET /producer/143
GET /.../...
Contract
(V-2003)
Producer
(143)
...
(...)

SOA y RESTful Web Services



Créditos

- Essential Software Architecture. Ian Gorton
- MDA, Model Driven Architecture. Lea Kutnoven.
- OMG’s Model Driven Architecture. Davide Buscaldi.