

Introducción a las arquitecturas C/S

Curso 2020/21

Javier Albert Segui

Índice

- Definición del Concepto Cliente/Servidor – C/S
- Arquitectura C/S
- Características Sistema C/S
- Beneficios Sistema C/S
- Problemas Sistema C/S
- Generaciones Sistema C/S
- Modos de llamada C/S
- Modalidades C/S
- Computación en la Nube

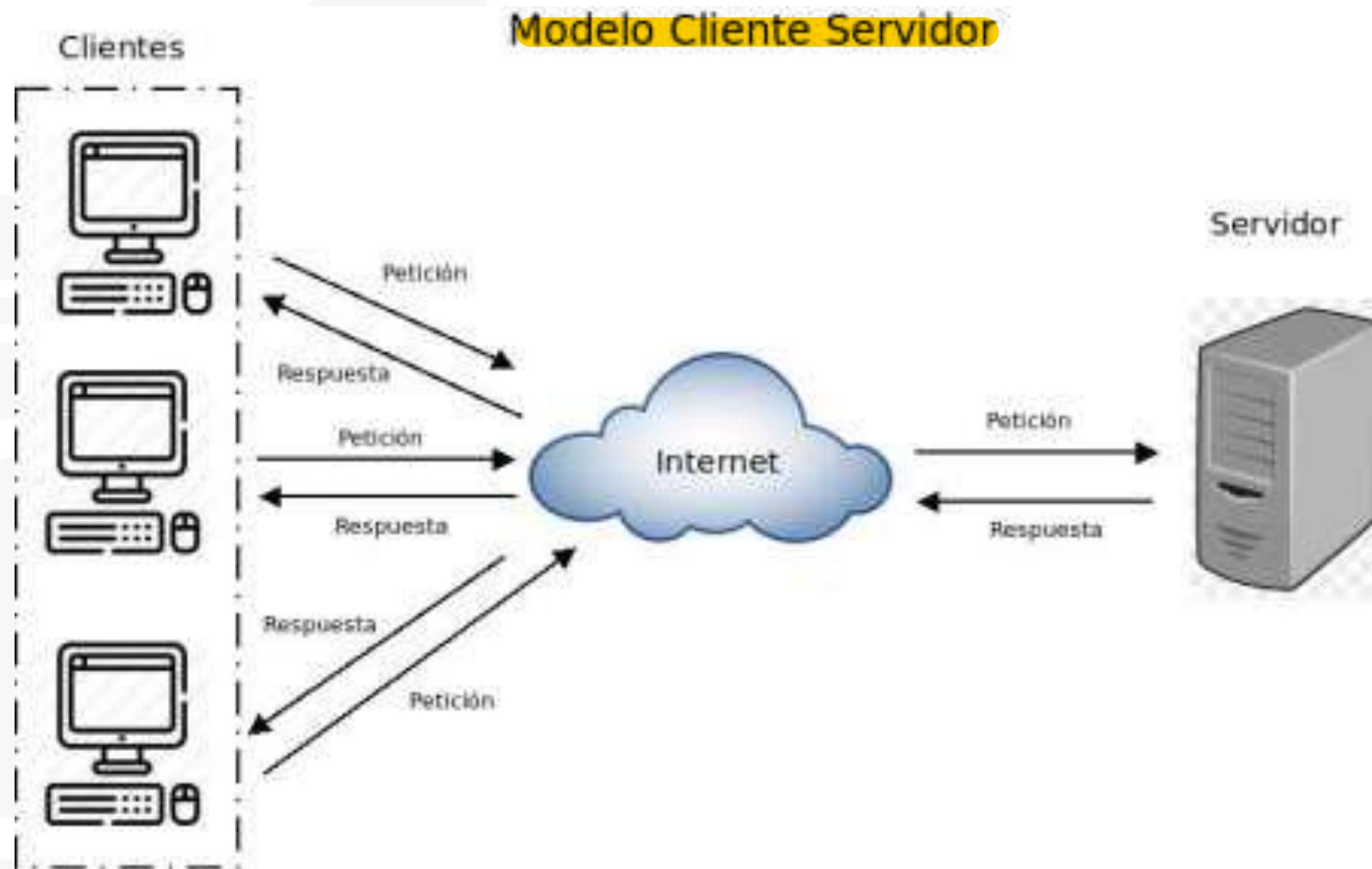
Definición C/S

- Es la integración distribuida de un sistema en red, con los
 - Recursos,
 - Medios
 - Aplicaciones
- Definidos de forma modular para atender las solicitudes de los clientes compartiendo datos, procesos e información de forma transparente

Definiciones C/S

- **Clients:** son aquellas “entidades” que necesitan Servicios
- **Servidores:** Proporcionan estos Servicios, son objetos separados desde un punto de vista lógico y se comunican a través de una red de Comunicaciones para realizar una o varias tareas de forma conjunta.
- **Middleware:** es un software distribuido para la interacción entre el cliente y el servidor.

Arquitectura Cliente / Servidor



Características Sistemas C/S

- **Recursos compartidos:** Un único servidor puede interactuar con distintos clientes simultáneamente, por tanto, necesita controlar el acceso a sus recursos para garantizar la seguridad.
- **Transparencia:** Las aplicaciones clientes direccionan el servidor de manera lógica sin importar su localización.
- **Separación de funciones (modularidad):** La lógica se separa en distintos módulos funcionales para distribuir la carga entre procesadores.
- **Entornos heterogéneos:** Las aplicaciones deberían ser independientes del procesador y el sistema operativo utilizado.

Características Sistemas C/S

- **Encapsulación de servicios:** Las funcionalidades deben ser encapsuladas para que el acceso sea conocido y accesible. Internamente la función puede cambiar sin necesidad de cambiar los clientes.
- **Protocolos asimétricos:** El cliente es quien inicia la comunicación mientras el servidor esta esperando de forma pasiva.
- **Intercambio de mensajes:** La comunicación entre los clientes y servidores se basa en mensajes. El cliente envía **peticiones** y recibe **respuestas** del servidor.
- **Integridad:** El código y los datos se mantienen centralizados por lo tanto es mas sencillo proteger la integridad de ellos.

Beneficios Sistemas C/S

- Escalabilidad multidimensional: Los sistemas C/S deben ser escalables tanto horizontal como verticalmente.
 - Escalabilidad horizontal: El añadir o eliminar clientes genera impactos de rendimiento
 - Escalabilidad vertical: cambio en el servidor.
- Escalabilidad = economía
- HW y SW Heterogéneo = Integración
 - Despliegues independientes del Cliente y Servidor
 - Cliente y Servidor usan el HW y SSOO más adecuados para su función.
- Robustez
- Amigabilidad / Usabilidad

Problemas Sistema C/S

- **Software:** Escasas herramientas para la evaluación de carga
- **Comunicación:** Necesidad de tener mecanismos de comunicación idénticos en distintas plataformas.
Sockets, RPC...
- **Seguridad:** Hay que verificarla en ambos extremos.
- **Red:** congestión, pérdida de datos,
- **Duplicidad:** Distintas localizaciones, duplicidad de servidores y servicios, actualizaciones.....

Ejemplos C/S

- Fileservers
- Servidores de Bases de datos
- Servidores de transacciones
- Servidores de groupware
- Servidores Web

Generaciones Sistemas C/S



Generaciones Sistemas C/S

- Generación 1- Arquitectura C/S de una capa
 - Basados en Mainframes y miniordenadores a los que se conectaban terminales “tontos”
 - Basados en redes LAN, posteriormente WAN y MAN.
 - Comunicaciones homogéneas
 - Costosos de implementar, desarrollar y mantener.

Generaciones Sistemas C/S



Generaciones Sistemas C/S

- Generación 2 – Arquitectura C/S de 2 capas
 - Funciones distribuidas. Coordinación entre clientes y servidores
 - Algunas funciones se integran en los Gestores de Bases de datos.
 - Arquitecturas complejas, ausencia de estándares, falta de herramientas de gestión.
 - Desarrollo costoso

Generaciones Sistemas C/S

- Generación 3 – Arquitectura C/S de 3 capas
 - Aparece el concepto de *front-end* o *capa de presentación* que es el responsable de la presentación.
 - El componente *back-end* proporciona acceso a los servicios dedicados
 - Un componente intermedio *middle-tier* que permite compartir y controlar la lógica de negocio.

Funciones del servidor

- Esperar peticiones de los clientes
- Atender solicitudes simultáneas
- Priorizar la atención de las solicitudes
- Capacidad de realizar acciones en segundo plano.
- Robustez
- Escalabilidad y extensibilidad.

Funciones del servidor

- Requisitos del SSOO.

- Basicos

- Alto nivel de concurrencia
 - Slots de pequeño tamaño
 - Prioridades
 - Mecanismos de concurrencia
 - Mecanismos de comunicacion entre procesos
 - Threads
 - Sistemas de ficheros de altas prestaciones
 - Sistemas de gestion de memoria eficientes
 - Extensibilidad sin recompilaciones, reinicios....

Funciones del servidor

- Requisitos del SSOO.
 - Extendidos
 - Distintos protocolos de comunicaciones
 - Acceso transparente a recursos compartidos
 - Sistemas de directorio centralizado y global
 - Servicios de autenticación y seguridad
 - Gestión de la configuración, monitorización y alerta

Características del cliente

- Clientes de 3 tipos: sin GUI, con GUI, con OOUI
- Requisitos del SSOO cliente:
 - Implementar mecanismo de envío/recepción de mensajes
 - Implementar mecanismos de transferencia de archivos
 - Multitarea (prioridades, comunicación entre procesos, threads...)
 - Portabilidad
 - Robustez

Middleware

- Su función es que todo funcione de forma TRANSPARENTE
- Este middleware nos da capacidades de:
 - Comunicación a través de la red
 - Envío y recepción de ficheros
 - Servicio de directorios global
 - Mecanismos de seguridad distribuidos

