DECSAI
Departamento de Ciencias de la Computación e I.A.
Universidad de Granada

Middleware y sistemas distribuidos

© Fernando Berzal, berzal@acm.org

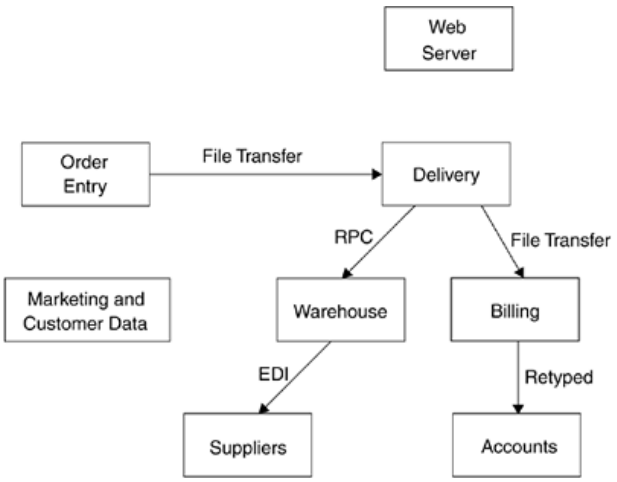
Middleware

- Motivación
- Componentes asociados al middleware
- Arquitectura IT
- Estándares
  - Sockets TCP/IP
  - RPC [Remote Procedure Call], e.g. RMI
  - Colas de mensajes
  - Middleware orientados a objetos, e.g. CORBA & DCOM
  - Middleware basado en componentes, e.g. EJB & .NET
  - Servicios, e.g. servicios web
- Tendencias: "Comoditización" & microservicios
- Apéndice: ZooKeeper

Middleware

Motivación

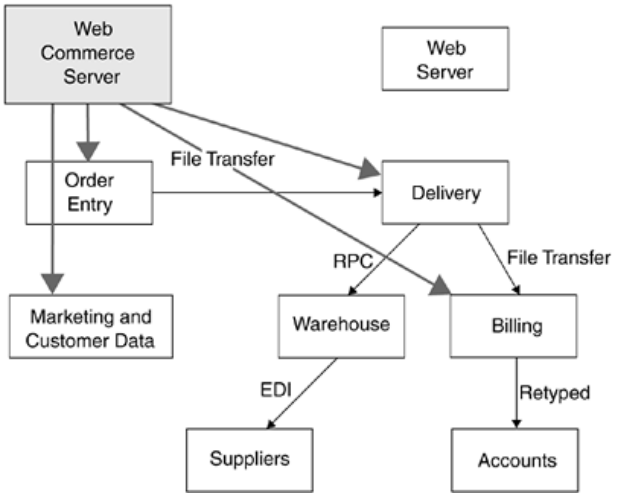
Empresa que vende productos…



Middleware

Motivación

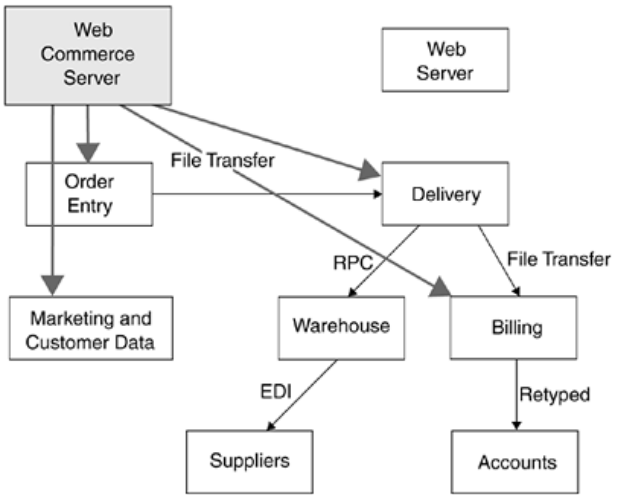
… decide crear su portal de comercio electrónico:



Middleware

Motivación

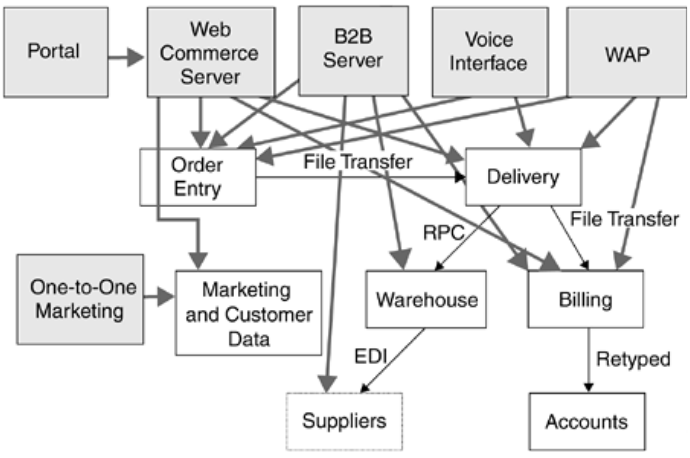
… decide crear su portal de comercio electrónico:



Middleware

Motivación

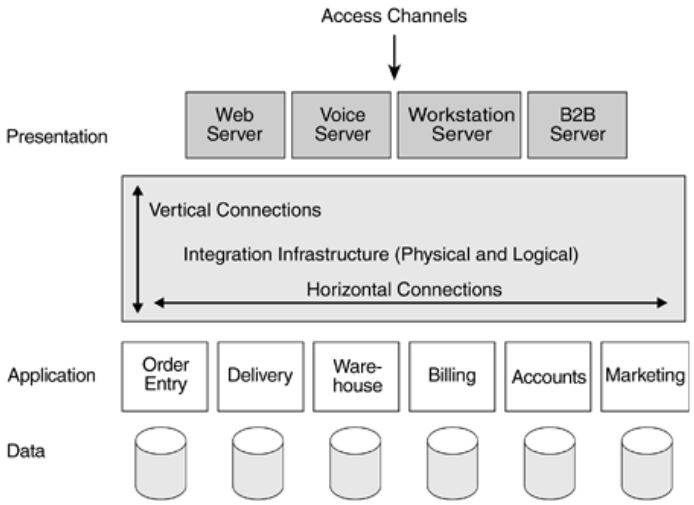
Con el tiempo, su sistema va siendo más complejo:



Middleware

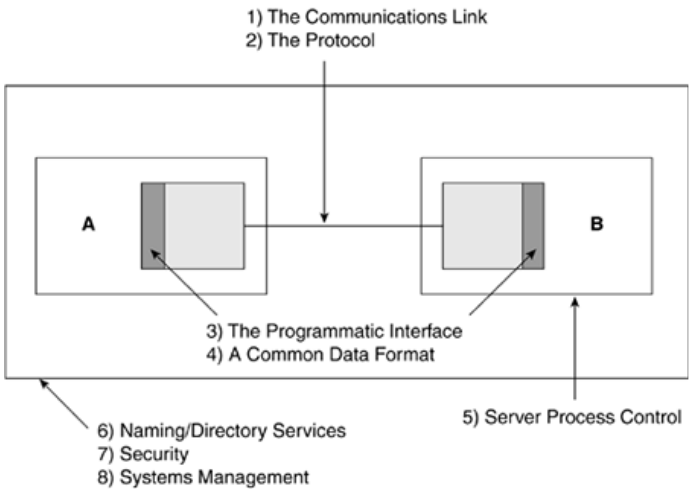
Motivación

Arquitectura IT



Middleware

Componentes asociados al middleware



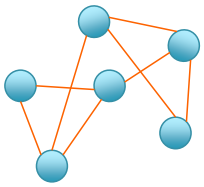
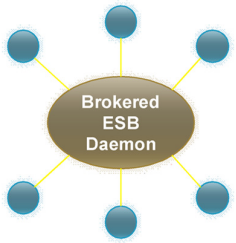
Middleware

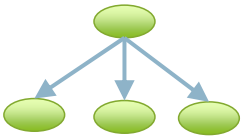
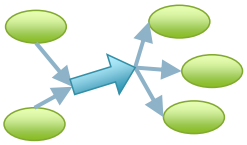
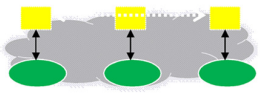
Protocolos

Point-to-Point | Client/Server | Publish/Subscribe | Queuing | Data-Centric





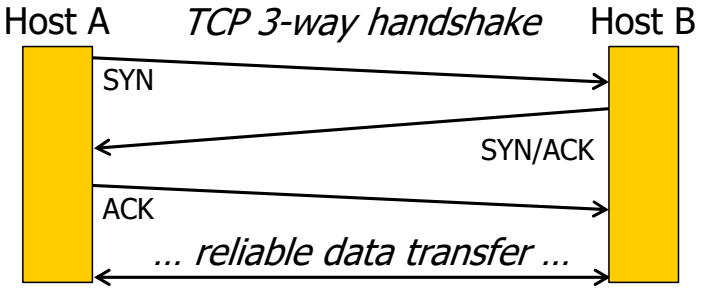
Data-Centric
Publish/Subscribe (DCPS)
DataBus

rti Your systems. Working as one.

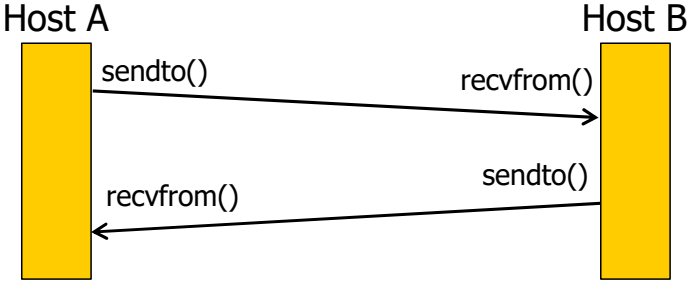
Middleware

Protocolos orientados a conexión vs. Protocolos no orientados a conexión

TCP



UDP



Protocolos: Codificación de los datos

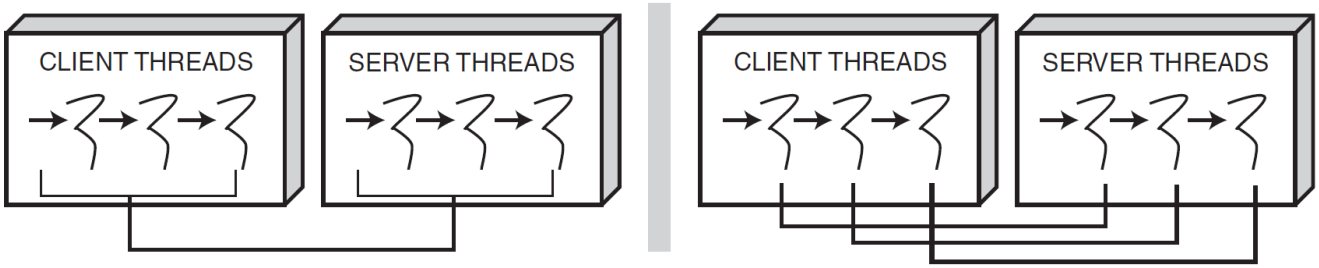
Textual
p.ej. HTTP

```
GET /cs282/ka.jpg HTTP/1.0
Connection: Keep-Alive
User-Agent: Mozilla/5.0 [en] …
Host: xar.dre.vanderbilt.edu
Accept: image/gif, image/jpeg, */*
Accept-Language: en Accept-Charset:
iso-8859-1,*,utf-8
...
```

Binaria
p.ej. GIOP

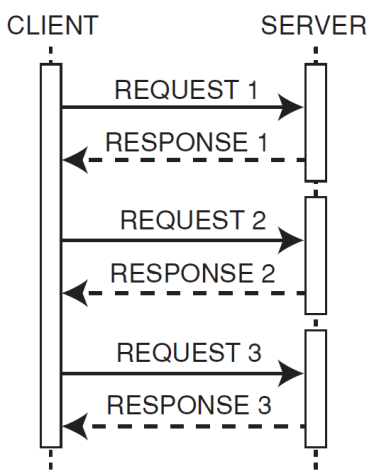
```
module GIOP {
  enum MsgType { Request, Reply, ... };
  struct MessageHeader {
    char magic[4];
    Version GIOP_version;
    octet byte_order, message_type;
    unsigned long message_size;
  };
  ...
```

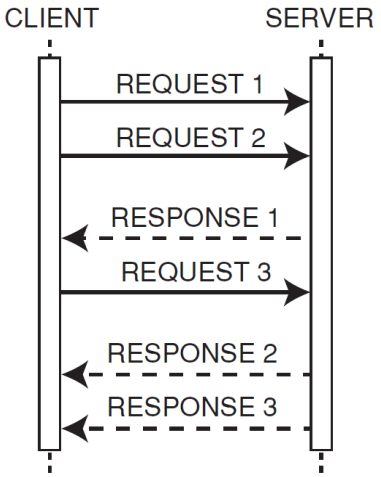

Protocolos: Multiplexación de conexiones



Middleware

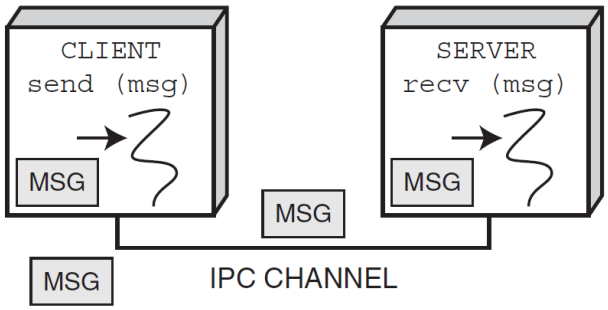
Protocolos:
Intercambio de mensajes síncrono/asíncrono

