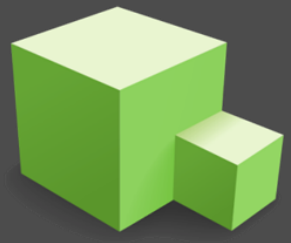


sidelab®

Laboratorio de Software y Entornos de Desarrollo

DESARROLLO DE APLICACIONES WEB - TEMA 2

Tecnologías de desarrollo de aplicaciones web

Micael Gallego

Correo: micael.gallego@urjc.es

Twitter: @micael_gallego

Blog: http://micaelgallego.github.io

Universidad Rey Juan Carlos

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

Tecnologías de desarrollo de aplicaciones web

- **Introducción**
- Arquitecturas de aplicaciones web
- Tecnologías del cliente
- Tecnologías del servidor
- Bases de datos
- Sistemas gestores de contenido

Introducción

- El desarrollo de aplicaciones web ha evolucionado enormemente en la última década, tanto desde el punto de vista del **desarrollo de software** como a nivel de **administración de sistemas**
- **Desarrollo de software**
  - Se han creado multitud de tecnologías, *frameworks* de desarrollo de aplicaciones, bibliotecas, aplicaciones configurables, arquitecturas, modelos de publicación de versiones (*release*)…
- **Administración de sistemas**
  - Se ha evolucionado enormemente en la administración de sistemas, servicios de alojamiento, técnicas de escalabilidad, monitorización, gestión de centros de procesos de datos…

Desarrollo de software

- La evolución ha tenido como resultado que hay 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 disciplina**
- Existen dos enfoques en el desarrollo de aplicaciones web:
  - Creación de webs con tecnologías de desarrollo
  - Creación de webs con sistemas gestores de contenido

Desarrollo de software

- Creación de webs con tecnologías de desarrollo
  - **Arquitecturas de aplicaciones web:** Una aplicación web puede tener diferentes arquitecturas. Esto determina cómo se usan las diferentes tecnologías existentes
  - **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.

Desarrollo de software

- Creación de webs con 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 **utiliza un software ya desarrollado** y se personaliza y adapta a las necesidades
  - A las aplicaciones de este tipo se las denomina **Sistemas Gestores de Contenido** (CMSs).

Administración de sistemas

- **Internet** y las **aplicaciones web** han hecho evolucionar la administración de sistemas en muchos aspectos
  - Para que una aplicación web funcione necesita que el sistema donde se instale disponga de un **servidor web** y habitualmente una **base de datos**
  - Como la web tiene que estar disponible para los usuarios de Internet, habitualmente se instala en **sistemas que se alquilan a terceros**: alojamiento en la nube (*cloud*)
  - Como las aplicaciones web pueden tener un número **muy grande de usuarios** y tienen que estar **siempre disponibles**, se utilizan técnicas de **escalabilidad** y **tolerancia a fallos**

Se verá en el Tema3

Tecnologías de desarrollo de aplicaciones web

- Introducción
- **Arquitecturas de aplicaciones web**
- Tecnologías del cliente
- Tecnologías del servidor
- Bases de datos
- Sistemas gestores de contenido

Arquitecturas de aplicaciones web

- La **arquitectura básica de una aplicación web** está formada por los siguientes elementos:
  - **Un navegador:** Hace de cliente y realiza peticiones solicitando recursos a los servidores web. Cuando hace una petición a un servidor y le contesta enviándole un recurso, se lo muestra al usuario.
  - **Un servidor web:** Recibe peticiones de clientes (navegadores) y responde a esas peticiones enviado un recurso o notificando un error si el recurso no existe.
  - **El protocolo http:** Es el protocolo basado en TCP/IP que se utiliza para que el navegador realice las peticiones al servidor web y este responda.
  - **HTML:** Es el formato básico de los documentos de la web. Es un formato textual, basado en etiquetas que permite estructurar el contenido de la página.

Arquitecturas de aplicaciones web

- La arquitectura de las aplicaciones web ha **evolucionado** mucho en los últimos años
- No todas las aplicaciones web tienen la **misma arquitectura**
- Las arquitecturas se diferencias principalmente en lo **estática** o **dinámica** que sea la web
- Una web puede ser **dinámica en el cliente y/o en el servidor**
- Las tecnologías utilizadas:
  - **Dinamismo en cliente:** JavaScript
  - **Dinamismo en servidor:** Java EE, .NET, PHP, Ruby on Rails, Python Django, Groovy, Node.js, Scala Play…

Arquitecturas de aplicaciones web

- **Cliente estático (Sin JavaScript)**
  - Servidor estático
  - Servidor dinámico (3 capas)
- **Cliente dinámico (Con JavaScript)**
  - Servidor estático
  - Servidor dinámico
    - JavaScript para efectos gráficos
    - JavaScript con peticiones en segundo plano (AJAX)
    - Single Page Application con REST