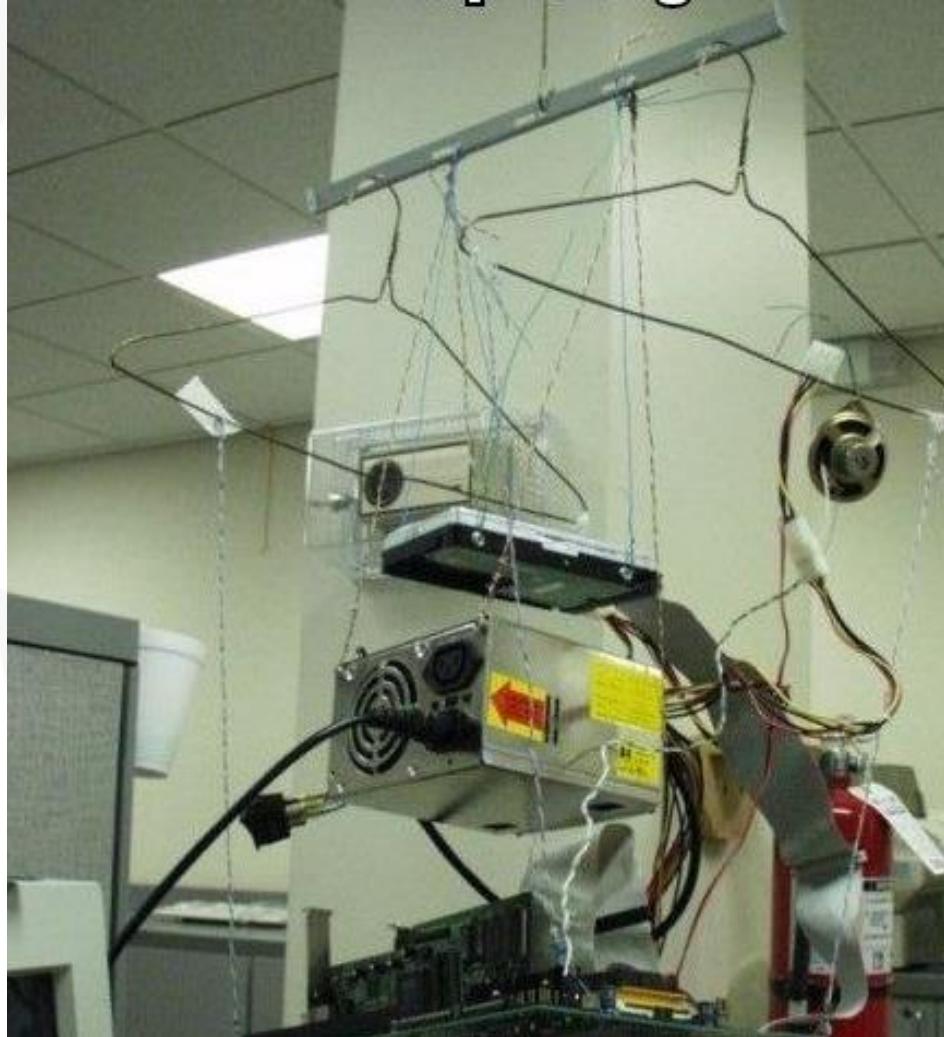
Llamadas en el C/S

- Llamada **síncrona**: Es el modo mas sencillo de gestionar y usar las llamadas entre Cliente y Servidor.
 - Cuando se realiza una llamada al servidor, el cliente se coloca en modo espera hasta recibir una respuesta.
- Llamada **asíncrona**: El cliente no entra en modo espera al llamar a un servicio.
 - Cuando el servicio termina, informa a la aplicación de forma automática.
 - La aplicación no tiene que esperar para continuar con otros procesos.

Protocolos C/S

- **Sockets:** Protocolo propio de la comunicación C/S
- **RPC (Remote Procedure Call):** nos permite ejecutar un procedimiento remoto localizado en un servidor.
- **RMI (Remote Method Invocation):** Es un protocolo de invocación de métodos de JAVA
- **CORBA:**
 - Common Object Request Broker Arquitecture.
 - Definicion ESTANDARD
- **Web Services:** Mecanismo para crear Servicios distribuidos basados en el protocolo HTTP mediante SOAP o REST

Cloud computing



Cloud Computing

- Es un paradigma que nos permite ofrecer **servicios** a través de una **red**, generalmente **Internet**

Características

- Agilidad
- Costo
- Escalabilidad
- Independencia
- Rendimiento
- Seguridad
- Mantenimiento

Ventajas

- Integración de servicios
- Rápida implementación
- Servicios ubicuos
- Actualización automática
- Green Computing

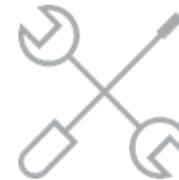
Desventajas

- Centralización
- Disponibilidad
- Madurez
- Seguridad
- Escalabilidad

Servicios en la nube



IaaS
Infrastructure as a Service
host



PaaS
Platform as a Service
build



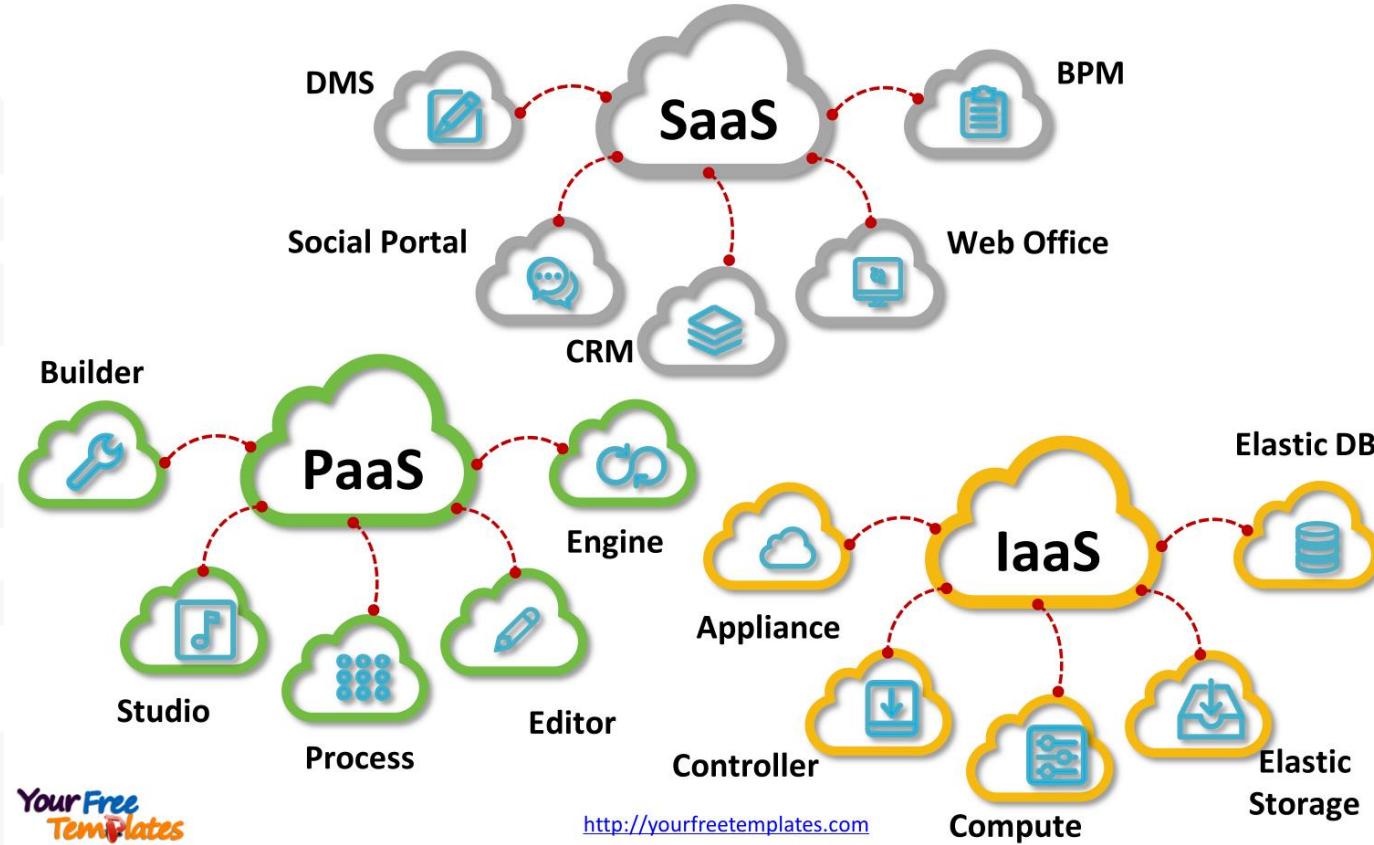
SaaS
Software as a Service
consume

Pizza as a Service

Pizza as a Service



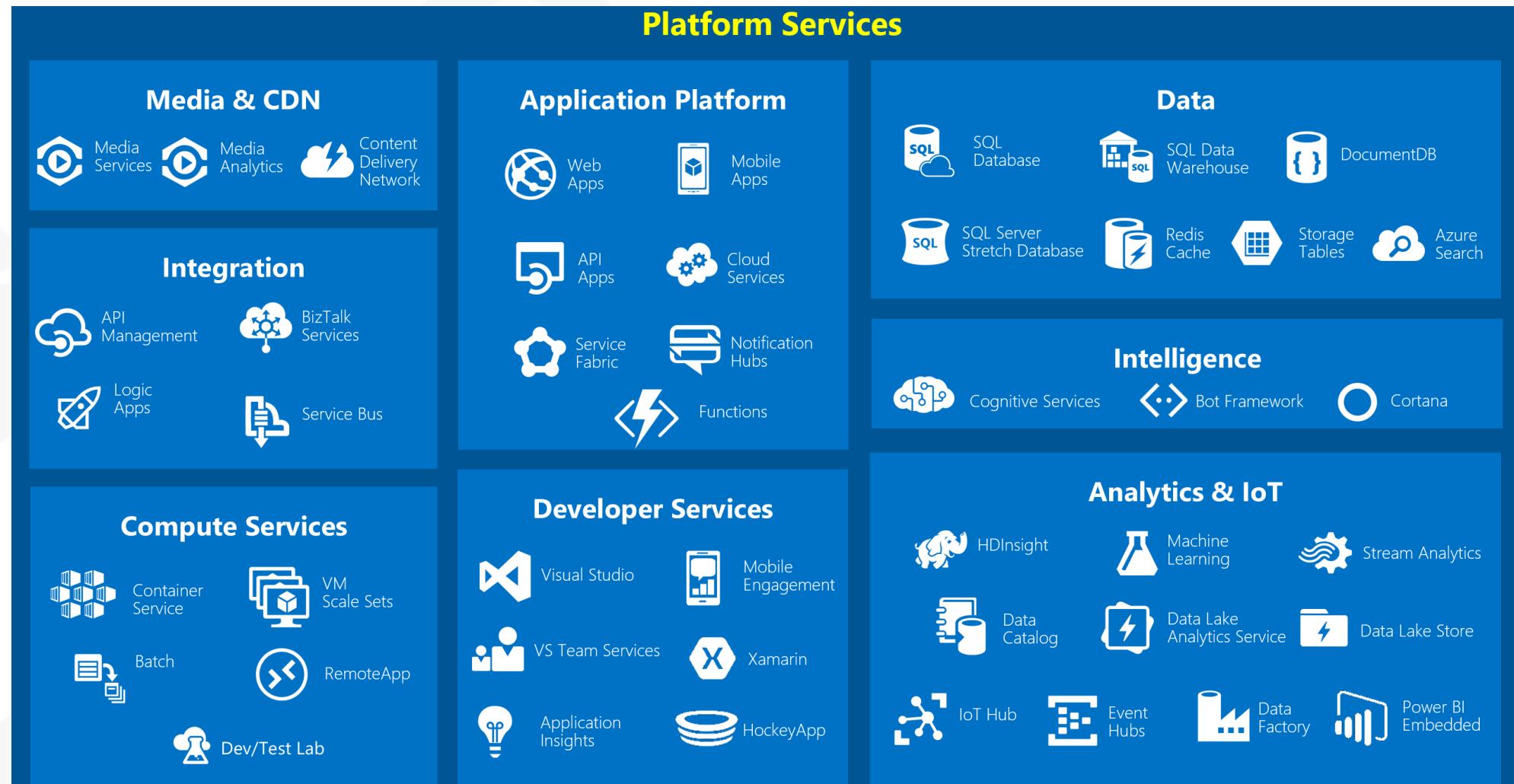
Servicios en la nube



IaaS



PaaS



SaaS



facebook



twitter

flickr



Google buzz



LinkedIn



salesforce.com.
Success. Not Software.

