

# Generación de servicios en red

Programación de servicios y procesos

Enero/Febrero 2020 (gmq.psp2019@gmail.com)

# Generación de servicios en red

- **Servicios en red**
- Programación de aplicaciones cliente-servidor
- Protocolos a nivel de aplicación
- Librerías de clases y componentes
- Procedimientos remotos
- Javadoc

# Servicios en red

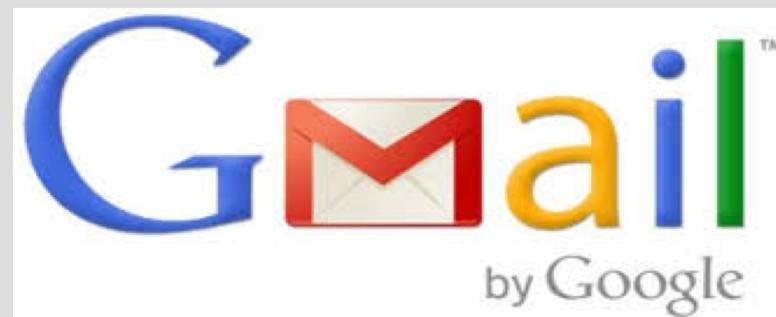
- Servicio
  - Palabra polisémica
  - Un servicio software es aquel capaz de ofrecer una funcionalidad concreta y definida
  - Los servicios en red son aquellos que utilizan una red de comunicaciones para interactuar o para transmitir información
  - Hay muchos servicios en red muy diferentes entre ellos

# Servicios en red

- Ejemplos de servicios en red
  - Netflix: micro-servicios
  - Whatsapp
  - AWS
  - Gmail
    - AntiSpam
    - Taggeado
    - Escritura predictiva
    - Lectura de mensaje
    - Adjuntar mensajes

# Servicios en red

- Servicio
  - Gmail
    - Escritura predictiva
    - Identificador de idioma
    - Respuesta inmediata
    - Buscador de expresiones habituales

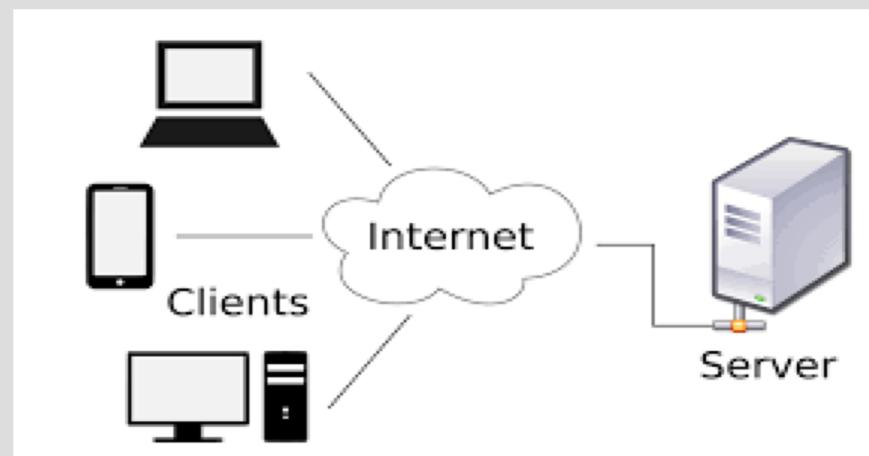


# Generación de servicios en red

- Servicios en red
- **Programación de aplicaciones cliente-servidor**
- Protocolos a nivel de aplicación
- Librerías de clases y componentes
- Procedimientos remotos
- Javadoc

# Programación de aplicaciones cliente/servidor

- Cuando se diseña un sistema con la arquitectura Cliente/Servidor se deben definir:
  - Funcionalidad del servidor
  - Tecnología de las comunicaciones
  - Protocolo de nivel de aplicación



# Programación de aplicaciones cliente/servidor

- Funcionalidad del servidor
  - ¿Qué realiza el sistema que toma el rol de servidor?
  - Mensajes/request
    - Cambian en número y formato según funcionalidad
  - Sockets: servidor de tiempo
  - Adaptables a cualquier ámbito o tipo de negocio

# Programación de aplicaciones cliente/servidor

- Funcionalidad del servidor
  - ¿Qué realiza el sistema que toma el rol de servidor?
  - Mensajes/request
    - Cambian en número y formato según funcionalidad
  - Sockets: servidor de tiempo
  - Adaptables a cualquier ámbito o tipo de negocio

# Programación de aplicaciones cliente/servidor

- Tecnología de comunicaciones
  - ¿Sockets?
  - ¿Qué librerías se utiliza?
  - ¿Reintentos o circuit breaker?
  - ¿TTL?
  - ¿Es necesaria la comunicación sobre una capa segura TLS?

