INGENIERÍA WEB Y COMPUTACIÓN EN LA NUBE

Bloque1: Introducción a la ingeniería web

TEMA 1.2: TECNOLOGÍAS DE DESARROLLO DE APLICACIONES WEB

Boni García
boni.garcia@urjc.es

Universidad Rey Juan Carlos

Índice de contenidos

Índice de contenidos

1. Introducción

- El impacto de la Web ha propiciado la aparición de una gran cantidad de tecnologías, librerías, herramientas y estilos arquitectónicos para desarrollar una aplicación web
- Es conveniente conocer los elementos más importantes desde un punto de vista de alto nivel para tener una visión global de la programación web
- Existen dos enfoques en el desarrollo de aplicaciones web:
  - Creación de aplicaciones web con **tecnologías de desarrollo**
  - Creación de aplicaciones web con **sistemas gestores de contenido**

1. Introducción

Tecnologías de desarrollo

- **Tecnologías de cliente:** Tecnologías que permiten crear interfaces de usuario atractivos y permiten la comunicación con el servidor. Basadas en **HTML, CSS y JavaScript**.
- **Tecnologías de servidor:** Tecnologías que permiten implementar el comportamiento de la aplicación web en el **servidor**: lógica de negocio, generación de informes, compartir información entre usuarios, envío de correos, etc…
- **Bases de datos:** La gran mayoría de las webs necesitan guardar información. Las bases de datos son una parte esencial del desarrollo web.

1. Introducción

Arquitectura de aplicaciones web

- Existen diferentes **arquitectura** de aplicación web en función de las tecnologías que usan y cómo se usan:
  - Página web estática
  - Página web interactiva
  - Aplicación web con cliente estático
  - Aplicación web interactiva
  - Aplicación web con AJAX
  - Aplicación web SPA

1. Introducción

Sistemas gestores de contenido

- Existen aplicaciones web cuya principal funcionalidad es la **publicación de contenido**: blogs, páginas de empresas, organismos públicos, etc.
- Todas estas webs tienen **mucho en común**, prácticamente sólo se **diferencian en el contenido** y en el aspecto gráfico
- Para desarrollar este tipo de webs, en vez de desarrollar la web con técnicas de desarrollo, se **usa una aplicación ya creada que se puede personalizar y adaptar** (mayormente vía web)
- A las aplicaciones de este tipo se las denomina **Sistemas Gestores de Contenido** (**CMSs**).

Índice de contenidos

2. Arquitecturas de aplicaciones web

Arquitectura	Cliente	Servidor
Página web estática	Estático. HTML y CSS	Estático. Recursos en disco duro
Página web interactiva	Dinámico. JavaScript	Estático. Recursos en disco duro
Aplicación web con cliente estático	Estático. HTML y CSS	Dinámico. Ejecución código
Aplicación web interactiva	Dinámico. JavaScript	Dinámico. Ejecución código
Aplicación web con AJAX	Dinámico. JavaScript	Dinámico. Ejecución código
Aplicación web SPA	Dinámico. JavaScript	Dinámico. Ejecución código

2. Arquitecturas de aplicaciones web

Página web estática

- El navegador hace **petición** al servidor mediante **HTTP**
- El **servidor** transforma **URL** a ruta en **disco**
- El **servidor** devuelve el **fichero** de disco al **navegador**
- El navegador **visualiza** (*renderiza*) la página **HTML** con estilos CSS e imágenes (**sin JavaScript**).
- Cuando el usuario hace **clic en un enlace**, el navegador repite el proceso con la URL del link y **recarga** por **completo** la página web

2. Arquitecturas de aplicaciones web

Página web estática

- Con esta arquitectura el servidor siempre devuelve los **mismos recursos**
- Desde el punto de vista del servidor, la **web es estática**
- La web está formada por **HTML, CSS, imágenes, PDF, etc…** (pero no incluye JavaScript)
- La Web se diseñó con esta arquitectura
- Al principio todas las páginas web eran así (no existía el concepto de aplicaciones web)

2. Arquitecturas de aplicaciones web

Página web estática

- Actualmente esta arquitectura se usa principalmente para:
  - Páginas personales
  - Páginas de proyectos software
  - Documentación técnica (JavaDoc en Java, Maven site, etc…)

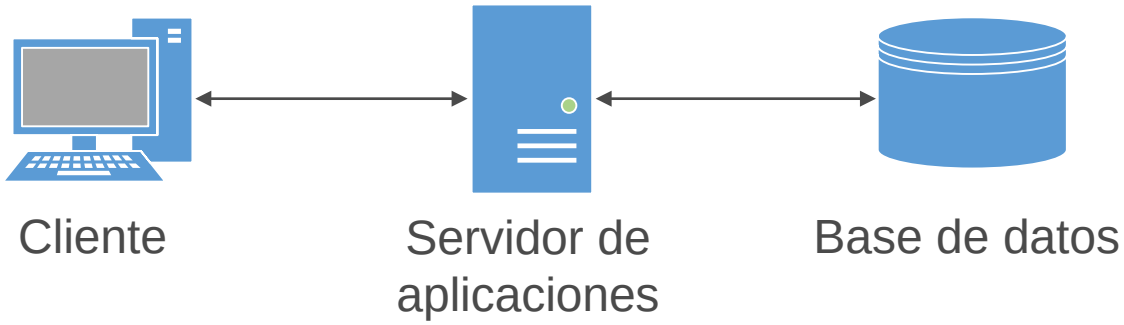
2. Arquitecturas de aplicaciones web

Página web interactiva

- El contenido de la página web está alojado en el **disco duro del servidor** (estático)
- El cliente es dinámico porque las páginas incluyen código **JavaScript** que se ejecuta en el **navegador**
- Este **JavaScript** se usa para incluir **efectos gráficos**:
  - Efectos gráficos que no se pueden implementar con **CSS**
  - **Mostrar u ocultar información** en función de los elementos que se seleccionan (para documentos largos)
  - **Menús** desplegables
  - Páginas adaptables para **móviles** (***responsive***)