angel.com



YAHOO

TypePad



InterCall

intuit

GoToMeeting®

Google

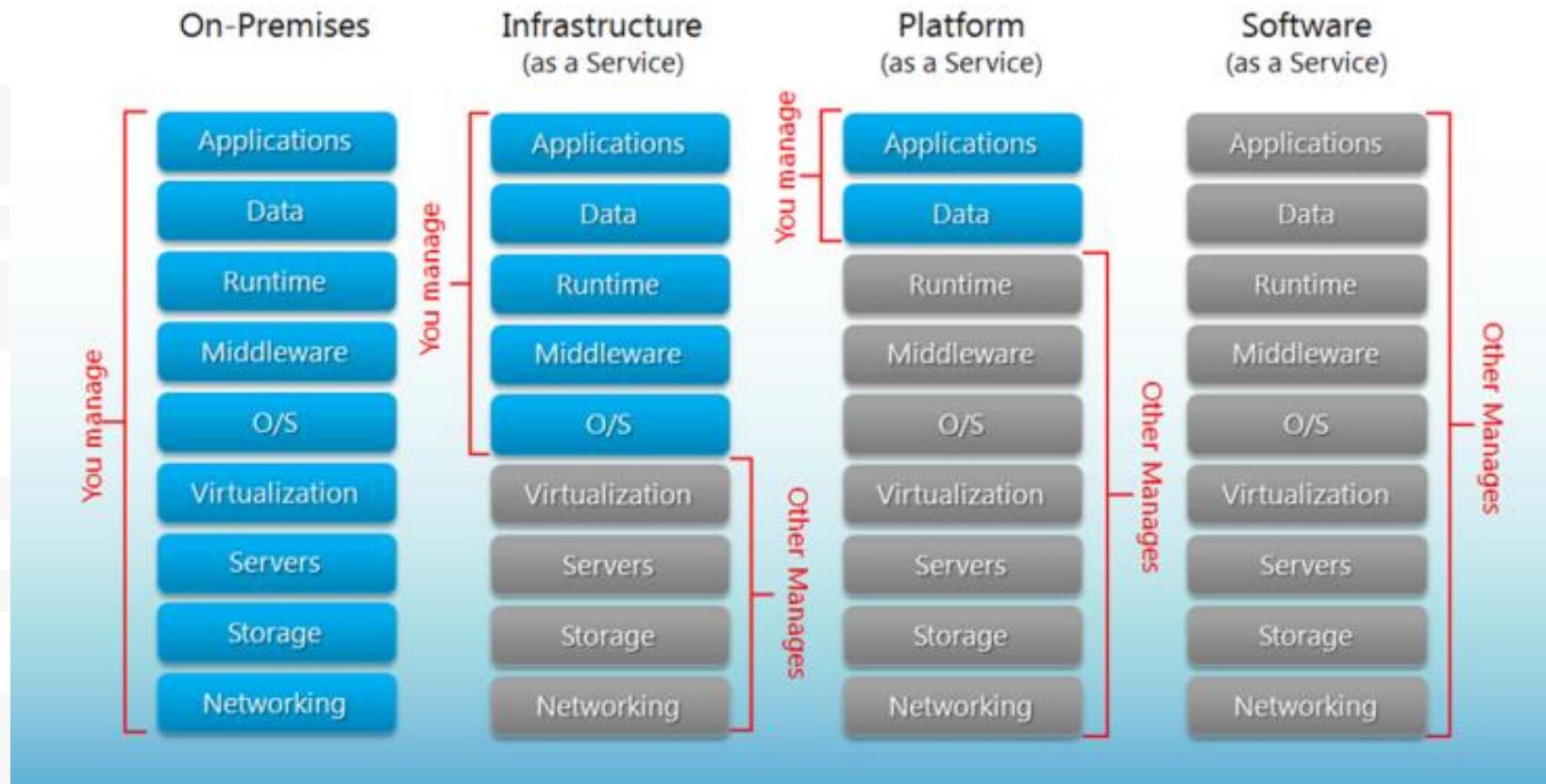
Constant Contact®
Connect. Inform. Grow.

facebook

ADP



Responsabilidades



Arquitectura Web

Curso 2020/21

Javier Albert Segui

Indice

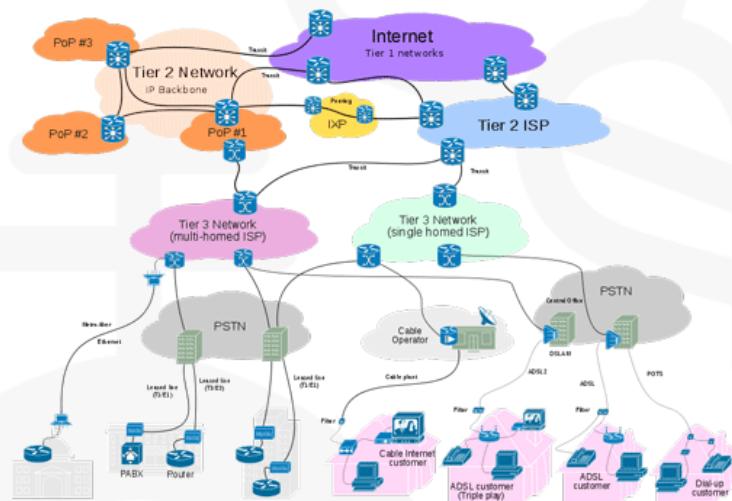
- Evolución y funcionamiento de Internet
- Servicios y protocolos de Internet
- Protocolo HTTP
- Desarrollo de aplicaciones Web
- Arquitectura Web

Evolución de Internet

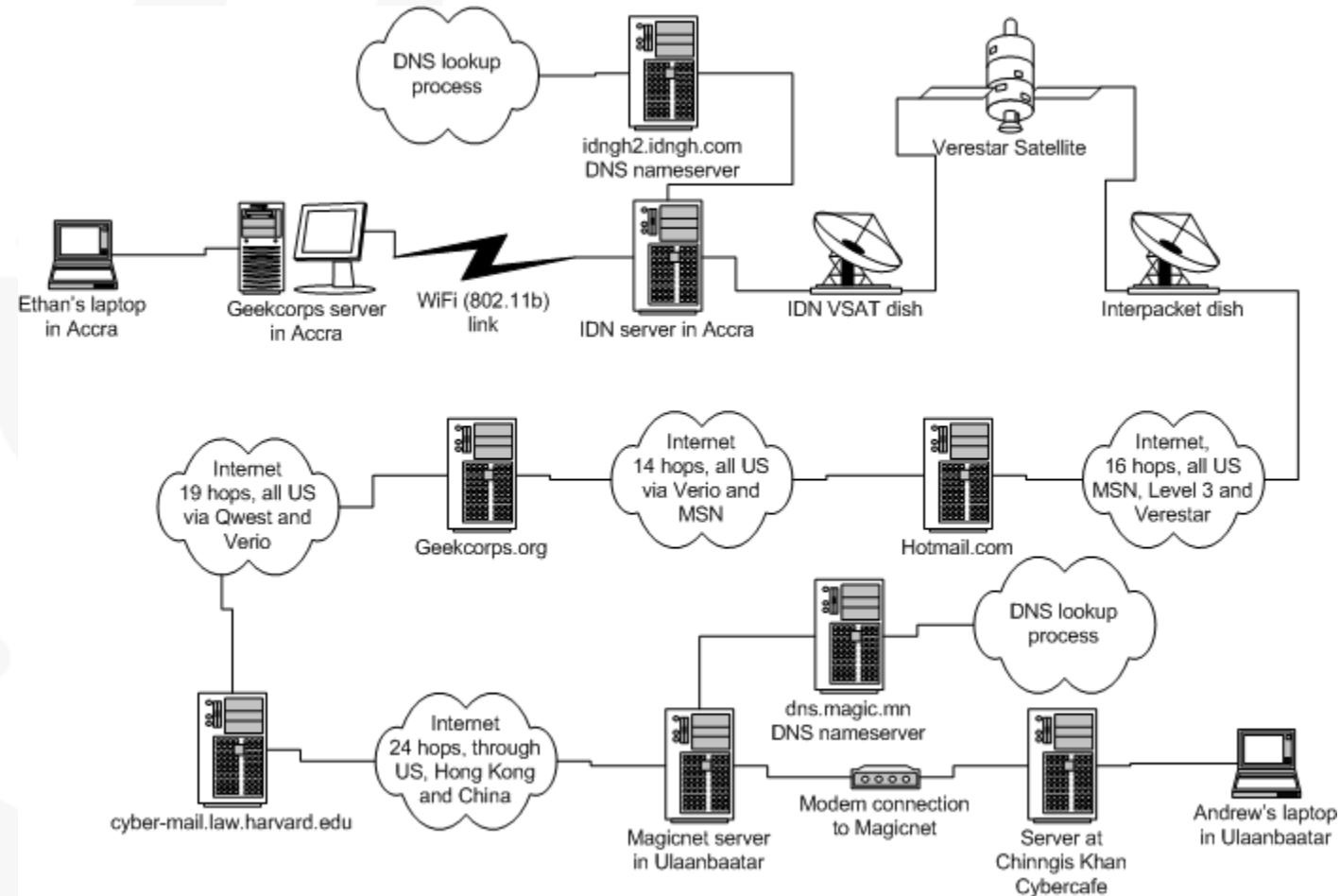
- Internet no es mas que una red de ordenadores de tamaño mundial, que usa como protocolo de comunicaciones TCP/IP
- Internet dio sus primeros pasos en Estados Unidos a finales de los años 60. Como evolución de la red del DoD para comunicación en caso de guerra, ARPAnet.