# Programación de aplicaciones cliente/servidor

- Protocolo nivel de aplicación
  - Se encuentra en la parte alta de la pila de protocolos OSI
  - Se elegirá un protocolo lo más adecuado posible a nuestra arquitectura
    - Peticiones REST: HTTPS
    - Envío de mensajes a un bróker RabbitMQ: AMQP
  - Cualquier tipo de información se debe poder enviar independiente del protocolo utilizado

# Programación de aplicaciones cliente/servidor

- Protocolo nivel de aplicación
  - Con estado (Statefull)
    - Las peticiones anteriores condicionan la petición actual
    - Concepto de sesión
  - Sin estado (Stateless)
    - Cada petición es independiente de las que se han hecho anteriormente

# Generación de servicios en red

- Servicios en red
- Programación de aplicaciones cliente-servidor
- **Protocolos a nivel de aplicación**
- Librerías de clases y componentes
- Procedimientos remotos
- Javadoc

# Protocolos de aplicación

- La capa de aplicación: define las aplicaciones de red y los servicios de Internet estándar que puede utilizar un usuario
- Posibilidad de acceder a los servicios de las demás capas
- Define los protocolos que utilizan las aplicaciones para intercambiar datos, como correo electrónico (POP y SMTP), gestores de bases de datos y protocolos de transferencia de archivos (FTP)

# Protocolos de aplicación

## Protocolos:

- FTP
- DNS
- DHCP
- HTTP (S)
- HTTPS
- POP / SMTP
- SSH
- TELNET
- LDAP



# Protocolos de aplicación

- FTP (File Transfer Protocol)
- DNS (Domain Name System - Sistema de nombres de dominio).
- DHCP (Dynamic Host Configuration Protocol - Protocolo de configuración dinámica de anfitrión).
- HTTP (HyperText Transfer Protocol) para acceso a páginas web.
- HTTPS (Hypertext Transfer Protocol Secure) Protocolo seguro de transferencia de hipertexto.
- POP (Post Office Protocol) para recuperación de [correo electrónico].
- SMTP (Simple Mail Transport Protocol) para envío de correo electrónico.
- SSH (Secure SHell)
- TELNET para acceder a equipos remotos.
- LDAP (Lightweight Directory Access Protocol).

# Protocolos de aplicación

- TelNet (Teletype Network)
  - Permite acceder a otra máquina para manejarla remotamente como si estuviéramos sentados delante de ella
  - Puerto 23
  - Acceso en modo terminal
  - Sin capa de seguridad
  - Habitual en sistemas UNIX (varios clientes conectados a un server)
  - Cliente TelNet: Putty
  - Comando: *telnet ip*

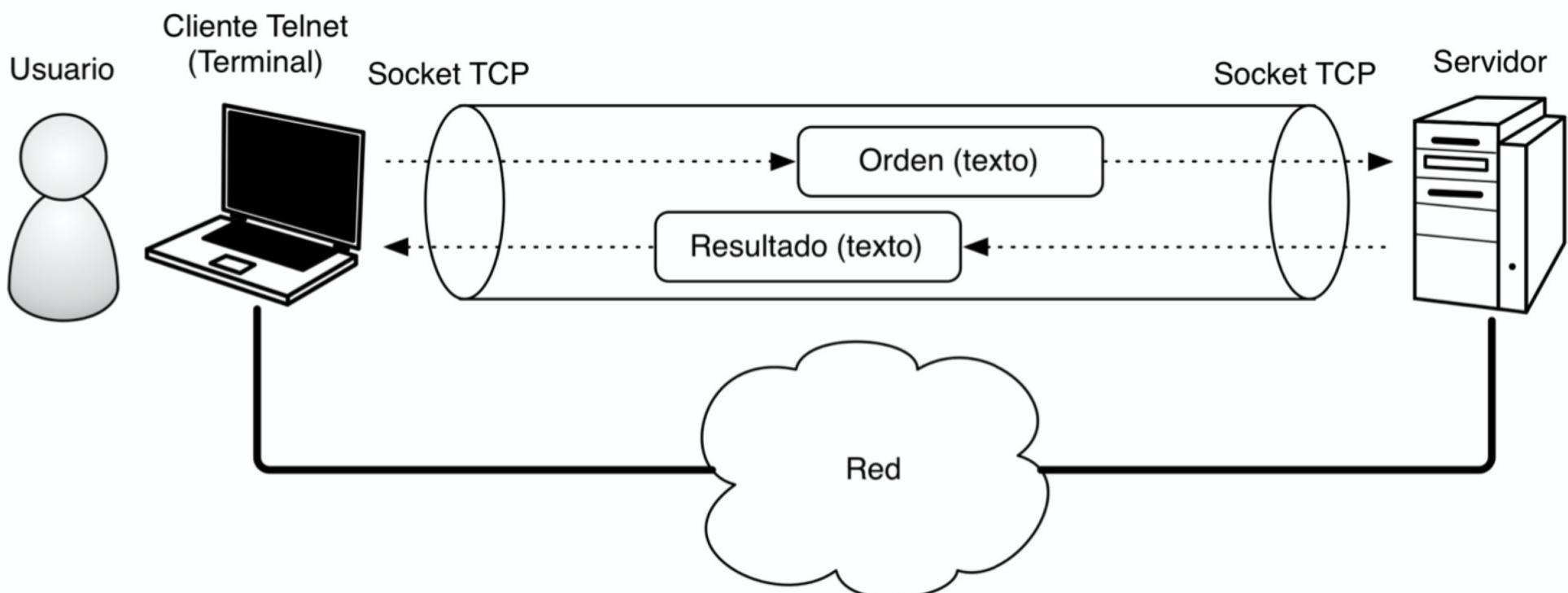
# Protocolos de aplicación

- TelNet
  - Desventajas
    - Vulnerable
    - Sin capa de seguridad
    - Sin modo de autenticación seguro
    - Sesiones no cifradas