Arquitecturas de aplicaciones web

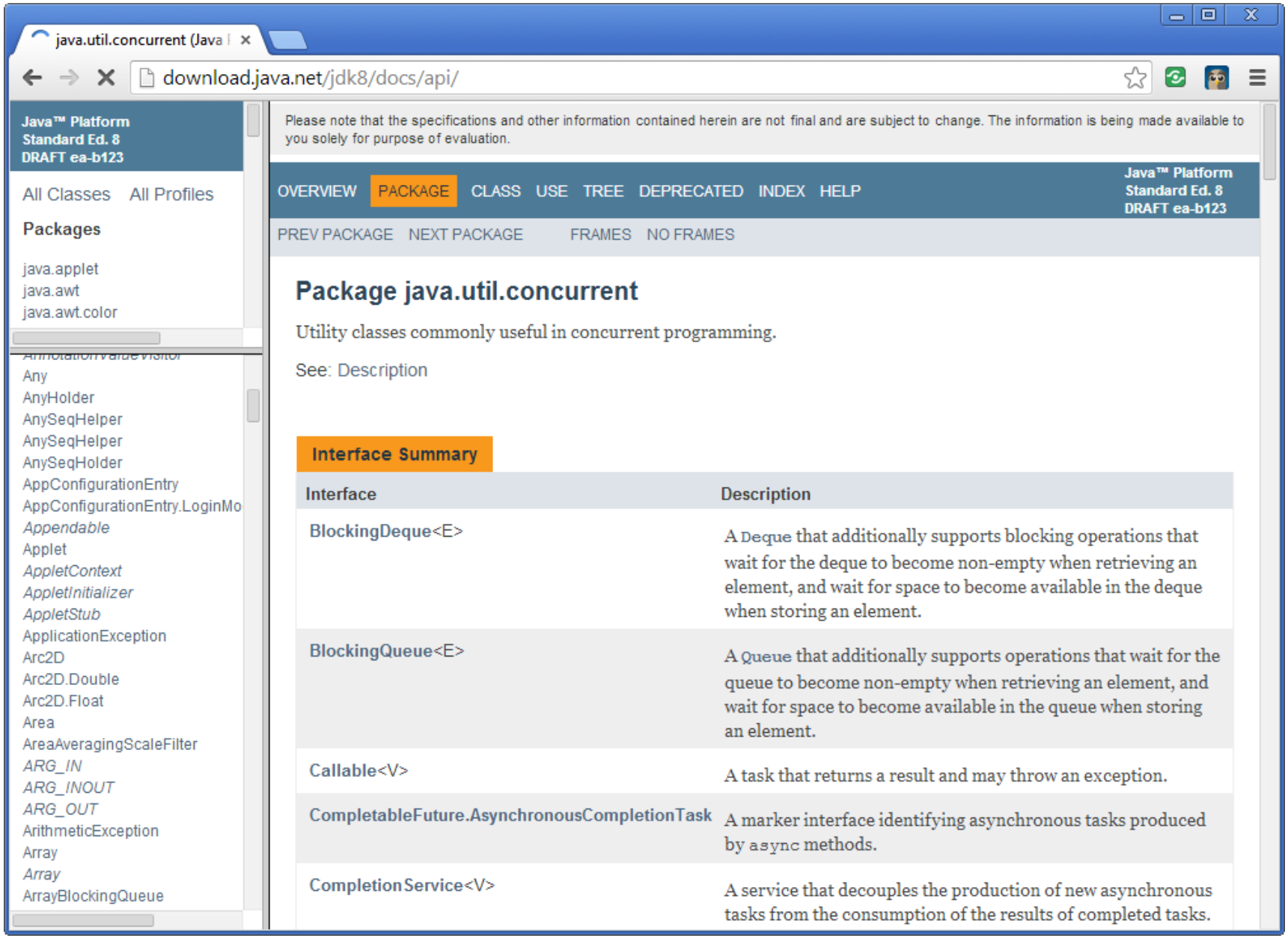
- **Arquitectura Cliente estático y Servidor estático**
  - El navegador hace **petición** al servidor mediante **http**
  - El **servidor** transforma **URL** a ruta en **disco**
  - El **servidor** devuelve el **fichero** de disco al **navegador**
  - El navegador **visualiza** (*renderiza*) la página **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

Arquitecturas de aplicaciones web

- **Arquitectura Cliente estático y Servidor estático**
  - 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)
  - A este tipo de web no se le suele llamar aplicación web porque nada es dinámico. Se denomina simplemente **página web**