Como funciona Internet

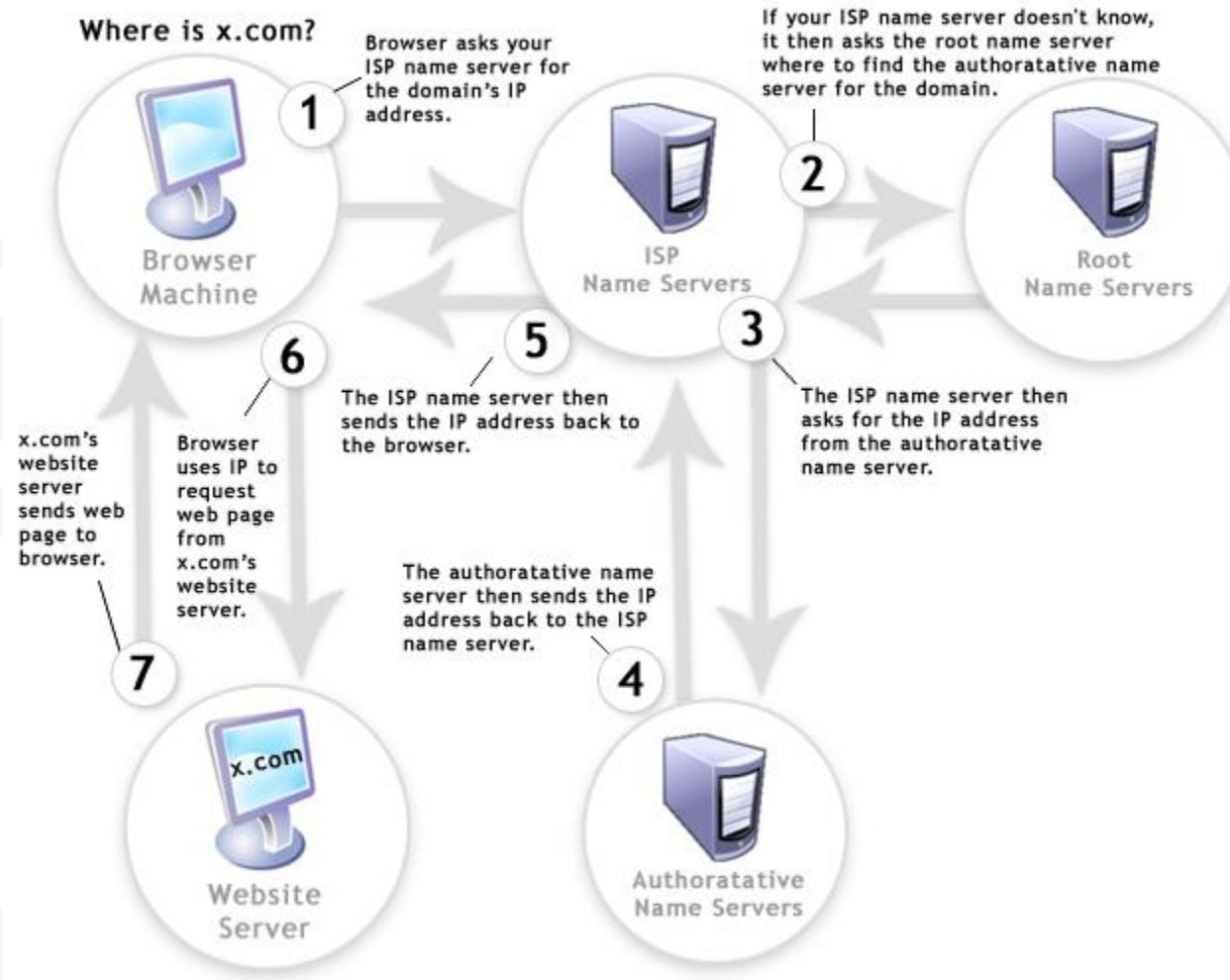
- Internet responde a una arquitectura Cliente/Servidor
- En el momento que utilizamos alguno de los servicios que Internet nos ofrece, se pone en marcha un extenso entramado de maquinas y aplicaciones que hacen posible ese funcionamiento.



Como funciona Internet



Como funciona Internet



Protocolos de Internet

- Los Servicios a los que podemos acceder en internet son muy diversos.
- Definiremos **Servicio** como un conjunto de programas y utilidades que nos permiten realizar una determinada tarea.
- Servicios:
 - FTP (File Transfer Protocol)
 - DNS
 - Gopher
 - News
 - Correo electrónico
 - Mensajería instantánea
 - Redes P2P
 - Web

Protocolos de Internet

- El proyecto **World Wide Web** nació en respuesta a la necesidad que la comunidad científica internacional tenía de nuevos sistemas de distribución de la información.
- Este fue uno de los objetivos que se planteó Tim Berners-Lee (ingeniero británico) cuando en 1989 presentó a sus superiores del **CERN** la propuesta original para el proyecto World Wide Web. (<http://www.w3c.org>)
- Desde el punto de vista del usuario, la Web consiste en un enorme conjunto, a nivel mundial, de documentos llamados páginas. Cada página puede contener vínculos (enlaces) con otras páginas relacionadas en cualquier lugar del mundo. Podemos afirmar que es un gran almacén o repositorio de información.

Protocolos de Internet

- Cada servidor Web tiene un demonio (proceso) que escucha en el puerto TCP80, esperando conexiones entrantes. Tras establecerse una conexión, el cliente envía una solicitud, el servidor genera una respuesta y finalmente se libera la conexión.
- El servicio Web responde a un modelo Cliente/Servidor, donde los clientes demandan “hipertextos” a los servidores. La web es un sistema hipermedia distribuido de acceso interactivo para los usuarios.
 - Hipertexto = Documentos con “punteros” a otros documentos.
 - Hipermedia = Hipertexto + información multimedia (imágenes, sonidos, vídeos,...)

El “lenguaje” de la Web

- El protocolo de Comunicaciones es **HTTP – HyperText Transfer Protocol**
- El lenguaje para poder crear documentos válidos es **HTML – HyperText Markup Language**
- El esquema de direcciones que permiten localizar Recursos en Internet es **URL – Universal Resource Locator**
- Las “herramientas” que interpretan el lenguaje HTML y lo presentan a los usuarios son los **Navegadores**

Protocolo HTTP

- Es la base sobre la que se sustenta la Web
- Es un protocolo del nivel de aplicación para sistemas de información multimedia distribuidos.
- Es un protocolo no orientado a estado, que permite el intercambio de información.
- Al ser un protocolo sin estado, se hace necesario que al crear aplicaciones en entornos web se necesite mantener el estado de las mismas a través de Sesiones.
- Basado en las RFC1945/2616/2774/7540.....

Protocolo HTTP

- Tiene las siguientes **propiedades**:
 - Se asienta sobre el paradigma **solicitud/respuesta**
 - La **comunicación** utiliza el protocolo **TCP/IP**
 - El puerto por defecto es **80**, pero puede utilizarse cualquier otro.
 - Los mensajes/transacciones son en texto plano lo que lo hace legible y fácil de procesar y depurar.
 - Está abierto a nuevos tipos de datos, utiliza tipos **MIME** (Multipart Internet Mail Extension) para determinar el tipo de datos que transporta.
 - Tiene un esquema de direccionamiento comprensible:
servicio://host:puerto/recurso

Protocolo HTTP

○ Cliente

- Get / HTTP/1.1
- Host: www.example.com
- User-Agent: Lynx/2.2
libwww/2.14

○ Servidor

- HTTP/1.1 200 OK
- Date: Mon, 23 May 2005 22:38:34
GMT
- Content-Type: text/html;
charset=UTF-8
- Content-Length: 155
- Last-Modified: Wed, 08 Jan 2003
23:11:55 GMT
- Server: Apache/1.3.3.7 (Unix) (Red-Hat/Linux)
- ETag: "3f80f-1b6-3e1cb03b"
- Accept-Ranges: bytes
- Connection: close

Protocolo HTTP

○ Comandos HTTP :

- GET Petición de recurso.
- POST Petición de recurso pasando parámetros.
- HEAD Petición de datos sobre recurso.
- PUT Creación o envío de recurso.
- DELETE Eliminación de recurso.
- TRACE Devuelve al origen la petición tal como se ha recibido en el receptor, para depurar errores.
- OPTIONS Sirve para comprobar las capacidades del servidor.
- CONNECT Reservado para uso en servidores intermedios capaces de funcionar como túneles.
- PATCH: Aplica modificaciones parciales a un recurso ya cargado.

Protocolo HTTP

○ Respuestas:

- Informational 1XX – Petición recibida, continúa proceso.
- Successful 2XX – Petición procesada correctamente
- Redirection 3XX – Petición redirigida
- Client Error 4XX – No se puede procesar la petición porque es incorrecta
- Server Error 5XX – El servidor ha fallado en el proceso de la petición.

Protocolo HTTP

○ Cabecera:

- Son metadatos que se envían en las transacciones de petición o respuesta para proporcionar información esencial sobre el mensaje en curso.
- Cada cabecera es especificada por un nombre de cabecera seguido por dos puntos, un espacio en blanco y el valor de dicha cabecera seguida por un retorno de carro.
 - Host: www.prueba.com
 - Accept: www/source
 - Accept: text/html
 - Accept: image/gif
 - User-Agent: Lynx/2.2 libwww/2.14

Desarrollo de aplicaciones Web

- Con la introducción de Internet y de la Web en concreto, se han abierto infinidad de posibilidades en cuanto al acceso a la información desde casi cualquier sitio.
- Esto representa un desafío a los desarrolladores de aplicaciones, ya que los avances en tecnología demandan cada vez aplicaciones más rápidas, ligeras y robustas que permitan utilizarse en la Web.

Desarrollo de aplicaciones Web

- El principio....
 - **CGI (Common Gateway Interface)** es una importante tecnología de la World Wide Web que permite a un cliente (navegador web) solicitar datos de un programa ejecutado en un servidor web. CGI especifica un estándar para transferir datos entre el cliente y el programa. Es un mecanismo de comunicación entre el servidor web y una aplicación externa cuyo resultado final de la ejecución son objetos MIME.
 - Las aplicaciones CGI fueron una de las **primeras prácticas de crear contenido dinámico para las páginas web**.
 - En una aplicación CGI, el servidor web pasa las **solicitudes** del cliente a un **programa externo**. Este programa puede estar escrito en cualquier lenguaje que soporte el servidor, aunque por razones de portabilidad se suelen usar lenguajes de script. La **salida de dicho programa es enviada al cliente** en lugar del archivo estático tradicional.

Desarrollo de aplicaciones Web

○ Actualidad:

- Microsoft (.NET + ASP + RAZOR + BLAZOR.....)
- Java (Servlets, JSP.....)
- Otros (PHP, NodeJS....)
- Cada vez hay mas frameworks que ayudan a los programadores a un Desarrollo mas rápido de aplicaciones para la web (JSF, Structs, Spring, Symfony....)

Arquitectura Web

- Suele presentar un **esquema de 3 niveles:**
 - Nivel 1: Capa de presentación.
 - Nivel 2: Capa de lógica de negocio.
 - Nivel 3: Capa de acceso a datos.