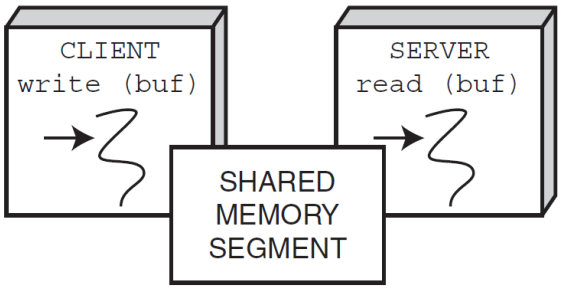




Middleware

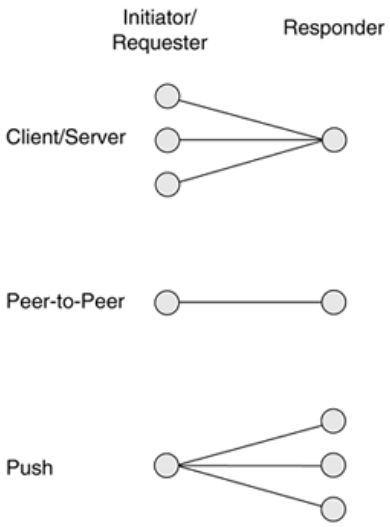
Protocolos:
Paso de mensajes vs. Memoria compartida





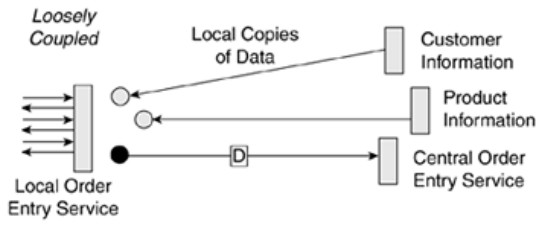
Middleware

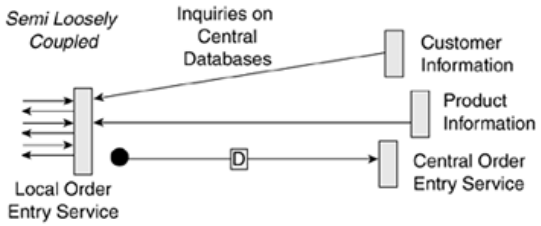
Protocolos

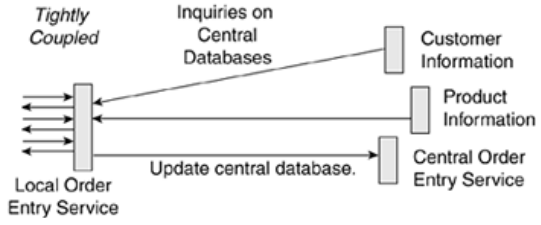


Middleware

Grados de acoplamiento

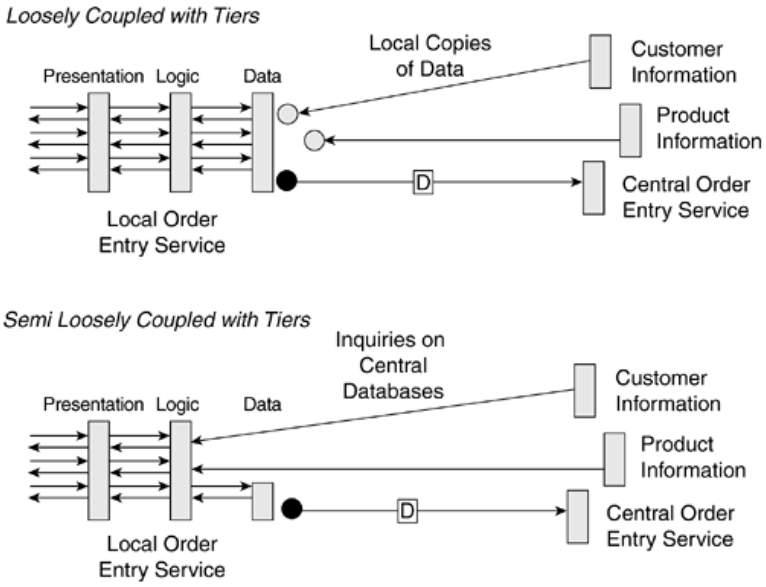




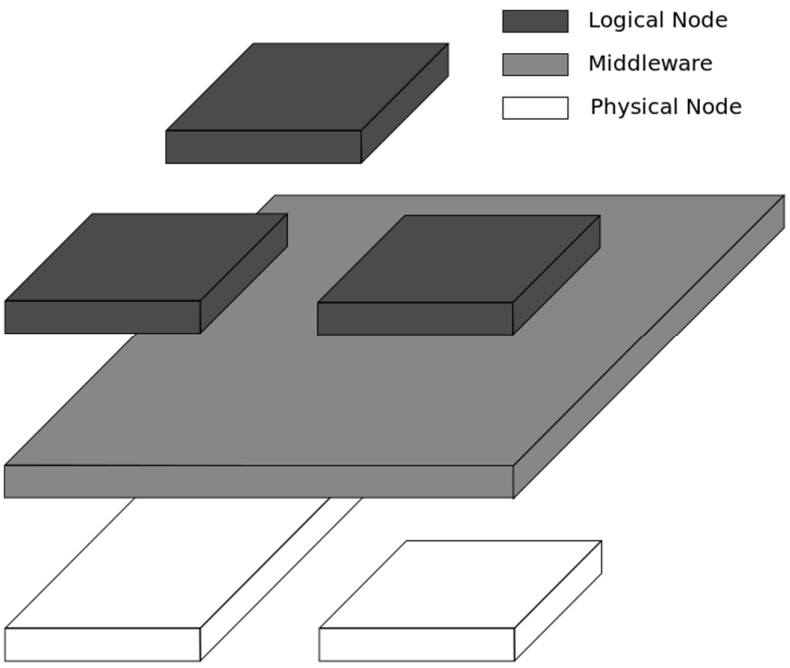


Middleware

Grados de acoplamiento en arquitecturas multicapa

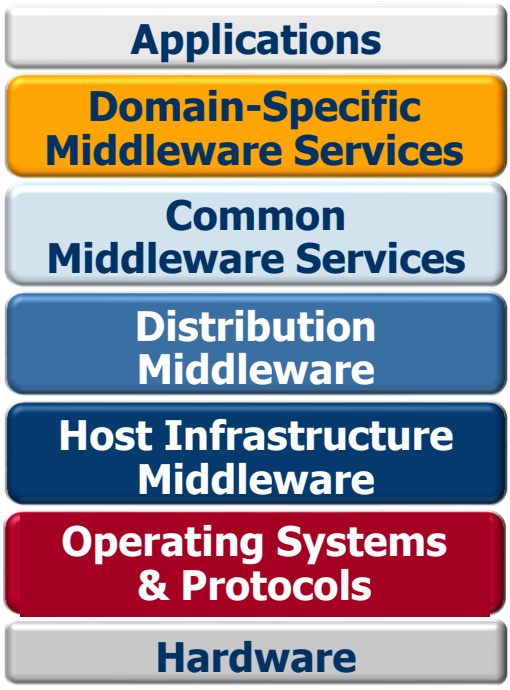


Middleware



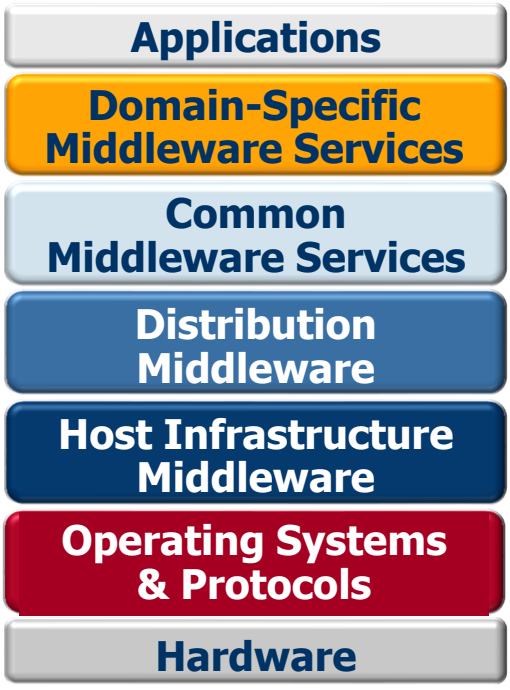
Middleware

Arquitectura basada en capas



Middleware

Arquitectura basada en capas



e-business, avionics, health care…

Transacciones, seguridad, recursos…

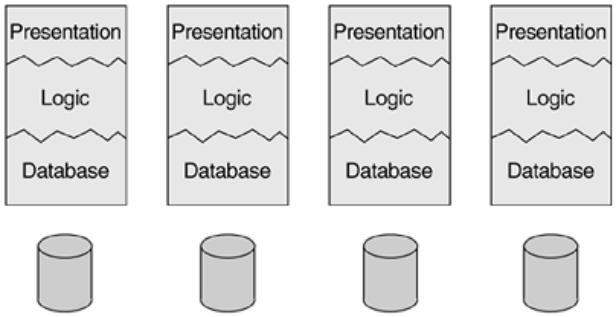
OMG CORBA & DDS, W3C Web Services, Java RMI, .NET Remoting…

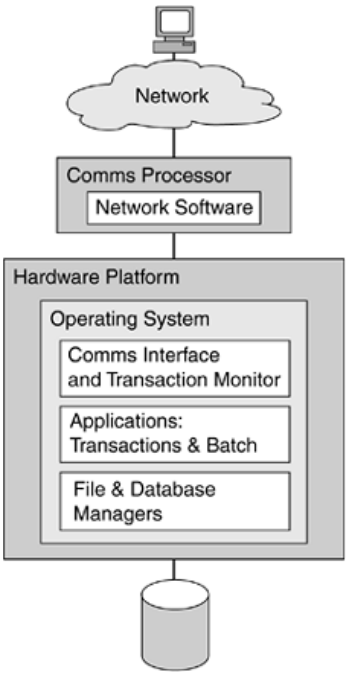
JVM (Java), CLR (.NET), ACE…

UNIX/Windows, TCP/IP…

Middleware

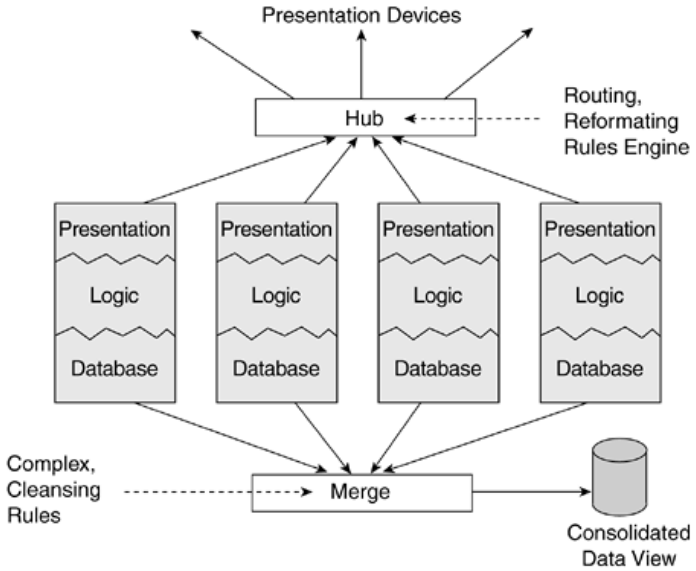
Silos…





Middleware

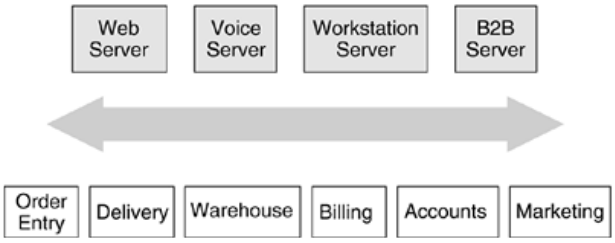
Arquitectura "surround"