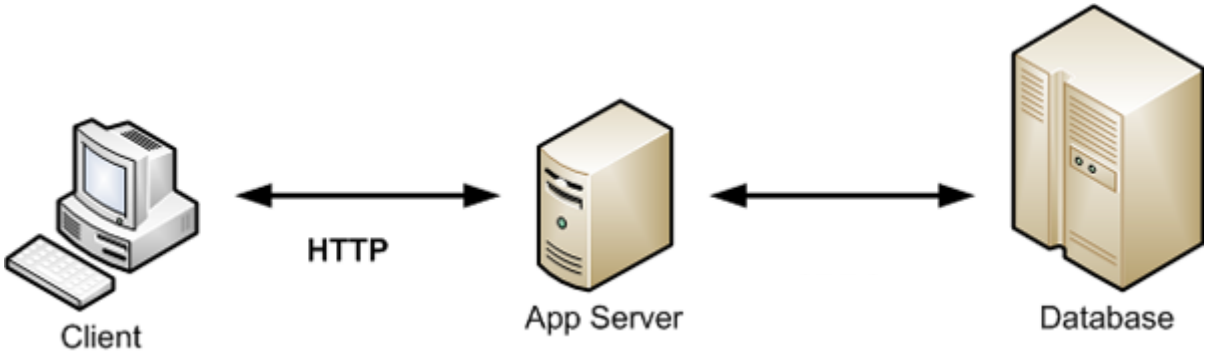
Arquitecturas de aplicaciones web

- **Arquitectura Cliente estático y Servidor estático**
  - La web se diseñó con esta arquitectura
  - Las primeras páginas web eran así
  - Todavía se sigue usando en muchas páginas web:
    - **Páginas personales o de proyectos básicas** (p.e. Tecnología de webs de github)
    - **Documentación técnica** (JavaDoc en Java, Maven site, etc…)


java.util.concurrent (Java
download.java.net/jdk8/docs/api/
Java™ Platform
Standard Ed. 8
DRAFT ea-b123
All Classes
All Profiles
Packages
java.applet
java.awt
java.awt.color
Any
AnyHolder
AnySeqHelper
AnySeqHelper
AnySeqHolder
AppConfigurationEntry
AppConfigurationEntry.LoginMo
Appendable
Applet
AppletContext
AppletInitializer
AppletStub
ApplicationException
Arc2D
Arc2D.Double
Arc2D.Float
Area
AreaAveragingScaleFilter
ARG_IN
ARG_INOUT
ARG_OUT
ArithmeticException
Array
Array
ArrayBlockingQueue
Please note that the specifications and other information contained herein are not final and are subject to change. The information is being made available to you solely for purpose of evaluation.
OVERVIEW
PACKAGE
CLASS
USE
TREE
DEPRECATED
INDEX
HELP
Java™ Platform
Standard Ed. 8
DRAFT ea-b123
PREV PACKAGE
NEXT PACKAGE
FRAMES
NO FRAMES
Package java.util.concurrent
Utility classes commonly useful in concurrent programming.
See: Description
Interface Summary
Interface
Description
BlockingDeque<E>
A Deque that additionally supports blocking operations that wait for the deque to become non-empty when retrieving an element, and wait for space to become available in the deque when storing an element.
BlockingQueue<E>
A Queue that additionally supports operations that wait for the queue to become non-empty when retrieving an element, and wait for space to become available in the queue when storing an element.
Callable<V>
A task that returns a result and may throw an exception.
CompletableFuture.AsynchronousCompletionTask
A marker interface identifying asynchronous tasks produced by async methods.
CompletionService<V>
A service that decouples the production of new asynchronous tasks from the consumption of the results of completed tasks.

Arquitecturas de aplicaciones web

- **Arquitectura Cliente estático y Servidor dinámico**



  - 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

Arquitecturas de aplicaciones web

- **Arquitectura Cliente estático y Servidor dinámico**
  - 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
  - También se pueden generar recursos de otro tipo (**imágenes, PDFs…**)
  - Si el usuario pulsa un link, se **recarga la página al completo**

Arquitecturas de aplicaciones web

- **Arquitectura Cliente estático y Servidor dinámico**
  - Es la arquitectura de las primeras "**aplicaciones web**"
  - Todavía sigue habiendo **muchas web** 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 clic
    - 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)

Arquitecturas de aplicaciones web

- **Arquitectura Cliente dinámico y Servidor estático**
  - 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***)

Arquitecturas de aplicaciones web

- **Arquitectura Cliente dinámico y Servidor dinámico**
  - 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, las aplicaciones se pueden dividir en tres tipos:
    - JavaScript para **efectos gráficos**
    - JavaScript con peticiones en segundo plano (**AJAX**)
    - ***Single Page Application*** con API REST

Arquitectura Cliente dinámico y Servidor dinámico

- **JavaScript para efectos gráficos**
  - En este caso, el dinamismo en el cliente se utiliza exactamente igual que con un **servidor estático**
  - **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**

Arquitectura Cliente dinámico y Servidor dinámico

- **JavaScript con peticiones en segundo plano (AJAX)**

  - JavaScript se puede usar para no tener que **recargar completamente** la página completa al pulsar un link
  - Con JavaScript se puede 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*)**

Arquitectura Cliente dinámico y Servidor dinámico

- **JavaScript con peticiones en segundo plano (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…)

Arquitectura Cliente dinámico y Servidor dinámico

- **JavaScript con peticiones en segundo plano (AJAX)**

  - Cuando el código **JavaScript** hace peticiones, el servidor puede devolver:
    - Contenido para ser incluido en la página directamente:
      - **Fragmentos de HTML** generados dinámicamente
      - **Recursos estáticos en disco:** Imágenes, PDF, HTML, etc…
    - Información que será interpretada por JavaScript para modificar la página (Mostrar un error, cambiar un color, …):
      - Información generada dinámicamente **estructura en XML** o **JSON** (un formato similar a XML).

