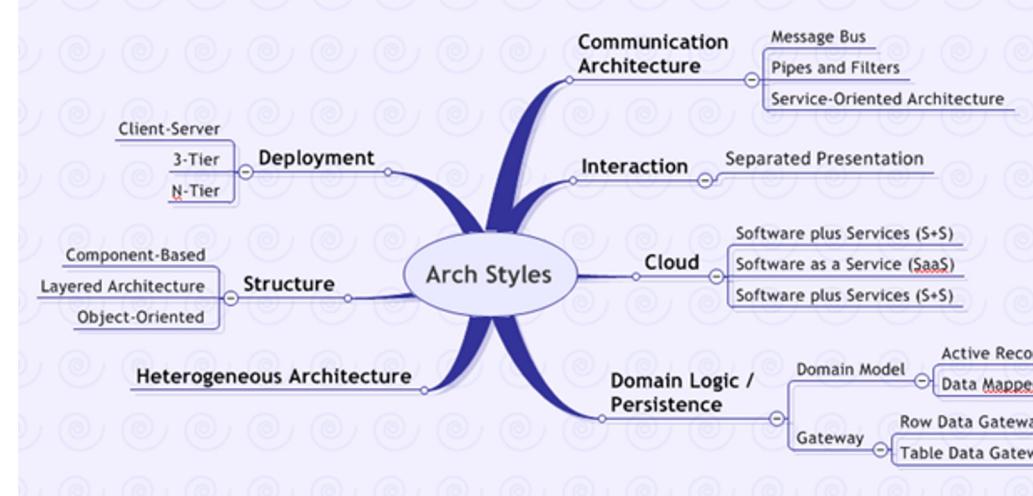
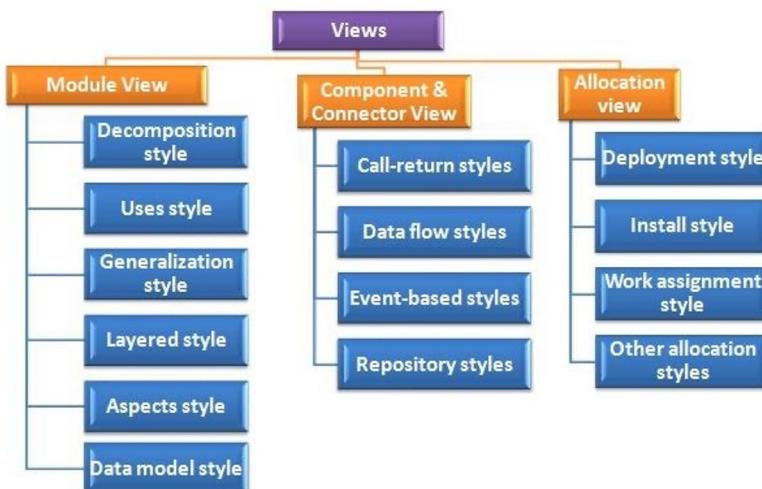
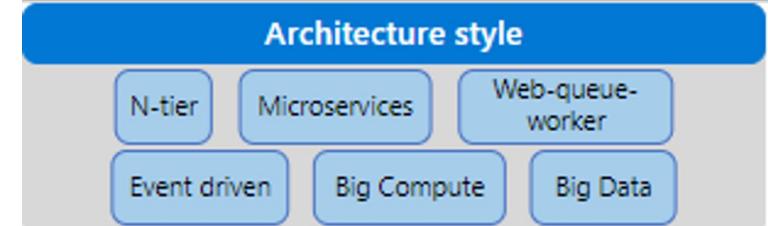
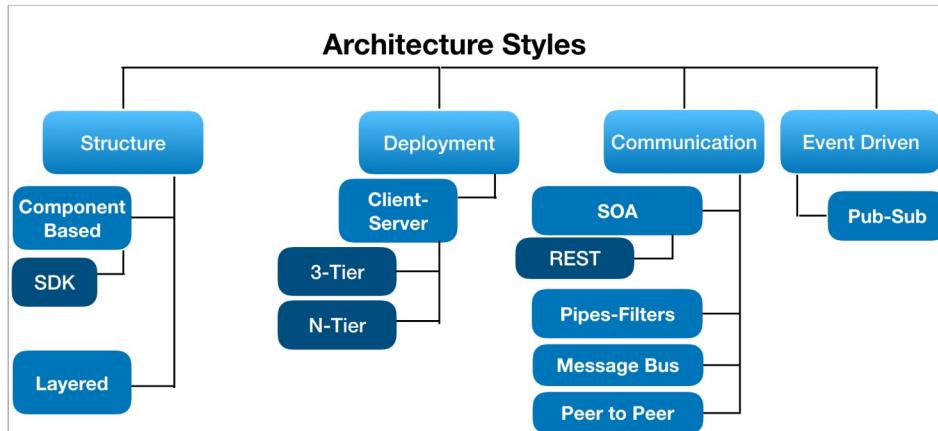


Connectors and Components

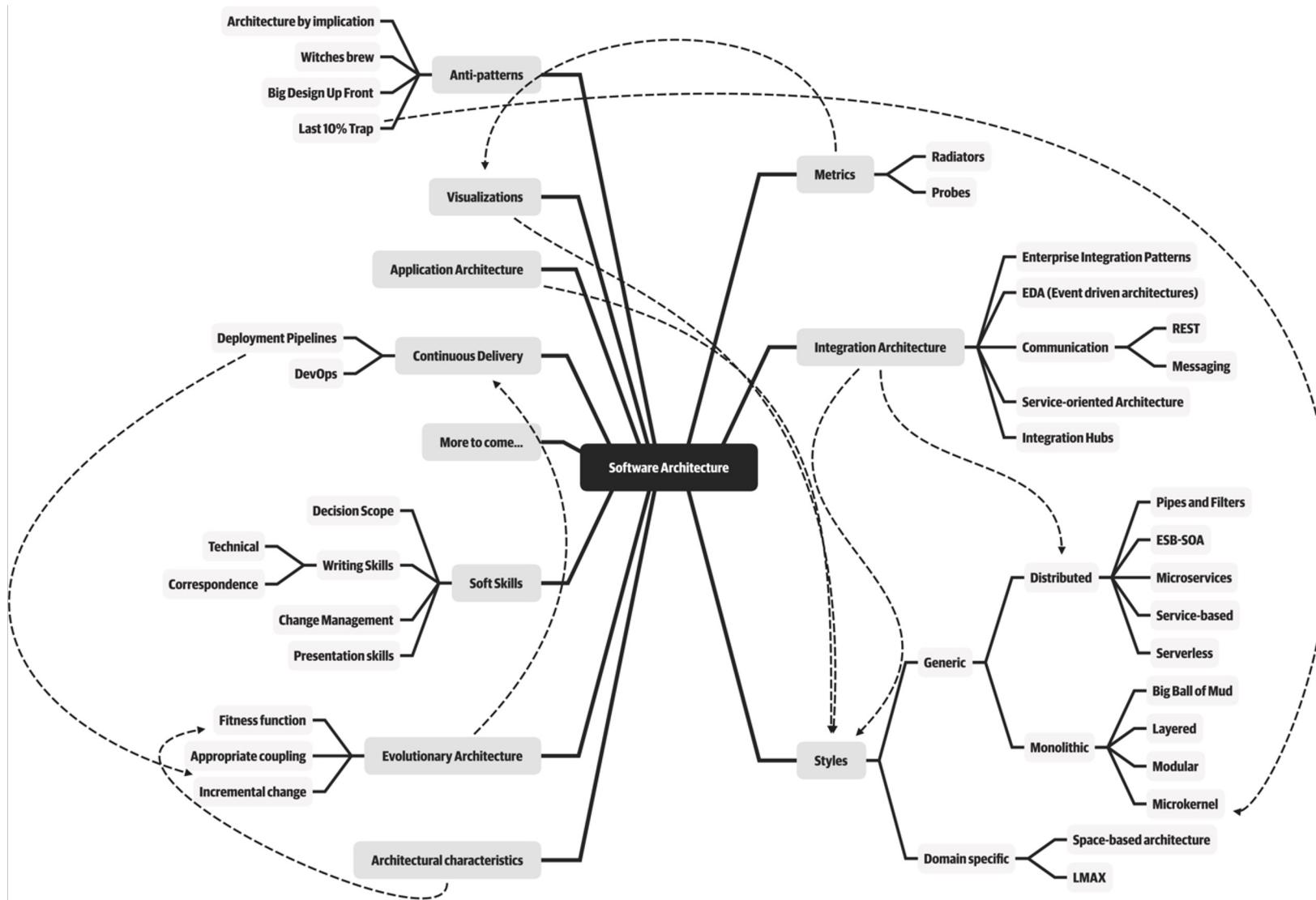
Estilos arquitectónicos

- Un estilo describe una clase de arquitectura o piezas de la misma
- Un «estilo» de arquitectura es un conjunto de decisiones de diseño arquitectural que son aplicables en un contexto de desarrollo específico, restringen las decisiones de diseño de un sistema a ese contexto y plantean como objetivo ciertas cualidades para el sistema resultante.
- An architectural style defines: a family of systems in terms of a pattern of structural organization; a vocabulary of components and connectors, with constraints on how they can be combined.
- Architectural styles are reusable 'packages' of design decisions and constraints that are applied to an architecture to induce chosen desirable qualities.

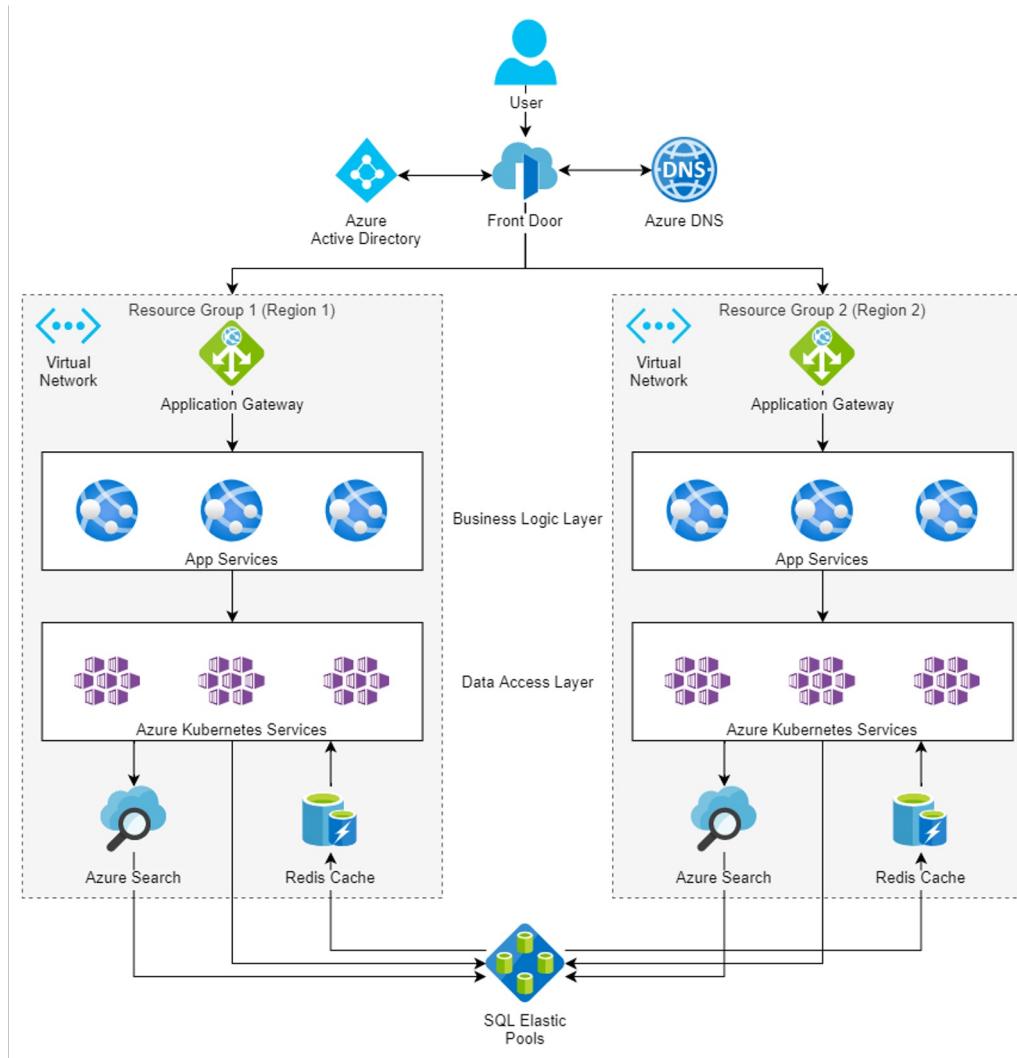
Estilos arquitectónicos



Estilos arquitectónicos - Arquitecto software



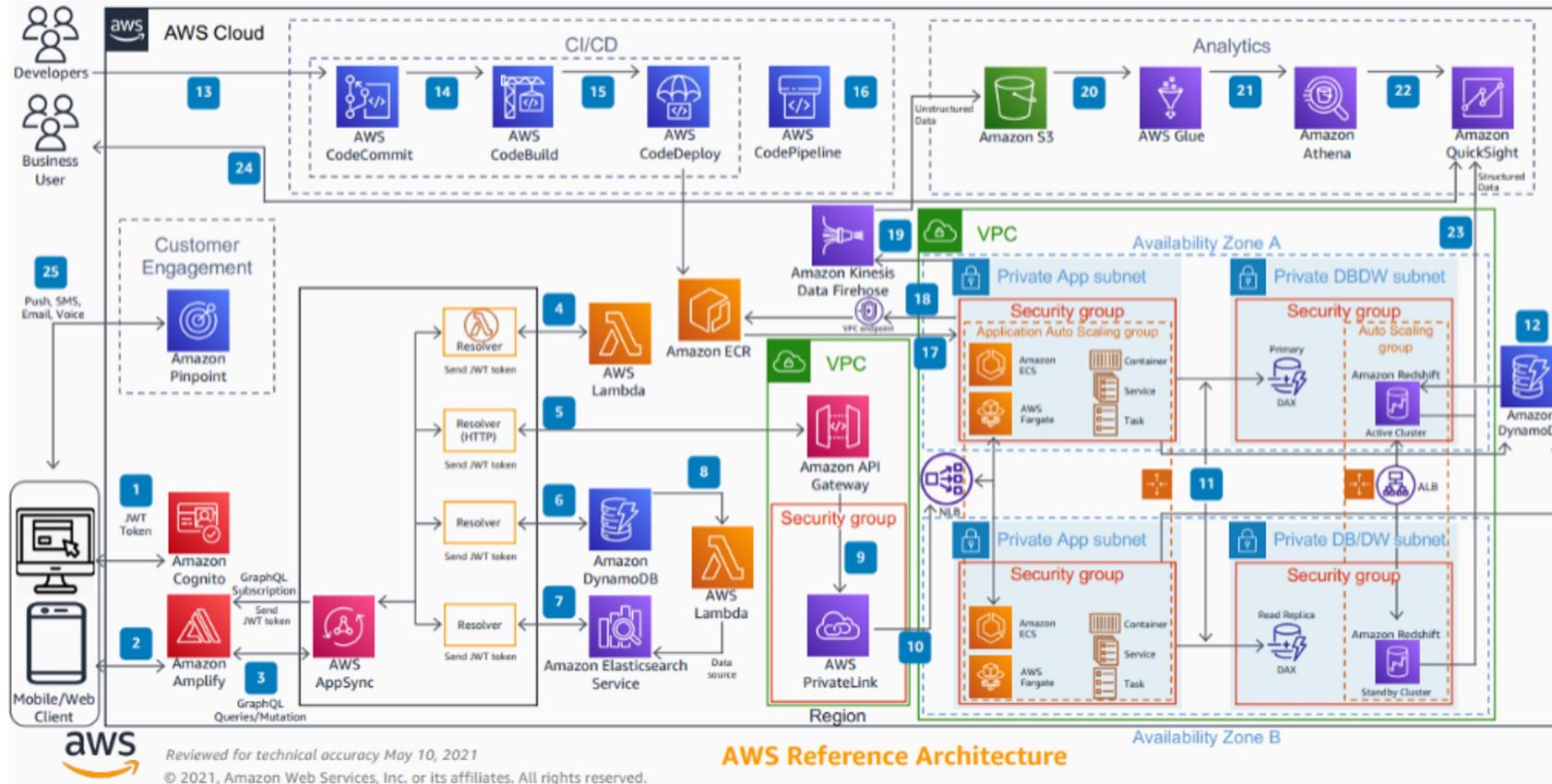
Estilos arquitectónicos



Estilos arquitectónicos

Modern Serverless Mobile/Web Application Architecture Integrated with CI/CD and Analytics Use Case