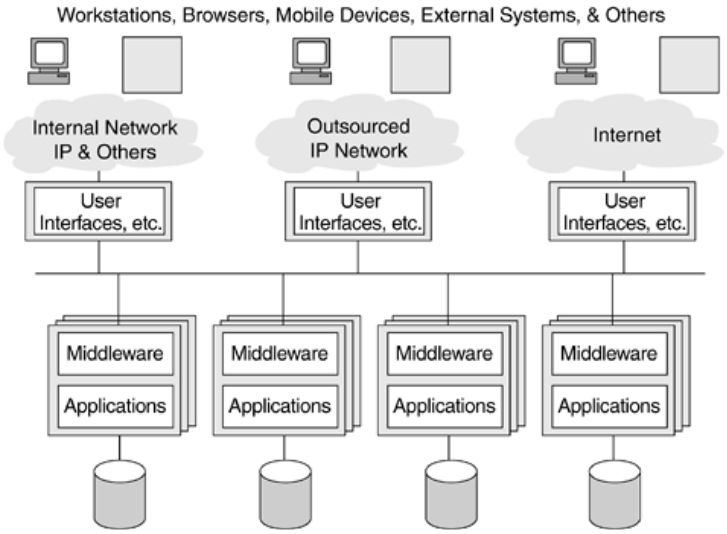


Middleware

Arquitectura en bus

e.g. Enterprise Service Bus

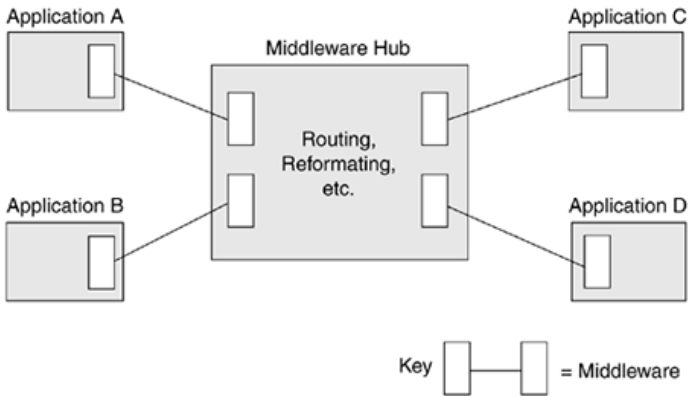




Middleware

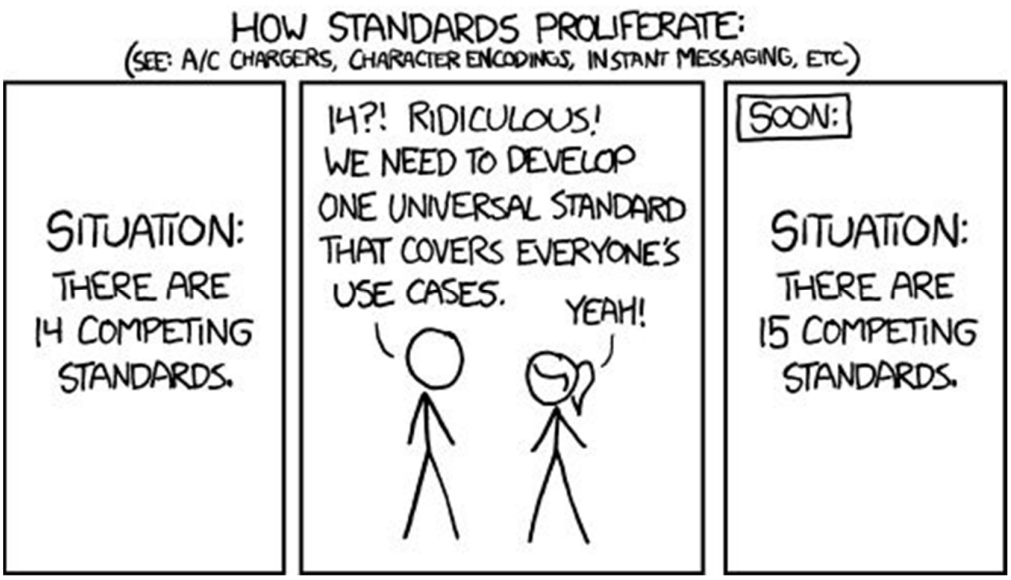
Interoperabilidad

Enterprise Application Integration



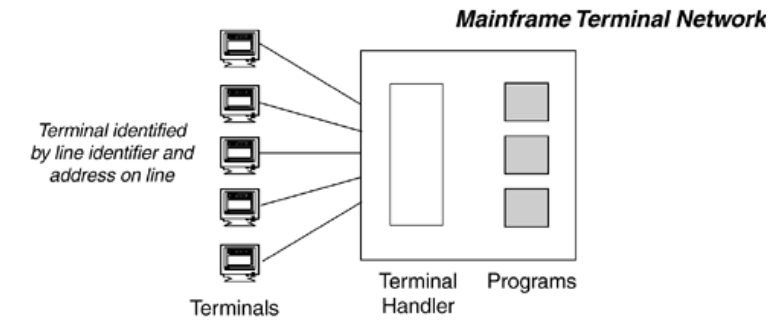
Middleware

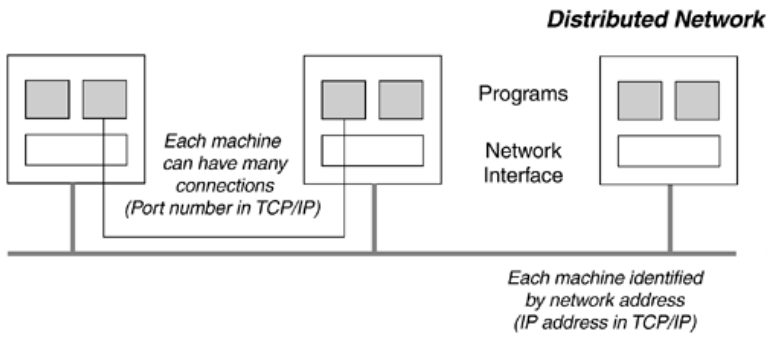
La estandarización del middleware



Middleware

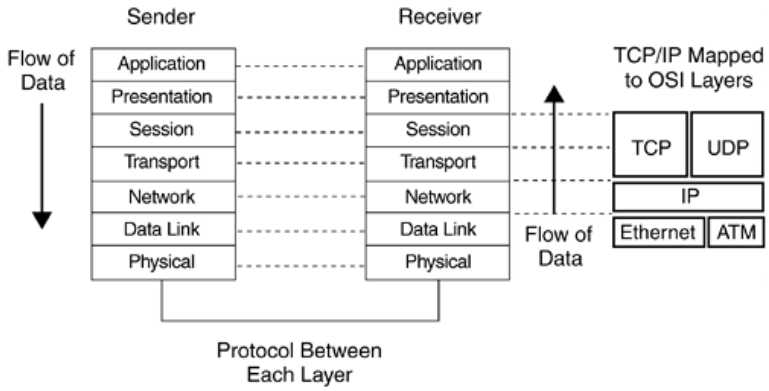
La estandarización del middleware





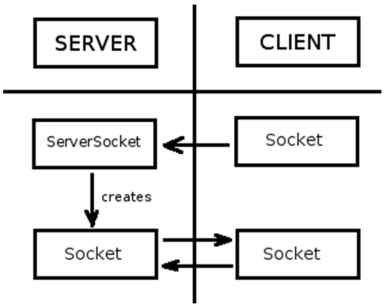
La estandarización del middleware

Sockets TCP/IP



La estandarización del middleware

Sockets TCP/IP en Java



Servidor

```
serverSocket = new ServerSocket(4321);
clientSocket = serverSocket.accept();
output = new PrintWriter(clientSocket.getOutputStream());
input  = new BufferedReader(new
       InputStreamReader(clientSocket.getInputStream()));
```

Cliente

```
socket = new Socket("elvex.ugr.es", 4321);
output = new PrintWriter(socket.getOutputStream());
input  = new BufferedReader(new
       InputStreamReader(socket.getInputStream()));
```

Middleware

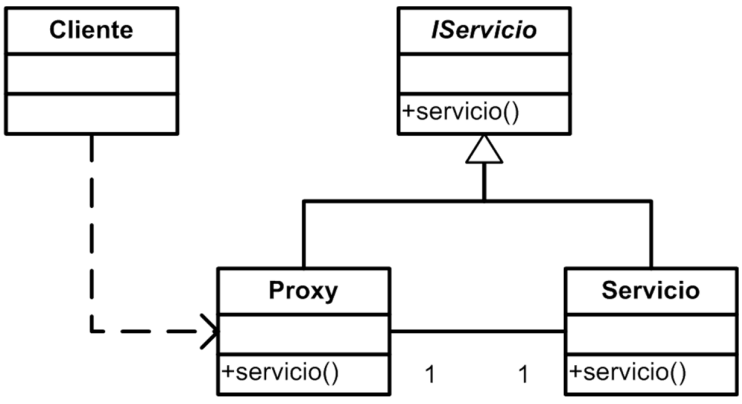
La estandarización del middleware

Características deseables del middleware:

- Facilidad de uso (vs. API a bajo nivel, e.g. sockets),
- Transparencia en cuanto a la localización (trasladar aplicaciones de una máquina a otra sin necesidad de recompilar el código).
- Integridad en la entrega de mensajes. (los mensajes no deberían perderse ni duplicarse).
- Integridad en el contenido de los mensajes (el contenido de los mensajes no debe corromperse).
- Independiencia del lenguaje (sistemas heterogéneos).

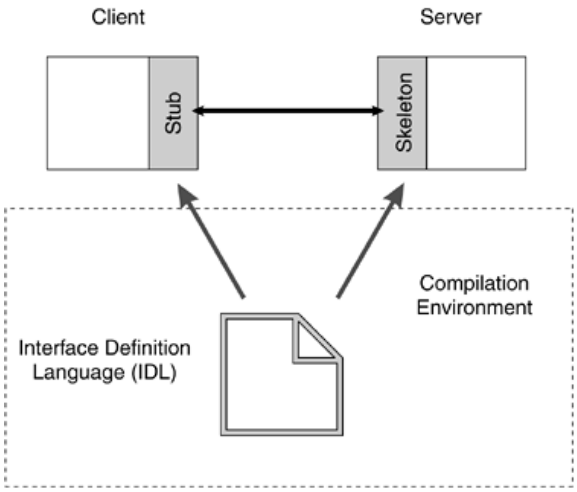
RPC [Remote Procedure Call]

RPC permite realizar la comunicación entre procesos como si se tratase de simples llamadas a funciones.



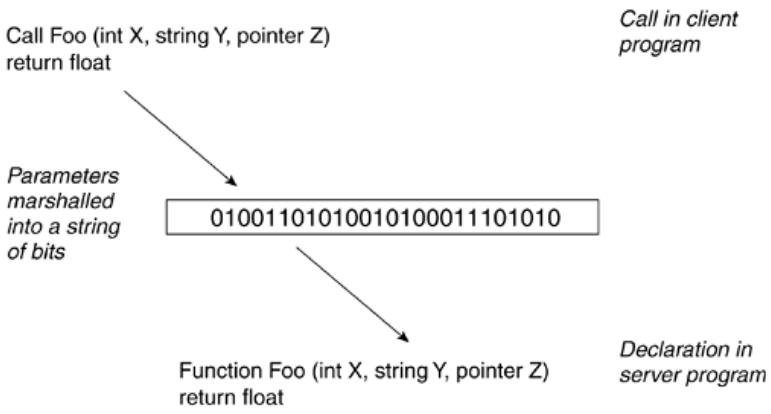
RPC [Remote Procedure Call]

Stubs & Skeletons



RPC [Remote Procedure Call]

Marshalling



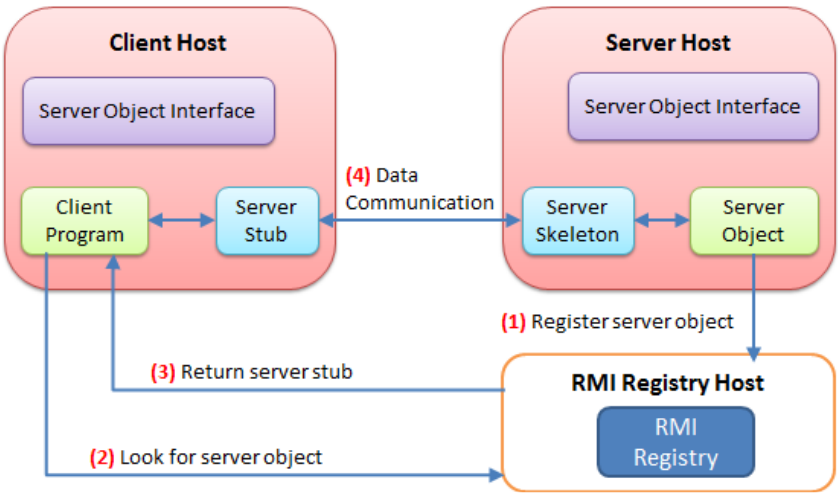
RPC [Remote Procedure Call]

Estándares

- **Java RMI** [Remote Method Invocation]
- **.NET Remoting** (en la plataforma .NET)
- **Windows RPC**, cumpliendo con el estándar OSF DCE [Open Software Foundation Distributed Computing Environment]

RPC [Remote Procedure Call]

Java RMI



RPC [Remote Procedure Call]

Java RMI

1. Interfaz remota

```
import java.rmi.Remote;
import java.rmi.RemoteException;

public interface Compute extends Remote
{
  Object executeTask(Task t)
          throws RemoteException;
}
```


RPC [Remote Procedure Call]

Java RMI

2. Objetos serializables

```
import java.io.Serializable;

public interface Task extends Serializable
{
    Object execute();
}
```

RPC [Remote Procedure Call]

Java RMI

3. Servidor RMI (1/2)

```
import java.rmi.*;
import java.rmi.server.*;

public class ComputeEngine
       extends UnicastRemoteObject  // Objeto remoto
       implements Compute           // Interfaz remota
{
  public ComputeEngine() throws RemoteException
  { super(); }

  public Object executeTask (Task t)
  { return t.execute(); }
…
```