Arquitectura Cliente dinámico y Servidor dinámico

- **JavaScript con peticiones en segundo plano (AJAX)**

  - Un servidor web **genera HTML** de forma dinámica cuando recibe peticiones **http**
  - Si se usa **AJAX**, el servidor **genera información en XML o JSON** cuando recibe peticiones **http**
  - Habitualmente, cuando un servidor web genera XML o JSON ante peticiones **http**, implementa una **API REST**

Arquitectura Cliente dinámico y Servidor dinámico

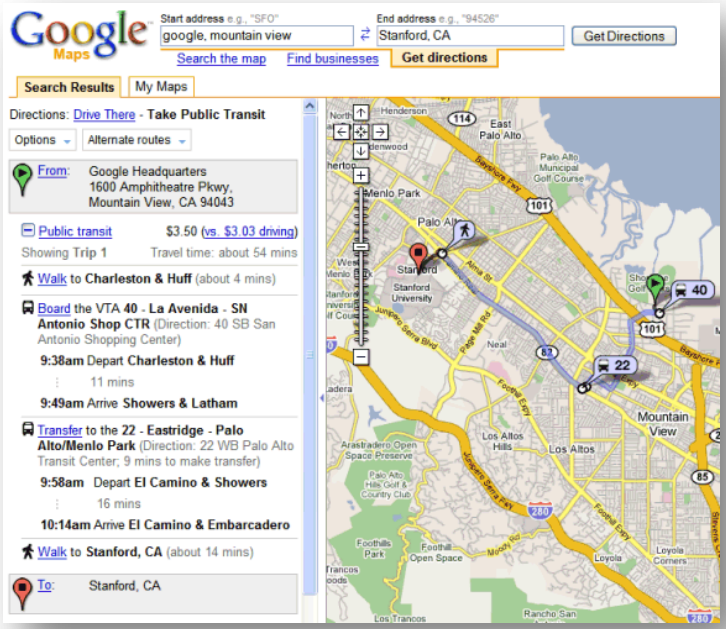
- **JavaScript con peticiones en segundo plano (AJAX)**
  - Existen **muchas aplicaciones** web que **no usan AJAX**
  - La mayoría de las aplicaciones que se han desarrollado en los **últimos años usan AJAX** en algunas de sus páginas por la mejora en la **experiencia del usuario**

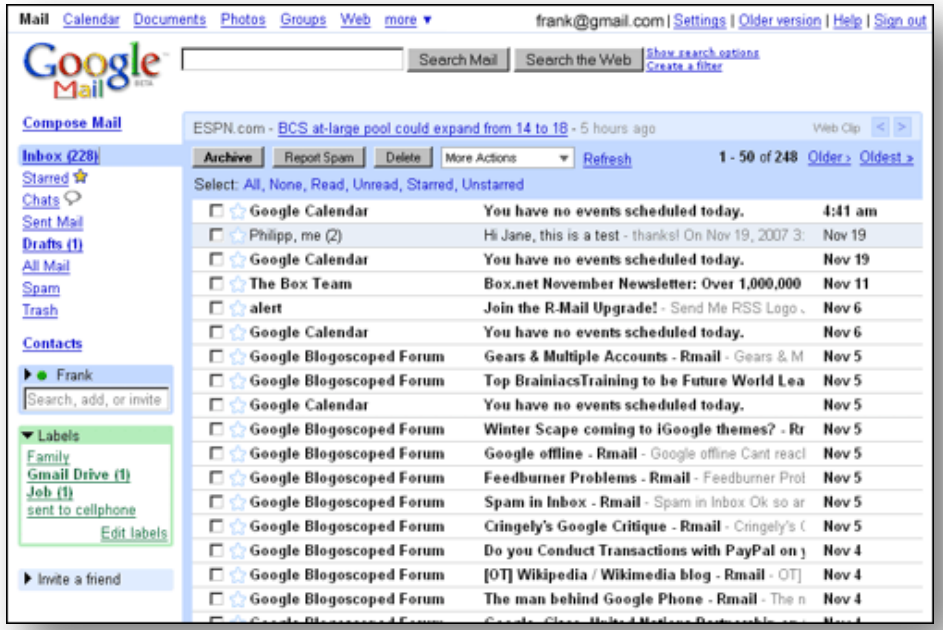
Arquitectura Cliente dinámico y Servidor dinámico