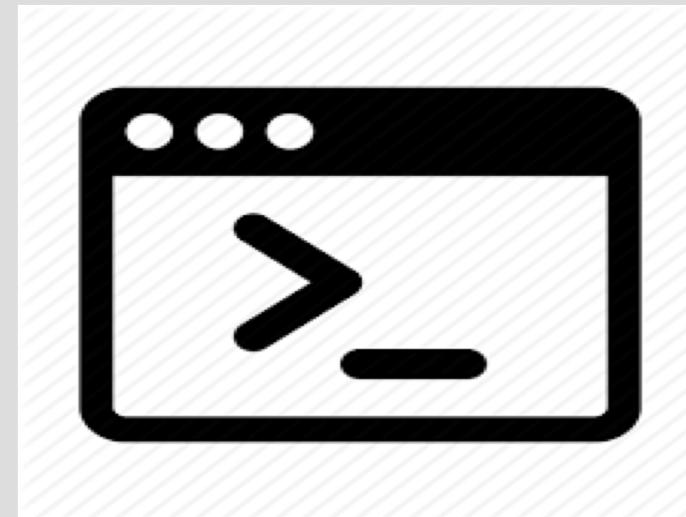
# Protocolos de aplicación

- TelNet



# Protocolos de aplicación

- SSH (Secure Shell)
  - Es el acceso remoto a un servidor por medio de un canal seguro en el que toda la información está cifrada.
  - Parecido a Telnet pero con capa de seguridad
  - Puerto 22

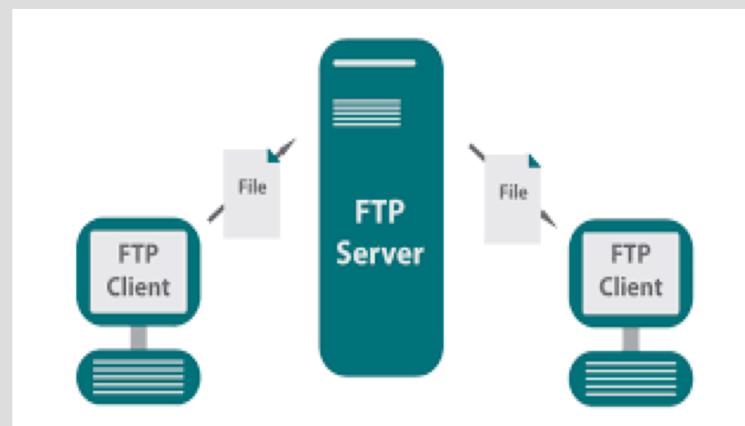


# Protocolos de aplicación

- FTP (File Transfer Protocol)
  - Desde un equipo cliente se puede conectar a un servidor para descargar archivos desde él o para enviarle archivos, independientemente del sistema operativo utilizado en cada equipo
  - Pensado para ofrecer la máxima velocidad en la conexión
  - Problemas de seguridad
    - Sin ningún tipo de cifrado, con lo que un posible atacante puede capturar este tráfico
  - Puerto 20-21

# Protocolos de aplicación

- FTP (File Transfer Protocol)

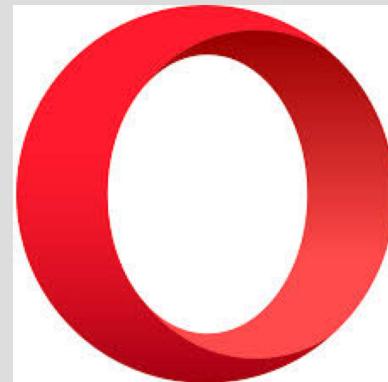


- Cliente: Filezilla



# Protocolos de aplicación

- HTTP(S) (Hyper Text Transfer Protocol)
  - El gran protocolo de internet, el más usado. Transferencia de información en la World Wide Web
  - Protocolo sin estado
  - Sesión a través de las cookies (información que un servidor puede almacenar en el sistema cliente)



# Protocolos de aplicación

- HTTP(S) (Hyper Text Transfer Protocol)
  - 1991: versión 0.9
    - Soporta solo un comando, GET
  - Mayo 1996: HTTP/1.0
    - Usado en proxies
  - Junio 1999: HTTP/1.1
    - Versión más usada actualmente
  - Febrero 2000: HTTP/1.2
  - Mayo 2015: HTTP/2
    - Enfocado a mejorar el empaquetado de datos
  - Diciembre 2019: borrador HTTP/3

# Protocolos de aplicación

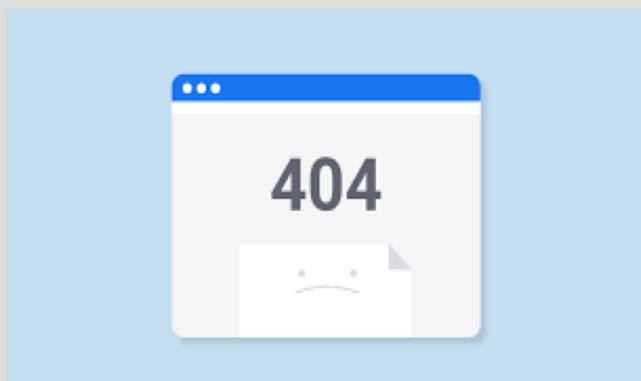
- HTTP(S) (Hyper Text Transfer Protocol)
  - Métodos
    - POST
    - GET
    - PUT
    - PATCH
    - DELETE
    - HEAD

# Protocolos de aplicación

- HTTP(S) (Hyper Text Transfer Protocol)
  - Respuestas
    - 1xx: Respuestas informativas. Indica que la petición ha sido recibida y se está procesando.
    - 2xx: Respuestas correctas. Indica que la petición ha sido procesada correctamente.
    - 3xx: Respuestas de redirección. Indica que el cliente necesita realizar más acciones para finalizar la petición.