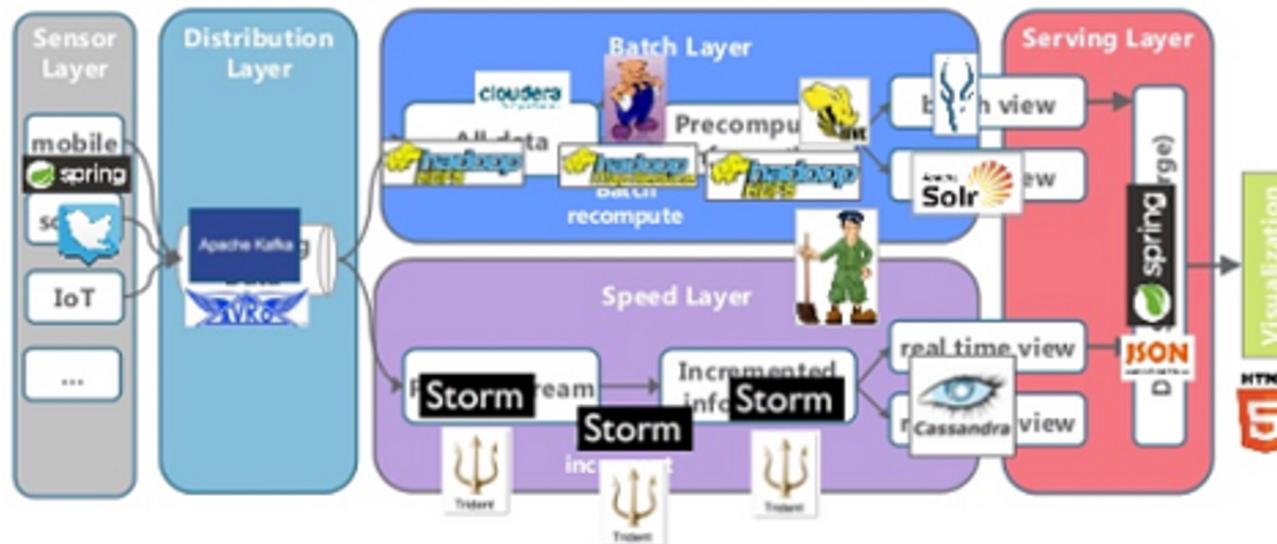
This diagram shows how to build a modern serverless mobile/web application in AWS for mobile/web clients, using AWS AppSync for frontend and ECS Fargate containers for backend application, along with Continuous Integration and Delivery and analytics to derive insight from application logs and structured data using the Amazon QuickSight dashboard.



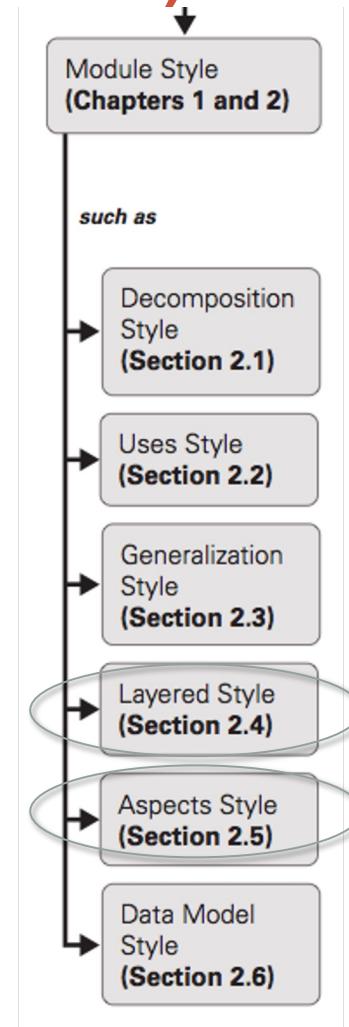
Estilos arquitectónicos

■ Lambda Architecture

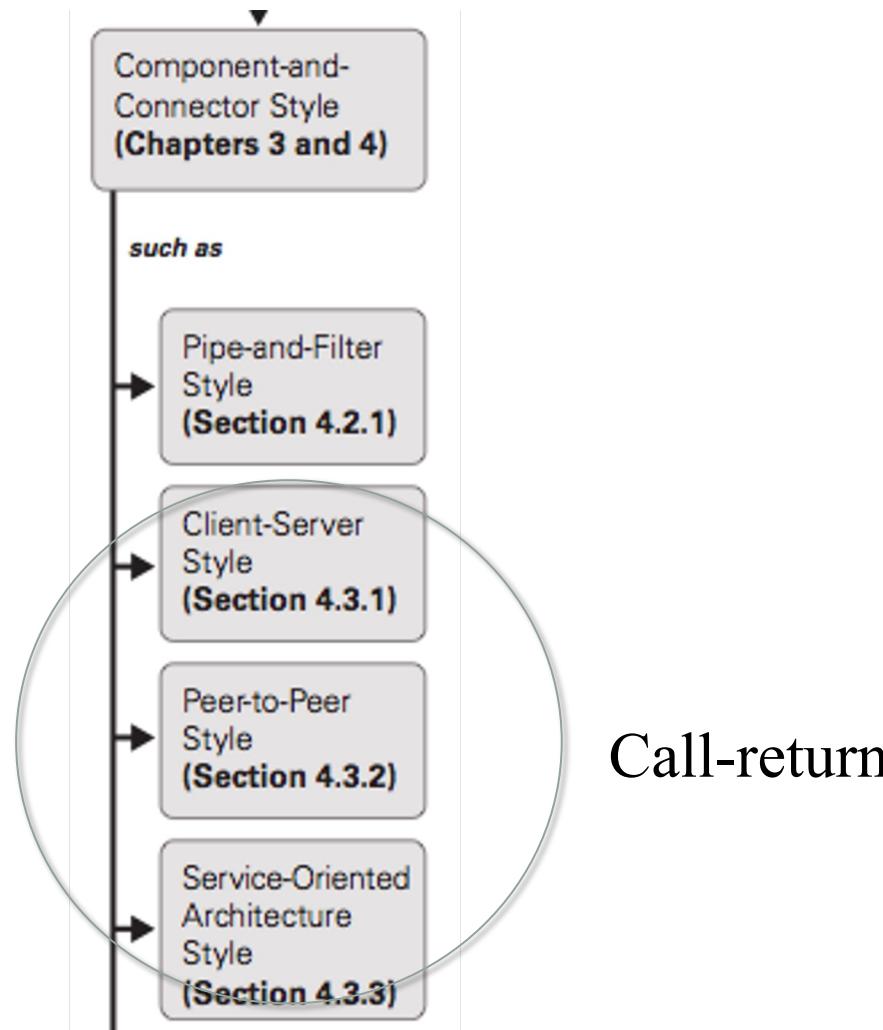
Open Source Frameworks for implementing a Lambda Architecture



Estilos arquitectónicos (vistos en cursos anteriores)