RPC [Remote Procedure Call]

Java RMI

3. Servidor RMI (2/2)

```
…
  public static void main(String[] args)
         throws Exception
  {
    System.setSecurityManager (new SecurityManager());

    String name = "//elvex.ugr.es/Compute";
    Compute engine = new ComputeEngine();

    Naming.rebind(name, engine);
  }
}
```

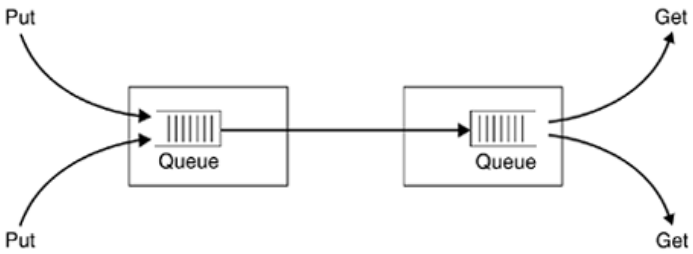
RPC [Remote Procedure Call]

Java RMI

4. Cliente RMI

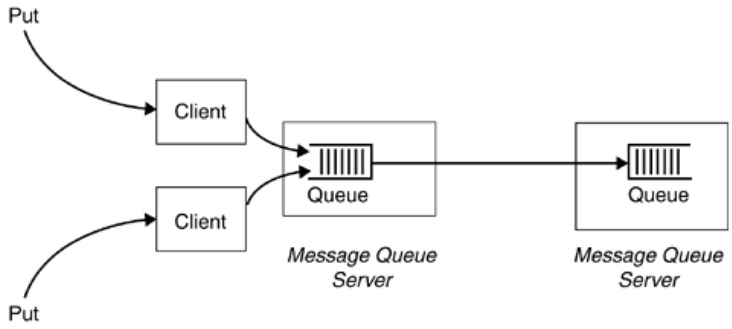
```
import java.rmi.*;

public class RemoteClient {
    …
    System.setSecurityManager (new SecurityManager());
    String name = "//elvex.ugr.es/Compute";
    Compute comp = (Compute) Naming.lookup(name);

    tarea = new XTask();
    resultado = (XResult) comp.executeTask(task);
    …
}
```


Colas de mensajes



Colas de mensajes

Client/Server Message Queueing

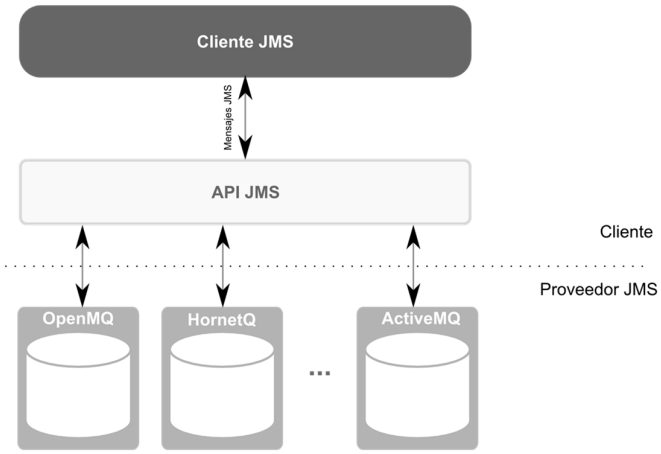


Colas de mensajes

JMS

Java Message Service

https://en.wikipedia.org/wiki/Java_Message_Service



Implementaciones:

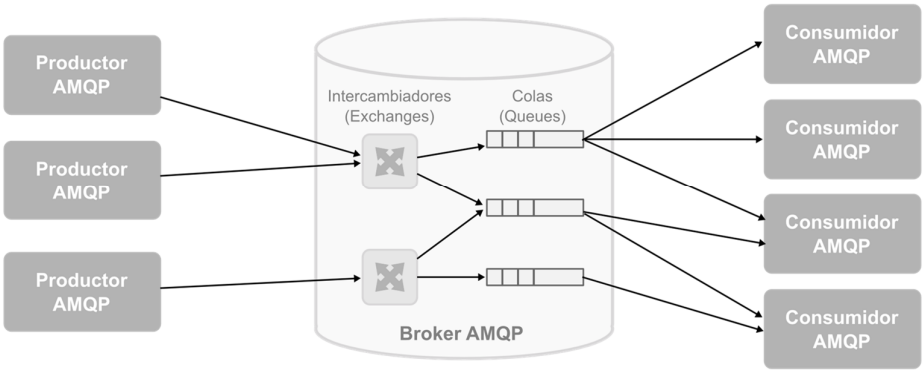
IBM WebSphere, Oracle Weblogic, Apache ActiveMQ…

Colas de mensajes

AMQP

Advanced Message Queuing Protocol

https://en.wikipedia.org/wiki/Advanced_Message_Queuing_Protocol



Implementaciones:

RabbitMQ, Apache ActiveMQ, Apache Qpid…

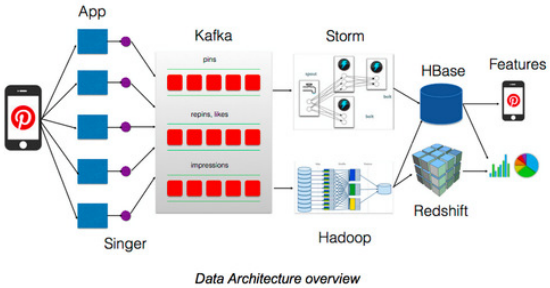
Colas de mensajes

Apache Kafka

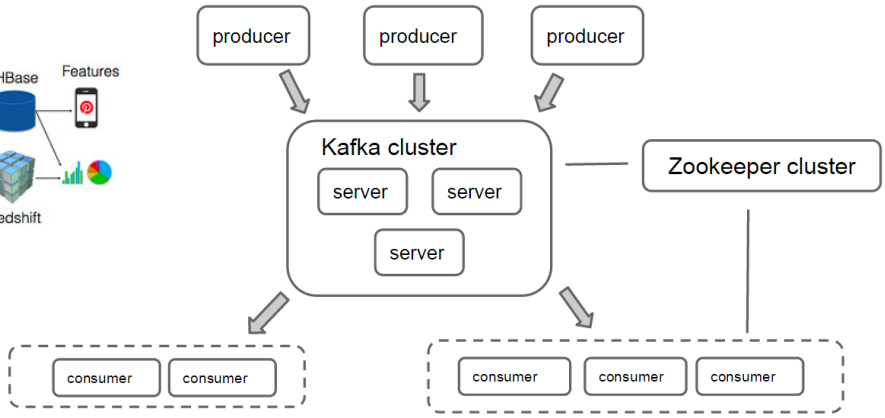
http://kafka.apache.org/

Apache Kafka

A high-throughput distributed messaging system.



Data Architecture overview



Desarrollado originalmente en LinkedIn.

Usado en Cisco, Netflix, PayPal, Spotify, Uber...

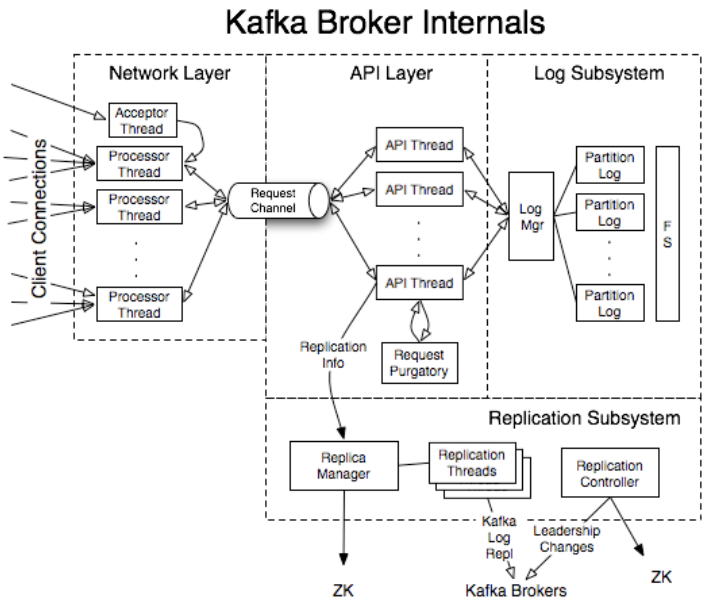
Colas de mensajes

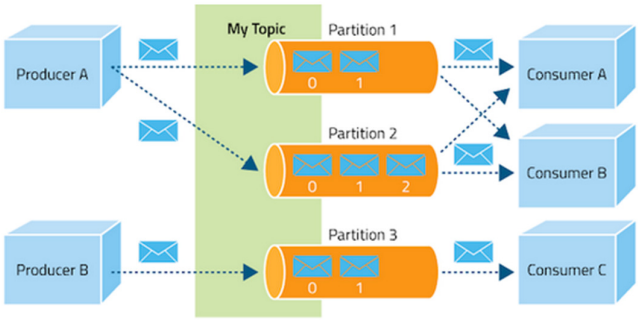
Apache Kafka

http://kafka.apache.org/

Apache Kafka

A high-throughput distributed messaging system.





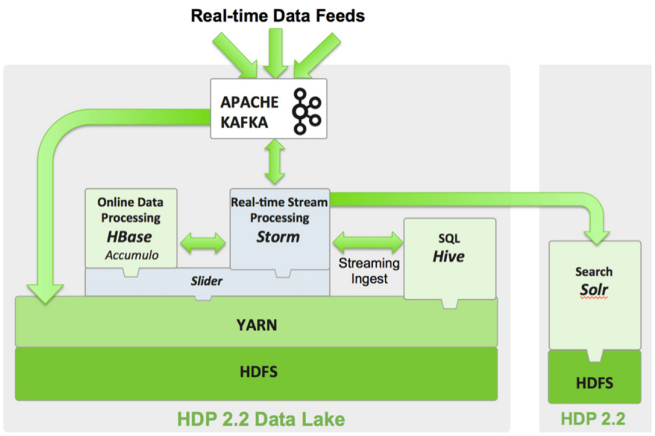
Colas de mensajes

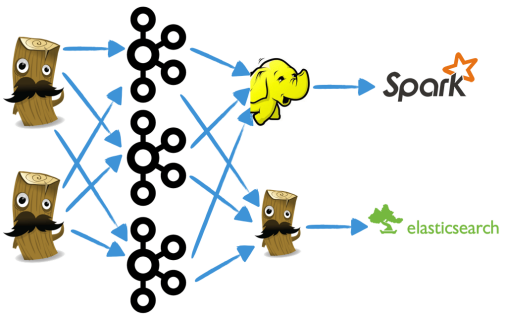
Apache Kafka

http://kafka.apache.org/

Apache Kafka

A high-throughput distributed messaging system.





Usado con Apache Storm [stream processing]: Yahoo!, Twitter…

Usado con Logstash, p.ej. elasticsearch: Wikimedia, GitHub, FDA, CERN…

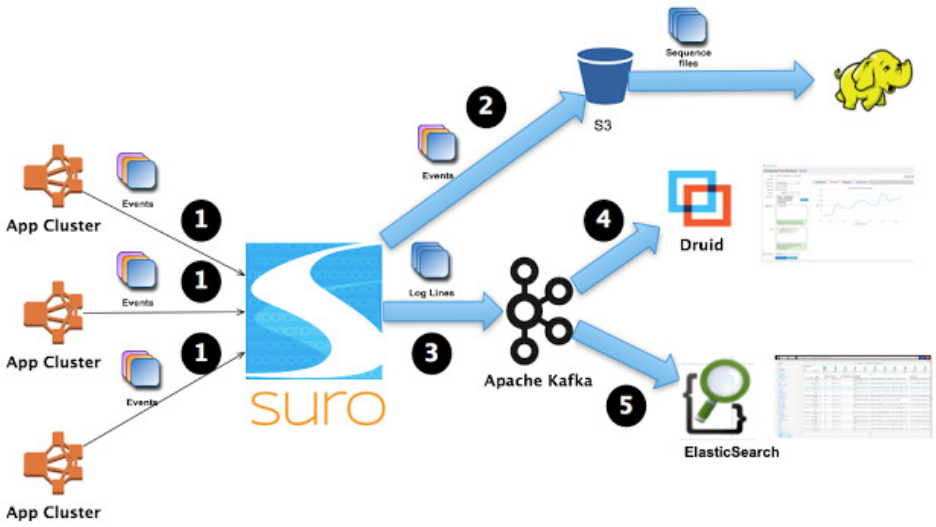
Colas de mensajes

Apache Kafka

http://kafka.apache.org/

Apache Kafka

A high-throughput distributed messaging system.