# Protocolos de aplicación

- HTTP(S) (Hyper Text Transfer Protocol)
  - Respuestas
    - 4xx: Errores causados por el cliente. Indica que ha habido un error en el procesado de la petición a causa de que el cliente ha hecho algo mal.
    - 5xx: Errores causados por el servidor. Indica que ha habido un error en el procesado de la petición a causa de un fallo en el servidor.



# Protocolos de aplicación

- HTTP(S) (Hyper Text Transfer Protocol)
  - Cabeceras
    - Metadatos que se envían en las peticiones o respuesta HTTP para proporcionar información esencial sobre la transacción en curso

method	path	protocol
GET	/tutorials/other/top-20-mysql-best-practices/	HTTP/1.1
<pre>Host: net.tutsplus.com User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US; rv:1.9.1 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q= Accept-Language: en-us,en;q=0.5 Accept-Encoding: gzip,deflate Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7 Keep-Alive: 300 Connection: keep-alive Cookie: PHPSESSID=r2t5uvjq435r4q7ib3vtdjq120 Pragma: no-cache Cache-Control: no-cache</pre>		

**HTTP headers as Name: Value**

# Protocolos de aplicación

- HTTP(S) (Hyper Text Transfer Protocol)
  - Podemos clasificar las cabeceras según su función.
    - Capacidades aceptadas por el que envía el mensaje:
      - Accept (indica el MIME aceptado),
      - Accept-Charset (indica el código de caracteres aceptado),
      - Accept-Encoding (indica el método de compresión aceptado),
      - Accept-Language (indica el idioma aceptado),
      - User-Agent (para describir al cliente),
      - Server (indica el tipo de servidor),
      - Allow (métodos permitidos para el recurso)

# Protocolos de aplicación

- HTTP(S) (Hyper Text Transfer Protocol)
  - Podemos clasificar las cabeceras según su función.
    - Cabeceras que describen el contenido:
      - Content-Type (indica el MIME del contenido)
      - Content-Length (longitud del mensaje)
      - Content-Range
      - Content-Encoding
      - Content-Language
      - Content-Location.
    - Cabeceras que hacen referencias a URLs:
      - Location (indica donde está el contenido),
      - Referer (Indica el origen de la petición).

# Protocolos de aplicación

- HTTP(S) (Hyper Text Transfer Protocol)
  - Podemos clasificar las cabeceras según su función.
    - Cabeceras que permiten ahorrar transmisiones:
      - Date (fecha de creación),
      - If-Modified-Since, If-Unmodified-Since,
      - If-Match, If-None-Match,
      - Expires, Last-Modified,
      - Cache-Control, Via,
      - Age, Retry-After.
    - Cabeceras para control de cookies:
      - Set-Cookie, Cookie
    - Cabeceras para autentificación:
      - Authorization, WWW-Authenticate

# Protocolos de aplicación

- POP (Post Office Protocol)
  - Obtiene los mensajes de correo electrónico almacenados en un servidor remoto, denominado Servidor POP.
  - Diseñado para recibir correo, que en algunos casos no es para enviarlo
  - Puerto 110
  - No se tienen que enviar tantas órdenes para la comunicación entre ellos. Funciona adecuadamente si no se utiliza una conexión constante a Internet o a la red que contiene el servidor de correo.

# Protocolos de aplicación

- POP (Post Office Protocol)
  - Un cliente que utiliza POP3:
    - se conecta al servidor
    - obtiene todos los mensajes
    - los almacena en la computadora del usuario como mensajes nuevos
    - los elimina del servidor y
    - finalmente se desconecta

# Protocolos de aplicación

- SMTP (Simple Mail Tranfer Protocol)
  - Intercambio de mensajes de correo electrónico entre computadoras u otros dispositivos (PDA, teléfonos móviles, impresoras, etc).
  - Limitaciones en cuanto a la recepción de mensajes en el servidor de destino
  - Puerto 587

# Protocolos de aplicación

- DNS (Domain Name Service)
  - Sistema de nomenclatura jerárquico descentralizado para dispositivos conectados a redes IP como Internet o una red privada.
  - DNS utiliza una base de datos distribuida y jerárquica que almacena información asociada a nombres de dominio en redes como Internet.
  - Los usuarios generalmente no se comunican directamente con el servidor DNS

# Protocolos de aplicación

- DNS (Domain Name Service)
  - La petición se envía al servidor DNS local del sistema operativo.
  - El sistema operativo comprueba si la respuesta se encuentra en la memoria caché
  - Si no se encuentra, se envía a servidores DNS
  - Se pueden utilizar servidores propios de su ISP, usar un servicio gratuito o contratar un servicio avanzado de pago

# Protocolos de aplicación

- DNS (Domain Name Service)
  - Google Public DNS
    - Servicio DNS gratuito ofertado por Google
    - Direcciones IPv4: 8.8.8.8 y 8.8.4.4
    - Direcciones IPv6: 2001:4860:4860::8888 y 2001:4860:4860::8844
    - Google prioriza el rendimiento y la seguridad en el servicio. Guarda la dirección IP del usuario (eliminada pasadas las 24 horas), su ISP y la información de localización de las mismas (mantenida permanentemente).
    - Desde enero de 2019 ofrece DNS mediante TLS.