Tecnologías del Lado Cliente Lenguajes de Marcado

- ¿Qué es un Lenguaje de Marcas?
- Orígenes
- Ejemplos de Lenguajes de Marcado
 - Lenguaje HTML
 - Lenguaje XHTML
 - Lenguaje XML

¿Qué es un Lenguaje de Marcas?

- Los Lenguajes de Marcas permiten codificar un documento. Mezclando la propia información con la incorporación de marcas o etiquetas (meta- información utilizada para la presentación de dicho documento)
- No es un Lenguaje de Programación.
- Surgen en los años 60 debido a la gran variedad de tipos de ficheros que existían y a los problemas relacionados con el intercambio.

Orígenes:

- A finales de los años 60, comienzos de los 70 la empresa IBM - International Business Machines, desarrolló el lenguaje GML - Generalized Markup Language.
- En el año 1986, nace SGML -Standard Generalized Markup Language, ISO 8879:1986 es un estándar de la ISO para definir lenguajes de marcado generalizados para documentos.

Lenguajes de Marcado

- Existen multitud de tipos de lenguajes de marcas, pero en concreto para la Tecnología Web, podemos hablar de dos fundamentalmente:
 - HTML
 - XHTML



- El lenguaje **HTML = HiperText Markup Language**, es usado como referente en la publicación de información en la web desde el año 1990/91 con Tim Berners-Lee.
- **HTML 2.0**
 - Se publica en **1995**. A pesar de su nombre, HTML 2.0 es el primer estándar oficial de HTML, es decir, el **HTML 1.0 no existió como estándar**. HTML 2.0 no soportaba tablas. Se simplificaba al máximo la estructura del documento para agilizar su edición.

Evolución HTML

- **HTML 3.2**
 - La versión HTML 3.2 se publicó en **1997** y es la primera recomendación de HTML publicada por el **W3C**. Esta revisión incorporó los últimos avances de las páginas web desarrolladas hasta 1996, como eran los **Applets de Java**.
 - El Consorcio World Wide Web (W3C) es una comunidad internacional donde las distintas instituciones y el público en general, **trabajan conjuntamente** para desarrollar **estándares** en el ámbito Web.
 - Liderado por el inventor de la Web Tim Berners-Lee y el Director Ejecutivo (CEO) Jeffrey Jaffe, la misión del W3C es guiar la Web hacia su máximo potencial.
<http://www.w3.org/> En España: <https://www.w3c.es/>

Evolución HTML

HTML 4.01

- La última especificación oficial del W3C de HTML se publicó en diciembre de 1999 y se denomina HTML 4.01.
- Desde la publicación de HTML 4.01, la actividad de estandarización de HTML se detuvo y el W3C se centró en el desarrollo de un nuevo estándar XHTML.
- Por ello, en el año 2004, las empresas Apple, Mozilla y Opera mostraron su preocupación por la falta de interés del W3C en el HTML y decidieron organizarse en un Grupo de Trabajo denominado WHATWG (Web Hypertext Application Technology Working Group). <http://www.whatwg.org/>. Cuyo resultado fue la culminación del HTML en su última versión – 5 (2008) y ya junto con el W3C a partir del 2010.

Ejemplo

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <title> EJEMPLO 1 </title>
  </head>
  <body>
    <h1>Encabezado</h1>
    <p> esto es un ejemplo de html</p>
  </body>
</html>
```

- Fuente : https://es.wikipedia.org/wiki/HTML_4.0

Evolución HTML

- **HTML 5**
 - En el año 2008 nace la versión 5 y en la actualidad ya está disponible el borrador de la versión 5.3.
 - Esta nueva versión del HTML podemos afirmar que es un compendio de:
 - [Estructura + Contenido] + Apariencia + Comportamiento
 - Los desarrolladores actuales del Lenguaje HTML son:
 - World Wide Web Consortium – **W3C**
 - Internet Engineering Task Force – **Organización Internacional de normalización (RFCs)**
 - Web Hypertext Application Technology Working Group - **WHATWG**



Ejemplo

```
1. <!DOCTYPE html>
2. <html lang="es">
3.   <head>
4.     <meta charset="utf-8" />
5.     <title>Hola Mundo!</title>
6.   </head>
7.   <body>
8.     <h1>Hola Mundo!</h1>
9.   </body>
10. </html>
```



Evolución XHTML

- El lenguaje XHTML = eXtensible HiperText Markup Language, es una extensión del lenguaje HTML para ser compatible con el lenguaje XML.
- XHTML 1.0
 - Es una re-formulación del lenguaje HTML 4 en XML, donde se definen tres DTD's – Documents Type Definition.
 - Semántica idéntica de los elementos del HTML 4

Estructura de documento XHTML

```
<?xml version="1.0" encoding="Codificación"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html>
  <head>
    <title> Aquí va el título de la página </title>
  </head>
  <body>
    <p> Aquí va el contenido de la página web </p>
    <p> en párrafos con texto, tablas, imágenes, formularios, etc. </p>
  </body>
</html>
```

XML

- El lenguaje XML es un meta-lenguaje que permite definir lenguajes de marcas, desarrollado por el W3C, que sirve para organizar y almacenar datos en forma legible.
- Permite definir la gramática de lenguajes específicos.
- XML no sirve únicamente para su aplicación en Internet, sino que se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas.

XML

- Alrededor de la tecnología XML, existen otras que la complementan y extienden su funcionalidad, como por ejemplo:
 - DTD. Document Type Definition, define los tipos de elementos, atributos y entidades permitidas, y puede expresar algunas limitaciones para combinarlos. Los documentos XML que se ajustan a su DTD son denominados válidos.

Evolución XML

XML

– DTD

```
<!ELEMENT addressbook (contact)+>
<!ELEMENT contact (name, address+, city, state, zip, phone, email, web, company)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT address (#PCDATA)>
<!ELEMENT city (#PCDATA)>
<!ELEMENT state (#PCDATA)>
<!ELEMENT zip (#PCDATA)>
<!ELEMENT phone (voice, fax?)>
<!ELEMENT voice (#PCDATA)>
<!ELEMENT fax (#PCDATA)>
<!ELEMENT email (#PCDATA)>
<!ELEMENT web (#PCDATA)>
<!ELEMENT company (#PCDATA)>
```

- **XML Schema.** Un XML Schema es algo similar a un DTD. Define qué elementos puede contener un documento XML, cómo están organizados y qué atributos y de qué tipo pueden tener sus elementos.
- ¿Qué diferencia existe?
 - Usan sintaxis XML y no “ad hoc” como las DTDs
 - Son extensibles.

Evolución XML

XML Schema.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:element name="Libro">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="Título" type="xsd:string"/>
                <xsd:element name="Autores" type="xsd:string" maxOccurs="10"/>
                <xsd:element name="Editorial" type="xsd:string"/>
            </xsd:sequence>
            <xsd:attribute name="precio" type="xsd:double"/>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

Evolución XML

Ejemplo básico de XML

Codificar con XML el siguiente texto de manera que el marcado posibilite las búsquedas de información según los siguientes campos: destinatario del pedido, artículo pedido, dirección de entrega, fecha de entrega.

Entrega de un envío a Roberto Barchino Plata. El envío se compone del material Ipad Pro 11" 64GB. A entregar en la dirección Despacho N313, Escuela Politécnica Superior, Universidad de Alcalá, el día 01-10-2019.

Possible Solución

```
<?xml version="1.0" encoding="UTF-8"?>
<ENTREGA>
  <Destinatario>Joe Doe</Destinatario>
  <Articulo> Ipad Pro 64GB </Articulo>
  <Direccion>NY Morgue</Direccion>
  <Fecha> 01-10-2020 </Fecha>
</ENTREGA>
```

Tema 4

HTML 5 y CSS3

Curso 2020/21

Javier Albert Segui

- HTML 5.0
 - Características.
 - Novedades.
 - Enlaces de interés.
- CSS 3
 - Características
 - ¿Cómo funciona?
 - Enlaces de interés





Estructura



Contenido



Apariencia



Comportamiento



Universidad
de Alcalá

Características

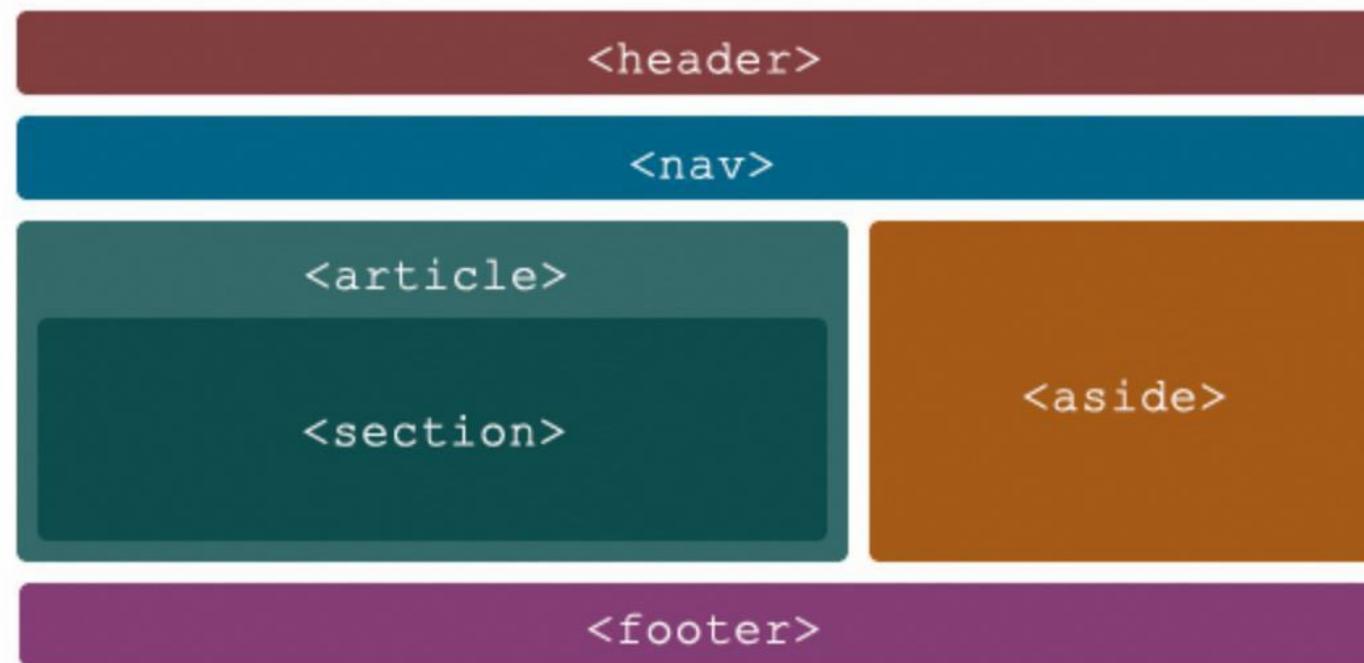
- 5^a versión del lenguaje HTML.
- Surge en el marco de colaboración entre:
 - W3C
 - WHATWG
- Recomendación del 28 de Octubre de 2014
- Los Navegadores no soportan, todavía, toda la especificación, <http://html5test.com/>:
 - Chrome, Explorer, Opera y FireFox

- Donde acceder a la especificación HTML
<http://www.w3.org/TR/html5/>
- Sintaxis básica

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Sample page</title>
  </head>
  <body>
    <h1>Sample page</h1>
    <p>This is a <a href="demo.html">simple</a> sample.</p>
    <!-- this is a comment -->
  </body>
</html>
```

Novedades

Nuevas etiquetas semánticas que ayudan a definir la estructura de una página HTML 5



Novedades

- **←HEADER→: contenido introductorio.** Un elemento header normalmente contiene una sección de encabezado (un elemento h1-h6), pero puede contener otro tipo de elementos, como una tabla de contenidos, un formulario de búsqueda o cualquier logo importante.
- **←NAV→:** Sección de una página que enlaza con otras páginas o partes de la misma.
- **←FOOTER→:** Representa el pie de una sección. Un pie contiene información acerca de la sección de la página, como quién la codificó, copyright y similares..