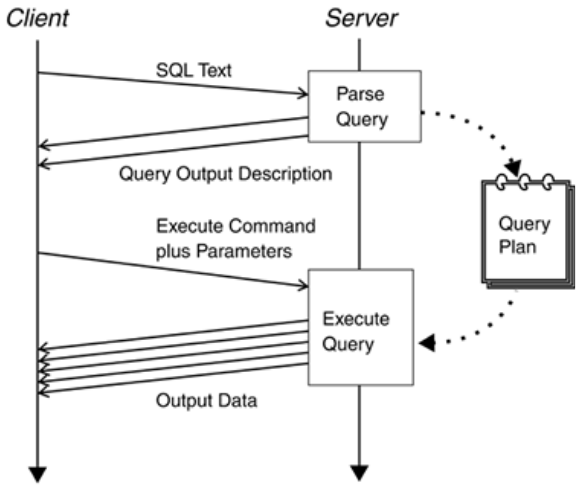
Suro = Netflix's data pipeline

https://github.com/netflix/suro

Colas de mensajes

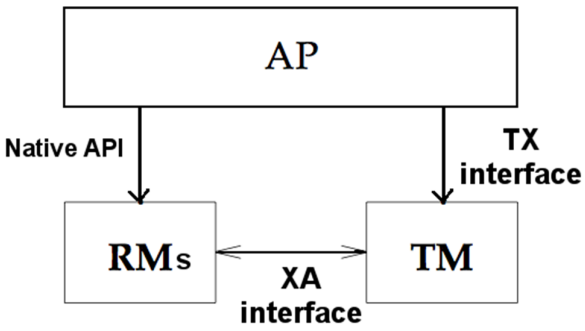
¿Por qué no usar bases de datos?

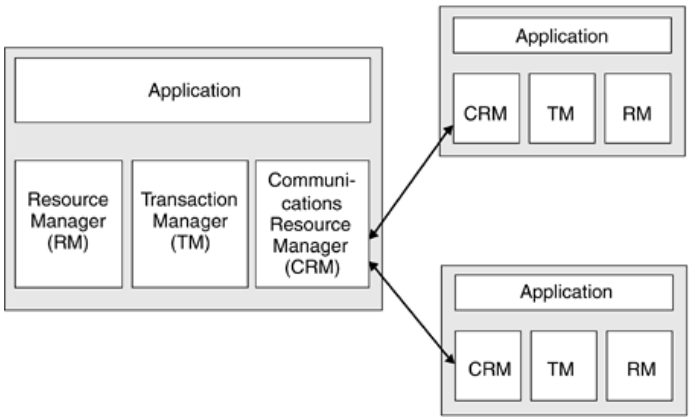
Limitaciones del uso de bases de datos como middleware



Middleware

Open Group DTP (antiguamente, X/Open)
DTP = Distributed Transaction Processing

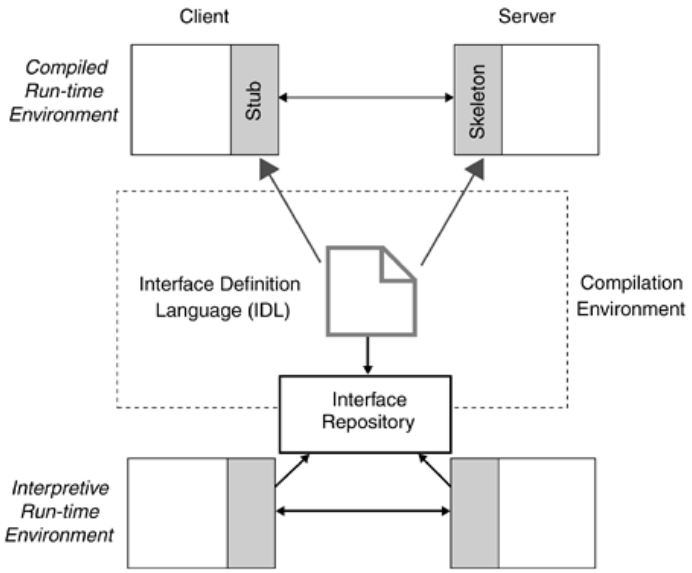




THE Open GROUP
Making standards work®

Middleware orientado a objetos

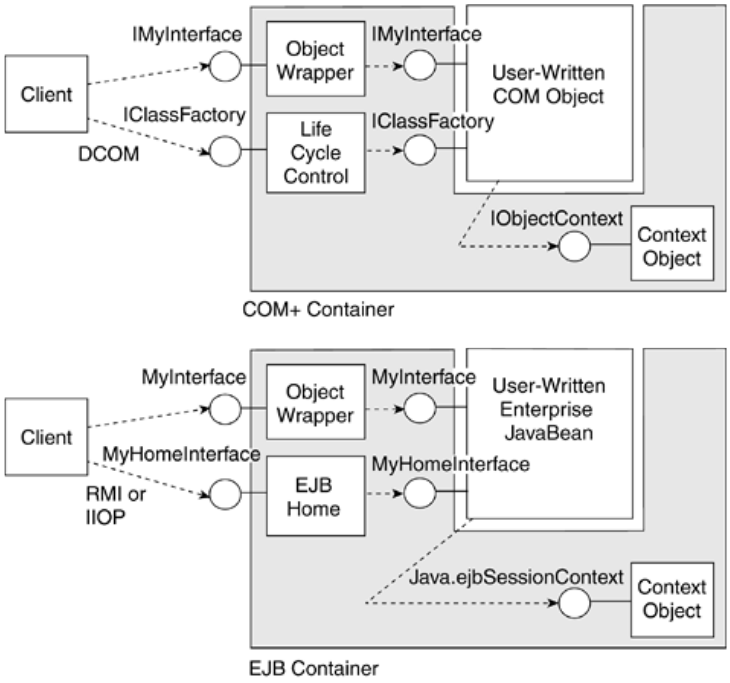
CORBA & DCOM



Middleware basado en componentes

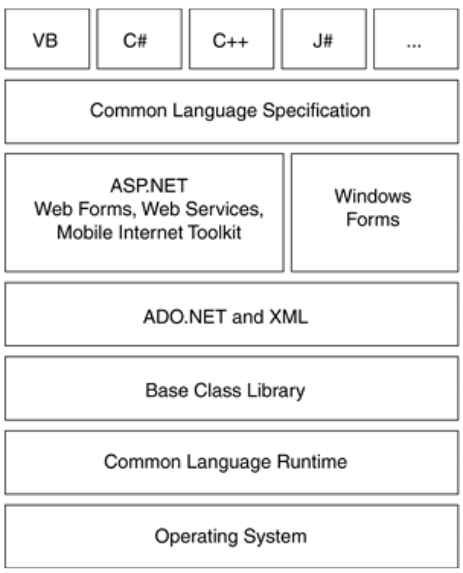
EJB [Enterprise JavaBeans]

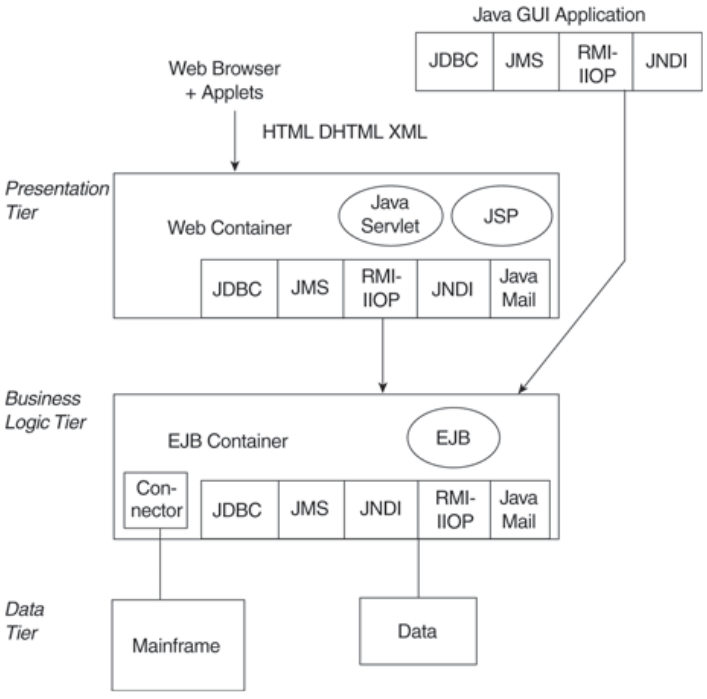
.NET Enterprise Services [COM+]



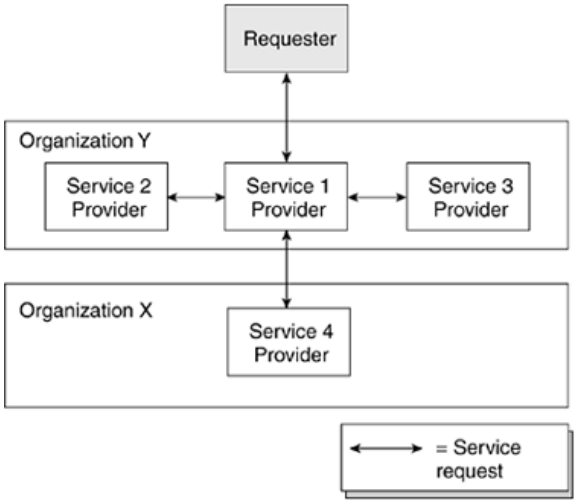
Middleware basado en componentes

Arquitecturas .NET & EJB

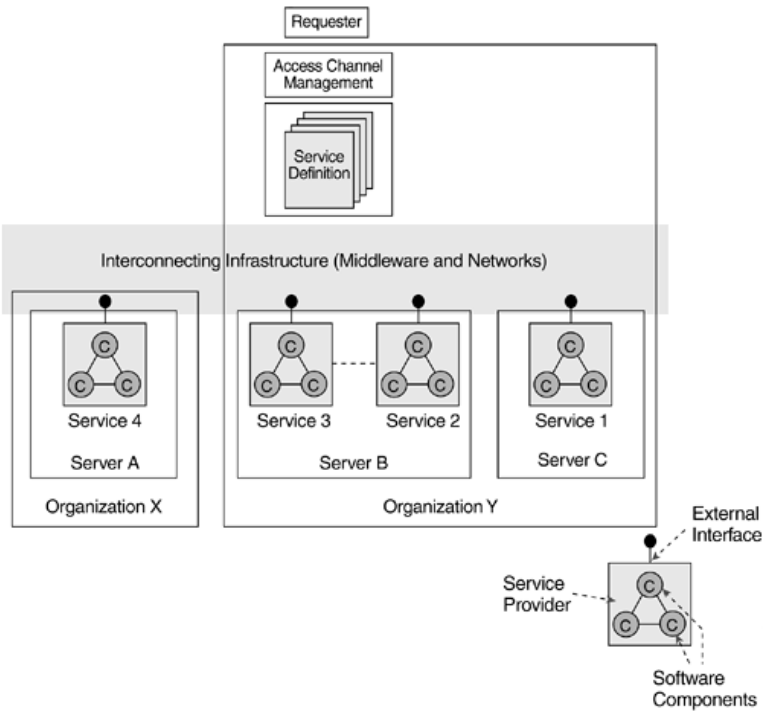




Servicios



Servicios



Servicios web

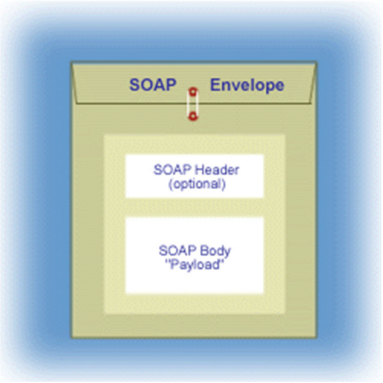
Directorio	UDDI
Inspección	DISCO
Descripción de servicios	WSDL
Paso de mensajes	SOAP
Formato de datos	XML
Comunicaciones	Internet

Más información y ejemplos…
http://elvex.ugr.es/decsai/csharp/distributed/web-services.xml

Servicios web

SOAP [Simple Object Access Protocol]

Solicitud

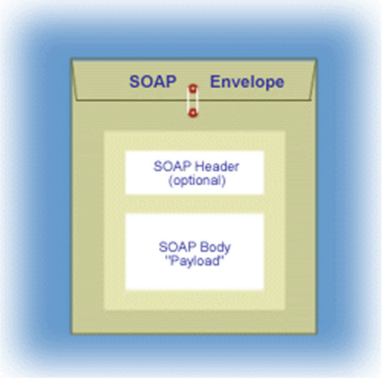



```
HTTP/1.1 POST /soap/myservice
Content-Type: text/xml
SOAPAction: MyInterface#MyComponentMethod

<SOAP:Envelope>
  <SOAP:Header>
    <MyHeader SOAP:mustUnderstand="0"> ... </MyHeader>
  </SOAP:Header>
  <SOAP:Body>
    <MyRequest>
      <argument>PI</argument>
    </MyRequest>
  </SOAP:Body>
</SOAP:Envelope>
```

Servicios web

SOAP [Simple Object Access Protocol]

Respuesta

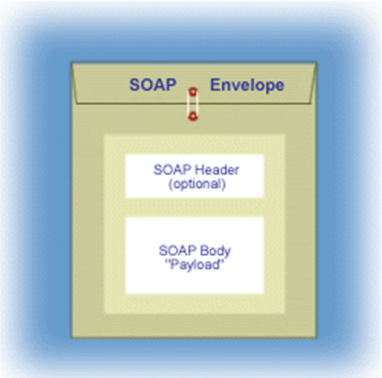



```
HTTP/1.1 200 OK
...
Content-Type:text/xml
Content-Length: XXX

<?xml version="1.0"?>
<SOAP:Envelope ...>
  <SOAP:Body>
    <MyRequestResult>
      <result>3.1416</result>
    </MyRequestResult>
  </SOAP:Body>
</SOAP:Envelope>
```