# Protocolos de aplicación

- DHCP (Dynamic Host Configuration Protocol)
  - Asigna dinámicamente una dirección IP y otros parámetros de configuración de red a cada dispositivo en una red para que puedan comunicarse con otras redes IP
  - Este servidor posee una lista de direcciones IP dinámicas y las va asignando a los clientes conforme estas van quedando libres, sabiendo en todo momento quién ha estado en posesión de esa IP, cuánto tiempo la ha tenido y a quién se la ha asignado después

# Protocolos de aplicación

- DHCP (Dynamic Host Configuration Protocol)
  - Métodos de asignación de direcciones IP:
    - Asignación manual o estática :Asigna una dirección IP a una máquina determinada.
    - Asignación automática :
      - Asigna una dirección IP a una máquina cliente la primera vez que hace la solicitud al servidor DHCP y hasta que el cliente la libera. Se suele utilizar cuando el número de clientes no varía demasiado.
      - Asignación dinámica
        - El único método que permite la reutilización dinámica de las direcciones IP.

# Protocolos de aplicación

- TLS (Transport Layer Security)
  - Evolución de SSL
  - Protocolos criptográficos, que proporcionan comunicaciones seguras por una red,
  - Se usan certificados X.509 y por lo tanto criptografía asimétrica para autenticar a la contraparte con quien se están comunicando
    - Clave pública

# Generación de servicios en red

- Servicios en red
- Programación de aplicaciones cliente-servidor
- Protocolos a nivel de aplicación
- **Librerías de clases y componentes**
- Procedimientos remotos
- Javadoc

# Librerías de clases y componentes

- En Java hay desarrolladas librerías que nos ayudan a trabajar con los servicios en red.
- El tráfico más habitual es el http/http
  - OkHttp
  - RestTemplate
  - Netty
- FTP
  - Apache Commons Net
- SMTP
  - JavaMail
- Telnet
  - Apache Commons Net

# Generación de servicios en red

- Servicios en red
- Programación de aplicaciones cliente-servidor
- Protocolos a nivel de aplicación
- Librerías de clases y componentes
- **Procedimientos remotos**
- Javadoc

# Procedimientos remotos

- RPC (Remote Procedure Call)
  - programa que utiliza una computadora para ejecutar código en otra máquina remota sin tener que preocuparse por las comunicaciones entre ambas
  - Gran avance sobre los sockets de Internet usados hasta el momento. De esta manera el programador no tenía que estar pendiente de las comunicaciones, estando estas encapsuladas dentro de las RPC.
  - Las RPC son muy utilizadas dentro de la comunicación cliente-servidor.
  - El cliente el inicia el proceso solicitando al servidor que ejecute cierto procedimiento o función y enviando este de vuelta el resultado de dicha operación al cliente.

# Procedimientos remotos

- RPC (Remote Procedure Call)
  - Se utiliza XML como lenguaje para definir el lenguaje de definición de la interfaz (IDL) y HTTP como protocolo de aplicación, dando lugar a lo que se conoce como servicios web.
  - Ejemplos de estos pueden ser SOAP o XML-RPC.
  - Java Remote Method Invocation
    - mecanismo ofrecido por Java para invocar un método de manera remota
    - proporciona un mecanismo simple para la comunicación de servidores en aplicaciones distribuidas basadas exclusivamente en Java

# Procedimientos remotos

- Java RMI
  - La arquitectura RMI puede verse como un modelo de cuatro capas.
    - Primera capa
      - la de aplicación y se corresponde con la implementación real de las aplicaciones cliente y servidor
      - llamadas a alto nivel para acceder y exportar objetos remotos
      - declarar dichos métodos en una interfaz que extienda `java.rmi.Remote`
      - "marcar" un objeto como remotamente accesible

# Procedimientos remotos

- Java RMI
  - Segunda capa
    - Hace de proxy
    - Las llamadas a objetos remotos y acciones junto con sus parámetros y retorno de objetos tienen lugar en esta capa
  - Tercera capa
    - Referencia remota
    - Establecimiento de las persistencias semánticas y estrategias adecuadas para la recuperación de conexiones perdidas
    - Conexión de tipo stream
  - Cuarta capa
    - De transporte. Protocolo JRMP (Java Remote Method Protocol)

# Procedimientos remotos

- Java RMI
  - Server RMI
    - Se crean referencias a objetos remotos para hacerlos accesibles esperando que el cliente los invoque.

```
public interface MiInterfazRemota extends java.rmi.Remote
{
    public void miMetodo1() throws java.rmi.RemoteException;
    public int miMetodo2() throws java.rmi.RemoteException;
}
```

# Procedimientos remotos

- Java RMI
  - **UnicastRemoteObject**: superclase para implementar objetos remotos
    - Da soporte punto-a-punto para referencias activas de objetos (invocaciones, parámetros y resultados) utilizando streams TCP
  - Registro RMI
    - Gestiona un conjunto de objetos remotos compartidos y se buscan ante las peticiones de los clientes

# Procedimientos remotos

- Java RMI
  - Cliente RMI
    - Tiene referencia a los objetos remoto
    - Realiza las llamadas a objetos remotos