2. Arquitecturas de aplicaciones web

Aplicación web con cliente estático



- Es un ejemplo de arquitectura de 3 capas:
  - Navegador: Capa de presentación
  - Servidor web: Capa de aplicación (lógica de negocio)
  - Base de datos: Capa de datos

2. Arquitecturas de aplicaciones web

Aplicación web con cliente estático

- Cuando el servidor web recibe una **petición**, dependiendo de la URL:
  - Devolver contenido del **disco**
  - Ejecutar código para **generar el recurso dinámicamente**
- Cuando se **ejecuta código**, normalmente se hacen **consultas a una base de datos** para recuperar la información
- Lo más habitual es que se **genere la página HTML** de forma dinámica (pero también puede generar imágenes, PDFs, etc…)
- Si el usuario pulsa un link, se **recarga la página al completo**

2. Arquitecturas de aplicaciones web

Aplicación web con cliente estático

- Es la arquitectura de las primeras **aplicaciones web**
- Todavía sigue habiendo **muchas webs** con esta arquitectura
- El contenido es dinámico, porque se **ejecuta código en el servidor** para generar dicho contenido
- La experiencia de usuario antes no era muy buena:
  - **Conexiones lentas** implican tiempos de carga apreciables en cada click
  - **La recarga completa** de la página ofrece una mala experiencia de usuario (página en blanco)
- Pero ha mejorado:
  - **Mayor velocidad** de Internet (menos tiempo de espera)
  - Navegadores muestran la **nueva página una vez cargada** (sin pasar por la página en blanco)

2. Arquitecturas de aplicaciones web

- La mayoría de las aplicaciones web **actuales** son **dinámicas** tanto en **cliente** como en **servidor**
- Dependiendo de cómo se use el **JavaScript** en el **cliente** se diferencian tres arquitecturas:
  - **Aplicación web interactiva**
  - **Aplicación web con AJAX**
  - **Aplicación web SPA**
  - s muestran la **nueva página una vez cargada** (sin pasar por la página en blanco)

2. Arquitecturas de aplicaciones web

Aplicación web interactiva

- El JavaScript se utiliza para crear efectos gráficos
- El **dinamismo en el cliente** se utiliza exactamente igual que en las **páginas web interactivas**
- **JavaScript** se diseñó, entre otras cosas, para añadir **efectos gráficos** básicos a las páginas cuando el **CSS** era muy limitado
- La **gran mayoría** de las aplicaciones web que existen en Internet siguen esta **arquitectura**