- ***Single Page Application* con API REST**
  - La técnica **AJAX** se puede llevar al **extremo** y que todo el contenido dinámico se cargue únicamente con **JavaScript**
  - En este caso, la **aplicación web** es un conjunto de **recursos HTML, CSS y JavaScript estáticos**, que se cargan en el navegador
  - El contenido **dinámico** se genera en el servidor únicamente como **XML o JSON** que se carga en segundo plano con **JavaScript** mediante peticiones a la **API REST** del servidor web

Arquitectura Cliente dinámico y Servidor dinámico

- ***Single Page Application*** **con API REST**
  - Google popularizó AJAX y SPA con Gmail y Maps





Arquitectura Cliente dinámico y Servidor dinámico

- ***Single Page Application* con API REST**
  - El cliente una web SPA es una **aplicación completa y autónoma,** que se descarga de la red al acceder a una **URL** y que se comunica con un **servidor usando REST**
  - 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

Arquitectura Cliente dinámico y Servidor dinámico

- ***Single Page Application* con API REST**
  - Las **aplicaciones autónomas** que se implementan con tecnologías HTML5 se consideran SPA:
    - **Móviles:** Apache Cordova, Firefox OS, ubuntu mobile…
    - **Escritorio:** Windows Store Apps, Google Chrome Apps…
    - **SmartTV:** Samsung, LG…
  - En la mayoría de estas tecnologías en código de la aplicación cliente (**HTML, CSS y JS**) puede estar contenido en el **paquete de instalación** o puede **descargarse** al iniciar la aplicación

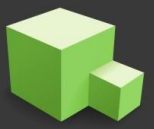
Tecnologías de desarrollo de aplicaciones web

- Introducción
- Arquitecturas de aplicaciones web
- **Tecnologías del cliente**
- Tecnologías del servidor
- Bases de datos
- Sistemas gestores de contenido

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**

Estándares web



- El W3C (*World Wide Web Consortium*) es una comunidad internacional que desarrolla estándares abiertos que aseguran el crecimiento de la Web a largo plazo

Scripting and Ajax

HTML & CSS

Gráficos

Accesibilidad

Audio & Video

Web Semántica

XML

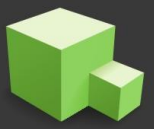
Servicios Web

http://www.w3.org

Estándares web

- **HTML** (*Hypertext Markup Language*) and **CSS** (*Cascading Style Sheets*) son dos de las tecnologías principales para la construcción de páginas web
  - **HTML** proporciona la información estructurada en secciones, párrafos, título, imágenes, etc…
  - **CSS** proporciona la distribución de los elementos y su estilo (colores, tipos de letra, fondos, efectos…)

Estándares web



- **HTML**
  - La versión actual es **HTML5**
  - Todavía no está finalizada, pero la mayoría de los navegadores **implementan** gran parte de la **especificación**
  - Ha supuesto una **revolución** para el dinamismo en el cliente porque ofrece muchas **librerías/tecnologías avanzadas**:
    - Multimedia: etiquetas vídeo, audio y canvas, webgl
    - Comunicaciones: websockets
    - Concurrencia: webworkers

Estándares web

- **CSS**
  - CSS es un lenguaje usado para definir la **presentación de un documento** estructurado escrito en HTML, XML, SVG o incluso interfaces de usuario de otras tecnologías (JavaFX)
  - Su versión actual es **CSS3** (aunque todavía no está finalizada)

Estándares web

- **Scripting**
  - Las páginas web pueden programarse con diversos lenguajes de script, aunque prácticamente sólo se usa **JavaScript**
  - Con JavaScript se puede modificar la página y ejecutar código cuando se interactúa con ella (a través del modelo de objetos del documento **DOM**)
  - Se hacen peticiones al servidor web en segundo plano y se actualiza el contenido de la web (**AJAX**)

Estándares web

- **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 JS de los navegadores, aunque actualmente todos son bastante **compatibles** entre sí (**en el pasado no fue así**)
  - Aunque algunos elementos de la sintaxis recuerden a Java, no tiene nada que ver con **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/

Estándares web

- **JavaScript**
  - Inicialmente era un lenguaje interpretado, pero actualmente se ejecuta con **máquinas virtuales** en los navegadores (**velocidad de ejecución y eficiencia de memoria**)
  - Características:
    - **Tipado dinámico** (habitual en los lenguajes de script)
    - **Funcional y orientado a objetos** (basado en prototipos)

Estándares web

- **DOM**
  - *Document Object Model*
  - Librería (API) para manipular el documento HTML cargado en el navegador
  - Es el equivalente en web a la librería de componentes gráficos
  - Permite la gestión de eventos, insertar y eliminar elementos, etc.