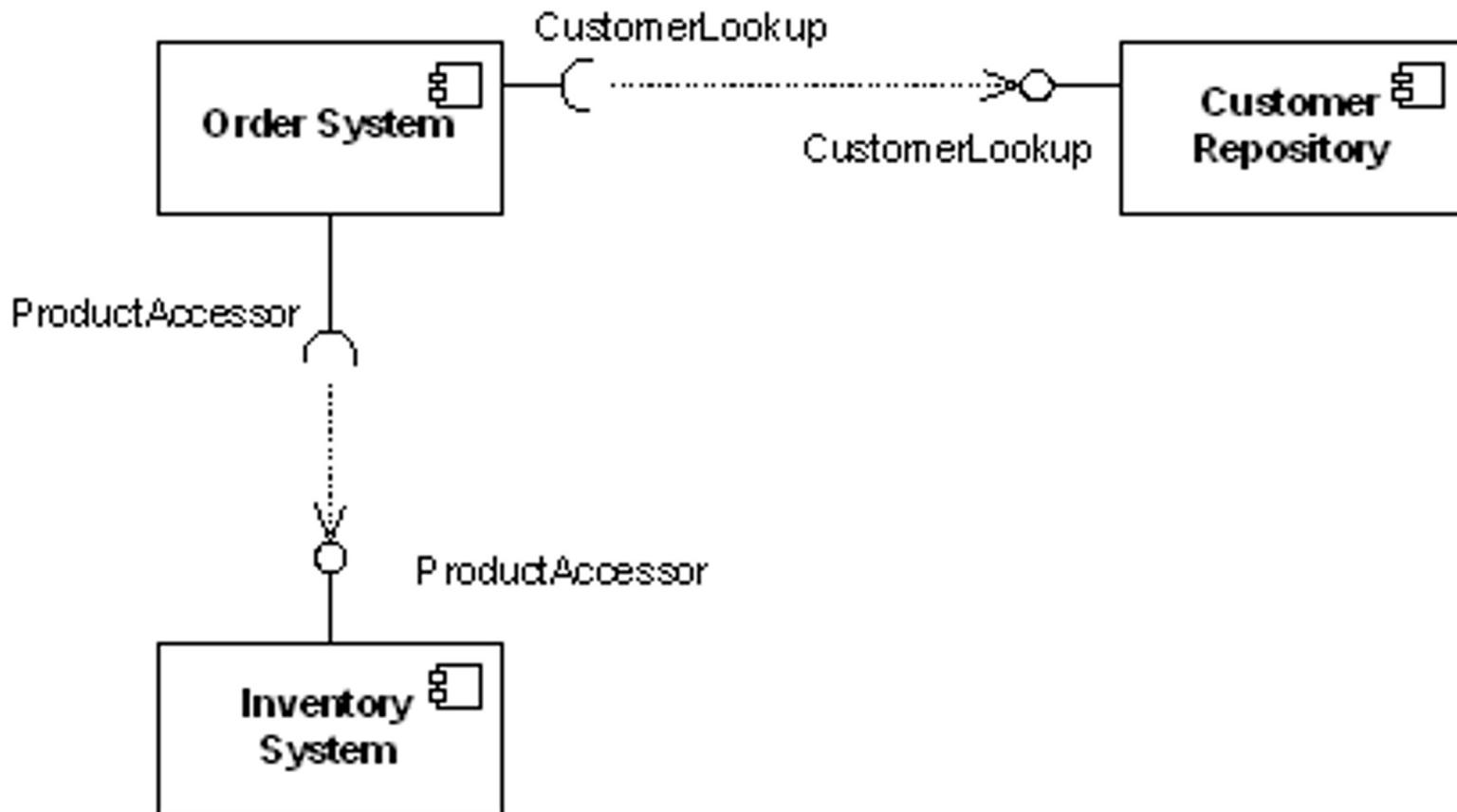
Estilos arquitectónicos



Componentes y conectores

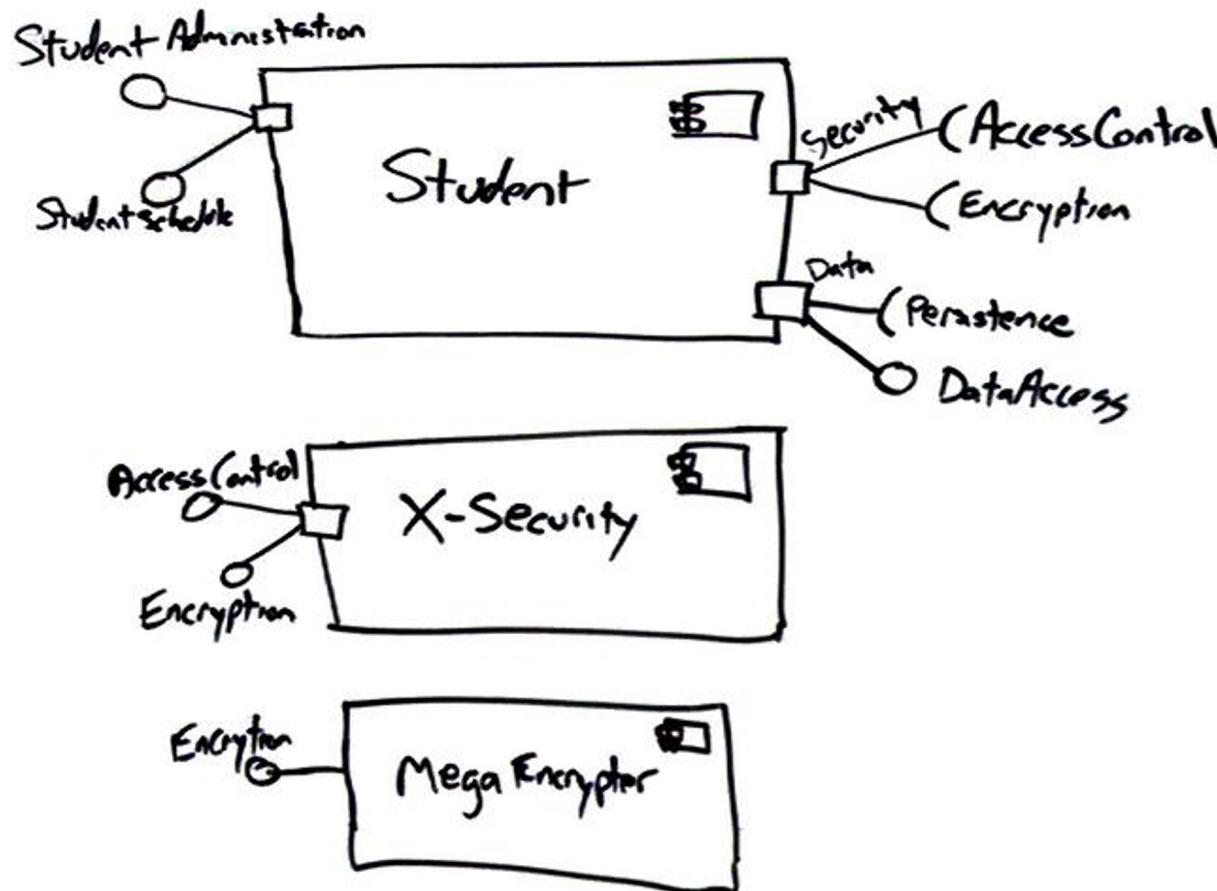


Componentes y Conectores



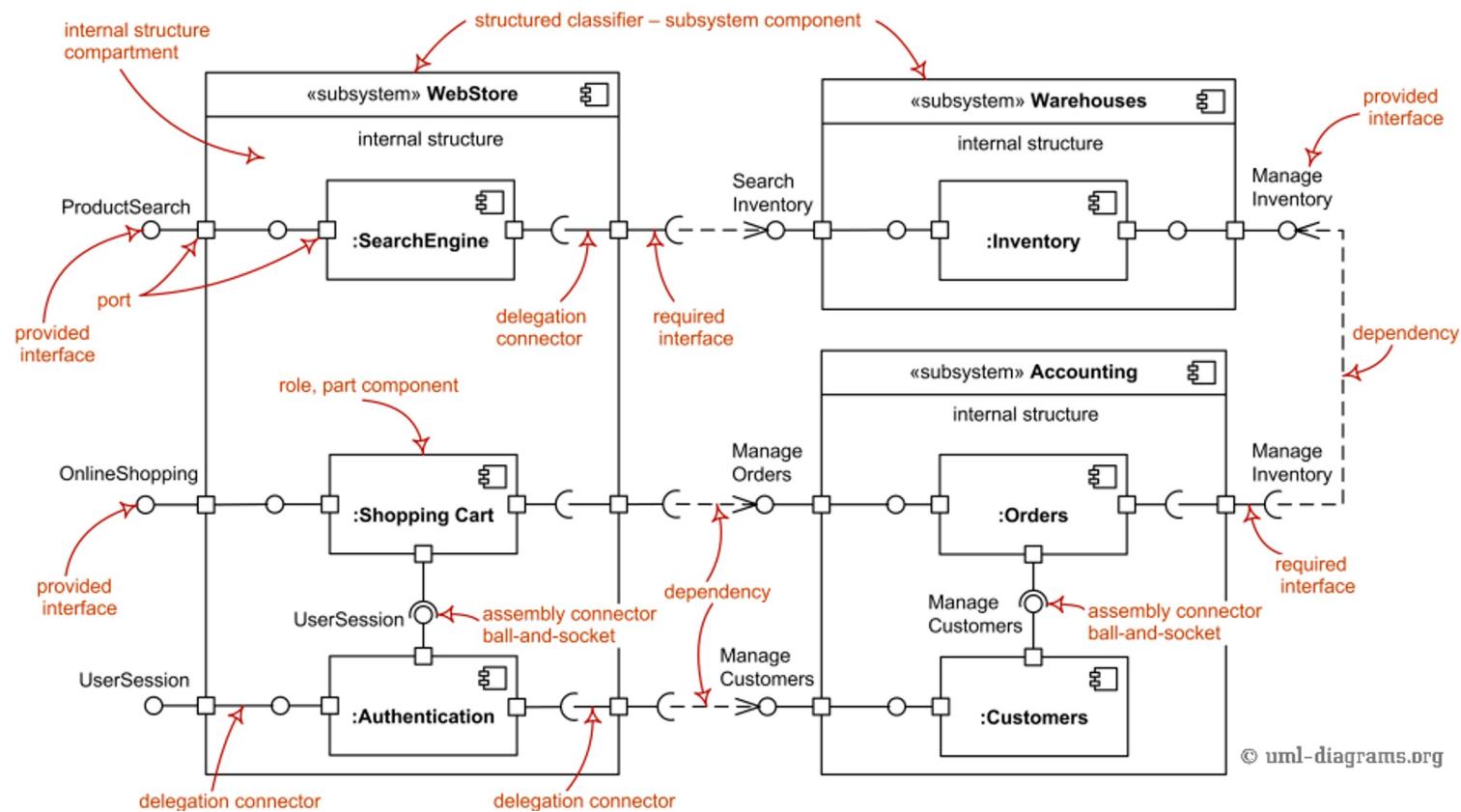
Componentes y Conectores

- Interfaces y Puertos

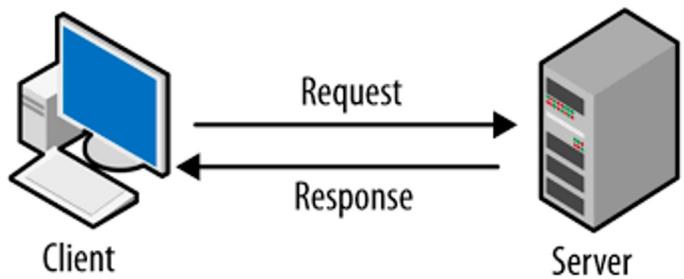


Componentes y Conectores

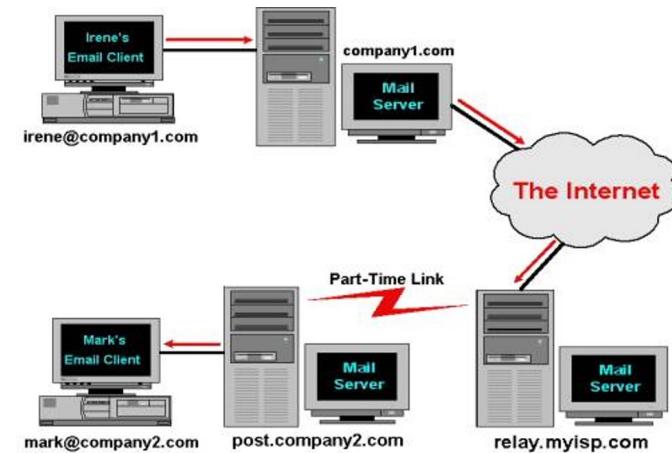
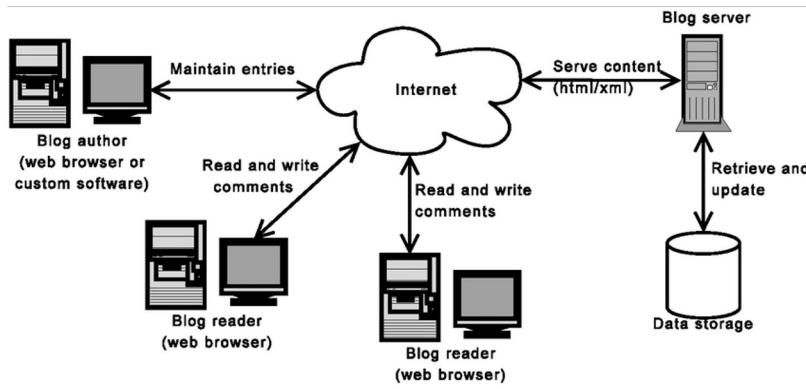
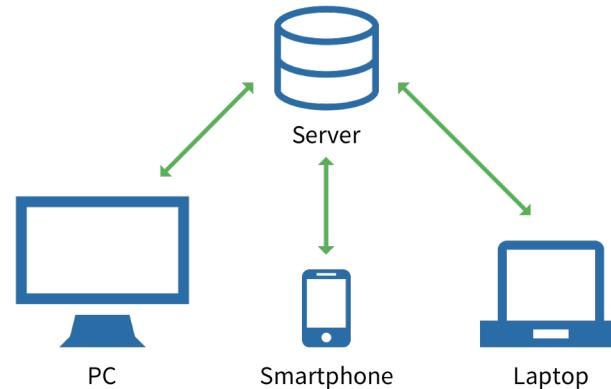
- Representación



Ejemplos - estilo cliente/servidor aplicado.



Client-Server Model

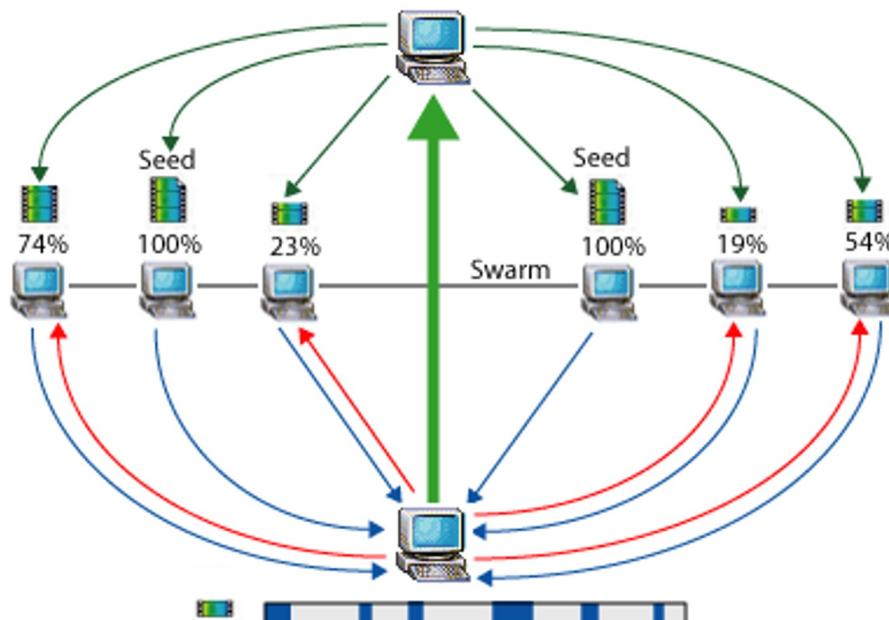


Continuamos la próxima semana!

Estilos arquitectónicos

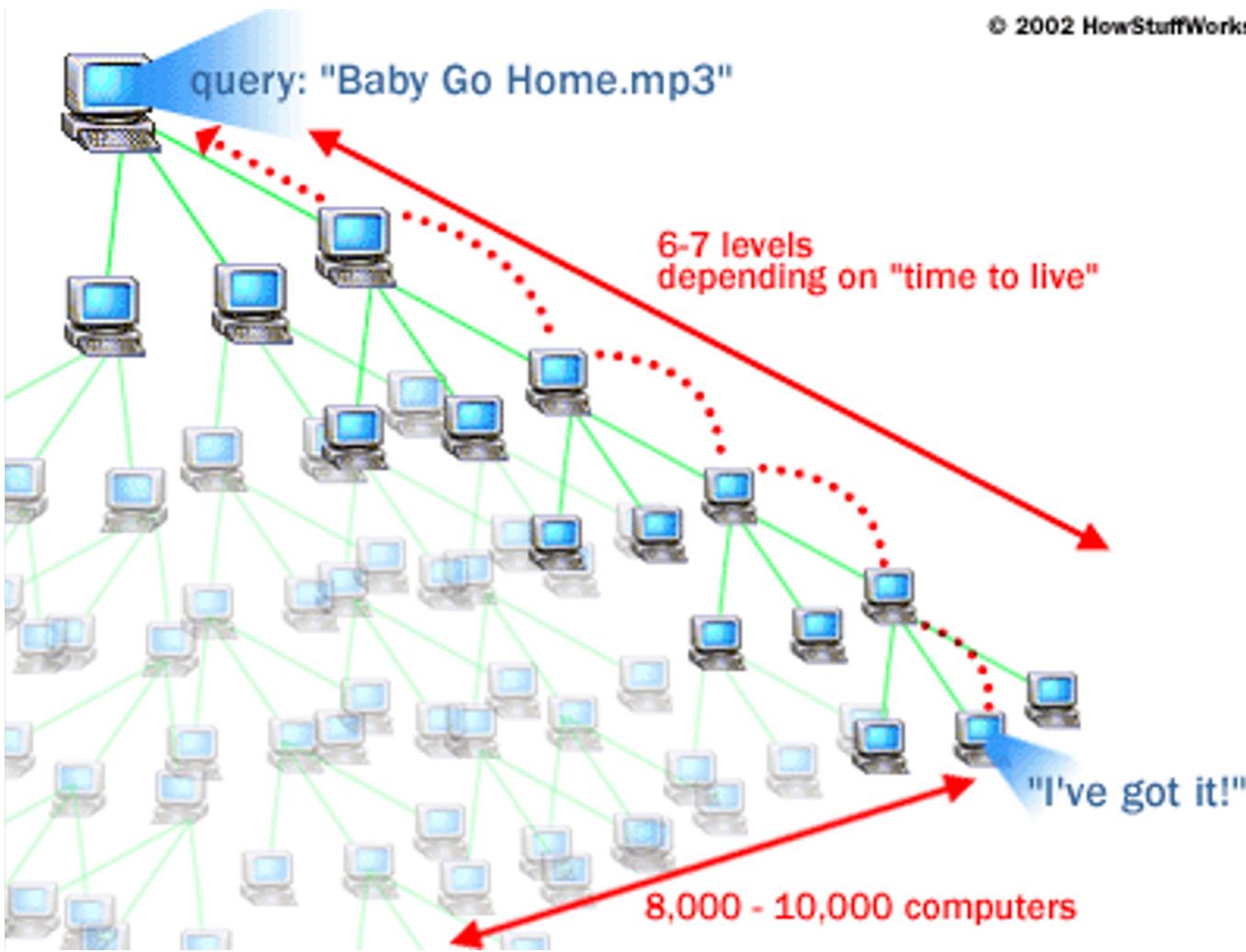
P2P (Punto a punto)

BitTorrent tracker identifies the swarm and helps the client software trade pieces of the file you want with other computers.

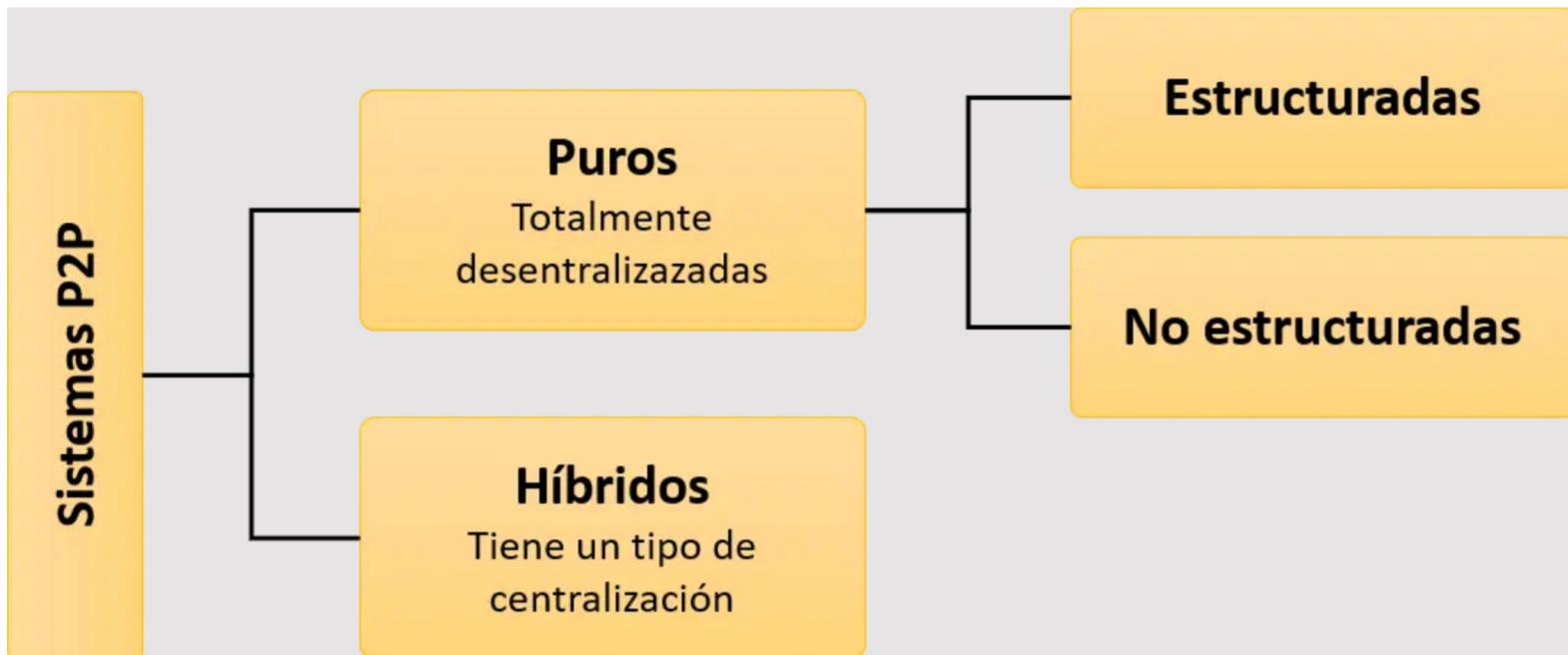


Computer with BitTorrent client software receives and sends multiple pieces of the file simultaneously.