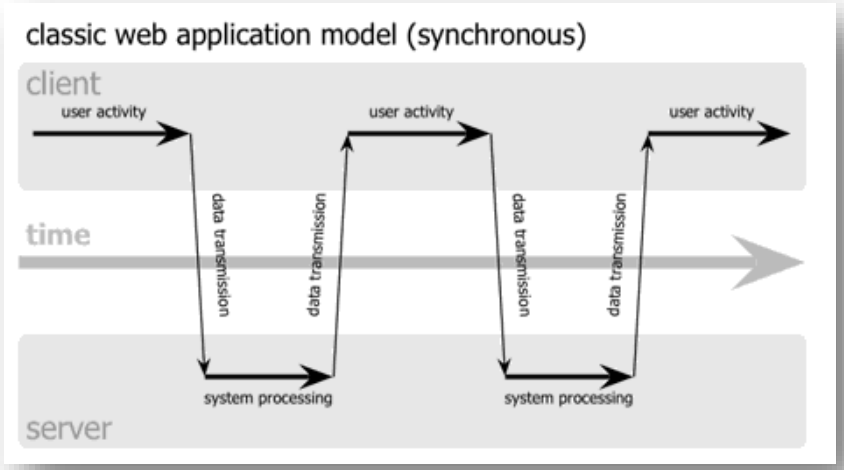
2. Arquitecturas de aplicaciones web

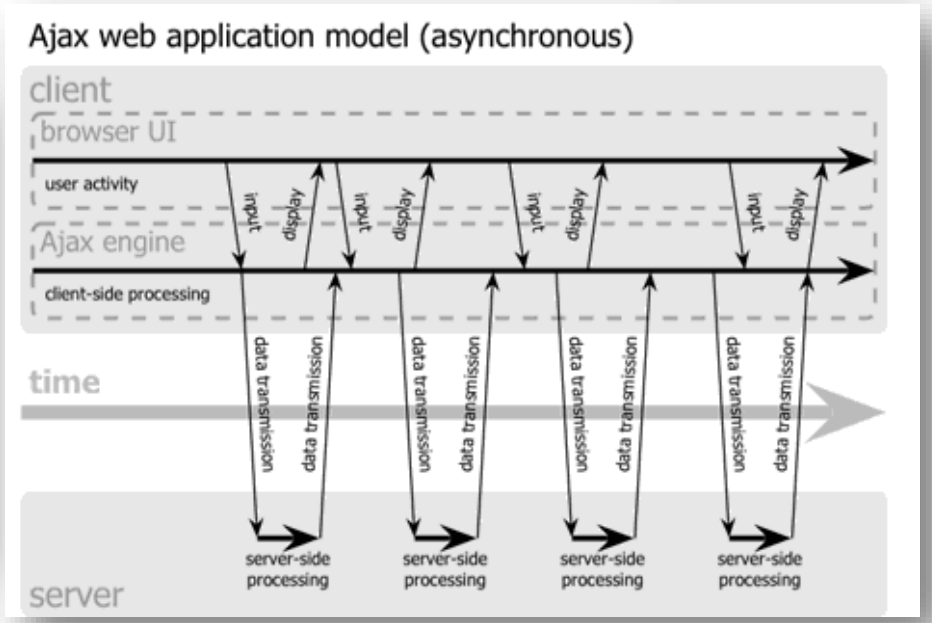
Aplicación web con AJAX

- JavaScript se usa para no tener que **recargar completamente** la página al pulsar un link
- Permite hacer petición al servidor web en **segundo plano** (oculta al usuario)
- Cuando llega al navegador el **resultado de la petición**, el código JavaScript **actualiza** aquellas **partes** de la **página** necesarias
- A esta técnica se la conoce como **AJAX** (***Asynchronous JavaScript And XML***)

2. Arquitecturas de aplicaciones web

Aplicación web con AJAX





2. Arquitecturas de aplicaciones web

Aplicación web con AJAX

- Usar **AJAX** en una página **mejora** mucho la **experiencia de usuario**
- No es necesario **recargar la página al completo**, sólo aquellas partes que cambian (p.e. se puede dejar el menú fijo)
- La página se puede **cargar por partes**, primero la información **importante** y en segundo plano otros elementos **complementarios** (p.e. los botones de compartir, los comentarios en un blog…)
- Se puede dar **realimentación** al usuario de formas más **adecuadas** (cuadro de diálogo, error de validación en un formulario, quitar el icono de carga de un recurso, etc…)

2. Arquitecturas de aplicaciones web

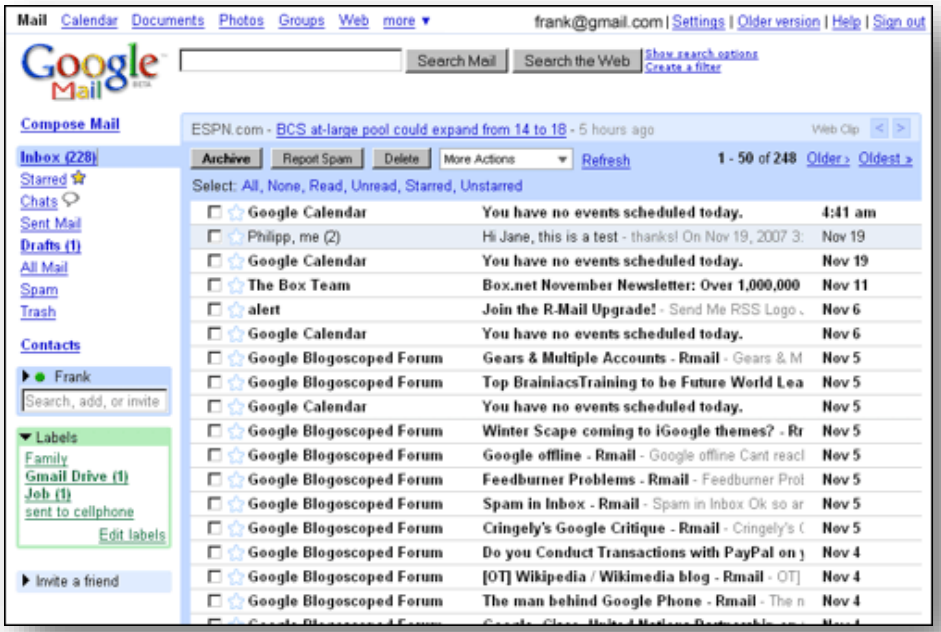
Aplicación web SPA

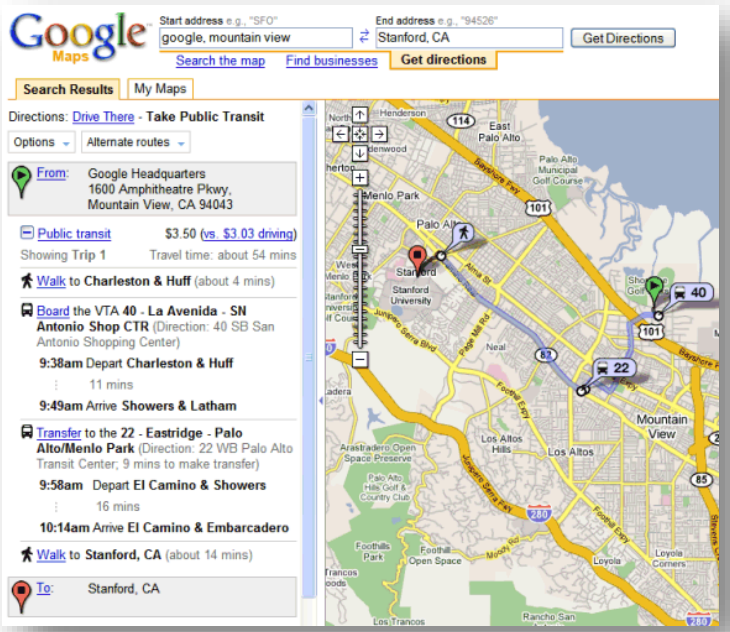
- **SPA (*Single Page Application*)**
- La técnica **AJAX** se puede llevar al **extremo** y que **todo** el contenido dinámico se cargue con **JavaScript en segundo plano**
- Existe una **única página** cuyo contenido va cambiando según el usuario interactúa con botones, pestañas, etc.
- El **botón de atrás** del navegador funciona porque se "emula" una navegación por páginas cuando se evoluciona por los estados de la aplicación