Estándares web

- **Librerías JavaScript**
  - Existen multitud de **bibliotecas** (APIs) JavaScript para el desarrollo de aplicaciones
  - Las más utilizadas son:
    - **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

Estándares web

- **Librerías JavaScript**
  - Además de librerías, 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

http://www.lostiemposcambian.com/blog/javascript/backbone-vs-angular-vs-ember/

Tecnologías no estándar en la web

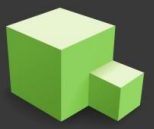
- La web ha avanzado y evolucionado gracias a tecnologías no estándar incluidas en los navegadores mediante ***plugins***
- Algunas llegaron a convertirse en **estándares "de facto"**
- La tecnología no estándar por excelencia de la web es **Adobe Flash** (aunque ha habido otras)
- La llegada de los dispositivos móviles, consolas y televisiones conectadas (SmartTVs) y la estandarización de **HTML5** han hecho que estas tecnologías no estándar estén en desuso

Tecnologías no estándar en la web

- Adobe Flash
  - 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
  - Es posible que haya cosas que **a día de hoy sólo se puedan hacer con flash** y no se puedan implementar con la tecnología HTML5

http://active.tutsplus.com/articles/roundups/10-flash-things-you-can%E2%80%99t-do-with-html5/

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
  - Ha sido acusada de de que **no es eficiente, no es abierta** y por tanto, **no es el futuro** de la web (Abril 2010 - Steve Jobs por el iPhone y iPad)
  - Adobe lo ha acabado reconociendo y no seguirá apostando por Flash como la herramienta básica de la web interactiva (Nov 2011)

http://www.apple.com/hotnews/thoughts-on-flash/
http://blogs.adobe.com/conversations/2011/11/flash-focus.html

Tecnologías no estándar en la web

- **Java Applets**
  - Los applets de Java fueron los precursores de Flash
  - Debido a prácticas anticompetitivas de Microsoft y que Sun Microsystems estaba más centrada en los servidores de aplicaciones hace mucho tiempo que está en desuso
- **Microsoft Silverlight**
  - La apuesta de Microsoft para competir con Adobe Flash
  - Soporte muy limitado en plataformas diferentes a Windows
  - El navegador web de Metro en Windows 8 no soportará plugins, por tanto, no tendrá soporte para Flash ni para Silverlight

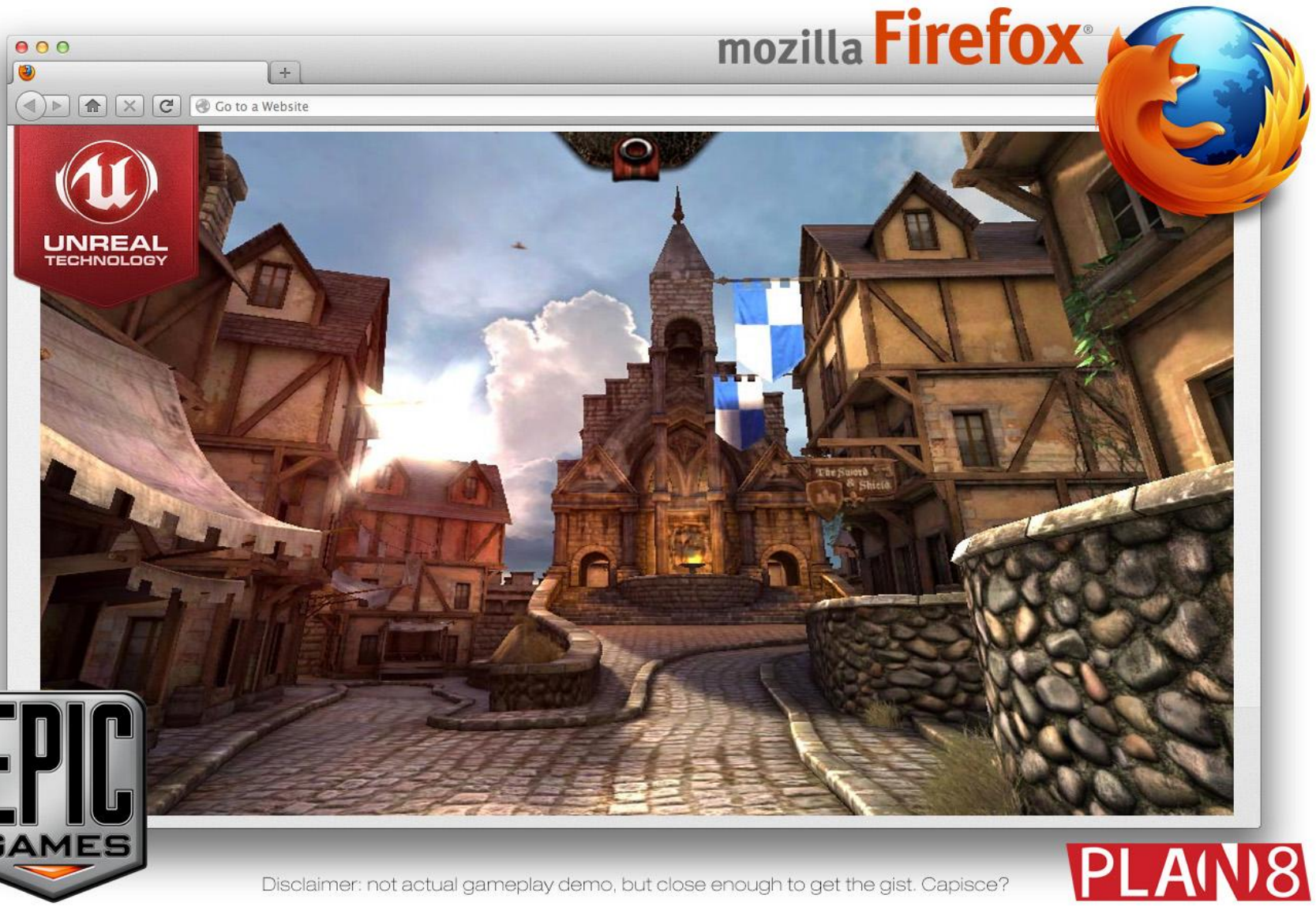
http://www.infoq.com/news/2011/09/Metro-Plug-ins