# Procedimientos remotos

- Java RMI
  - Registro RMI
    - Servidor que permite a una aplicación buscar objetos que están siendo exportados para su uso mediante llamadas a métodos remotos.
    - Una vez que el objeto ha sido localizado, ya se puede utilizar utilizando la misma sintaxis que una llamada a un método local
  - Por defecto, en el puerto 1099
    - Ejecución: (%java\_home%\bin en PATH)
      - Windows: start rmiregistry
      - Unix: rmiregistry &

En el caso de Windows, se debe ejecutar:

```
start rmiregistry 1234
```

Y en el caso de Linux:

```
rmiregistry &
```

# Procedimientos remotos

- Ejercicio
  - Dada la interfaz RMICalculator, se pide
    - Convertirla en una interfaz remota
    - Realizar un servidor remoto que exponga sus métodos
    - Crear un cliente que realice peticiones RMI al servidor de la calculadora



# Generación de servicios en red

- Servicios en red
- Programación de aplicaciones cliente-servidor
- Protocolos a nivel de aplicación
- Librerías de clases y componentes
- Procedimientos remotos
- ~~Javadoc~~
- Servicios Web

# Servicios web

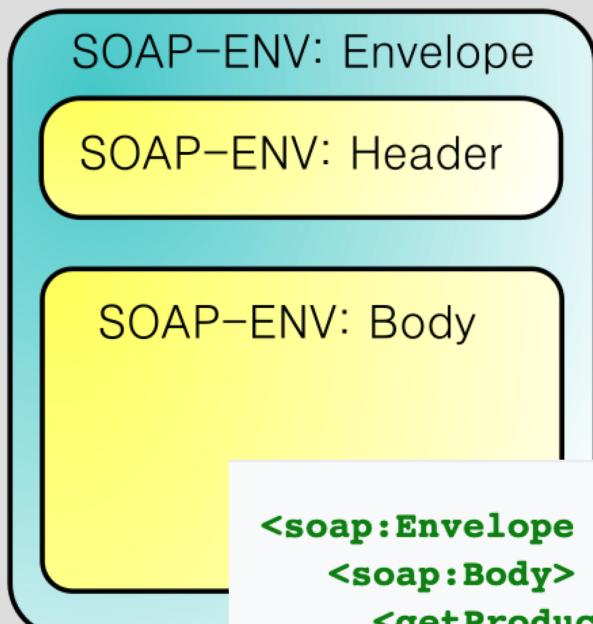
- SOAP (Simple Object Access Protocol)
  - Protocolo estándar que define cómo dos objetos en diferentes procesos se comunican mediante intercambio de datos XML
  - Deriva de un protocolo anterior llamado XML-RPC
  - Uno de los protocolos más utilizados en Servicios Web
  - Stateless

# Servicios web

- SOAP (Simple Object Access Protocol)
  - Características:
    - Extensibilidad
    - Neutralidad: corriendo sobre TCP puede ser utilizado por cualquier protocolo de aplicación (HTTP, JMS, ...)
    - Independencia: del modelo de programación, de arquitectura, ...

# Servicios web

- SOAP (Simple Object Access Protocol)
  - Formato de mensaje



```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <getProductDetails xmlns="http://warehouse.example.com/ws">
      <productId>827635</productId>
    </getProductDetails>
  </soap:Body>
</soap:Envelope>
```

# Servicios web

- SOAP (Simple Object Access Protocol)
  - Ventajas:
    - Interoperabilidad: independiente del lenguaje de programación
    - Escalable: gracias a la transmisión de mensajes vía HTTP.
    - Cualquier lenguaje y ejecutado en cualquier plataforma.
    - Autenticación o modo anónimo.
    - Posibilidad de utilizar cualquier protocolo de transporte capaz de transmitir texto, típicamente HTTP o SMTP.

# Servicios web

- SOAP (Simple Object Access Protocol)
  - Desventajas:
    - XML es más lento que Corba
    - Depende del WSDL (Web Services Description Language).
    - Se basa en un contrato cerrado, surgiendo incompatibilidades al evolucionarlo

# Servicios web

- SOAP (Simple Object Access Protocol)
  - Fichero WSDL:
    - Describe un servicio web
    - El servidor y el cliente cumplen el wsdl
    - Definen:
      - Tipo de datos
      - Mensajes
      - Puertos
      - Binding
    - Eclipse: File → New File → WSDL File

# Servicios web

- SOAP (Simple Object Access Protocol)
  - Fichero WSDL:
    - Type

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

    <xsd:element type="xsd:string" name="content" />
    <xsd:element type="xsd:string" name="endDate" />
    <xsd:element type="xsd:string" name="startDate" />

</xsd:schema>
```

- ComplexType

# Servicios web

- SOAP (Simple Object Access Protocol)
  - Fichero WSDL:
    - Type
    - ComplexType

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <xsd:element name="ClassifiedList">
    <xsd:complexType>

      <xsd:sequence>
        <xsd:element name="ClassifiedAd" maxOccurs="unbounded"
                     type="ClassifiedAdType"/>

      </xsd:sequence>

    </xsd:complexType>
  </xsd:element>

  <xsd:complexType name="ClassifiedAdType">
    <xsd:sequence>
      <xsd:element type="xsd:string" name="content" />

      <xsd:element type="xsd:string" name="endDate" />
      <xsd:element type="xsd:string" name="startDate" />
    </xsd:sequence>
  </xsd:complexType>

</xsd:schema>
```