2. Arquitecturas de aplicaciones web

Aplicación web SPA

- Google popularizó AJAX y SPA con Gmail y Maps

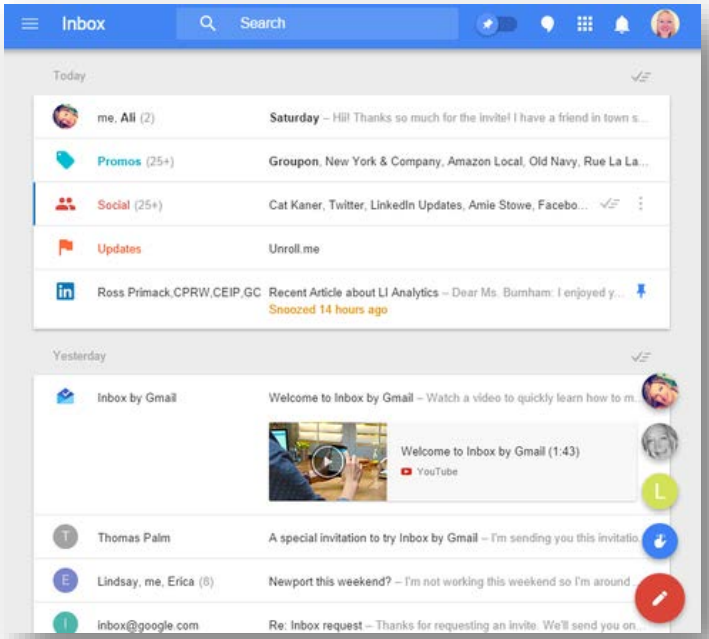


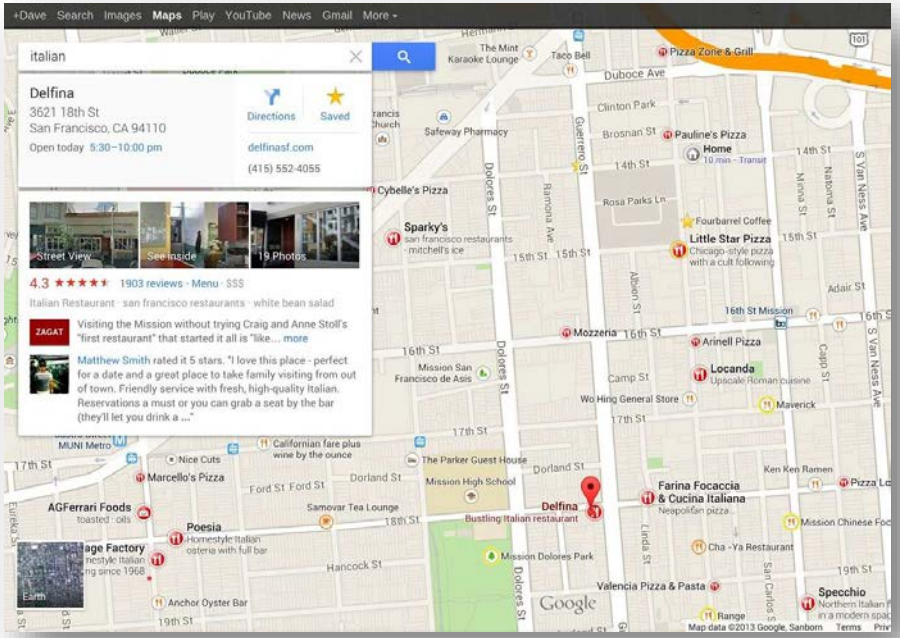


2. Arquitecturas de aplicaciones web

Aplicación web SPA

- Google popularizó AJAX y SPA con Gmail y Maps





Índice de contenidos

3. Tecnologías del cliente

- El cliente web por excelencia es el **navegador web**
- Existen un conjunto de **estándares web**, definidos por el **W3C**, que todo navegador debería implementar
- Existen un conjunto de **tecnologías no estándar** que algunos navegadores implementan para la construcción de aplicaciones **avanzadas** y acceso a contenido **multimedia**

3. Tecnologías del cliente

Estándares web

- El W3C (*World Wide Web Consortium*) es una comunidad internacional que desarrolla estándares abiertos para la Web

Scripting and Ajax

HTML & CSS

Gráficos

Accesibilidad

Audio & video

Web semántica

XML

Servicios web

http://www.w3.org

3. Tecnologías del cliente

Estándares web - HTML

- La versión actual es **HTML5**
- Ha supuesto una **revolución** para el dinamismo en el cliente porque ofrece muchas **librerías/tecnologías avanzadas**:
  - Multimedia: etiquetas vídeo, audio y canvas
  - Comunicaciones: websockets
  - Concurrencia: webworkers

3. Tecnologías del cliente

Estándares web - CSS

- CSS es un lenguaje usado para definir la **presentación de un documento** estructurado escrito en HTML
- Su versión actual es **CSS3**

3. Tecnologías del cliente

Estándares web - JavaScript

- Las páginas web se pueden dinamizar con **JavaScript**
- Se puede modificar la página y ejecutar código cuando se interactúa con ella mediante la API **DOM** (*Document Object Model*)
- JavaScript es un lenguaje de programación basado en el estándar **ECMAScript** de **ECMA** (otra organización diferente al **W3C**)
- Hay ligeras **diferencias** en la implementación de JavaScript de los navegadores, aunque actualmente todos son bastante **compatibles** entre sí
- Aunque algunos elementos de la sintaxis recuerden a Java, el lenguaje es muy diferente a **Java**. El **nombre JavaScript** se eligió al publicar el lenguaje en una época en la que Java estaba en auge y fue principalmente por marketing

http://www.ecma-international.org/

3. Tecnologías del cliente

Estándares web - Librerías JavaScript

- Existen multitud de **bibliotecas** (APIs) JavaScript para el desarrollo de aplicaciones
- Algunas de las más **populares**:
  - **jQuery**: es un recubrimiento de la API DOM que aporta facilidad de uso, potencia y compatibilidad entre navegadores. Se usa para gestionar el interfaz (la página) y para peticiones **AJAX**.
  - **underscore.js:** Librería para trabajar con estructuras de datos con un enfoque funcional. También permite gestionar plantillas (*templates*) para generar HTML partiendo de datos