Gnutella



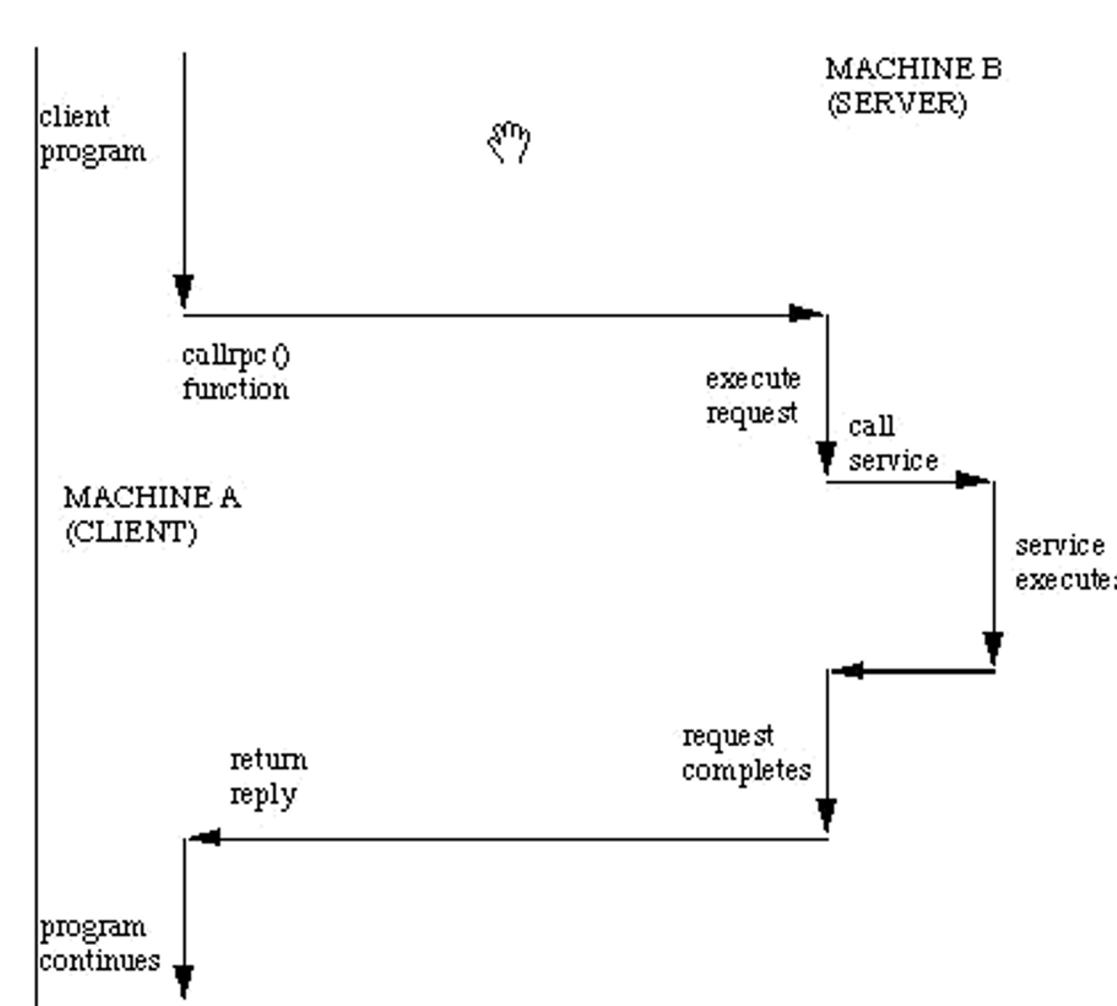
Clasificación de la arquitectura



Alternativas de implementación.

- Bajo nivel:
 - Sockets + protocolos propios.
- Alto nivel:
 - RPC (Remote Procedure Call)
 - ROI (Remote Object Invocation)

RPC



RPC

```
int
rpc_call (char *host /* Name of server host */,
          u_long progrnum /* Server program number */,
          u_long versnum /* Server version number */,
          xdrproc_t inproc /* XDR filter to encode arg */,
          char *in /* Pointer to argument */,
          xdr_proc_t outproc /* Filter to decode result */,
          char *out /* Address to store result */,
          char *nettype /* For transport selection */);
```

RPC

```
#include <stdio.h>
#include <utmp.h>
#include <rpc/rpc.h>
#include <rpccsvc/rusers.h>

/* a program that calls the RUSERSPROC
 * RPC program
 */

main(int argc, char **argv)
{
    unsigned long nusers;
    enum clnt_stat cs;
    if (argc != 2) {
        fprintf(stderr, "usage: rusers hostname\n");
        exit(1);
    }

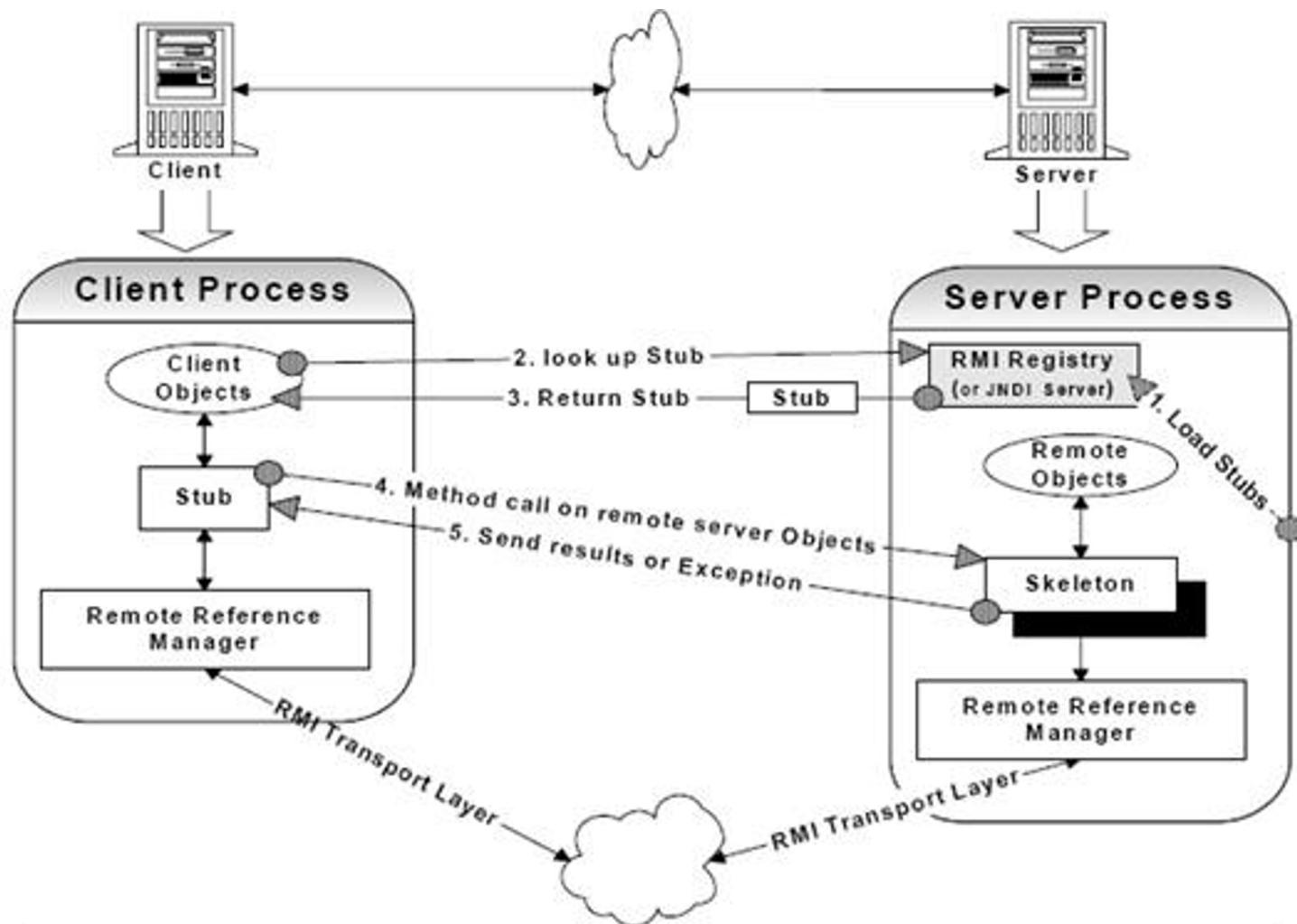
    if( cs = rpc_call(argv[1], RUSERSPROG,
                      RUSERSVERS, RUSERSPROC_NUM, xdr_void,
                      (char *)0, xdr_u_long, (char *)&nusers,
                      "visible") != RPC_SUCCESS ) {
        clnt_perrno(cs);
        exit(1);
    }

    fprintf(stderr, "%d users on %s\n", nusers, argv[1] );
    exit(0);
}
```

Remote Object Invocation

- Java RMI (Remote Method Invocation)
- .NET Remoting / DCOM.

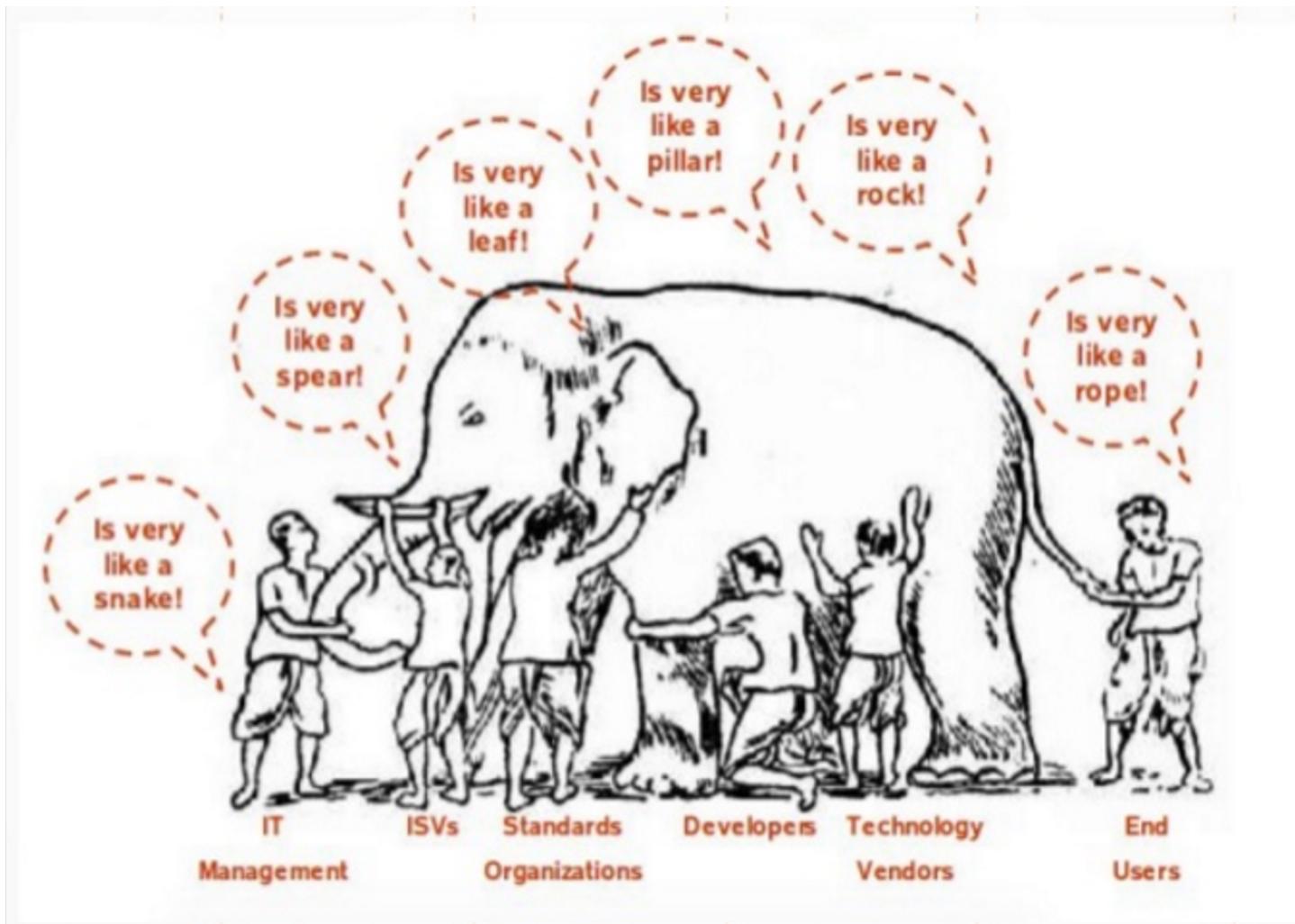
RMI



RPC vs Remote Object Invocation

RPC	R O I
Abstracción a nivel procedural (procedimiento remoto)	Abstracción a nivel de objeto (objetos remotos).
Soportado por múltiples plataformas.	Protocolos propietarios => no compatible con otras plataformas.
Complejidad para el envío de parámetros y respuestas: alta.	Complejidad para el envío de parámetros y respuestas: muy baja
No hay noción de estado.	Se tienen referencias a objetos remotos exactamente iguales con el mismo comportamiento de los objetos locales.

SOA



SOA (architectural style)

- *Service-oriented architecture (SOA) is a paradigm where software components are created with concise interfaces, and each component performs a discrete set of related functions. With its well-defined interface and contract for usage, each component, provides a service to other software components.*

SOA vs Cliente/Servidor – P2P

Cliente/Servidor

- Lógica principalmente en el cliente.
- Clientes 'gordos' y únicos.
- Lenguaje: 3GL/4GL

SOA

- Lógica: principio de abstracción.
- Bajo acoplamiento, sin estado: clientes simples.
- Lenguaje: estándares independientes de cualquier plataforma.

SOA

- Principios
 - Bajo acoplamiento de los servicios
 - Abstracción de servicios
 - Autonomía de los servicios
 - Ausencia del concepto de estado en los servicios.
 - Facilidad de descubrimiento
 - Compatibilidad con estándares

Bajo acoplamiento

Within the [service-orientation design paradigm](#), **service loose coupling** is a design principle^[1] that is applied to the services^[2] in order to ensure that the service contract is not tightly coupled to the service consumers and to the underlying service logic and implementation. This results in service contracts that could be freely evolved without affecting either the service consumers or the service implementation.^[3]

Abstracción

Service abstraction is a design principle that is applied within the [service-orientation design paradigm](#) so that the information published in a service contract is limited to what is required to effectively utilize the service^[1]. The service contract should not contain any superfluous information that is not required for its invocation. Also that the information should be limited to the serviced contract (technical contract and the [SLA](#)) only, no other document or medium should be made available to the service consumers other than the [service contract](#) that contains additional service related information.

Autonomía

Service autonomy is a design principle that is applied within the [service-orientation design paradigm](#), to provide services with improved independence from their execution environments.^[1] This results in greater reliability, since services can operate with less dependence on resources over which there is little or no control.