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
- Si es estrictamente necesario usar **Flash**, es conveniente conocer la cantidad de usuarios que **no podrán acceder a la web porque sus dispositivos no son compatibles con esta tecnología**
- **HTML5** avanza muy rápido. 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/

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
- Si es estrictamente necesario usar **Flash**, es conveniente conocer la cantidad de usuarios que **no podrán acceder a la web porque sus dispositivos no son compatibles con esta tecnología**
- **HTML5** avanza muy rápido. Se ha convertido en la **tecnología estándar** para multitud de plataformas diferentes


mozilla Firefox
Go to a Website
UNREAL
TECHNOLOGY
EPIC
GAMES
Disclaimer: not actual gameplay demo, but close enough to get the gist. Capisce?
PLAN8

Tecnologías de desarrollo de aplicaciones web

- Introducción
- Arquitecturas de aplicaciones web
- Tecnologías del cliente
- **Tecnologías del servidor**
- Bases de datos
- Sistemas gestores de contenido

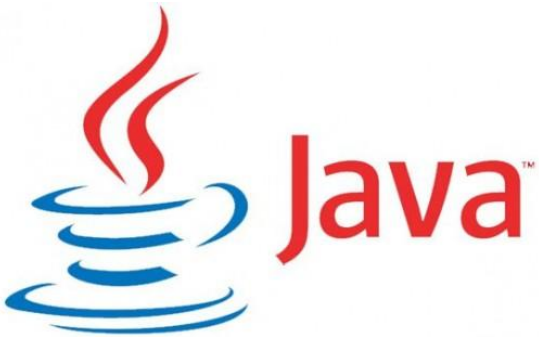
Tecnologías del servidor

- Los **estándares son muy importantes en los navegadores web** (cliente) 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 servidor con la tecnología de su elección
- En el servidor, se utilizan **tecnologías**, propietarias o abiertas, para el desarrollo de aplicaciones web

Tecnologías del servidor

- Existen multitud de tecnologías de construcción de aplicaciones en el servidor
  - Más usadas: PHP, Java EE, ASP.NET
  - Menos usadas: Ruby on Rails, Grails (Groovy), Django (Python), Perl, ColdFusion, muchas más

Java EE



- 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/

Java EE

- **Estándares en Java EE**
  - Java tiene una organización de estandarización propia llamada *Java Community Process* (JCP)
  - En ella se definen estándares abiertos que se pueden implementar con licencia libre o propietaria
  - Estándares web: Java EE, Servlets, JSP, JDBC, JPA, JSF, EJBs…
- **Bibliotecas y frameworks en Java EE**
  - Existen multitud de implementaciones independientes que pueden seguir o no un estándar
  - Ejemplos: Spring, Hibernate, GWT, Vaadin, Google Closure, Struts, Apache Tiles…

Java EE

- **Estándares más importantes en Java EE**
  - **Servlets**: Estándar para ejecutar código Java ante una petición web en un servidor Java EE
  - **JSP (*Java Server Pages*)**: Estándar que permite mezclar en un documento código Java y HTML para generar páginas web de forma dinámica
  - **JDBC (*Java Database Conectivity*)**: Estándar para conexión a bases de datos relacionales desde Java
  - **JPA (*Java Persistence API*)**: Estándar para la correspondencia objeto-relacional (ORM, *Object Relational Mapping*)
  - **JSF (*Java Server Faces*)**: Estándar de construcción de aplicaciones web basadas en componentes reutilizables
  - **EJB (*Enterprise JavaBeans*)**: arquitectura manejada para la construcción de aplicaciones web (transacciones, seguridad, distribución…)