3. Tecnologías del cliente

Estándares web - Librerías JavaScript

- También existen **frameworks del alto nivel** que estructuran una aplicación de forma completa. Especialmente en **aplicaciones SPA**
- Los más populares son Angular.js, Backbone.js y Ember

3. Tecnologías del cliente

Tecnologías no estándar en la Web

- Adobe Flash
  - Es una tecnología **propietaria y cerrada**
  - Es **gratuita** para los usuarios, pero los desarrolladores y servidores que usen ciertas características tienen que **pagar** licencia
  - Es una tecnología usada principalmente para incrustar contenido multimedia interactivo en páginas web
  - Durante muchos años fue la única forma de tener interactividad, animaciones, vídeos, juegos… en la Web
  - Fue acusada de que **no era eficiente, ni abierta** y por tanto, **no es el futuro** de la Web (Steve Jobs, Abril 2010)
  - Adobe lo acabó reconociendo y no la desarrolló más (Nov 2011)

3. Tecnologías del cliente

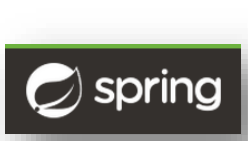
Conclusiones

- Si no hay un motivo importante, todas las aplicaciones web deberían implementarse con **estándares**
- En un mundo con **multitud de dispositivos** conectados a la red, es la única forma de la web sea accesible desde **todos** ellos
- **HTML5** se ha convertido en la **tecnología estándar** para multitud de plataformas diferentes
- Para saber qué estándares soporta cada versión de cada navegador, se puede usar la web http://caniuse.com/

Índice de contenidos

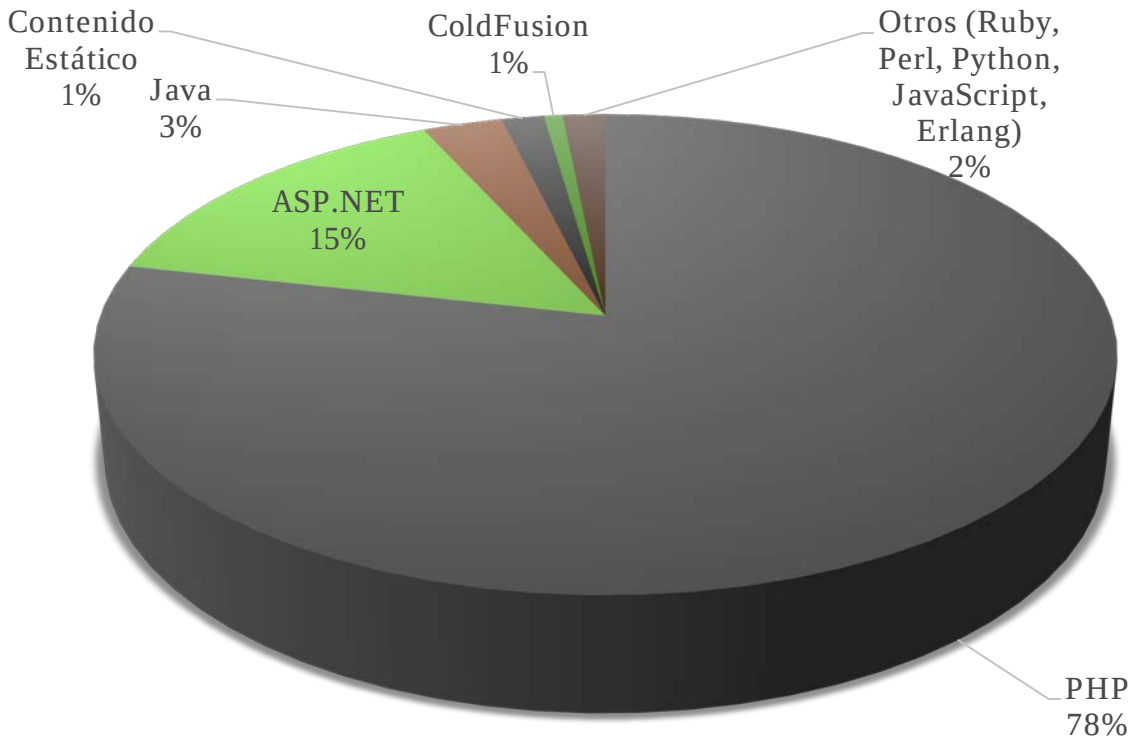
4. Tecnologías del servidor

- Los **estándares son muy importantes en los navegadores web** porque la web tiene que ser compatible con cualquier dispositivo
- En cambio los **estándares no son necesarios en el servidor**, porque cada organización desarrollará su aplicación en el servidor con la tecnología de su elección
- En el servidor, se pueden usar multitud de **tecnologías**