Ausencia de estado

Service statelessness is a design principle that is applied within the [service-orientation design paradigm](#), in order to design [scalable services](#) by separating them from their [state](#) data whenever possible.^[1] This results in reduction of the resources consumed by a service as the actual state data management is delegated to an external component or to an architectural extension. By reducing resource consumption, the service can handle more requests in a reliable manner.^[2]

Facilidad de descubrimiento (metadatos)

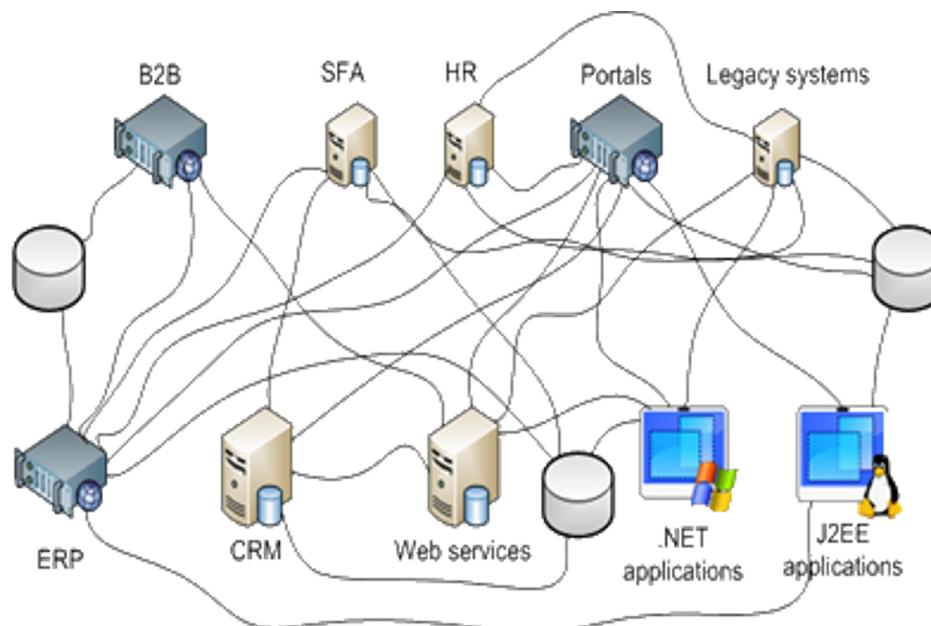
Discoverability is the ability of something, especially a piece of content or information, to be found. Discoverability is a concern in [library and information science](#), many aspects of digital media, software and web development, and in marketing, since a thing cannot be used if people cannot find it or don't understand what it's for. [Metadata](#), or "information about information," such as a book's title, a product's description, or a website's keywords, affects how discoverable something is. Organizing information by putting it into alphabetical order or including it in a search engine is an example of how to improve discoverability. Discoverability is related to, but different from, [accessibility](#) and [usability](#), other qualities that affect the usefulness of a piece of information.

SOA – Service Oriented Architecture

- Estilo arquitectónico
- Enfocado a sistemas distribuidos

SOA e Integración

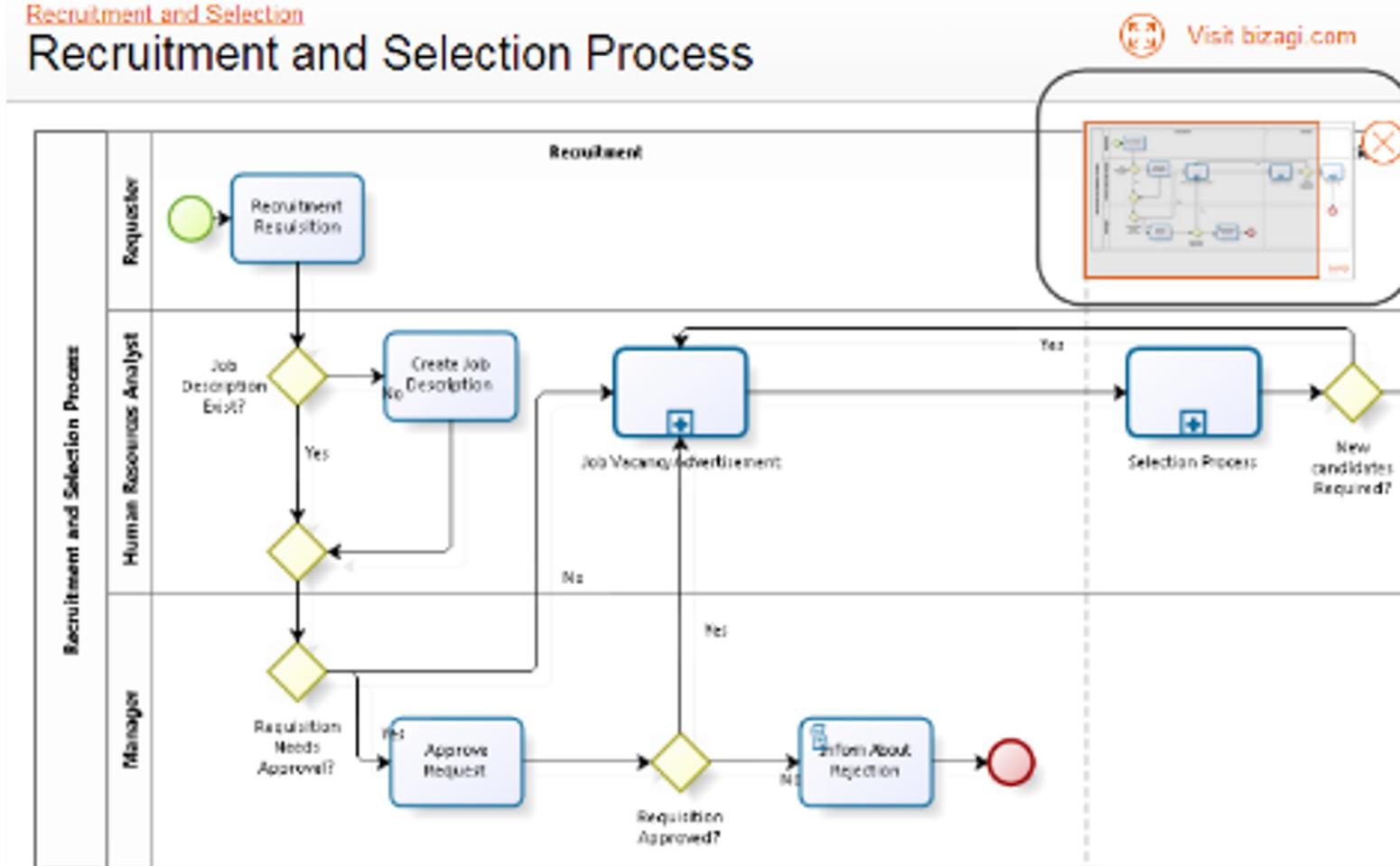
- ESB – Bus de servicios



SOA y Procesos de Negocio Automatizados

Recruitment and Selection

Recruitment and Selection Process

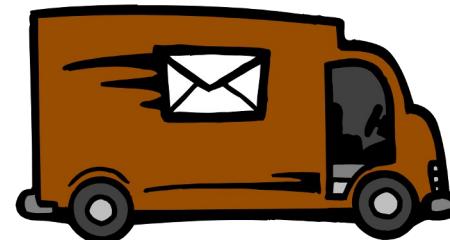


Visit bizagi.com

SOA – HTTP-based protocol:SOAP

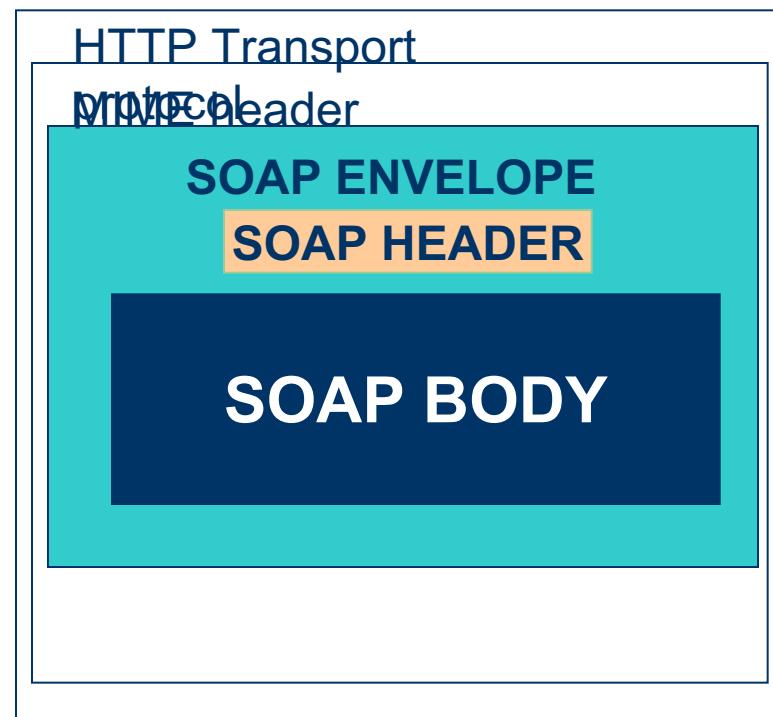
Simple Object Access Protocol

- SOAP: ‘sobre’ estándar para el envío de mensajes sobre HTTP.
- SOAP: protocolo para transferir dichos mensajes entre aplicaciones distribuidas.



SOAP: Envelope

- Mensaje SOAP:
 - An Envelope
 - A Header (optional)
 - A Body



SOAP: Envelope

```
<?xml version='1.0' encoding='UTF-
8'?>
<SOAP-ENV:Envelope xmlns:SOAP_ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsi="http://www.w3c.org/1999/XMLSchema-instance"
    xmlns:xsd="http://www.w3c.org/1999/XMLSchema">
    <SOAP-ENV:Header>
        </SOAP-ENV:Header>
        <SOAP_ENV:Body>
            </SOAP_ENV:Body>
    </SOAP-ENV:Envelope>
```

SOAP: Request envelope

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Body>
    <qk:query xmlns:qk="http://www.sosnoski.com/quakes">
      <qk:min-date>2001-07-25T19:58:24.918</qk:min-date>
      <qk:max-date>2001-09-01T16:00:59.4</qk:max-date>
      <qk:min-long>134.90671</qk:min-long>
      <qk:max-long>178.3323</qk:max-long>
      <qk:min-lat>-47.337864</qk:min-lat>
      <qk:max-lat>-18.00817</qk:max-lat>
    </qk:query>
  </SOAP:Body>
</SOAP:Envelope>
```

SOAP: response envelope

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Body>
    <qk:results xmlns:qk="http://www.sosnoski.com/quakes" count="1">
      <qk:result-set>
        <qk:area-name>Vanuatu Islands</qk:area-name>
        <qk:regions count="1">
          <qk:regions>
            <qk:region ident="rgn189" index="189">LOYALTY ISLANDS REGION</qk:region>
          </qk:regions>
        </qk:regions>
        <qk:quakes count="3">
          <qk:quakes>
            <qk:quake time="2001-08-10T15:05:12" millis="1300" latitude="-22.217" longitude="170.571" de
            <qk:quake time="2001-08-10T20:02:50" millis="4500" latitude="-22.262" longitude="170.506" de
            <qk:quake time="2001-08-30T08:12:27" millis="2400" latitude="-22.928" longitude="169.787" de
          </qk:quakes>
        </qk:quakes>
      </qk:result-set>
    </qk:results>
  </SOAP:Body>
</SOAP:Envelope>
```

Servicios Web

- Servicio remoto accesible a través de SOAP.
- Descrito por un documento WSDL.
- Sin estado (acorde con principios SOA).