# Servicios web

- SOAP (Simple Object Access Protocol)
  - Una aplicación SOAP debe correr en un Server
    - Tomcat
    - Instalación en eclipse
    - Jboss
    - ...
  - Librerías de desarrollo
    - Axis
    - Apache CXF:
      - Instalación en eclipse

# Servicios web

- SOAP (Simple Object Access Protocol)
  - Se requieren de varias librerías que nos ayuden con la comunicación entre cliente y servidor
  - Gestión de dependencias/librerías
    - Maven: xml
    - Gradle: groovy



# Servicios web

- SOAP (Simple Object Access Protocol)
  - Maven y CXF

```
<properties>
    <cxfr.version>2.2.3</cxfr.version>
</properties>

<dependencies>
    <dependency>
        <groupId>org.apache.cxf</groupId>
        <artifactId>cxf-rt-frontend-jaxws</artifactId>
        <version>${cxfr.version}</version>
    </dependency>
    <dependency>
        <groupId>org.apache.cxf</groupId>
        <artifactId>cxf-rt-transports-http</artifactId>
        <version>${cxfr.version}</version>
    </dependency>
    <!-- Jetty is needed if you're are not using the CXFServlet -->
    <dependency>
        <groupId>org.apache.cxf</groupId>
        <artifactId>cxf-rt-transports-http-jetty</artifactId>
        <version>${cxfr.version}</version>
    </dependency>
</dependencies>
```

# Servicios web

- SOAP (Simple Object Access Protocol)
  - Dependencias
    - Spring web
    - Spring context
      - Concepto de bean



# Servicios web

- SOAP (Simple Object Access Protocol)
  - Client
    - SoapUI
      - Nos permite realizar peticiones SOAP a partir de un fichero WSDL sin necesidad de programar ni una línea de código



# Servicios web

- SOAP (Simple Object Access Protocol)
  - HelloWorld WS
    - Create Dynamic Web Project
    - Maven pom.xml
      - CXF & Spring Dependencies
    - Define WSDL
    - Generate Server
      - CXF generates .java
    - Implement WS
    - Deploy on Tomcat

# Servicios web

- SOAP (Simple Object Access Protocol)
  - URLs desplegadas en Tomcat
    - Información del servicio web y sus operaciones:  
<http://localhost:8080/helloWorldPSP/services>
    - Endpoint del Servicio Web. Necesario en SoapUi  
<http://localhost:8080/helloWorldPSP/services/helloWorldPSPSOAP>
    - Fichero wsdl desplegado en tomcat  
<http://localhost:8080/helloWorldPSP/wsdl/helloWorldPSP.wsdl>

# Servicios web

- Ejercicio SOAP
  - Partiendo del WS HelloWorld
    - Añade una segunda operación “sayGoodBye”
      - Recibirá como argumento dos string:
        - El primer argumento se llamará *name*
        - El segundo argumento se llamará *procedence*
      - La respuesta será una cadena de texto:  
“Good bye \${name}, from \${procedence}”

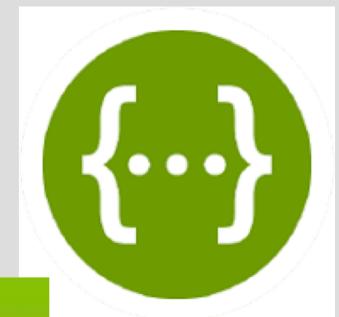


# REST

- REST (Representational State Transfer)
  - Se centra en los recursos, el modelo. No en las acciones como en RPC
  - Stateless
  - SOAP vs REST:
    - Alta de usuario                       vs     Usuario (POST)
    - Consultar usuario                      vs     Usuario (GET)
    - Dar de baja un usuario              vs     Usuario (DELETE)
    - Modificar un usuario                vs     Usuario (PUT)