Servicios web

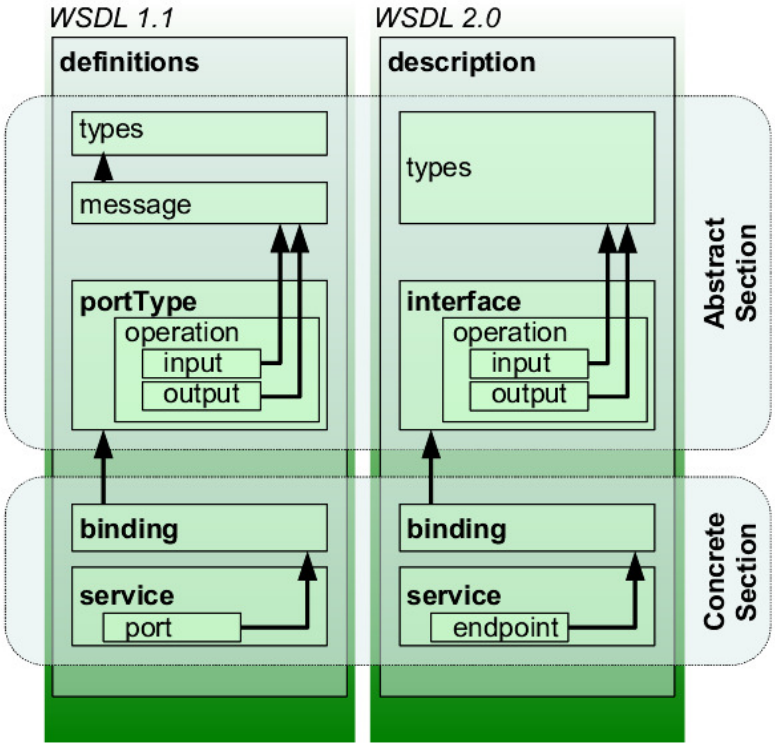
SOAP [Simple Object Access Protocol]

SOAP fault




```
HTTP/1.1 POST /soap/myservice
Content-Type: text/xml
SOAPAction: MyInterface#MyComponentMethod

<SOAP:Envelope>
  <SOAP:Body>
    <SOAP:Fault>
      <faultcode>Server.InvalidArg</faultcode>
      <faultstring>Type is wrong</faultstring>
      <detail/>
    </SOAP:Fault>
  </SOAP:Body>
</SOAP:Envelope>
```

WSDL [Web Services Description Language]



WSDL [Web Services Description Language]

```
<?xml version="1.0" encoding="UTF-8"?>
<description xmlns="http://www.w3.org/ns/wsdl"
             xmlns:tns="http://www.tmsws.com/wsdl20sample"
             xmlns:whttp="http://schemas.xmlsoap.org/wsdl/http/"
             xmlns:wsoap="http://schemas.xmlsoap.org/wsdl/soap/"
             targetNamespace="http://www.tmsws.com/wsdl20sample">

<documentation>
    This is a sample WSDL 2.0 document.
</documentation>

   <types> <!-- Abstract types (XML Schema) -->
      <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
                 xmlns="http://www.tmsws.com/wsdl20sample"
                 targetNamespace="http://www.example.com/wsdl20sample">
         <xs:element name="request"> ... </xs:element>
         <xs:element name="response"> ... </xs:element>
      </xs:schema>
   </types>
```

Servicios web

WSDL [Web Services Description Language]

```
<interface name="Interface1"> <!-- Abstract interface -->
   <fault name="Error1" element="tns:response"/>
   <operation name="Get" pattern="http://www.w3.org/ns/wsdl/in-out">
      <input messageLabel="In" element="tns:request"/>
      <output messageLabel="Out" element="tns:response"/>
   </operation>
</interface>
<!-- Concrete Binding Over HTTP -->
<binding name="HttpBinding" interface="tns:Interface1"
         type="http://www.w3.org/ns/wsdl/http">
   <operation ref="tns:Get" whttp:method="GET"/>
</binding>
<!-- Concrete Binding with SOAP-->
<binding name="SoapBinding" interface="tns:Interface1"
    type="http://www.w3.org/ns/wsdl/soap"
    soap:protocol="http://www.w3.org/2003/05/soap/bindings/HTTP/"
    wsoap:mepDefault="http://www.w3.org/2003/05/soap/mep/request-response">
   <operation ref="tns:Get" />
</binding>
```

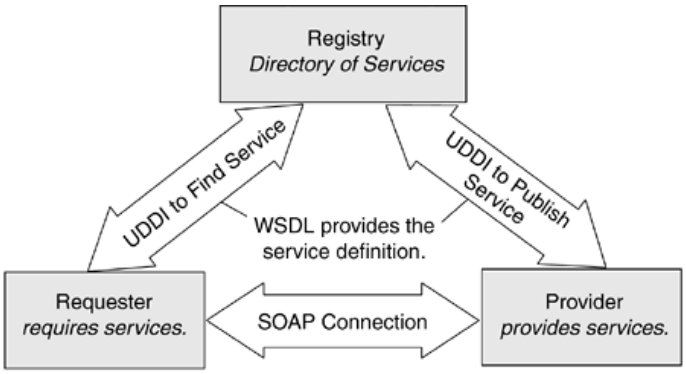

Servicios web

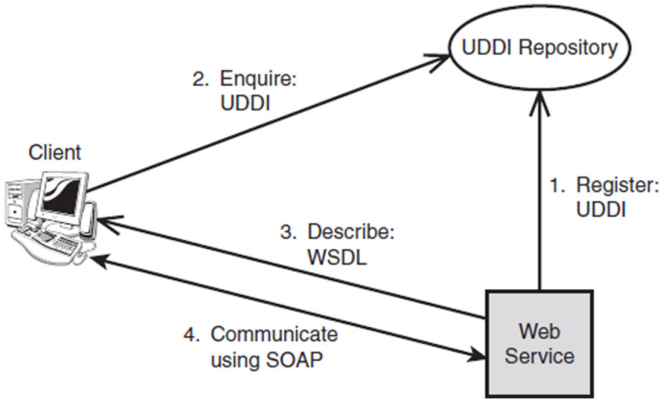
WSDL [Web Services Description Language]

```
<!-- Web Service offering endpoints for both bindings-->
<service name="Service1" interface="tns:Interface1">
   <endpoint name="HttpEndpoint"
             binding="tns:HttpBinding"
             address="http://www.example.com/rest/"/>
   <endpoint name="SoapEndpoint"
             binding="tns:SoapBinding"
             address="http://www.example.com/soap/"/>
</service>
</description>
```

Servicios web

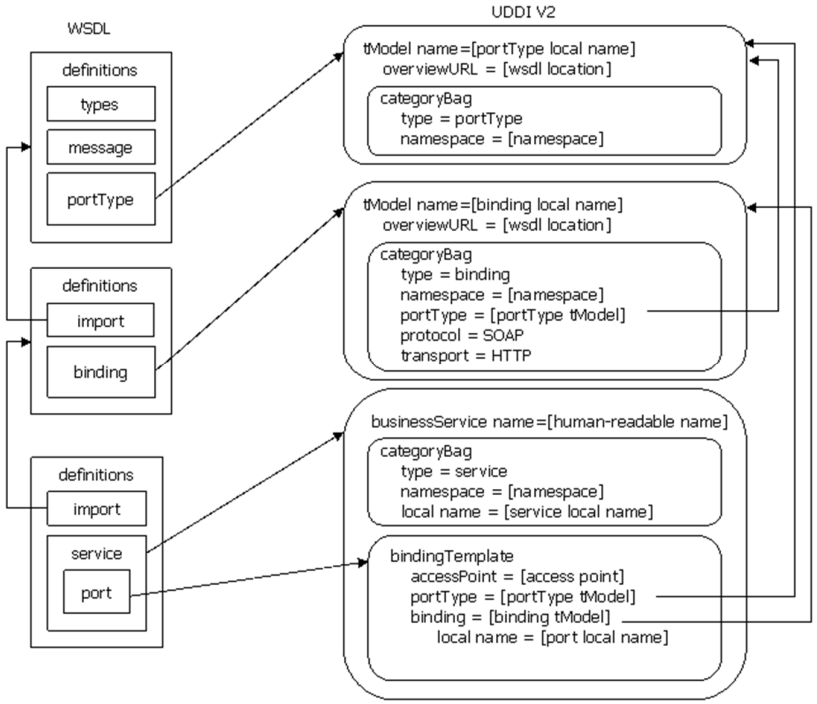
UDDI [Universal Description, Discovery, and Integration]
Descubrimiento de servicios





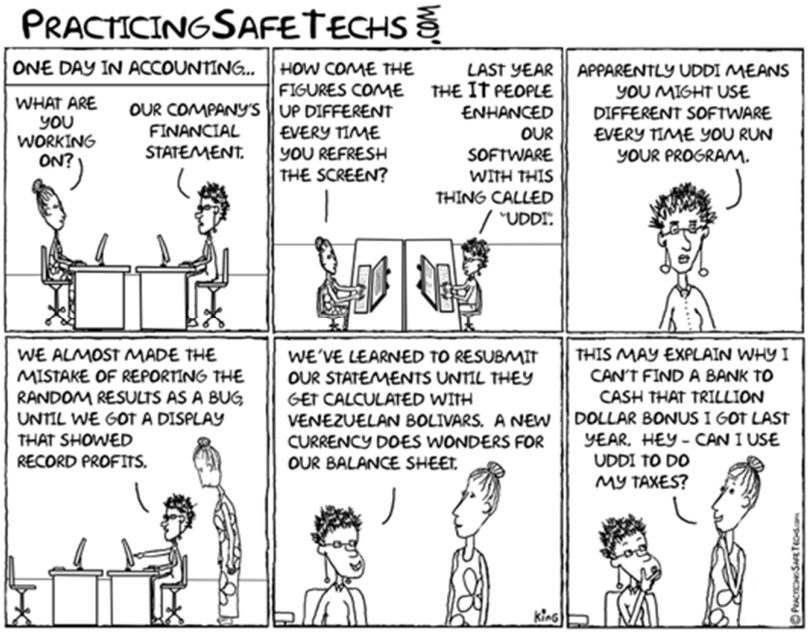
Servicios web

UDDI [Universal Description, Discovery, and Integration]

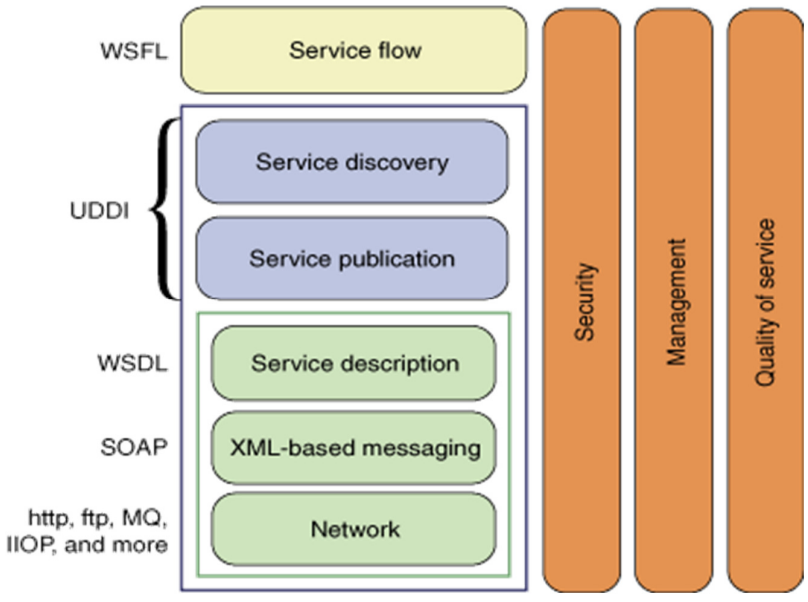


Servicios web

UDDI [Universal Description, Discovery, and Integration]
Descubrimiento de servicios



Servicios web



Servicios web

EJEMPLO: Uso de servicios web

```
// Amazon Product Advertising API

String AMAZON_ASSOCIATE_TAG = "ikor0c7-20";
String AWS_ACCESS_KEY_ID = "XXXXXXXXXXXXXXXXXXX";
String AWS_SECRET_KEY = "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx";
String ENDPOINT = "webservices.amazon.com"; // .es | .co.uk

// Set up the signed requests helper

SignedRequestsHelper helper = SignedRequestsHelper.getInstance
           (ENDPOINT, AWS_ACCESS_KEY_ID, AWS_SECRET_KEY);
```

Servicios web

EJEMPLO: Uso de servicios web

```
// Amazon Product Advertising API
// The helper can sign requests in map form,
// where the request parameters are stored in a map.

Map<String, String> params = new HashMap<String, String>();
        params.put("Service", "AWSECommerceService");
        params.put("Version", "2011-08-01");
        params.put("AssociateTag", AMAZON_ASSOCIATE_TAG);
        params.put("Operation", "ItemLookup");
        params.put("ItemId", isbn);
        params.put("ResponseGroup",
"Small,Images,ItemAttributes,OfferFull,ShippingCharges");
String requestUrl = helper.sign(params);
```