Web Services - WSDL

- A WSDL document is an XML document

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions>
    <types>
        <!-- define the types here using XML Schema -->
    </types>
    <message>
        <!-- XML messages the web service uses are defined here -->
    </message>
    <portType>
        <!-- define the input and output parameters here -->
    </portType>
    <binding>
        <!-- define the network protocol here -->
    </binding>
    <service>
        <!-- location of the service -->
    </service>
</definitions>
```

REST

- Estilo arquitectónico (no es un protocolo ni una herramienta) para la interoperabilidad. No es una tecnología ni un producto.
- Manejo explícito de HTTP.

REST – Elementos

Nouns (Resources)

i.e., <http://example.com/employees/12345>



Verbs

i.e., GET

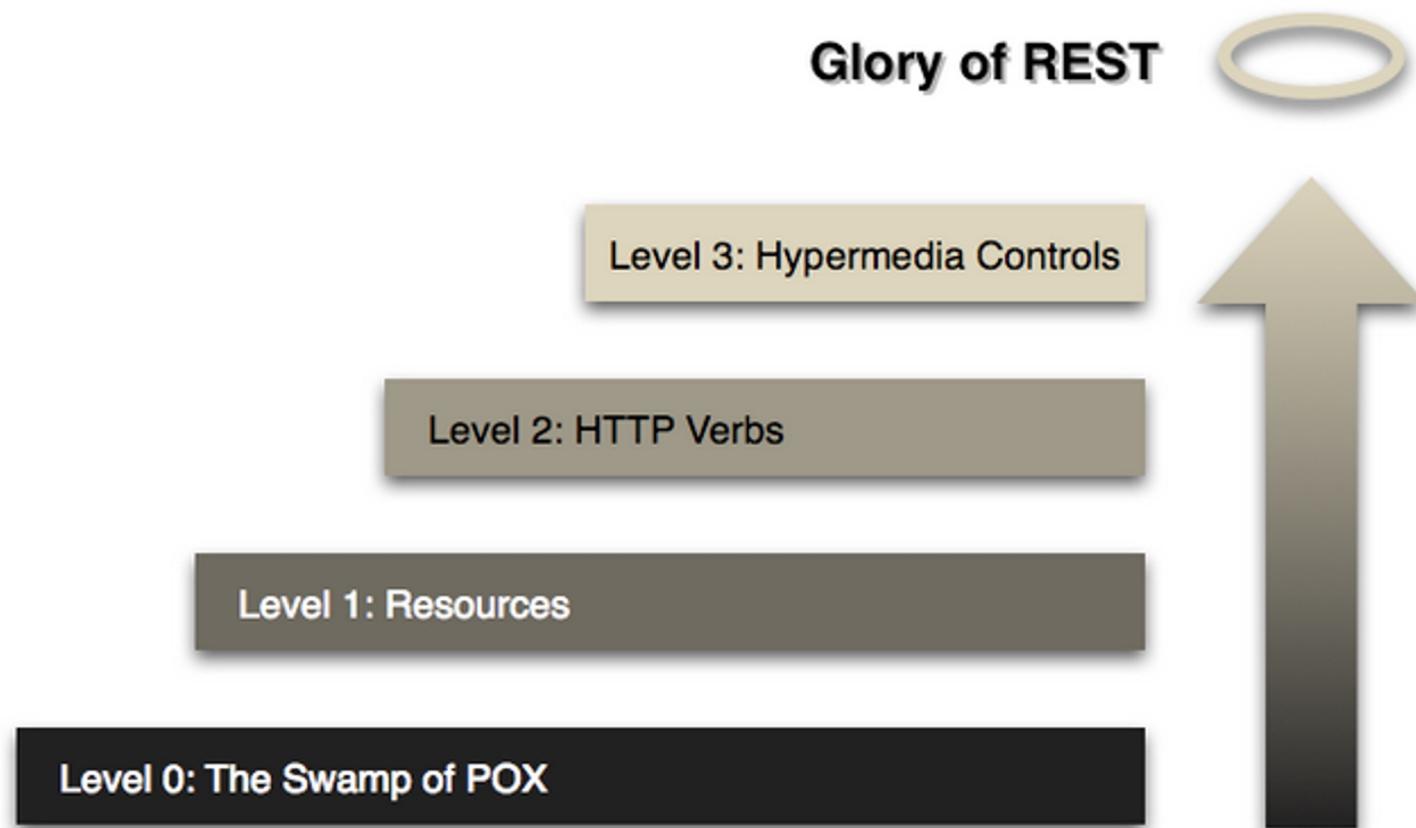