# REST

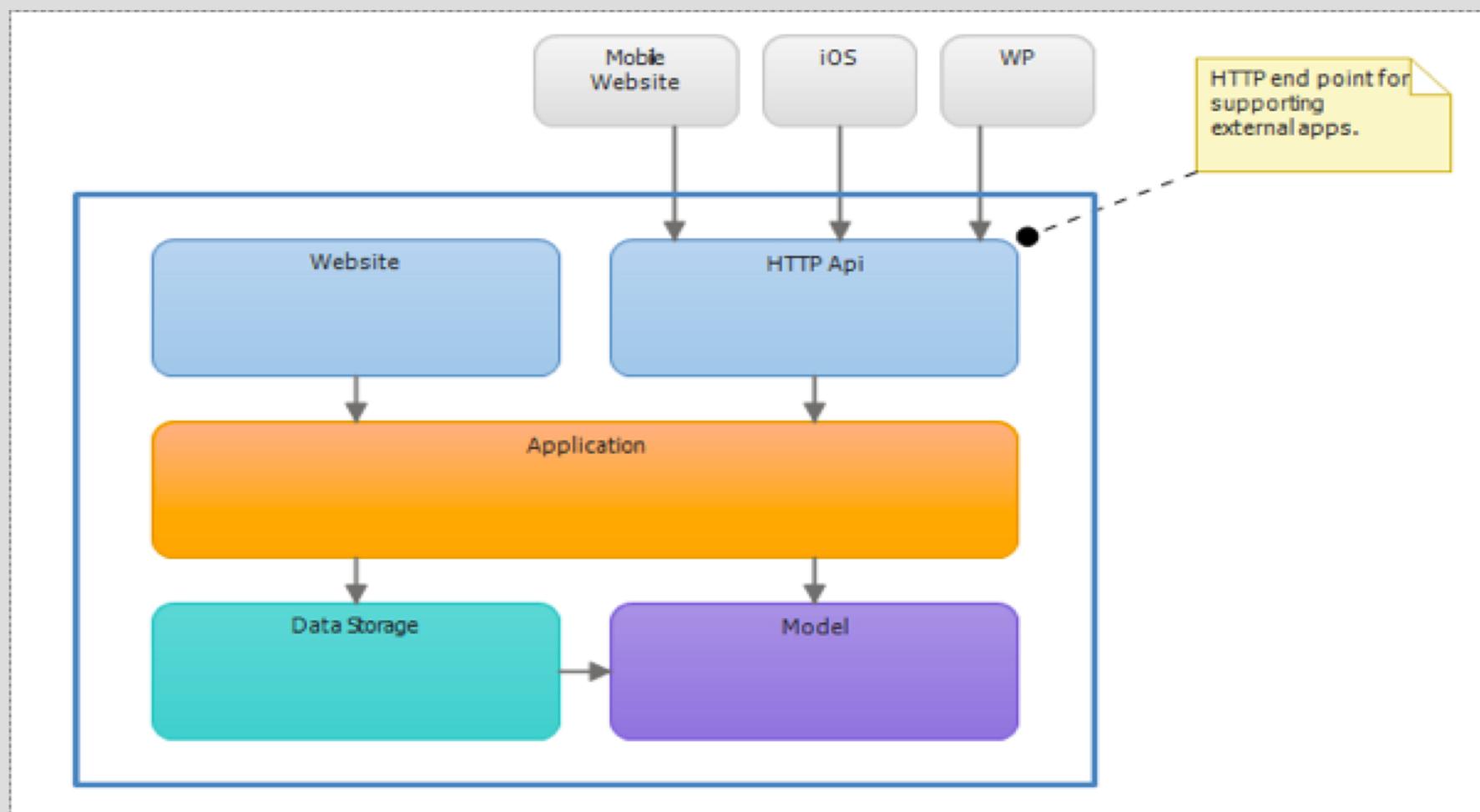
- REST (Representational State Transfer)
  - Definición de API
    - Swagger : formato yml



The image shows a screenshot of a Swagger Editor interface. On the left, there is a code editor window displaying a YAML (YML) file. The file contains definitions for a Petstore API, including models like Pet and Category, and operations for creating, reading, updating, and deleting pets. On the right, there is a generated API documentation page titled "Swagger Petstore". The page includes a "Try it out" button, a "Documentation" link, and links for "Definitions", "Responses", and "Parameters". Below the main content, there are three download links: "API", "OpenAPI", and "JSON".

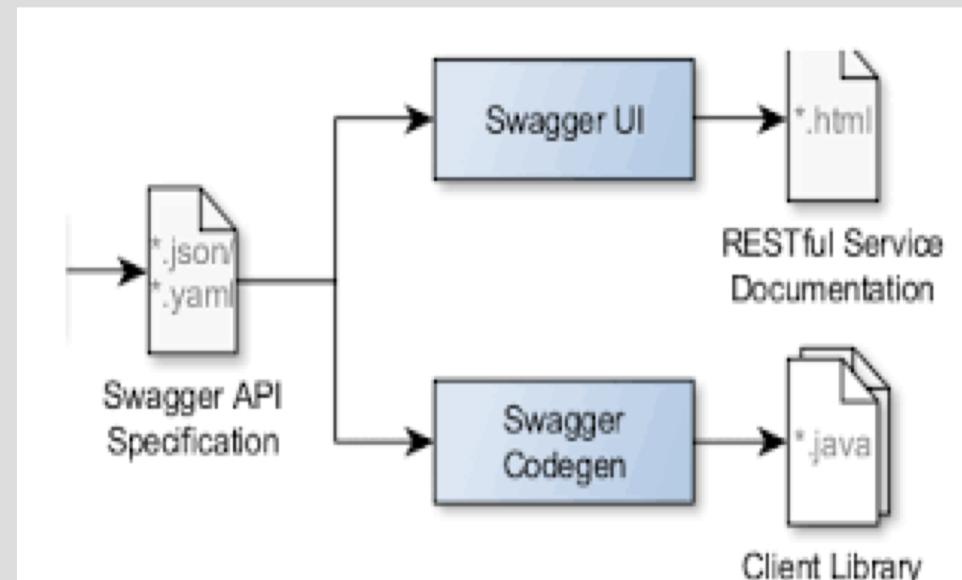
# REST

- REST (Representational State Transfer)



# REST

- REST (Representational State Transfer)
  - Swagger Codegen Maven Plugin
    - Genera el código del controlador, los DTOS y clientes.
    - Disponible en varios lenguajes



# Bibliografía

- Programación de Servicios y Procesos, Editorial RA-MA
- Wikipedia
- Oracle RMI docs
- <https://cxf.apache.org/docs/using-cxf-with-maven.html>
- <https://cxf.apache.org/docs/cxf-eclipse-plugin-instructions>