Servicios web

EJEMPLO: Uso de servicios web

```
// Amazon Product Advertising API
// Alternative string form, where the requests parameters
// have already been concatenated into a query string.

String queryString = "Service=AWSECommerceService"
                   + "&Version=2009-03-31“
                   + “&AssociateTag=" + AMAZON_ASSOCIATE_TAG
                   + "&Operation=ItemLookup"
                   + "&ResponseGroup=Small"
                   + "&ItemId=" + isbn;

String requestUrl = helper.sign(queryString);
```

Servicios web

EJEMPLO: Uso de servicios web (Java, a bajo nivel)

```
// Amazon Product Advertising API
DocumentBuilderFactory dbf =
       DocumentBuilderFactory.newInstance();
DocumentBuilder db = dbf.newDocumentBuilder();
Document doc = db.parse(requestUrl);

book.isbn = getElementValue(doc, "ASIN");
book.title = getElementValue(doc, "Title");

NodeList authorNodes = getElements(doc,"Author");
book.authors = new String[authorNodes.getLength()];

for (int i=0; i<book.authors.length; i++)
    book.authors[i] = authorNodes.item(i).getTextContent();
```

Servicios web

EJEMPLO: Uso de servicios web (Java, a bajo nivel)

```
// Amazon Product Advertising API
…
book.publisher = getElementValue(doc,"Publisher");
book.date = getElementValue(doc,"PublicationDate");
book.pages = getElementValue(doc,"NumberOfPages");
book.binding = getElementValue(doc,"Binding");

// URL
book.url = getElementValue(doc,"DetailPageURL");

// Image
Element imageElement = getElement(doc,"LargeImage");
if (imageElement!=null)
   book.image = getElementValue(imageElement,"URL");
```

Servicios web

EJEMPLO: Uso de servicios web (Java, a bajo nivel)

```
// Amazon Product Advertising API
…
// Prices (XPath)

XPathFactory xPathfactory = XPathFactory.newInstance();
XPath xpath = xPathfactory.newXPath();
XPathExpression expr = xpath.compile(
       "//Offers/Offer/OfferListing/Price/FormattedPrice");
String amazonPrice =
       ((NodeList) expr.evaluate(doc,XPathConstants.NODESET))
       .item(0).getTextContent();

book.offers.add( amazonPrice, book.url );
```

Servicios web

EJEMPLO: Uso de servicios web (Java, a bajo nivel)

```
// Rutinas auxiliares

private NodeList getElements (Document doc, String tag)
{
  return doc.getElementsByTagName(tag);
}

private Element getElement (Document doc, String tag)
{
  NodeList nodelist = doc.getElementsByTagName(tag);
  return ( (nodelist.getLength()>0) ?
           (Element) nodelist.item(0) : null);
}
```

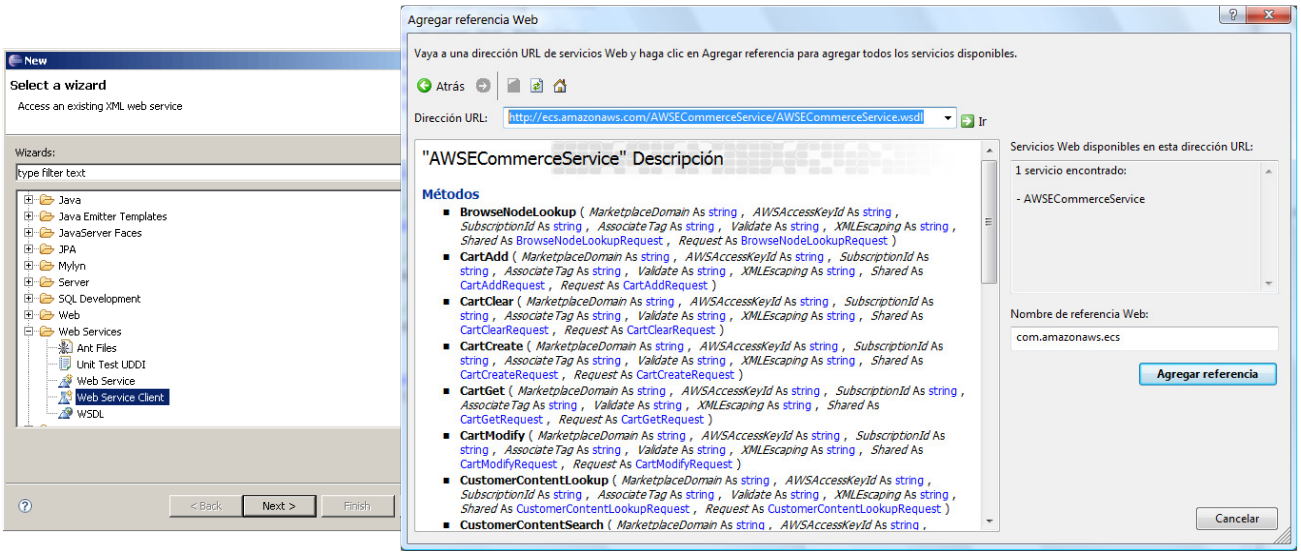
Servicios web

EJEMPLO: Uso de servicios web (Java, a bajo nivel)

```
// Rutinas auxiliares

private String getElementValue (Document doc, String tag)
{
  NodeList nodelist = doc.getElementsByTagName(tag);
  return ( (nodelist.getLength()>0) ?
           nodelist.item(0).getTextContent() : null );
}

private String getElementValue (Element root, String tag)
{
  NodeList nodelist = root.getElementsByTagName(tag);
  return ( (nodelist.getLength()>0) ?
           nodelist.item(0).getTextContent() : null );
}
```

Servicios web

Ejemplo: Uso de servicios web (con ayuda del IDE)



Añadimos un "cliente de servicio web" (Eclipse)
o una "referencia web" (Visual Studio)
y el IDE nos genera los stubs necesarios…

Servicios web

Rendimiento

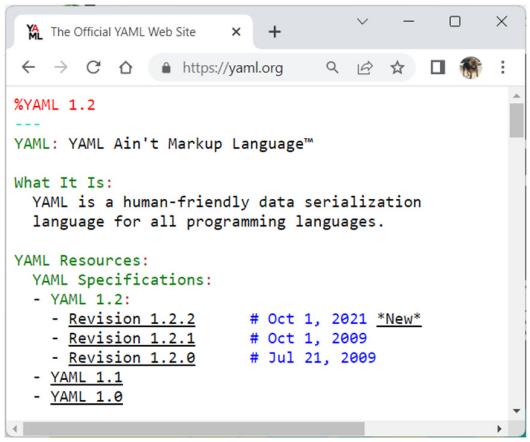
- Protocolo SOAP basado en **XML**
  (consume más ancho de banda y tiempo de CPU, ya que los mensajes requieren su interpretación [parsing]; a cambio, es legible, lo que facilita las labores de desarrollo y depuración).

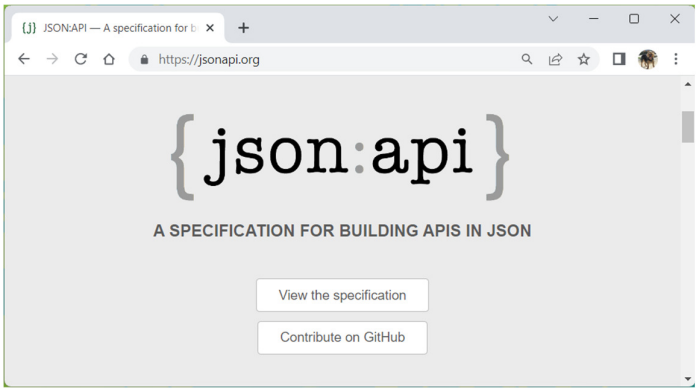
- **Mecanismos de transporte** alternativos:
  HTTP es el más común (entre otras cosas, porque permite atravesar cortafuegos sin problemas) aunque es más ineficiente que otras alternativas.

HTTP APIs

OAS [OpenAPI Specification]

Especificación de APIs en YAML o JSON





Herramientas:

- Swagger (https://swagger.io/)
- Postman (https://www.postman.com/)

HTTP APIs

HTTP APIs

Swagger
Supported by SMARTBEAR

Swagger Editor — File ▾ Edit ▾ Generate Server ▾ Generate Client ▾ About ▾

```yaml
swagger: '2.0'
basePath: /
paths:
  /api/login:
    post:
      responses:
        '201':
          description: Successful login.
          schema:
            $ref: '#/definitions/user.login'
        '401':
          description: 'Wrong credentials: incorrect user/password.'
          schema:
            $ref: '#/definitions/error'
      operationId: post_login
      parameters:
        - name: payload
          required: true
          in: body
          schema:
            $ref: '#/definitions/user.identification'
      security: []
      tags:
        - default
  /api/logout:
    post:
      responses:
        '201':
          description: Successful logout.
          schema:
            $ref: '#/definitions/user.logout'
        '403':
          description: Unauthenticated request / No active session.
          schema:
            $ref: '#/definitions/error'
      operationId: post_logout
      security:
        - token: []
      tags:
        - default
```

POST /api/login

Parameters — Try it out

Name	Description
payload * required object (body)	Example Value | Model `{ "user": "demo", "password": "xxxxxxxx" }` Parameter content type: application/json

Responses — Response content type: application/json

Code	Description
201	Successful login. Example Value | Model `{ "user": "demo", "token": "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx" }`
401	Wrong credentials: incorrect user/password. Example Value | Model `{ "error": "Error message." }`

HTTP APIs

Swagger
Supported by SMARTBEAR

```yaml
definitions:

  error:
    required:
      - error
    properties:
      error:
        type: string
        description: Error message
        example: Error message.
    type: object
```

A partir de la especificación del API, se puede generar tanto el cliente como el servidor para multitud de lenguajes de programación.

https://editor.swagger.io/

```yaml
user.identification:
  required:
    - password
    - user
  properties:
    user:
      type: string
      description: User name
      example: demo
    password:
      type: string
      description: User password
      example: xxxxxxxx
  type: object
user.login:
  required:
    - token
    - user
  properties:
    user:
      type: string
      description: User name
      example: demo
    token:
      type: string
      description: Session token
      example: xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  type: object
user.logout:
  required:
    - token
  properties:
    token:
      type: string
      description: Session token
      example: xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  type: object
```

Alternativas...

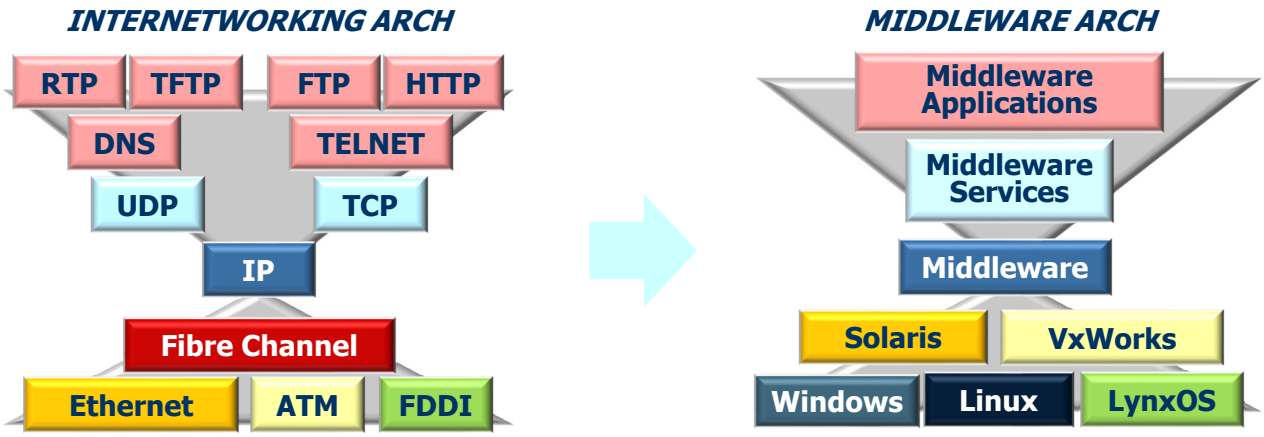
Tendencias

"Comoditización" de IT
Proceso económico por el cual los bienes, que tienen valor económico y se distinguen en términos de atributos, terminan convirtiéndose en una "commodity" a los ojos del mercado o los consumidores (un bien que es intercambiable con otros productos del mismo tipo).

- Énfasis más en la integración que en la programación.
- Convergencia y estandarización de tecnologías.
- Economías de escala.