Novedades

<ARTICLE>:

Este elemento representa un contenido completo, con el objetivo de ser distribuido y reutilizado. Por tanto, permite encapsular contenidos que tiene significado en sí mismo.

- <SECTION>: Representa una sección genérica de un documento. Una sección, en este contexto, es un grupo temático de contenido (Aparecen en dentro de la definición de <ARTICLE>)
- <ASIDE>: Una sección de una página que consiste en contenido tangencial, puede considerarse separado de este contenido.

Novedades

- Incorpora etiquetas (canvas 2D y 3D, audio y vídeo) con codecs para mostrar directamente **elementos multimedia** (lucha entre codecs libres o privados).
- Etiquetas para manejar grandes **conjuntos de datos** y generar tablas dinámicas (Datagrid, Details, Menu, Command).
- Mejoras en los **formularios**:
 - Nuevos tipos de datos (eMail, number, url, datetime).
 - Validación del contenido sin necesidad de javascript.

Novedades

- **Visores:**
 - MathML (fórmulas matemáticas) y SVG (gráficos vectoriales).
 - Se deja abierto para interpretar otros lenguajes XML.
- **Drag & Drop:**
 - Arrastras objetos.
- **Almacenamiento local**
- **Ubicación geográfica**

Novedades

- **Web semántica:**

- Etiquetas para manejar la Web Semántica: header, footer, article, nav, time (fecha del contenido), link rel=“ (tipo de contenido que se enlaza).
- Los buscadores podrán indexar e interpretar esta meta información.

Novedades

- **Nuevas APIs:**
 - Drag & Drop. Mediante eventos.
 - Trabajar Off-Line. Permite descargar todos los contenidos necesarios y trabajar en local.
 - Geoposicionamiento para dispositivos que lo soporten.
 - API Storage. Facilidad de almacenamiento persistente en local, con bases de datos (basadas en SQLite)

Novedades

- Nuevas APIs :
 - WebSockets. API de comunicación bidireccional entre páginas. Similar a los Sockets de C.
 - WebWorkers. Hilos de ejecución en paralelo.
 - System Information API. Acceso al hardware a bajo nivel: red, ficheros, CPU, Memoria, puertos USB, cámaras, micrófonos ... muy interesante pero con numerosas salvedades de seguridad.

Novedades

- Etiquetas nuevas:

- <article>, <aside>, <audio>, <canvas>, <command>, <datalist>, <details>, <dialog>, <embed>, <figure>, <footer>, <header>, <hgroup>, <mark>, <meter>, <nav>, <output>, <progress>, <ruby>, <rp>, <rt>, <section>, <source>, <time>, <video>

Novedades

- Etiquetas **eliminadas**:
 - Los navegadores no están teniendo en cuenta la eliminación para no perder cuota de mercado
`<acronym>, <applet>, <basefont>, <big>, <center>, <dir>, , <frame>, <frameset>, <isindex>, <noframes>, <s>, <strike>, <tt>, <xmp>`
- Etiquetas **cambiadas**:
 - `<a> href |target |rel |hreflang |media |type,`
`, <cite>, <hr>, <i>, <input> (añadidos 13 elementos a type),`
`<small>, <u>`

Enlaces de interés

- Referencia etiquetas:
 - <http://www.w3schools.com/tags/default.asp>
- Especificaciones:
 - <http://www.w3.org/TR/html5/>
 - <https://html.spec.whatwg.org/multipage/>

CSS - Características

- CSS (Cascading Style Sheets) es un lenguaje usado para definir la **presentación/apariencia** de un documento HTML.
- Por tanto, la idea es **separar la estructura** de un documento de su **presentación**.
- Cuando utilizamos las etiquetas HTML, tienen un estilo por defecto definido, CSS redefine estos atributos para mejorar la presentación de una página.



¿Cómo funciona: Sintaxis?

- En HTML
- Con CSS

```
<body bgcolor="#FF0000">
```

```
body {background-color: #FF0000;}
```

`selector {property: value;}`

↑
A qué etiqueta(s)
HTML se aplica
la propiedad (por
ejemplo, "body")

↖ el valor de la propiedad
"background-color"
podría ser, por ejemplo,
rojo ("#FF0000", valor
en hexadecimal)

La propiedad, por ejemplo,
podría ser el color de fondo
("background-color")

¿Cómo funciona: uso?

- Podemos utilizar CSS en un documento HTML de tres maneras diferentes. Se recomienda utilizar el tercero.
 - Primer método: en línea - inline (elemento HTML) atributo style:
`<h1 style="color: blue;">esto es un título</h1>`
 - Segundo método: cabecera y etiqueta style:

```
<html>
  <head>
    <title>Ejemplo</title>
    <style type="text/css">
      body {background-color: #FF0000;}
    </style>
  </head>
  <body>
    <p>Esta es una página con fondo rojo</p>
  </body>
</html>
```

¿Cómo funciona: uso?

- Podemos utilizar CSS en un documento HTML de tres maneras diferentes. Se recomienda utilizar el tercero.
- – Tercer método, externo (enlace a un fichero con la definición de los estilos (*.css)).

```
<html>
  <head>
    <title>Mi documento</title>
    <link rel="stylesheet" type="text/css" href="style/style.css" />
  </head>
  <body>
    ...
  </body>
</html>
```



CSS – Enlaces de Interés

- Página web CSS (Cascading Style Sheets) del W3C
 - <https://www.w3.org/Style/CSS/>
- Tutoriales de CSS del W3C
 - <https://www.w3.org/Style/Examples/011/firstcss>
 - <https://www.w3.org/Style/LieBos2e/enter/>
 - <http://es.html.net/tutorials/css/>

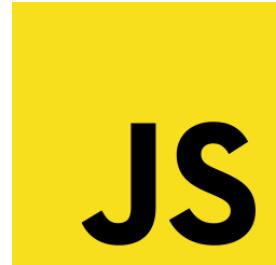
Javascript y frameworks en el lado del cliente

Javier Albert Segui

Curso 2020/21

Indice

- Lenguaje JavaScript
 - ¿Que es JavaScript?
 - Características
 - Ventajas e inconvenientes
 - Qué se puede hacer con JavaScript
 - Cómo incorporar código JavaScript en HTML
 - Modelo de Eventos
 - Clases en JavaScript
- Frameworks en el lado del cliente
 - ¿Que es un framework?
 - JQuery
 - React
 - Vue.js
 -



JavaScript

- Es un lenguaje de programación interpretado.
- Se define como:
 - “Orientado a objetos”
 - Basado en prototipos
 - Débilmente tipado
 - Imperativo y dinámico
- Es un lenguaje derivado del ECMAScript, que es una especificación de lenguaje de programación basada en el javascript original de Netscape.
 - Jscript, JavaScript.....

JavaScript

- Está basado en Objetos, utiliza objetos integrados y extensibles, pero no utiliza los métodos clásicos de la POO de las clases y la herencia
- No es necesaria la declaración de los tipos de variables
- Las referencias entre objetos se verifican en tiempo de ejecución.

JavaScript

○ Ventajas

- Rápido desarrollo de aplicaciones
- Aprendizaje fácil
- Es independiente de la Plataforma
- Consumo mínimo de recursos

○ Inconvenientes

- Conjunto limitado de métodos
- Código visible
- Difícil depuración del código

JavaScript

- Cálculos sencillos
- Chequeo de formularios
- Interactividad en las páginas web, mediante el uso de eventos.

JavaScript

- Las formas básicas para incorporar código JavaScript en una página HTML son:
 - Embebiendo el código JavaScript en nuestra página

```
<SCRIPT type="text/javascript">  
    //Sentencias JavaScript  
</SCRIPT>
```

- Importando un fichero JavaScript en la cabecera de nuestra página

```
<SCRIPT type="text/javascript" src="fuente.js"></SCRIPT>
```

JavaScript – Modelo de eventos

- Los eventos **suceden a tres niveles:**
 - A nivel del documento **HTML**
 - A nivel de un **formulario individual**
 - A nivel de un **elemento de un formulario**
- El evento es gestionado por una sección de código en JavaScript (Gestor de Eventos)
- Declaración de Gestores de Eventos: similar a los atributos en HTML

```
<BODY onLoad="cargarfuncion()" onUnload="descargarfuncion()">

    <FORM name="nombre_del_formulario" ...
          onSubmit="función_o_sentencia">

        <INPUT type="button" name="mycheck" value="HA!" onClick=
              "alert('Te he dicho que no me aprietas')">
```

JavaScript – Modelo de eventos

Evento	Ocurre Cuando	Gestor
blur	El usuario quita el cursor de un elemento de formulario	onBlur
click	El usuario clica un link o un elemento de formulario	onClick
change	El usuario cambia el valor de un texto, un área de texto o selecciona un elemento.	onChange
focus	El usuario coloca el cursor en un elemento de formulario.	onFocus
load	El usuario carga una página en el Navegador	onLoad
Mouseover	El usuario mueve el ratón sobre un link	onMouseOver
Select	El usuario selecciona un campo del elemento de un formulario	onSelect
Submit	Se envía un formulario	onSubmit
Unload	Se descarga la página	onUnload