4. Tecnologías del servidor

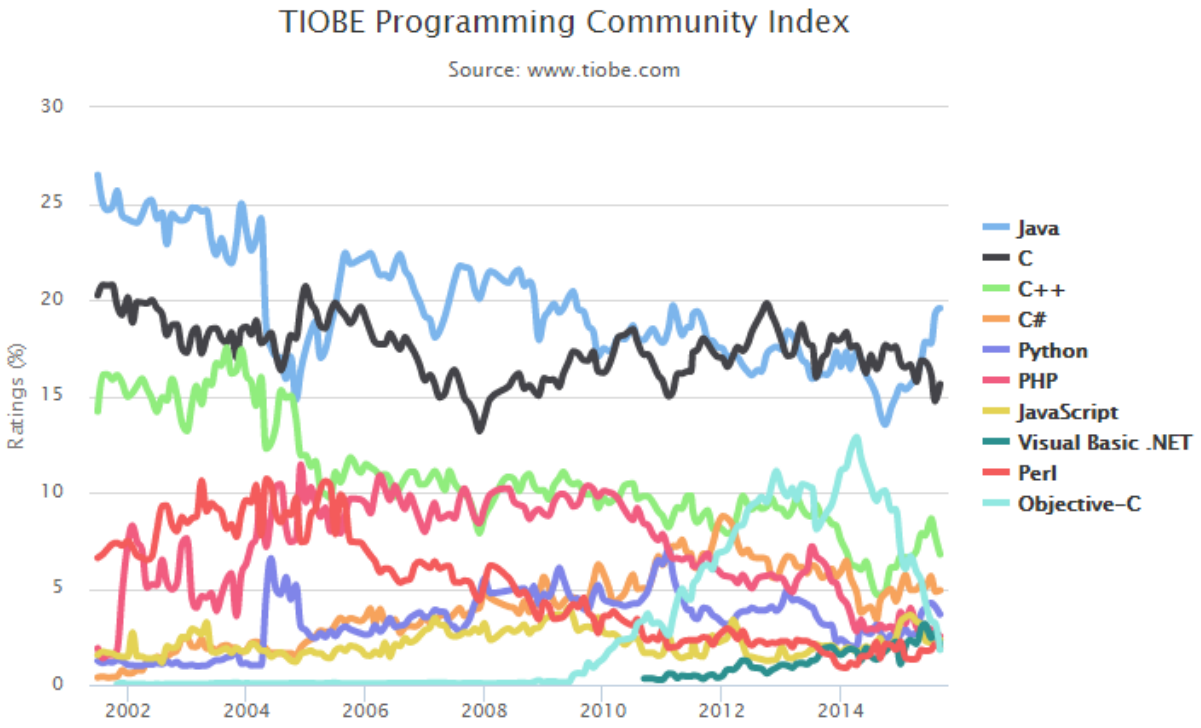
- Cuota de uso tecnologías del servidor (octubre 2015):



http://w3techs.com/technologies/overview/programming_language/all

4. Tecnologías del servidor

- Índice TIOBE (septiembre 2015):



http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

4. Tecnologías del servidor

Java Enterprise Edition

- Tecnología basada en **Java**
- Desarrollada por una coalición de empresas lideradas por **Oracle**, **IBM**, **Red Hat**, etc..
- Tecnología muy usada a nivel **empresarial**
- La mayoría de las **implementaciones** y **herramientas** para desarrollo son **software libre**
- Existen **comunidades** de desarrolladores y **empresas** que realizan **complementos, bibliotecas, herramientas**…

http://www.oracle.com/javaee/

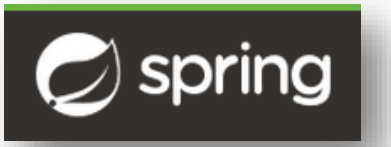
4. Tecnologías del servidor

Java Enterprise Edition

- **Estándares en Java EE**
  - Java tiene una organización que define estándares abiertos que cualquier empresa u organización puede implementar
  - Existen muchos estándares e implementaciones: Java EE, Servlets, JSP, JDBC, JPA, JSF, EJBs…
- **Frameworks en Java EE**
  - Existen multitud de implementaciones independientes de librerías y frameworks
  - Ejemplos: Spring, GWT, Struts, Apache Tiles…

4. Tecnologías del servidor

Java Enterprise Edition



- **Spring**
  - Spring es el framework de desarrollo de **aplicaciones empresariales** basado en tecnologías Java más popular
  - Está enfocado en desarrollo de **aplicaciones de servidor:**
    - Aplicaciones web, servicios REST y websockets
    - Análisis de datos
    - Procesado de tareas por lotes
    - Integración de sistemas

4. Tecnologías del servidor

PHP

- Desarrollado en 1994 por **Rasmus Lerdorf**
- Fue una de las **primeras tecnologías libres** que se popularizaron para desarrollo web
- Tecnología con un lenguaje propio llamado **PHP**
- Desarrollada por **PHP Group** con licencia libre **PHP license**
- Es la tecnología de programación que más sitios activos tiene en Internet
- Se integra normalmente con Apache y MySQL en entornos Linux en un paquete llamado **LAMP**
- **Facebook** es sin duda una muestra importante de la popularidad de PHP
- CMSs como **Drupal** y **Wordpress** también están implementados en PHP

http://www.php.net/

4. Tecnologías del servidor

ASP.NET

- Versión evolucionada del **ASP clásico**
- Integrada en la tecnología **.NET** de Microsoft junto con el lenguaje **C#**
- Licencia **propietaria** y para plataformas **Windows**
- Tiene una comunidad de desarrolladores más limitada que las otras alternativas

http://www.asp.net/

Índice de contenidos

5. Bases de datos

- Base de datos = conjunto ordenado de datos
  - La información está centralizada y es más sencillo realizar actualizaciones y copias de seguridad
- Sistema gestor de bases de datos (DBMS) = software que permite almacenar y consultar datos
- Existen muchos tipos de bases de datos, pero las más usadas son:
  - Bases de datos relacionales (RDBMS)
  - Bases de datos objeto-relacionales (ORDBMS)
  - Bases de datos NoSQL