Tendencias

"Comoditización" de IT



20th Century

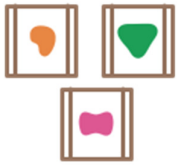
21st Century

Tendencias

Arquitecturas basadas en microservicios

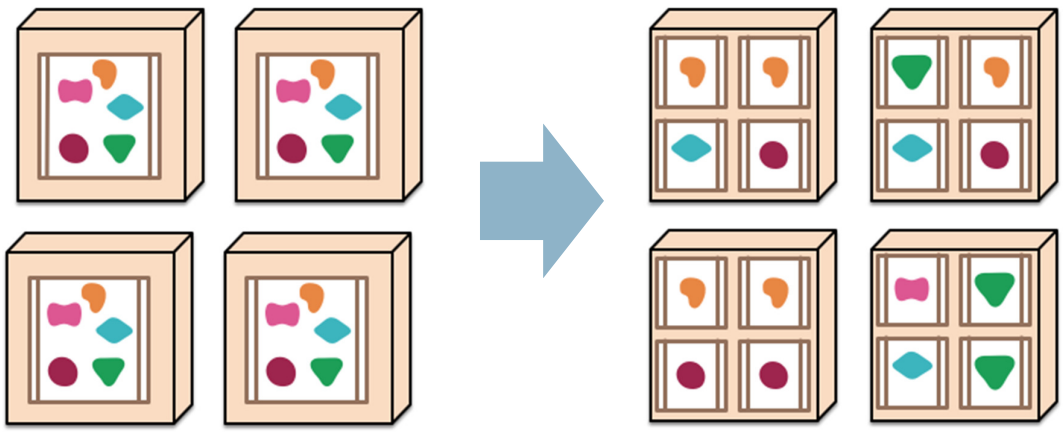
A monolithic application puts all its functionality into a single process...

A microservices architecture puts each element of functionality into a separate service...



... and scales by replicating the monolith on multiple servers

... and scales by distributing these services across servers, replicating as needed.



Tendencias

Arquitecturas basadas en microservicios

Características

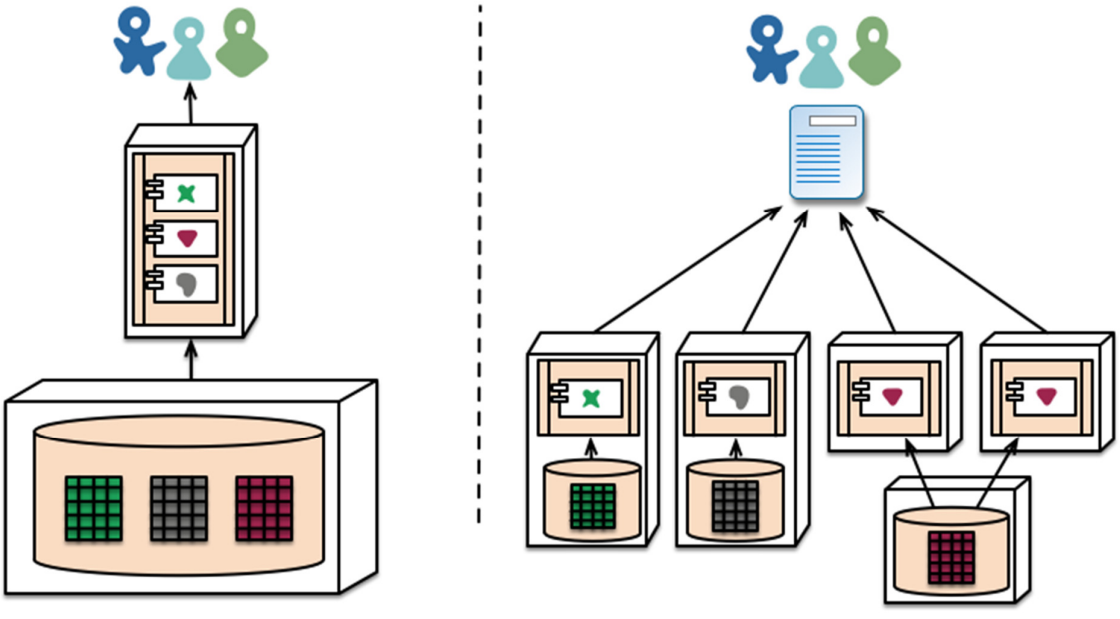
- "Componentización" vía servicios (web o RPC).
- Organización en torno a unidades de negocio (no alrededor de especialidades, e.g. GUI, BD…).
- Orientación a productos (no proyectos): "you build it, you run it".
- Acoplamiento débil (p.ej. HTTP): "smart endpoints & dumb pipes".
- Control descentralizado :-)
- Datos descentralizados :-(

http://martinfowler.com/articles/microservices.html

Tendencias

Arquitecturas basadas en microservicios

Descentralización… para lo bueno y para lo malo.

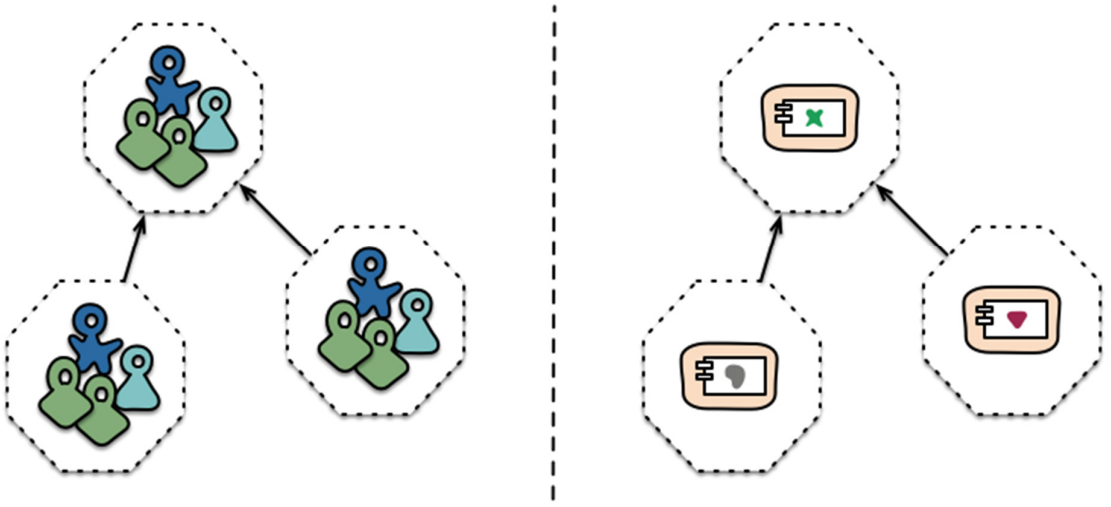


Tendencias

Arquitecturas basadas en microservicios

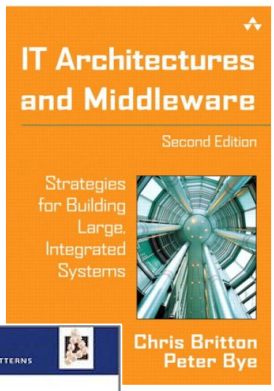
¿Cómo de grandes son los microservicios?

Amazon: "Two Pizza Team"
(el equipo completo puede alimentarse con 2 pizzas, i.e. no más de una docena de personas por equipo).

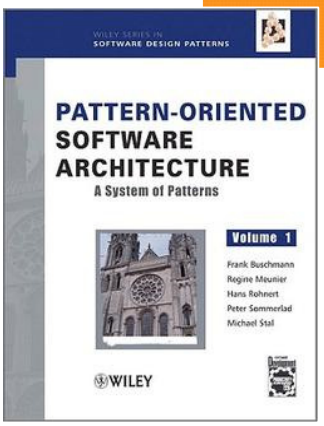


Bibliografía

- Chris Britton & Peter Bye: **IT Architectures and Middleware** Addison-Wesley, 2nd edition, 2004 ISBN 0-321-24694-2

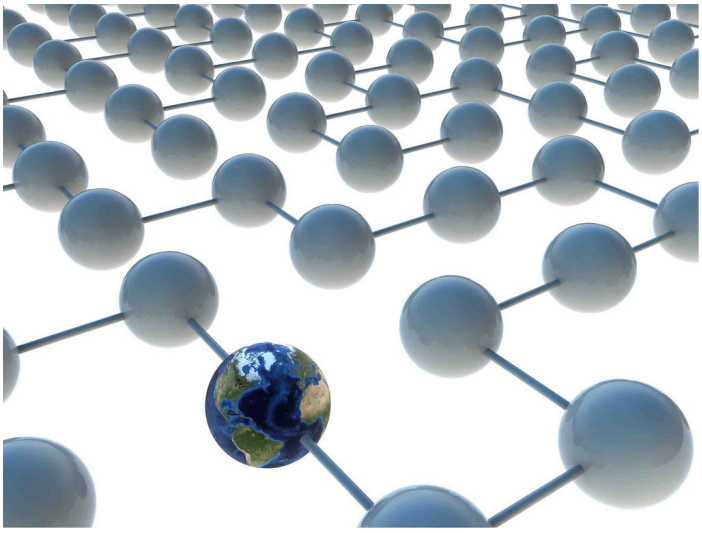


- Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad & Michael Stal: **Pattern-Oriented Software Architecture**. **Volume 1: A System of Patterns** Wiley, 1996. ISBN 0471958697



Curso recomendado

Pattern-Oriented Software Architectures for Concurrent and Networked Software
Douglas C. Schmidt (Vanderbilt University)
https://www.coursera.org/course/posasoftware



coursera

ZooKeeper

https://zookeeper.apache.org/

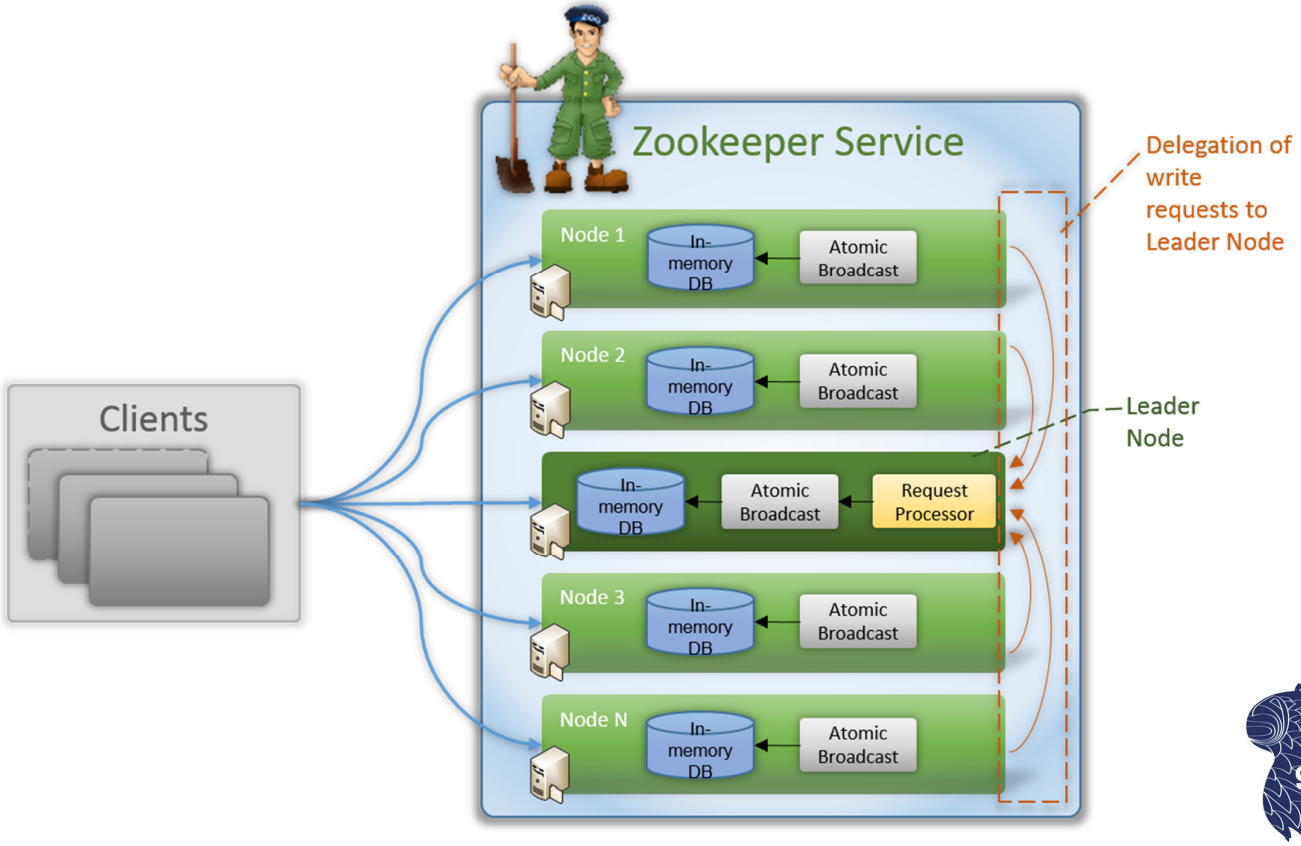
Objetivo:

Coordinación de procesos distribuidos**.**

Función:

Servicio centralizado para
mantener información de configuración,
elección de líderes [master election],
detección de fallos [crash detection],
gestión de grupos [group membership]…
Proporciona algoritmos asíncronos para sistemas distribuidos, difíciles de implementar correctamente.

ZooKeeper



ZooKeeper

Usos:

Apache Hadoop (map-reduce framework): YARN & HDFS
Apache HBase (NoSQL database)
Apache Solr (search engine)
Apache Storm (streaming)
Yahoo! Fetching Service (crawler)
Facebook Messages
eBay
Twitter
Netflix
Zynga
...