JavaScript

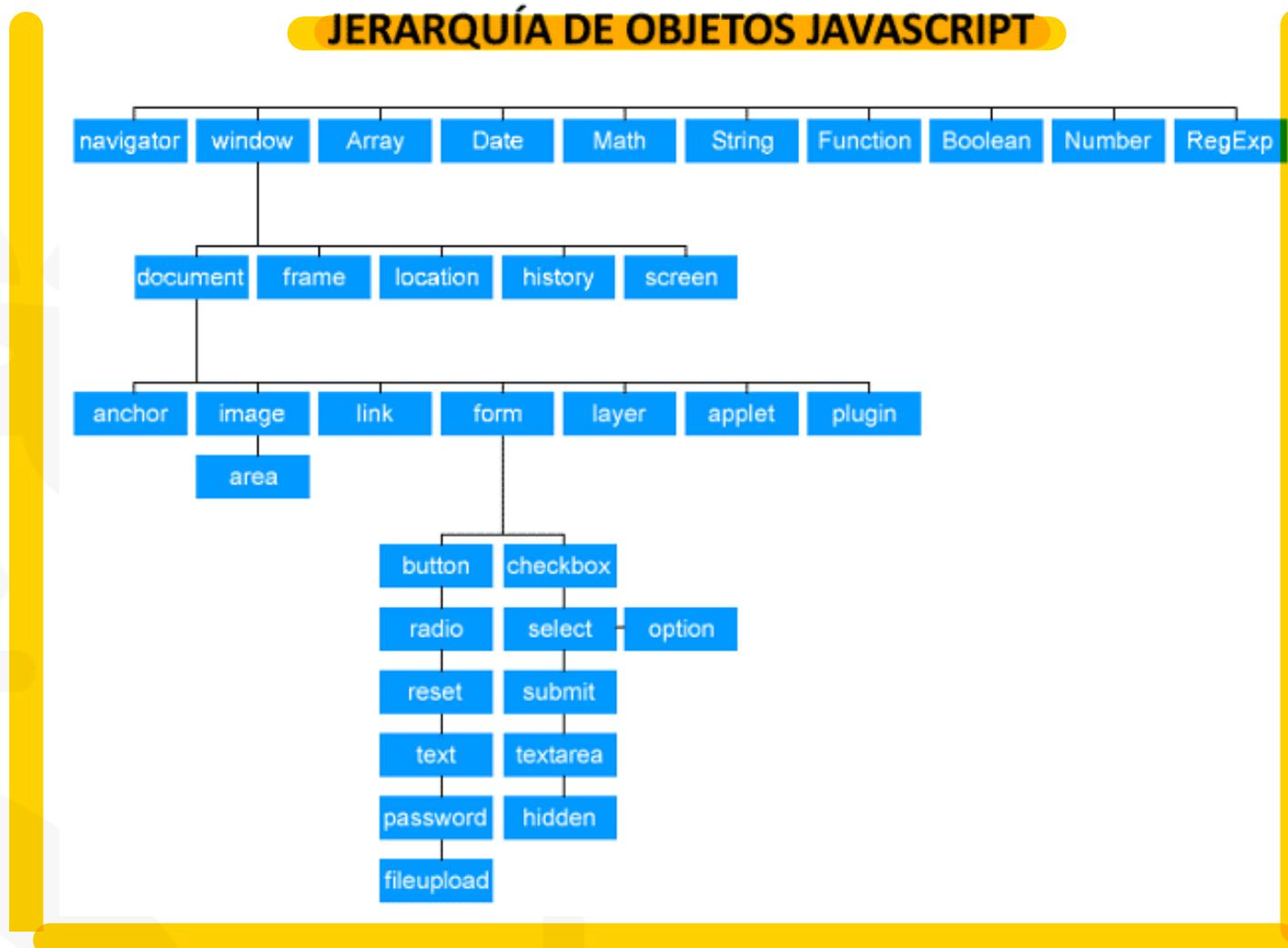
```
<HTML>
<HEAD>
  <SCRIPT>
    <!--
      function Alarma() { alert("Hola que tal estás"); return true;
    }
    // -->
  </SCRIPT>
</HEAD>
<BODY>
<A HREF="eventos.html" onMouseOver="Alarma()">
</BODY>
</HTML>
```

Pasa por aquí encima

JavaScript

- Clases Predefinidas
 - Clase String: Cada vez que se asigna una cadena de caracteres a una variable, se crea un objeto de la clase String
 - Clase Math: Se usa para efectuar cálculos matemáticos
 - Clase Date: Para el manejo de fechas y horas
 -
- Clases del Navegador
 - Tienen que ver con la navegación
- Clases del Documento HTML
 - Están asociadas con cualquier elemento de una página Web (link, ancla, formulario, etc.)
- Clases definidas por el usuario.

JavaScript



JavaScript

○ Clase Window

- Es el nivel más alto de la jerarquía de objetos de JavaScript.
- El resto de los objetos desciende siempre del objeto Window.
- Cada ventana se asocia a una estructura HTML de esa página que se refleja en el objeto document.
- Además, cada ventana se corresponde con alguna URL que se refleja en el objeto location. Cada ventana tiene una lista de documentos visitados que se han mostrado en esa ventana, que se agrupan en objeto history.

○ Los métodos de un objeto Window son:

- alert(string_mensaje)
- confirm(string_mensaje)
- open(URL_string, nombre_ventana)
- close()
- prompt(string_mensaje)

JavaScript

```
<HTML>
  <HEAD>
    <TITLE>Ejemplo sencillo de página HTML</TITLE>
  </HEAD>
  <BODY>
    <A name="principio">Este es el principio de la página</A> // ancla
    <HR>
    <FORM method="POST">
      <P> Introduzca su nombre:<INPUT type="text" name="me" size="70">
    </P>
      <INPUT type="reset" value="Borrar Datos">
      <INPUT type="submit" value="OK">
    </FORM>
    <HR>
    Clica aquí para ir al
    <A href="#principio">principio</A> de la página // link
  </BODY>
</HTML>
```

document.title

document.anchors[0].name

document.forms[0].method

document.forms[0].elements[1].value

document.links[0].href

Frameworks JavaScript

¿Qué es un framework?

- **Definición:** es una estructura conceptual y tecnológica de asistencia definida, normalmente, con artefactos o módulos concretos de software, que puede servir de base para la organización y desarrollo de software.
- Básicamente aglutanan bibliotecas de código y patrones de diseño y sirven como base para el desarrollo de aplicaciones.
- Ventajas:
 - Facilita la colaboración entre los miembros del proyecto
 - Normalmente nos da un esqueleto sobre el que construir nuestra aplicación

JQuery

- Es una librería multiplataforma de JavaScript que nos permite simplificar la manera de interactuar con el HTML, manipular los objetos del DOM, manejar eventos.....
- Se basa en la separación del HTML y del JavaScript.
- Intenta eliminar las incompatibilidades entre navegadores
- Es extensible
- JQueryUI: es una librería adicional a JQuery para la creación de interface de usuario ricas
- JQueryMobile: es una librería adicional de JQuery para la creación de aplicaciones en tabletas y móviles.

Angular

- Es un framework para aplicaciones web de código abierto, mantenido por Google que se utiliza para crear y mantener aplicaciones de una página (SPA).
- Se basa en crear una serie de etiquetas personalizadas dentro de nuestro código html que la librería se encarga de procesar.

React

- Es una librería JavaScript diseñada para la creación de **interfaces de usuario**, centrándose principalmente en las aplicaciones de una sola página (**SPA**)
- Es software libre, inicialmente **creada por FBK**.
- Características:
 - **DOM virtual**, se usa para determinar que partes del DOM han cambiado y actualizar solo esas partes de la página.
 - **Propiedades**: son atributos de configuración de los componentes.
 - **Estado**: los componentes pueden tener o no tener estado.
 - Tiene un “lenguaje” propio parecido a **HTML → JSX**

Vue.JS

- Es un framework JavaScript de código abierto para la construcción de interfaces de usuario y SPA.
- Cuenta con una arquitectura de adaptación gradual y composición de componentes.
- La biblioteca “central” solo se centra en la capa de vista de la aplicación.
- Para añadir otras funcionalidades se añaden componentes.

Polymer

- Es una biblioteca JavaScript de código abierto para la creación de aplicaciones web utilizando componentes web.
- Desarrollada por Google utilizando los principios de material design.

Recursos

- o <https://www.w3schools.com/>
- o <http://jquery.com>
- o <http://angular.io>
- o <https://vuejs.org/>
- o <https://es.reactjs.org/>

Tecnologías en el Servidor

Java - Servlets

Curso 2020/21

Javier Albert Segui

Índice

Arquitectura Web - Recordatorio

- La Arquitectura de las aplicaciones Web suelen presentar un esquema de tres niveles:
 - El **primer nivel** consiste en la capa de **presentación** que incluye no sólo el **navegador**, sino también el **servidor web** que es el responsable de dar a los datos un formato adecuado.
 - El **segundo nivel** está referido habitualmente a algún tipo de programa o script - **lógica de negocio**.
 - **Finalmente, el tercer nivel** proporciona al segundo los **datos** necesarios para su ejecución.

Java EE

- Java Platform Enterprise Edition (antes J2EE, ahora Jakarta EE) es un **estándar** para el desarrollo de **aplicaciones empresariales** (portables, robustas, escalables y seguras) usando tecnología Java.
- Jakarta EE es un conjunto de **especificaciones**, no es un producto. Los productos que cumplen con la especificación son realizados por terceras empresas u organizaciones.
- Toda la información acerca de esta especificación la podemos encontrar en <https://eclipse-ee4j.github.io/jakartaee-platform/>

Jakarta EE

- Ahora la versión de esta especificación es la v8 publicada en septiembre/17
- Incluye:
 - Jakarta WebSocket
 - Jakarta Servlet
 - Jakarta Server Page
 - Jakarta JSON Processing
 - Jakarta Standard Tag Library
 - Jakarta Server Faces
 - Jakarta RESTful Web Services

Tecnologías Java

- En JAVA tenemos dos tecnologías distintas para el Desarrollo de aplicaciones en un entorno Web:
 - Servlets
 - JSP
- Aunque distintas, están íntimamente ligadas, dado que la tecnología JSP al final se transforma en un servlet que son las aplicaciones que se ejecutan sobre el servidor de aplicaciones.