Representations

i.e., XML

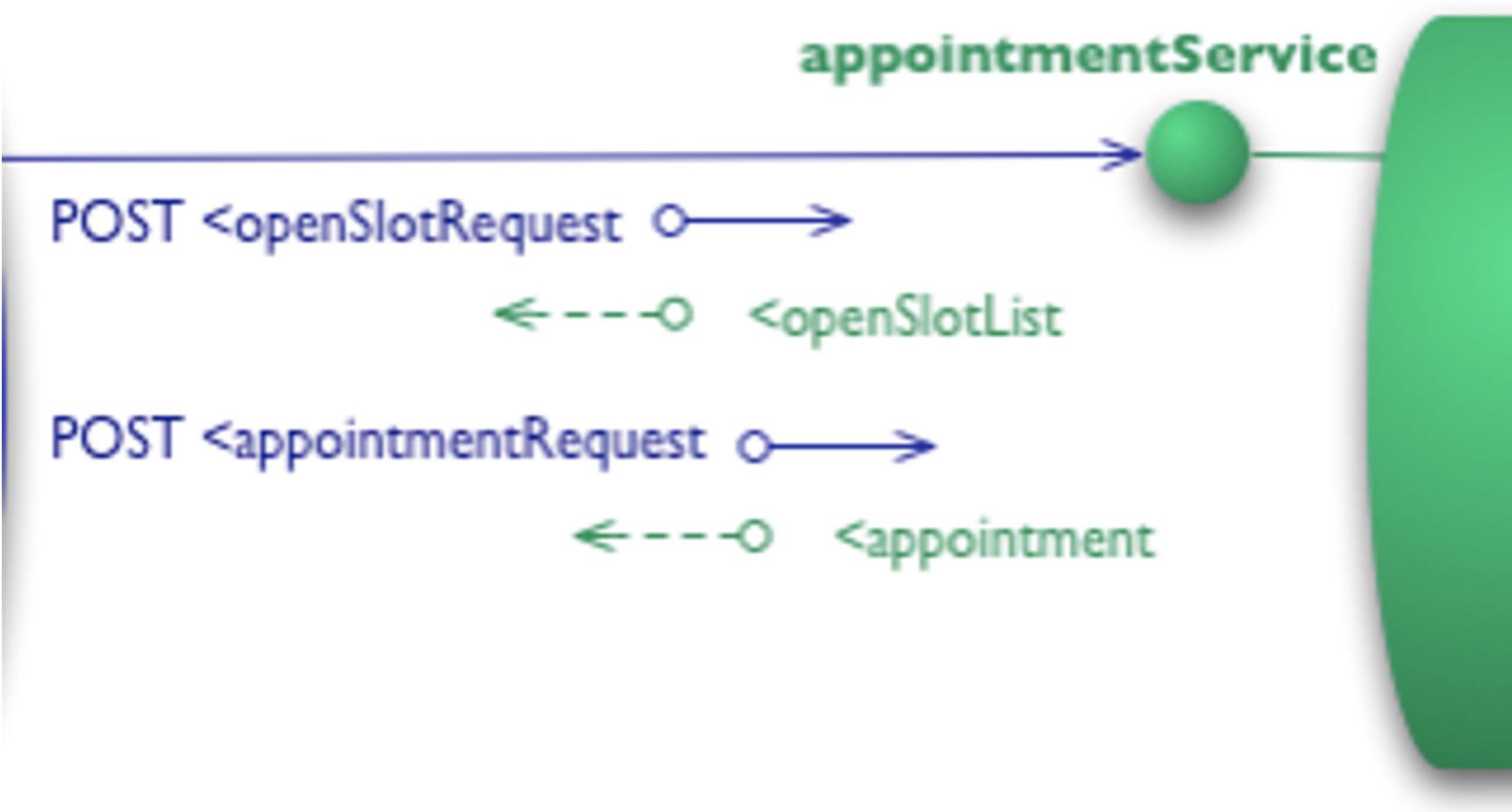
REST (Representaciones)

- XML
- JSON
- ...
- [Mime-types]

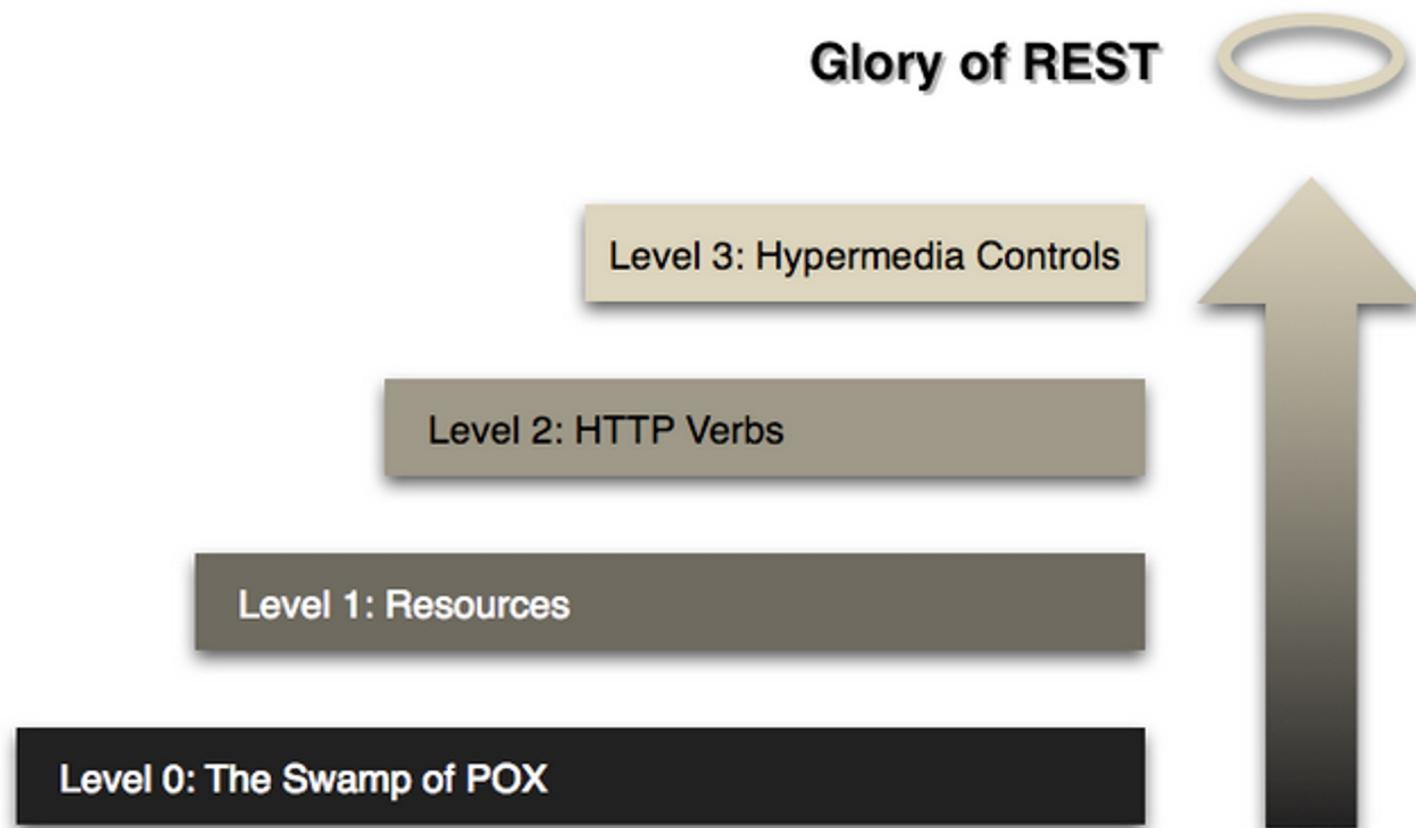
Niveles de madurez REST



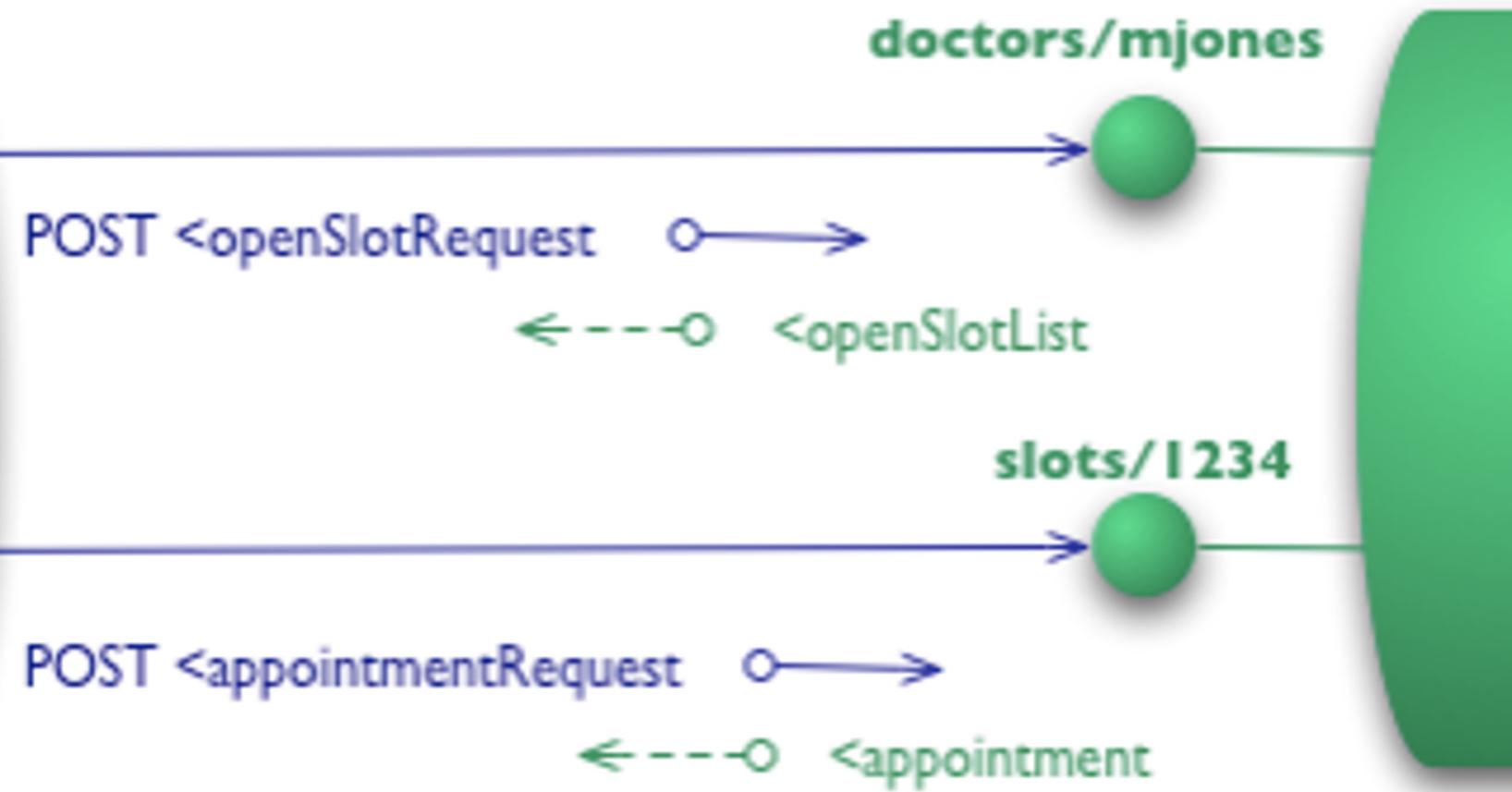
Nivel 0



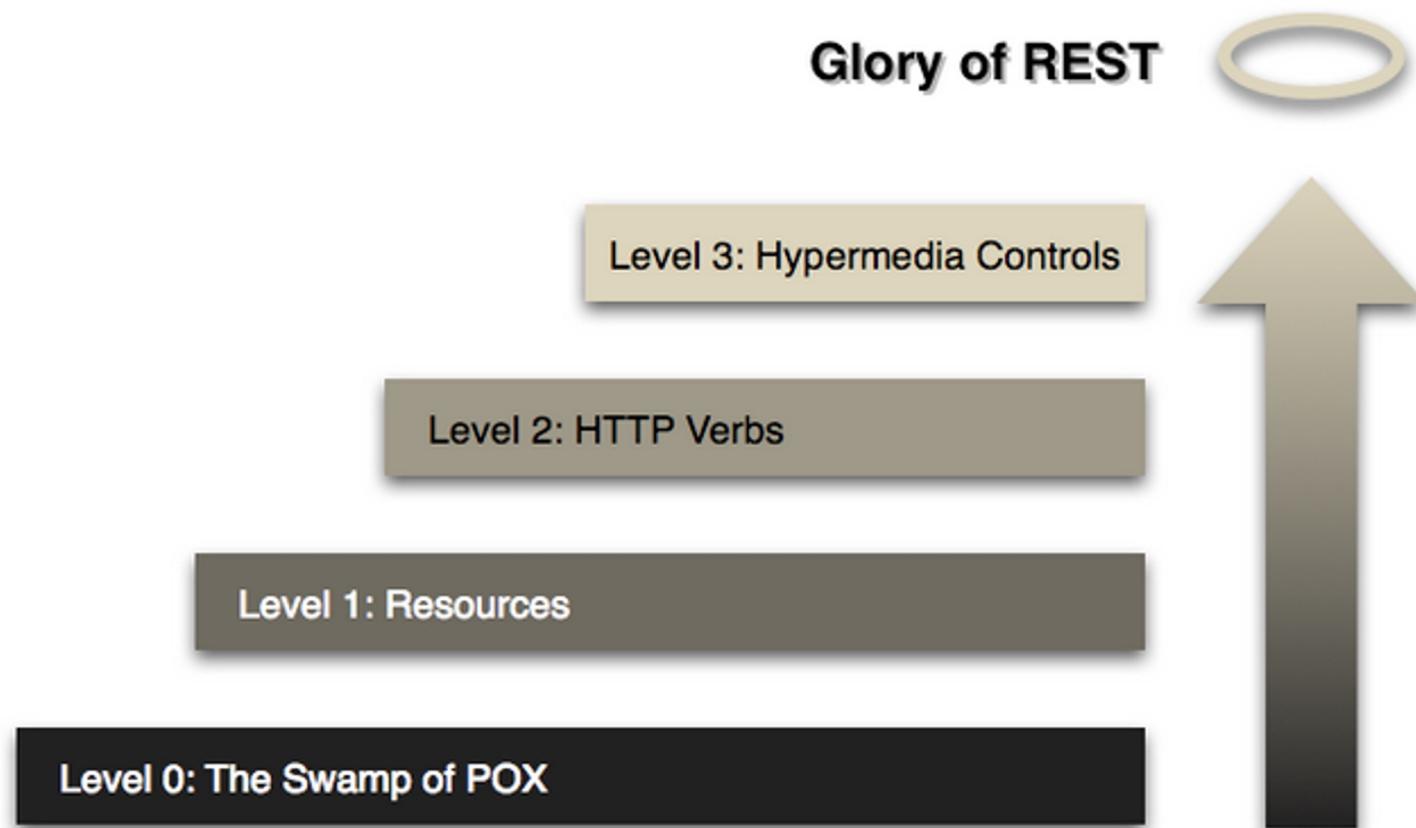
Niveles de madurez REST



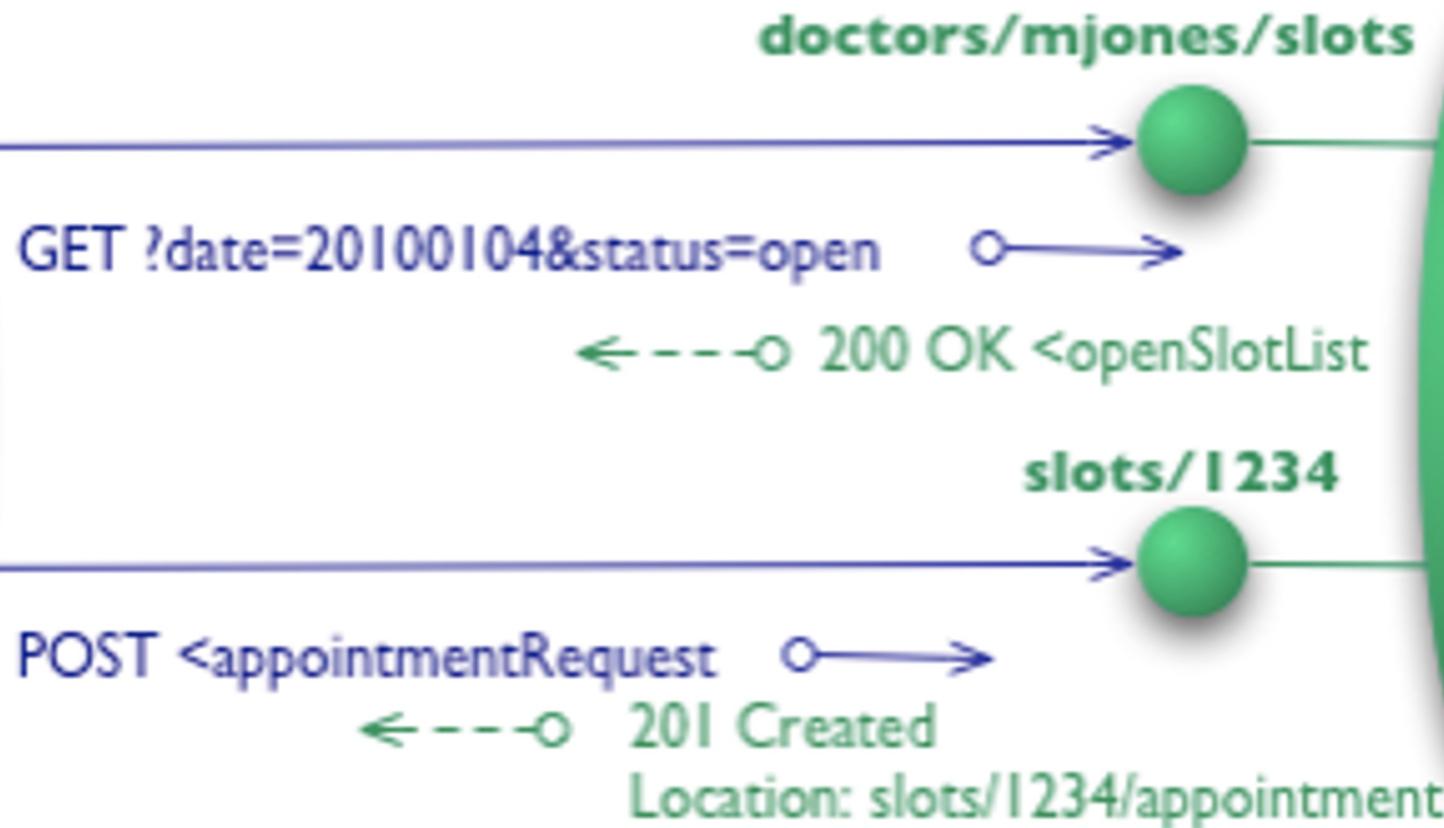
Nivel 1



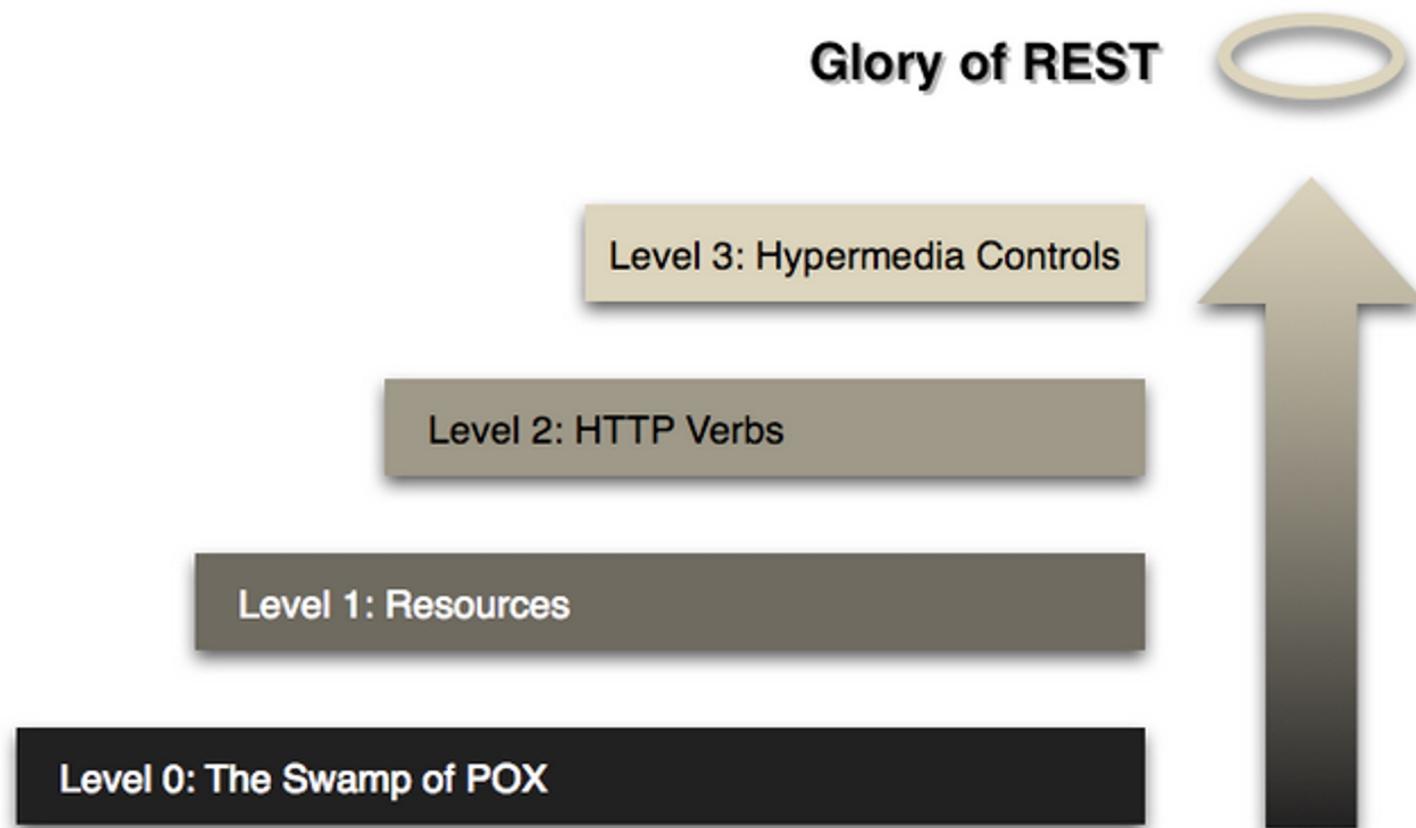
Niveles de madurez REST



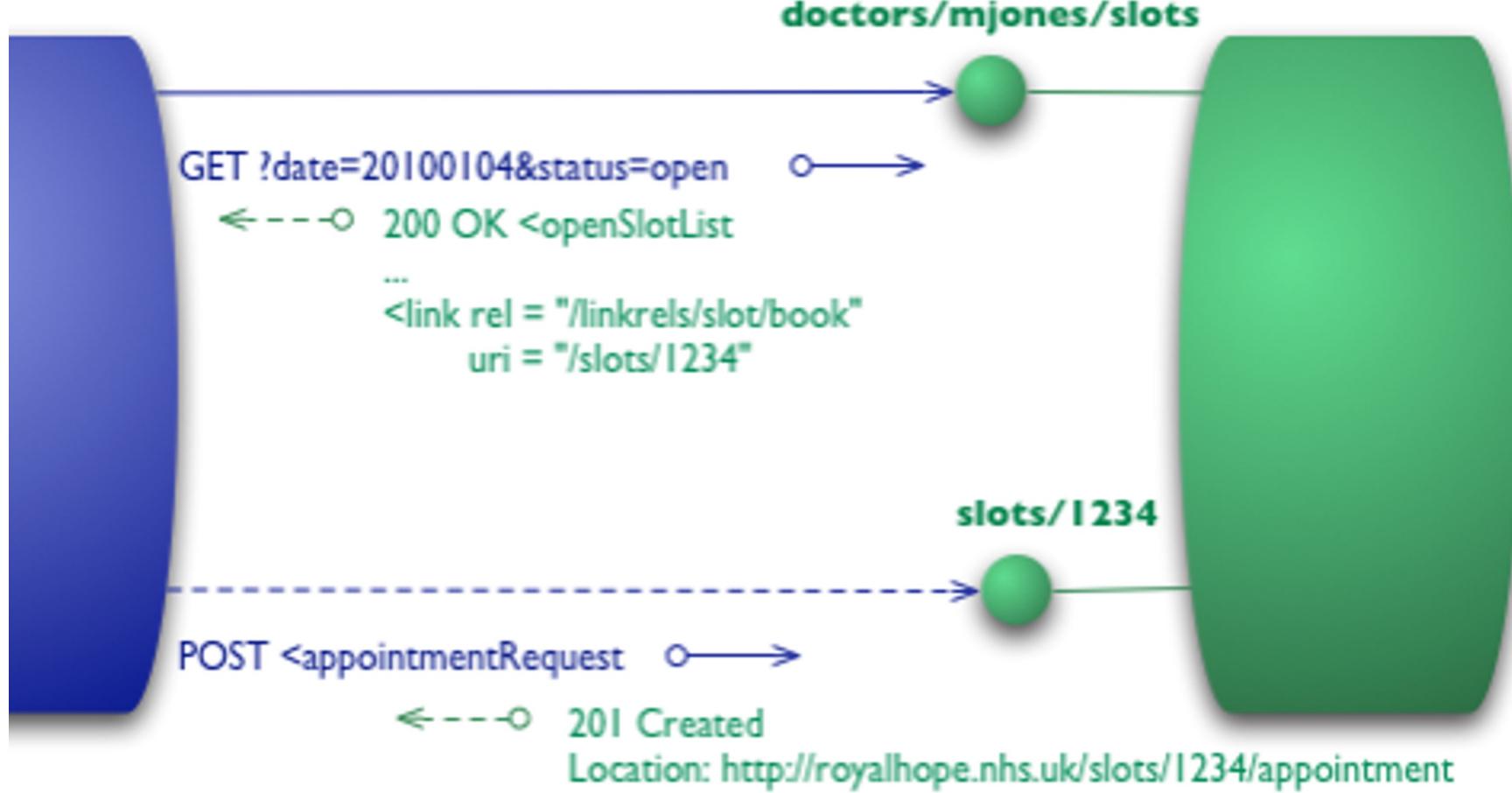
Nivel 2



Niveles de madurez REST

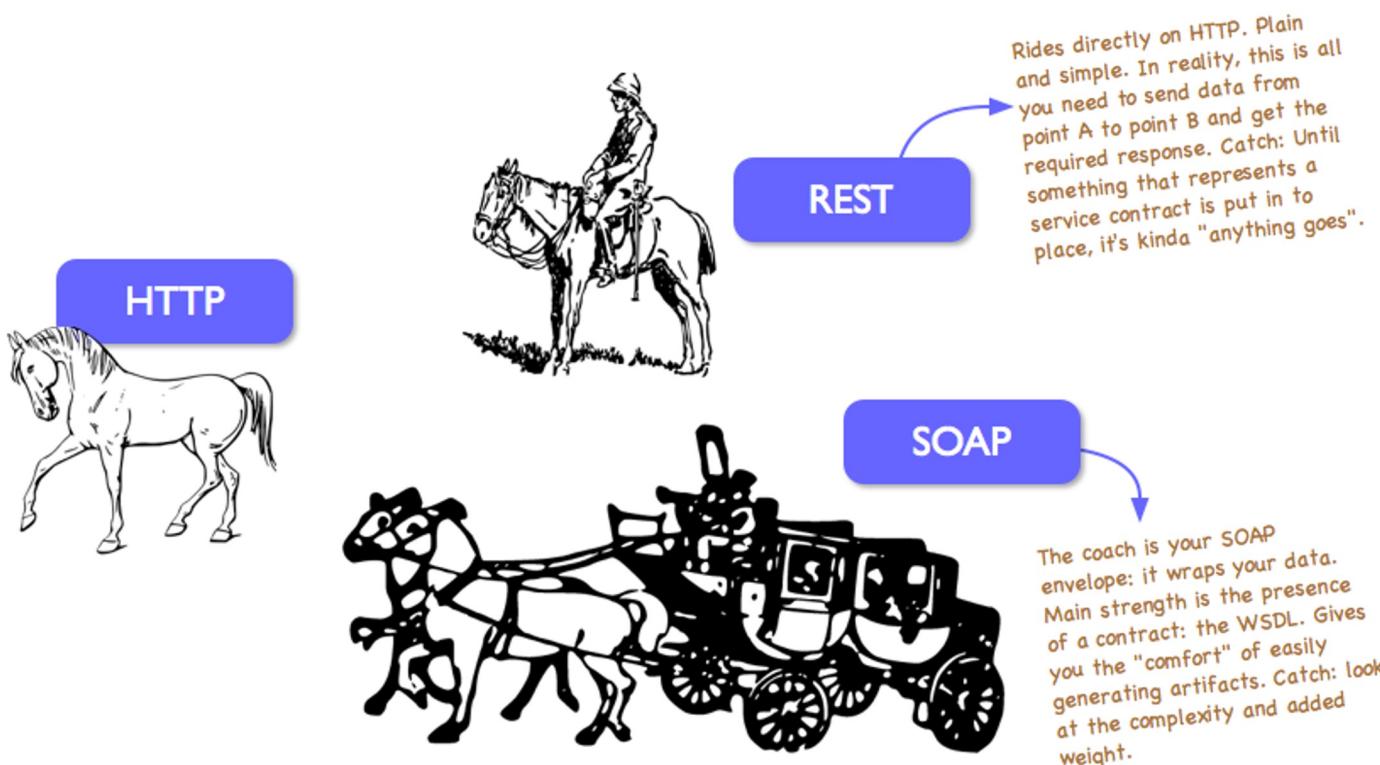


Nivel 3



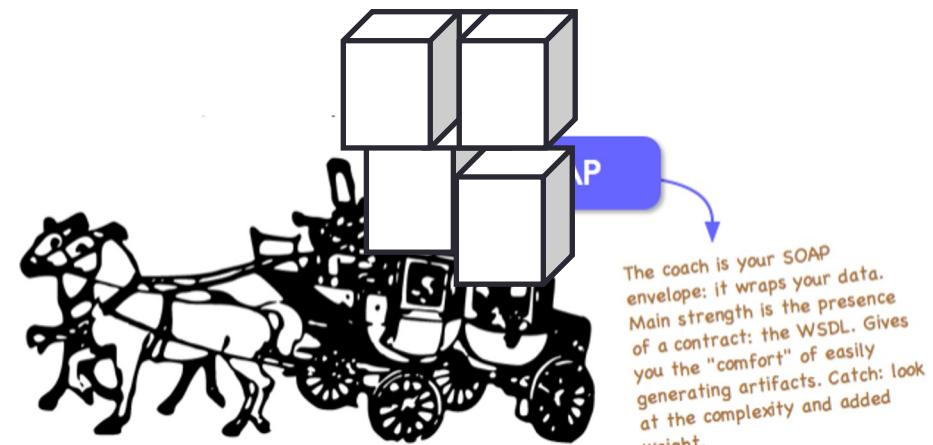
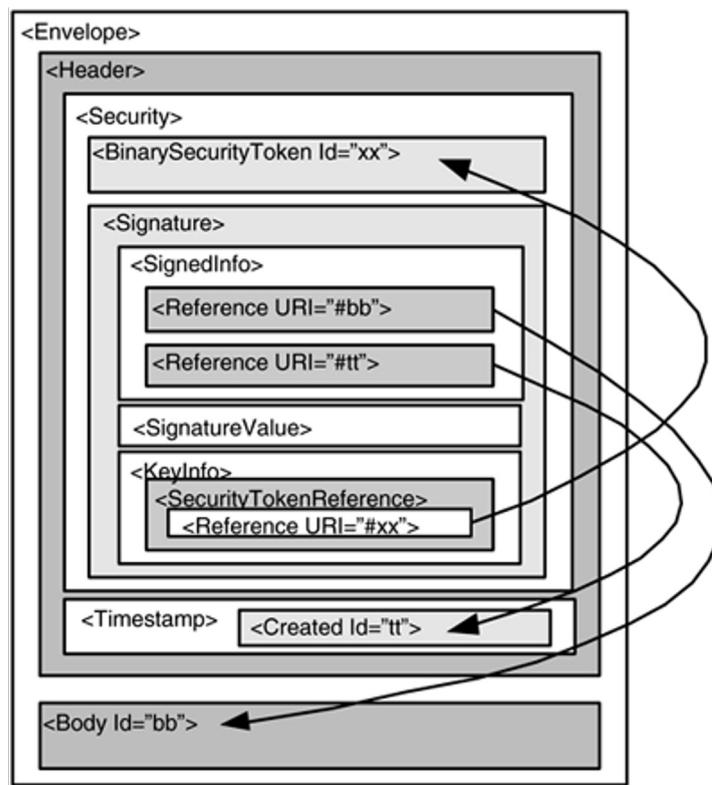
SOAP vs REST

- Tamaño de las peticiones



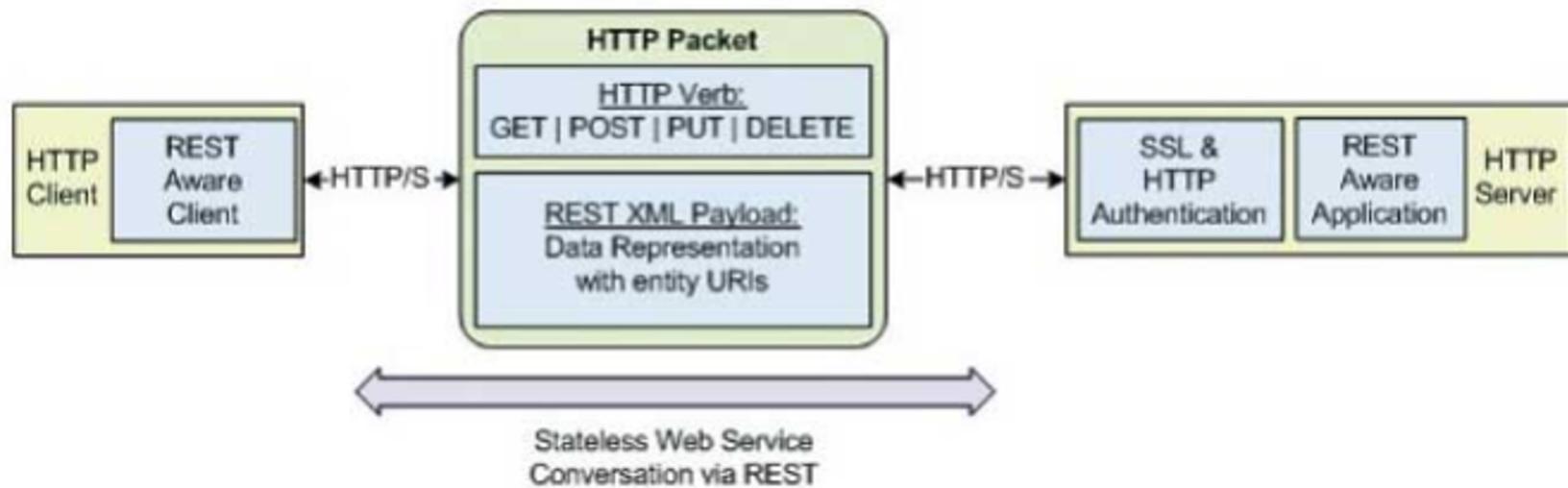
SOAP vs REST

- Seguridad. SOAP: SSL+HTTPS + WS-Security



SOAP vs REST

- Seguridad REST: SSL+HTTPS



Desarrollo de APIs REST - Python