5. Bases de datos

Bases de datos relacionales

- MySQL (Software Libre) - http://www.mysql.org
- Derby (Software Libre) - http://db.apache.org/derby
- H2 (Software libre) - http://www.h2database.com/
- HSQL (Software libre) - http://hsqldb.org/
- MS SQL Server (Comercial) - http://www.microsoft.com/sql
- PostgreSQL (Software Libre) - http://www.postgresql.org/
- Oracle (Comercial) - http://www.oracle.com

RDBMS

ORDBMS

5. Bases de datos

Bases de datos relacionales - MySQL

- Sistema gestor de base de datos multiplataforma
- Desarrollado en C
- Licencia código abierto GPL
- Herramienta interactiva para hacer consultas y crear bases de datos
- Muy popular en el desarrollo web
- Propiedad de Oracle

http://www.mysql.org/

5. Bases de datos

Bases de datos relacionales – H2

- Sistema gestor de base de datos multiplataforma
- Implementado en Java
- Licencia código abierto MPL 2.0 y EPL 1.0
- Soporte de un subconjunto de SQL 99 y 2003
- Dispone de driver JDBC para Java
- Se puede usar **en memoria**, ideal para desarrollo y pruebas

http://www.h2database.com/

5. Bases de datos

Bases de datos NoSQL

- El término NoSQL ("no sólo SQL") define una clase de DBMS que difieren del clásico modelo relacional:
  - No utilizan estructuras fijas como tablas para el almacenamiento de los datos
  - No usan el modelo entidad-relación
  - No suelen permitir operaciones JOIN (para evitar sobrecargas en búsquedas)
  - Arquitectura distribuida (los datos pueden estar compartidos en varias máquinas mediante mecanismos de tablas Hash distribuidas)
- Este tipo de bases de datos coincide con la explosión de usuarios que han experimentados algunas aplicaciones (por ejemplo Facebook, Twitter, YouTube, etc)

5. Bases de datos

Bases de datos NoSQL

- Pueden manejar **gran cantidad de datos** ("*Big Data*"): al usar una arquitectura distribuida, en muchos casos mediante tablas Hash
- Se ejecutan en máquinas con **pocos recursos**
- **Escalabilidad horizontal**: para mejorar el rendimiento de estos sistemas simplemente se consigue añadiendo más nodos
- **No genera cuellos de botella**: las consultas SQL complejas requieren un nivel de ejecución aún más complejo que ante muchas peticiones puede ralentizar el sistema

5. Bases de datos

Bases de datos NoSQL

- Cuándo usar NoSQL:
  - Cuando el volumen de los datos crece muy rápidamente en momentos puntuales (> Terabyte)
  - Cuando la escalabilidad de la solución relacional no es viable tanto a nivel de costes como a nivel técnico
  - Cuando tenemos elevados picos de uso del sistemas
  - Cuando el esquema de la base de datos no es homogéneo, es decir, cuando en cada inserción de datos la información que se almacena puede tener campos distintos

5. Bases de datos

Bases de datos NoSQL

- Hay 4 tipos principales de bases de datos NoSQL:
  1. Orientadas a documentos. Este tipo almacena la información como un documento, por ejemplo JSON, XML o BSON (*Binary JSON*)
  2. Orientadas a columnas. Funcionan de forma parecida a las bases de datos relacionales, pero almacenando columnas de datos en lugar de registros
  3. De clave-valor. Cada elemento está identificado por una llave única, lo que permite la recuperación de la información de forma muy rápida
  4. En grafo. La información se representa como nodos de un grafo y sus relaciones con las aristas del mismo