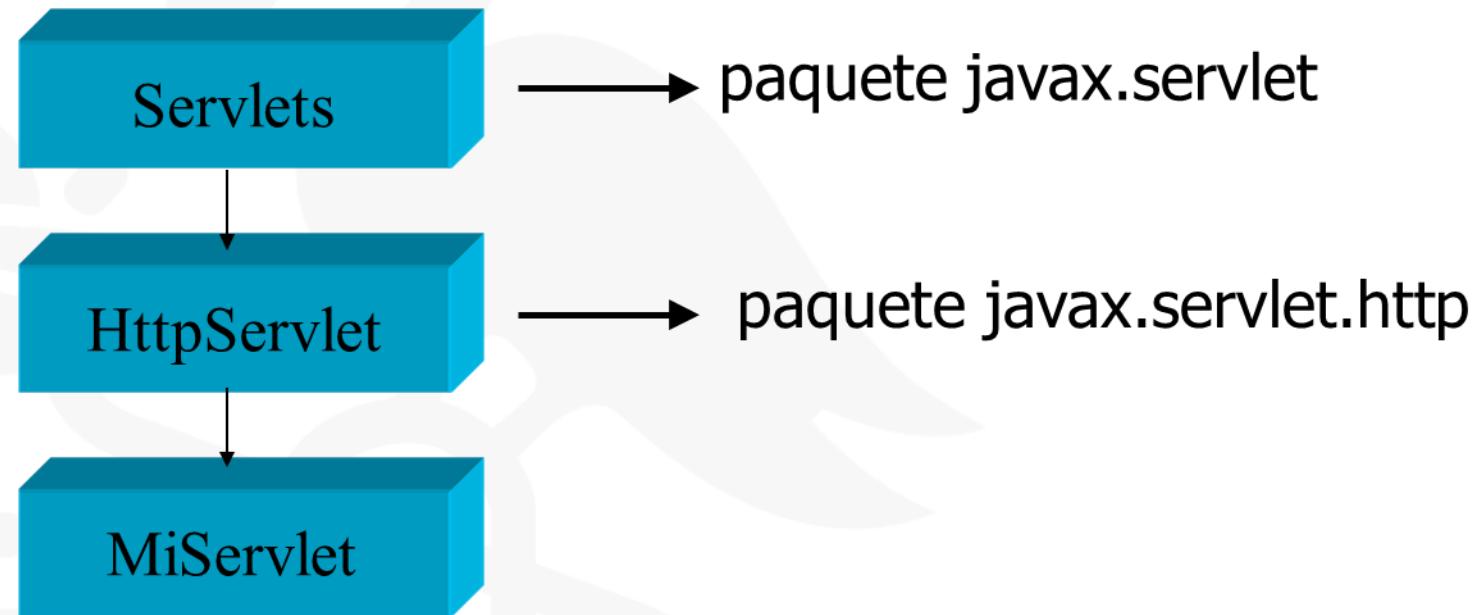
Servlets

- Se puede definir un **Servlet** como un **programa JAVA** que se ejecuta en un **entorno distribuido en red**, como un **servidor web**, y que recibe y responde a las peticiones de un cliente a través del protocolo **HTTP**.
- Son la contraparte a tecnologías como ASP.net y PHP
- Los **Servlets** son un **reemplazo efectivo** para los **CGI** en los servidores que los soporten ya que proporcionan una forma de **generar documentos dinámicos** utilizando las **ventajas de la programación en Java** como **conexión a base de datos**, **manejo de peticiones concurrentes**, **programación distribuida**, etc.
 - Por ejemplo, un servlet podría ser responsable de procesar los datos desde un formulario en HTML como registrar la transacción, actualizar una base de datos, contactar algún sistema remoto y retornar un documento dinámico o redirigir a otro servlet u alguna otra cosa.

Servlets

- Los servlets son independientes del servidor y de la Plataforma en que se ejecuta.
- En el servidor tendremos que tener una maquina virtual JAVA, además de las librerías necesarias para el soporte de los servlets.

Servlets - API



Servlets - API

- Hay dos paquetes dentro del API de Servlets
 - javax.servlet
 - javax.servlet.http

- **Javax.servlet**

- Interfaz Servlet
 - Interfaz ServletRequest
 - Interfaz ServletResponse
 - Interfaz ServletConfig
 - Interfaz ServletContext
 - Interfaz SingleThreadModel
 - Clase GenericServlet

- javax.servlet.http**

- Interfaz HttpServlet
 - Interfaz HttpServletRequest
 - Interfaz HttpServletResponse
 - Interfaz HttpSession

Servlets - API

○ Clase HttpServlet

- Extiende de la clase GenericServlet y proporciona una implementación de la interfaz Servlet mucho más específica para el protocolo HTTP.
Esta es la clase que extienden la mayoría de los servlets que hay en la actualidad.
- Métodos (throws ServletException, IOException)
 - Gestionan el servicio
 - GET/ POST service (HttpServletRequest req, HttpServletResponse res)
 - GET/ POST processRequest (HttpServletRequest req, HttpServletResponse res) → NetBeans
 - Implementan operaciones propias de HTTP
 - GET doGet (HttpServletRequest req, HttpServletResponse res)
 - POST doPost (HttpServletRequest req, HttpServletResponse res)

Servlet - API

○ **Javax.servlet.http.HttpServletRequest**

- Una interfaz que encapsula la funcionalidad de las peticiones que hace el cliente.
- **IMPORTANTE:** Uno de los métodos de esta interfaz es el método `getParameter (String Name)` utilizado en la mayoría de los servlets que recuperan valores de los de los formularios HTML rellenados por los clientes de las aplicaciones.
- Un objeto `HttpServletRequest` se pasa como parámetro al método `service` de la clase `HttpServlet`.

Servlet - API

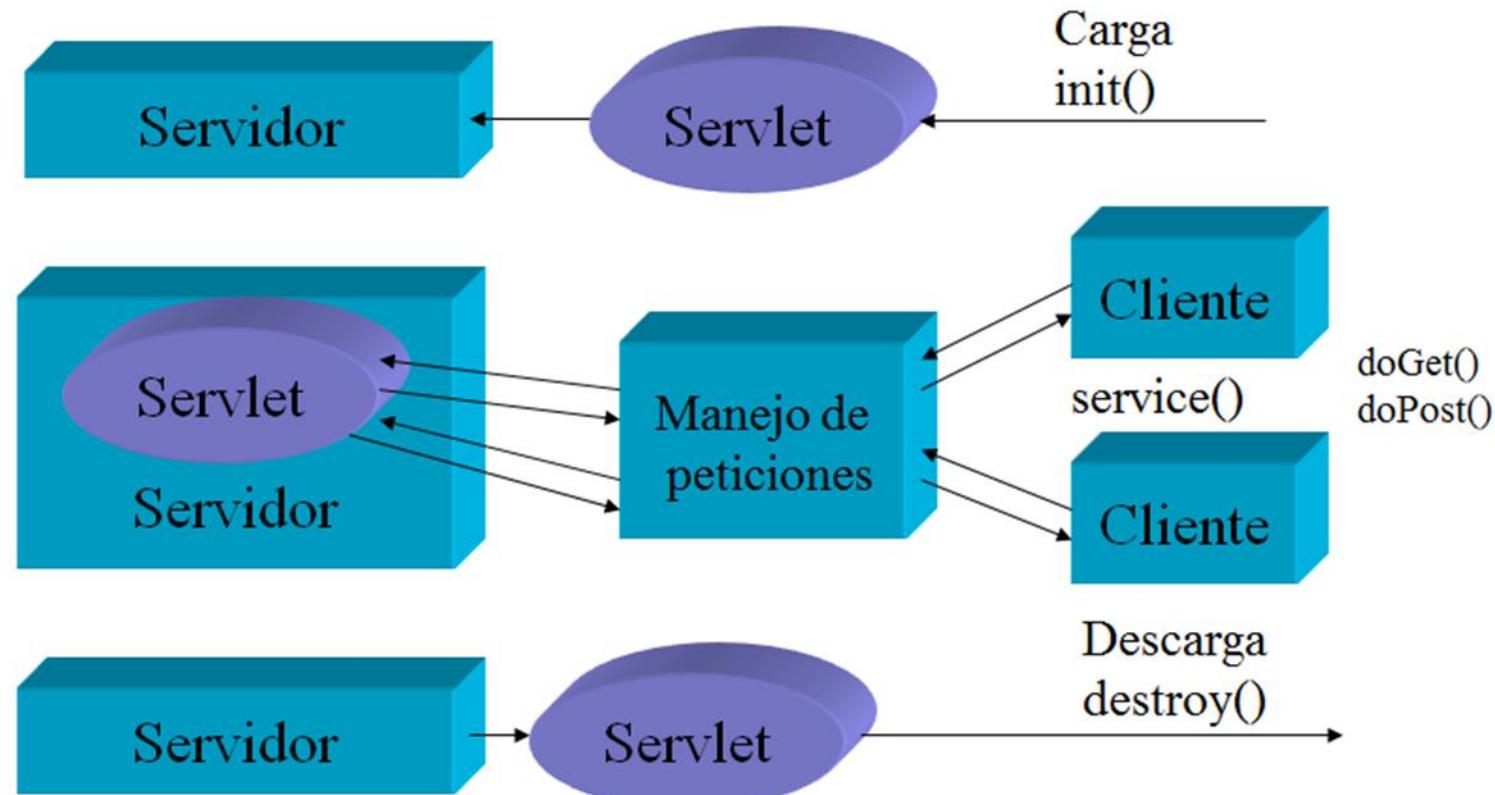
- **Javax.servlet.http.HttpServletResponse**

- Esta interfaz encapsula la funcionalidad de una respuesta HTTP, incluyendo el manejo de las cabeceras del propio protocolo.
- Interfaz que encapsula la funcionalidad de la respuesta MIME que será enviada al cliente.
- Un objeto HttpServletResponse se pasa como parámetro al método service de la clase HttpServlet.

Servlet – Ciclo de vida

- La interfaz que se utiliza para implementarlos define una serie de métodos para cada una de las etapas.
- El orden es el siguiente:
 - Cuando el servidor carga el Servlet invoca al método **init**.
 - Una vez ejecutado el método **init**, todas las peticiones serán atendidas por el método **service**.
 - Por último, cuando el servidor web descarga al Servlet se invocará al método **destroy**

Servlet – Ciclo de vida



Acceso a datos

- **JDBC**, Java DataBase Connectivity, es una API de Java que se usa para acceder a **Bases de Datos**, tanto locales como remotas.
- Existe una **independencia del SGBDR**: Si cambia el gestor se minimizan los cambios en la aplicación.
- **Lenguaje** de consulta **SQL**.
- El **API JDBC (java.sql)** está formada por cinco grupos:
 - Gestión de Drivers.
 - API para manejadores JDBC.
 - Excepciones.
 - API (trabajar con los datos)
 - Utilidades.

Gestión de Sesiones

- El protocolo HTTP **NO** posee la capacidad de almacenar **estados**, ¡es un protocolo sin estado!)
- Posibles soluciones:
 - Cookies.
 - Añadir información en la URL
 - Usar campos ocultos de formularios (HIDDEN)
 - Empleo del objeto HttpSession del servlet.

Gestión de Sesiones

- Los **servlets** proporcionan una **API** denominada **HttpSession**.
- Una interfaz de alto nivel **construida sobre cookies** y la reescritura de las urls (de manera transparente para el desarrollador).
- Permite **almacenar objetos de cualquier tipo**.

Gestión de Sesiones

- Pasos para el manejo de sesiones con Servlets
 - Crear un objeto HttpSession: Usaremos el método getSession de HttpServletRequest
 - Añadir / Recuperar información:
 - getAttribute("nombre_variable"): devuelve un Object en caso de que la sesión tenga ya contenido en esa variable, null si no se ha utilizado nunca.
 - setAttribute("nombre_variable",referencia): OJO si el objeto existe, se sobreescribe.
 - getAttributes():retorna un tipo Enumeration con los nombres de todos los atributos establecidos en la sesión del usuario.