```
from flask import abort

@app.route('/todo/api/v1.0/tasks/<int:task_id>', methods=['GET'])
def get_task(task_id):
    task = [task for task in tasks if task['id'] == task_id]
    if len(task) == 0:
        abort(404)
    return jsonify({'task': task[0]})
```

Desarrollo de APIs REST – Node.js

```
//File: controllers/tvshows.js
var mongoose = require('mongoose');
var TVShow = mongoose.model('TVShow');

//GET - Return all tvshows in the DB
exports.findAllTVShows = function(req, res) {
  TVShow.find(function(err, tvshows) {
    if(err) res.send(500, err.message);

    console.log('GET /tvshows')
    res.status(200).jsonp(tvshows);
  });
};
```



```
//GET - Return a TVShow with specified ID
exports.findById = function(req, res) {
  TVShow.findById(req.params.id, function(err, tvshow) {
    if(err) return res.send(500, err.message);

    console.log('GET /tvshow/' + req.params.id);
    res.status(200).jsonp(tvshow);
  });
};
```

Spring-MVC

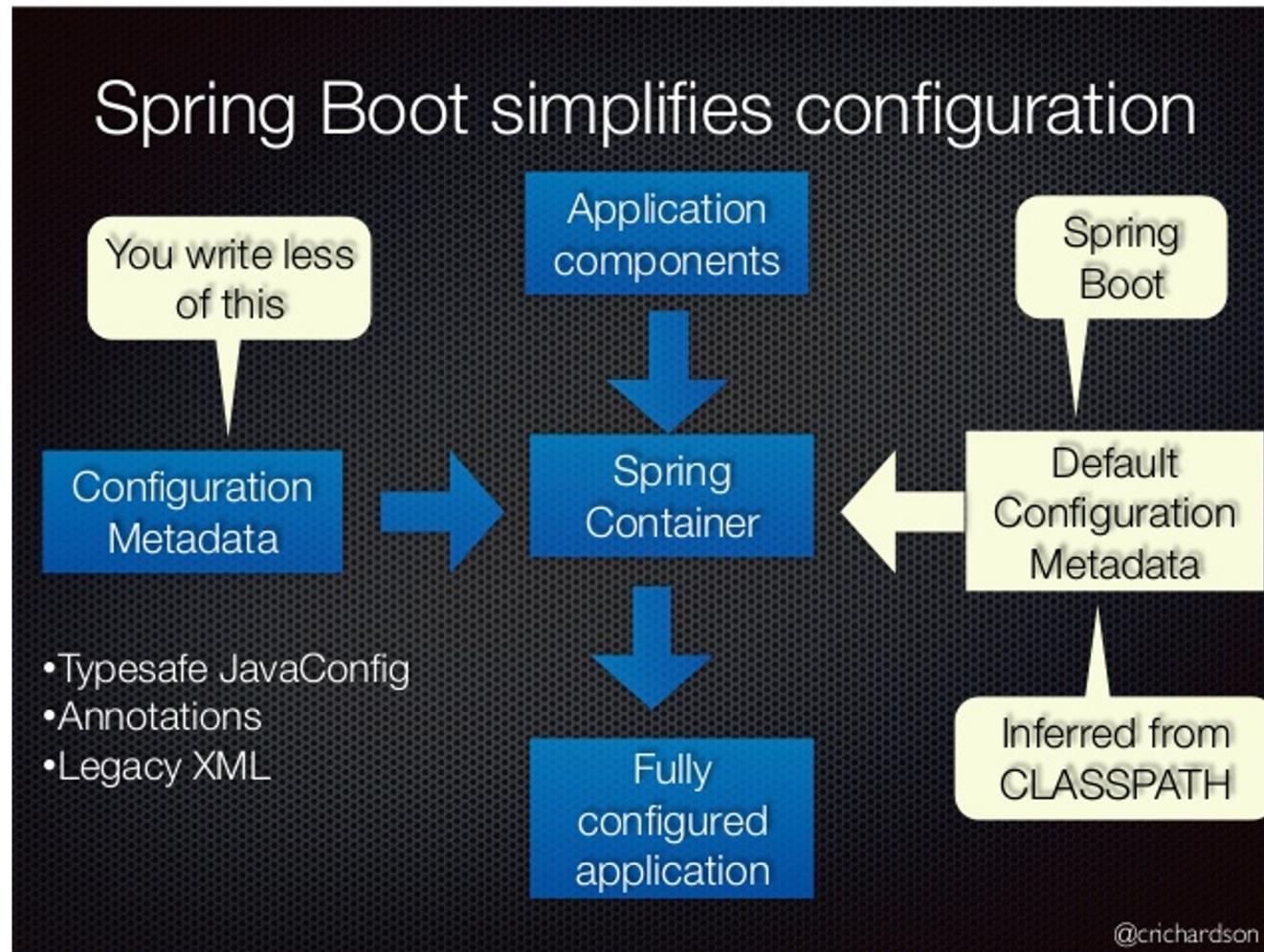


```
@RestController
public class GreetingController {

    private static final String template = "Hello, %s!";
    private final AtomicLong counter = new AtomicLong();

    @RequestMapping("/greeting")
    public Greeting greeting(@RequestParam(value="name", defaultValue="World")
        return new Greeting(counter.incrementAndGet(),
                            String.format(template, name));
    }
}
```

Spring-boot



Facilidad de descubrimiento (metadatos)

SOAP

- WSDL

```
- <wsdl:definitions name="MyCustomerTestService" targetNamespace="http://www.webservicetest.net">
  - <wsdl:types>
    - <xss:schema targetNamespace="http://www.webservicetest.net" elementFormDefault="qualified">
      + <xss:simpleType name="CustomerId"><xss:simpleType>
      + <xss:complexType name="Customer"><xss:complexType>
      + <xss:complexType name="Address"><xss:complexType>
    </xss:schema>
  </wsdl:types>
  - <wsdl:message name="getCustomerDataRequest">
    <wsdl:part name="CustomerRequest" type="tns:CustomerId"/>
  </wsdl:message>
  - <wsdl:message name="getCustomerDataResponse">
    <wsdl:part name="CustomerResponse" type="tns:Customer"/>
  </wsdl:message>
  - <wsdl:portType name="CustomerData">
    - <wsdl:operation name="getCustomerData">
      <wsdl:input message="tns:getCustomerDataRequest"/>
      <wsdl:output message="tns:getCustomerDataResponse"/>
    </wsdl:operation>
  </wsdl:portType>
  - <wsdl:binding name="CustomerSOAPBinding" type="tns:CustomerData">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    - <wsdl:operation name="getCustomerData">
      <soap:operation soapAction="" style="document"/>
      - <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      - <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  - <wsdl:service name="CustomerService">
    - <wsdl:port name="CustomerSOAPPort" binding="tns:CustomerSOAPBinding">
      <soap:address location="No Target Address"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

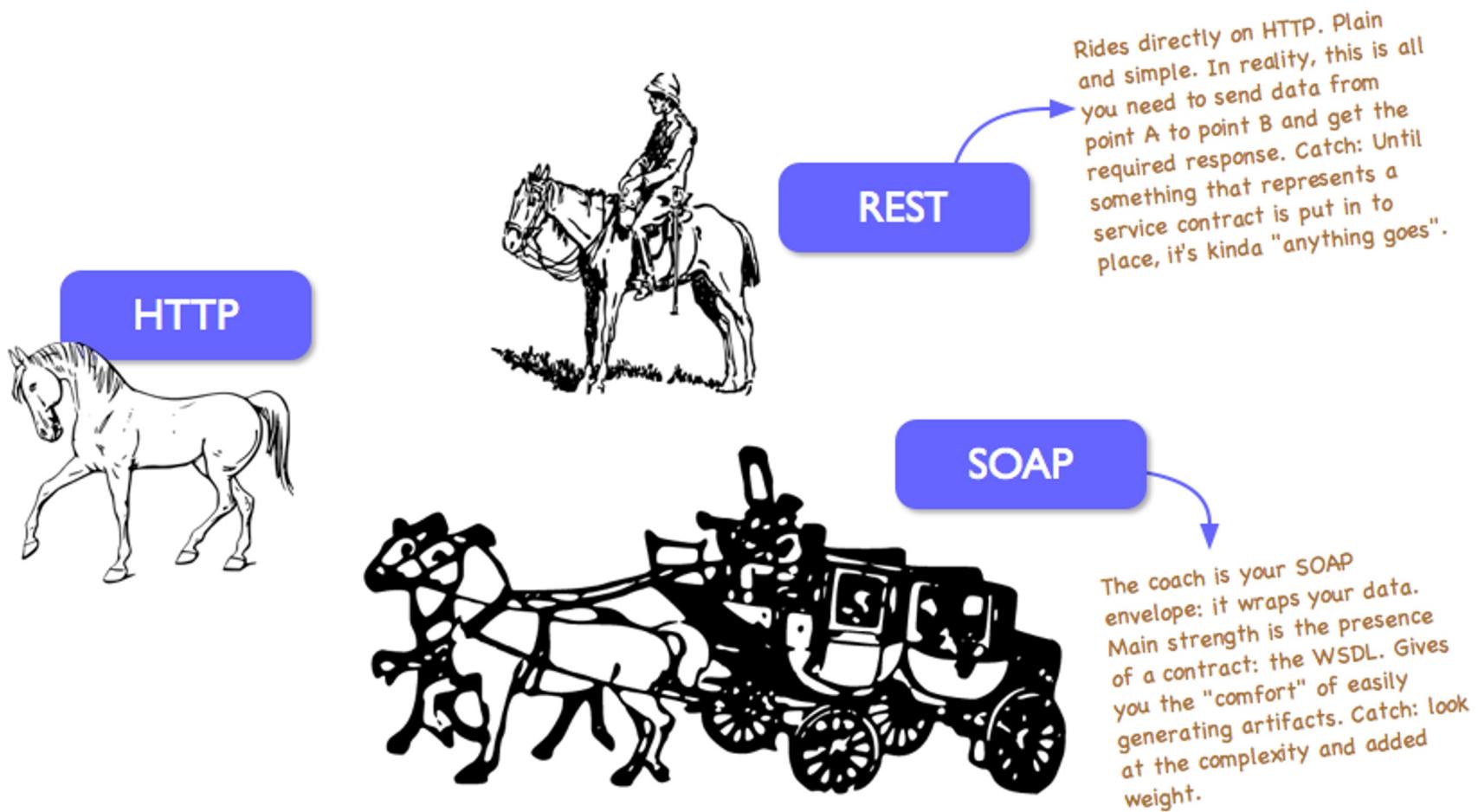


REST

- RAML

```
#%RAML 1.0
title: New API
mediaType: [ application/json, application/xml ]
types:
  Person:
  Another:
/list:
  get:
    responses:
      200:
        body: Person[]
/send:
  post:
    body:
      application/json:
        type: Another
```

Compatibilidad con estándares



Mejores prácticas – APIs REST

- Recursos, no funciones (Sustantivos, no verbos)
- Las peticiones GET NUNCA deben alterar estados.
- Recursos (sustantivos) en plural.
- Usar sub-recursos a manera de relaciones.
- Parámetros GET para: filtrado, ordenamiento, selección de campos y paginación.
- Versionamiento de APIs.
- Manejar errores con códigos HTTP.
- HATEOAS - Hypermedia as the Engine of Application State