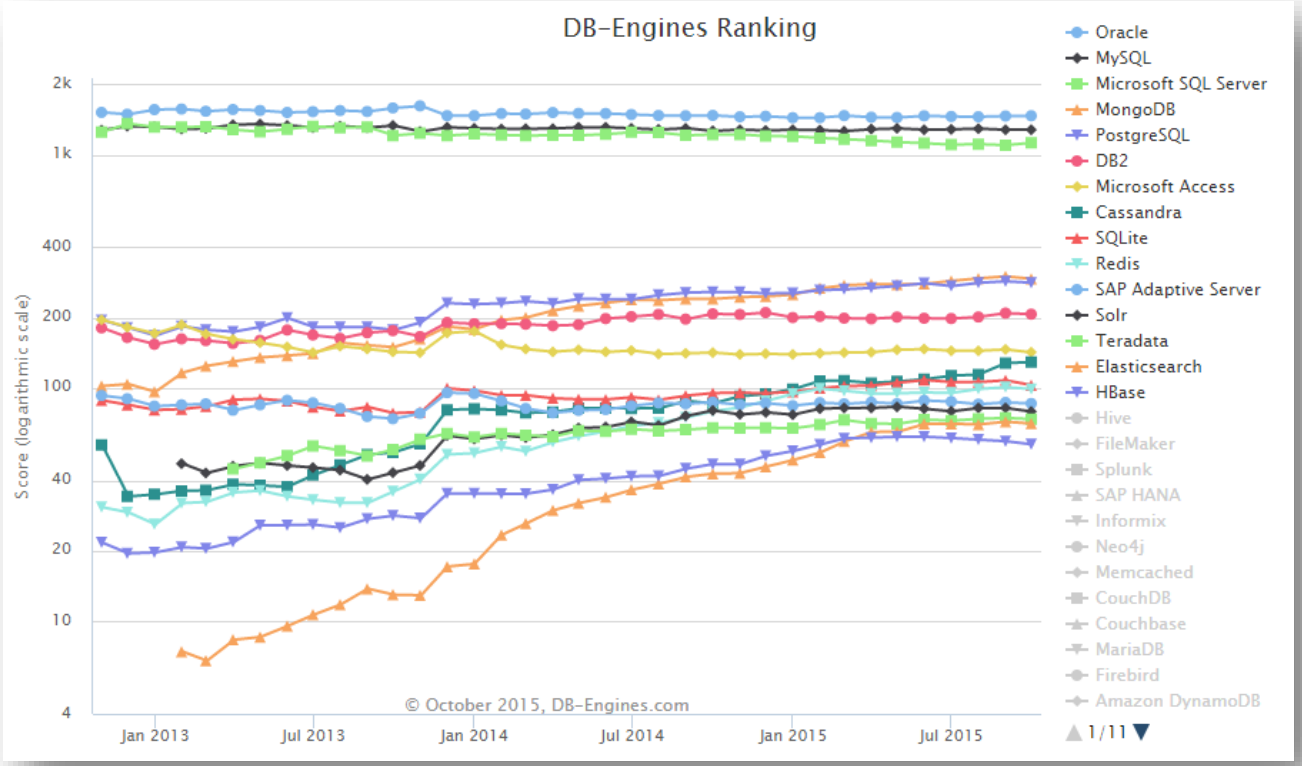
5. Bases de datos

Bases de datos NoSQL

Documento	Clave-Valor	Columna	Grafo
MongoDB	Redis	Cassandra	Neo4J
CouchDB	Membase	BigTable	FlockDB
RavenDB	Voldemort	Hbase (Hadoop)	InfiniteGraph
Terrastore	MemcacheDB	SimpleDB	InfoGrid
	Riak	Cloudera	Virtuoso

5. Bases de datos

Ranking de uso de bases de datos:



http://db-engines.com/en/ranking_trend

Índice de contenidos

6. Sistemas gestores de contenido

- **CMS (*Content Management System*)**
- Aplicación web genérica que permite la creación y administración de contenidos **vía web**
- El sistema permite manejar de manera **independiente el contenido y el diseño**, permite el cambio de diseño (con *templates* o *themes*)
- Los CMSs han evolucionado para convertirse en un **nuevo modelo de desarrollo de aplicaciones web** configurando y adaptando módulos con un interfaz web

6. Sistemas gestores de contenido

- Existen multitud de CMSs con enfoques y objetivos diferentes
- Ejemplos: **Drupal** (PHP), **Joomla** (PHP), **Wordpress** (PHP), Plone (JavaSript), Liferay (Java)

http://en.wikipedia.org/wiki/List_of_content_management_systems

Índice de contenidos

7. Servicios en la nube

- Los **servicios en la nube** se ofrecen bajo demanda y de forma escalable a través de la Web
- Podemos clasificar estos servicios en base a quien los consume y el nivel de abstracción de los mismos:
  - Servicios para desarrolladores:
    - **IaaS** (*Infrastructure as a Service*): Infraestructura como servicio (bajo nivel)
    - **PaaS** (*Platform as a Service*): Plataforma como servicio (nivel medio)
  - Servicios para usuarios finales:
    - **SaaS** (*Software as a Service*): Software como servicio (alto nivel)

7. Servicios en la nube

Infraestructura como servicio

- Es la capa de abstracción más baja del *cloud computing*
- Modelo de distribución de infraestructura normalmente mediante una plataforma de virtualización
- En lugar adquirir servidores, espacio en un centro de datos o equipamiento de redes, los clientes compran todos estos recursos a un proveedor de servicios
- La diferencia fundamental con el hosting virtual es que el provisionamiento de estos servicios se hacen de manera integral a través de la web

7. Servicios en la nube

Infraestructura como servicio

- Servicios típicos ofrecidos por un proveedor IaaS:
  - Servidores (*instances*)
  - Balanceadores de carga (*load balancer*)
  - Gestión de sistemas operativos (*images*)
  - Copias se seguridad de servidores
  - Almacenamiento de datos
  - Direcciones IP
  - Servidores DNS

7. Servicios en la nube

Infraestructura como servicio

- **Amazon Web Services** (AWS) es el proveedor más famoso y más completo en estos servicios
- AWS ofrece un conjunto de servicios y un modelo de precios que se ajusta a las necesidades de cada cliente
- El servicio central de AWS es Amazon EC2 (*Elastic Compute Cloud*), que es el nombre comercial del servicio de servidores virtuales o instancias

7. Servicios en la nube

Plataforma como servicio

- En el ***Platform as a Service*** **(PaaS)** se ofrece una plataforma para soportar el ciclo de vida completo de construcción y puesta en marcha de aplicaciones y servicios web
  - Servidores web, bases de datos, gestión de logs, monitorización…
- Los desarrolladores no se preocupan de la gestión de la plataforma, sólo se **preocupan de su software**
- La ventaja fundamental es que es **escalable** y **tolerante a fallos** de forma automática
- Cada proveedor ofrece unos **servicios diferentes**

7. Servicios en la nube

Plataforma como servicio

- Ejemplos de proveedores PaaS:
  - **Amazon Elastic BeanStalk**: Plataforma de ejecución de código Java, PHP, Node.js, Ruby, Python
  - **Google App Engine**: Permite desarrollar aplicaciones en Python, Java, y PHP
  - **Heroku**: Plataforma de desarrollo con Java, Node.js, Django
  - **OpenShift**: Plataforma para Node.js, Ruby, Python, PHP, Java

7. Servicios en la nube

Software como servicio

- El software como servicio engloba aquellos servicios en la nube que se ofrecen al usuario final vía web
- Desde un punto de vista técnico, muchos servicios del PaaS se podrían considerar como SaaS, pero con SaaS se incide en que el usuario final usa el servicio
- Si es de pago, se paga por uso o por número de usuarios
- Ejemplos: Google Apps, Microsoft 365, Dropbox, iCloud, …