Java EE

- **Servidores Java EE**
  - Toda aplicación web Java EE tiene que ejecutarse en una servidor de aplicaciones Java (aunque luego se integre en **Apache**, **NginX** o **IIS**)
  - Existen muchos tipos de **servidores**, dependiendo de sus funcionalidades/rendimiento y de su licencia/coste
  - Ejemplos: **Glassfish** (Oracle), **Tomcat** (Apache), **Jetty** (Eclipse), **JBoss** (RedHat), **WebSphere** (IBM), **WebLogic** (Oracle)

Java EE

- **Herramientas de desarrollo**
  - Para desarrollar aplicaciones Java EE se utilizan IDEs y plugins para ellos
  - **Eclipse**: Fundación Eclipse con multitud de plugins. Mucha diversidad, falta de integración. Software libre.
  - **Netbeans**: Oracle. Muy integrado. Software libre.
  - **IntelliJ**: Jetbrains. Muy integrado. Propietario

PHP

- Desarrollado en 1994 por **Rasmus Lerdorf**
- Tecnología con un lenguaje propio llamado **PHP**

- Desarrollada por **PHP Group** con licencia libre **PHP license**
- Es la tecnología de programación del lado del servidor con se han implementado más servidores de Internet
- Es multiplataforma
- Se integra normalmente con Apache y MySQL en entornos Linux en un paquete llamado **LAMP**

http://www.php.net/

PHP

- **Estándares y empresas en PHP**
  - No existe un organismo de estandarización, la tecnología evoluciona por la comunidad en **PHP Group**
  - No hay muchas empresas grandes que apoyan el desarrollo de PHP, pero **Zend** es muy relevante
  - **Facebook** es sin duda una muestra importante de la popularidad de PHP
  - CMSs como **Drupal** y **Wordpress** también están implementados en PHP

PHP

- **Bibliotecas y frameworks**
  - Existen multitud de **frameworks** para el desarrollo de aplicaciones PHP
  - Ejemplos: CakePHP, CodeIgniter, Zend, Symfony, Yii, Zeta Components, Horde

horde

symfony

CakePHP

CodeIgniter®

ZEND FRAMEWORK

http://www.phpframeworks.com/

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/

ASP.NET

- **Librerías y frameworks**
  - La mayoría de las librerías para ASP.NET son las oficiales proporcionadas por Microsoft
  - **Web Pages:** Tecnología similar a JSP y PHP que permite combinar HTML con código ASP
  - **Web Forms:** Tecnología de construcción de aplicaciones web basadas en componentes (similar a JSF de JavaEE)
  - **Data Access Layer (DAL):** Capa de acceso a los datos. Proporciona la misma funcionalidad que JDBC y JPA

Tecnologías de desarrollo de aplicaciones web

- Introducción
- Arquitecturas de aplicaciones web
- Tecnologías del cliente
- Tecnologías del servidor
- **Bases de datos**
- Sistemas gestores de contenido

Bases de datos

- Las bases de datos más populares para el desarrollo de aplicaciones web han sido las **bases de datos relacionales**
- Existen muchas bases de datos relacionales (comerciales y software libre): MySQL, Derby, Oracle, MS SQL Server, PostgreSQL

Apache Derby

MySQL

Microsoft

ORACLE

Bases de datos

- **MySQL**
  - http://www.mysql.org/
  - Sistema gestor de base de datos multiplataforma
  - Desarrollado en C
  - Licencia código abierto GPL
  - Soporte de un subconjunto de SQL 99
  - Herramienta interactiva para hacer consultas y crear bases de datos
  - Muy popular en el desarrollo web

Bases de datos

- Como las aplicaciones web tienen muchas necesidades de escalabilidad y tolerancia a fallos, hay una **nueva familia de bases de datos**
- Se denominan genéricamente **NoSQL**, que se puede interpretar como: **No SQL** o como **Not Only SQL**
- Algunas de las más famosas son:

Tecnologías de desarrollo de aplicaciones web

- Introducción
- Arquitecturas de aplicaciones web
- Tecnologías del cliente
- Tecnologías del servidor
- Bases de datos
- **Sistemas gestores de contenido**

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

Sistemas gestores de contenido

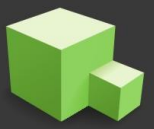
- Existen multitud de CMSs con enfoques y objetivos diferentes
- Ejemplos: Drupal (PHP), Joomla (PHP), Wordpress (PHP), Plone (JavaSript), Moodle (PHP), Liferay (Java)

moodle

Joomla!™ Drupal WordPress.com Plone LIFERAY Enterprise. Open Source. For Life.

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

Sistemas gestores de contenido



- **Joomla:** Principal ventaja es permitir editar el contenido de un sitio web de manera sencilla.
- **WordPress:** Es un sistema de gestión de contenido enfocado a la creación de blogs

Drupal

- Drupal fue originalmente creado por **Dries Buytaert**
- Comenzó como Sistema de tablón de anuncios
- Código libre con licencia GPL/GNU
- Escrito en PHP
- Modular y muy configurable
- Desarrollado y mantenido por una activa comunidad de usuarios
- www.drupal.org

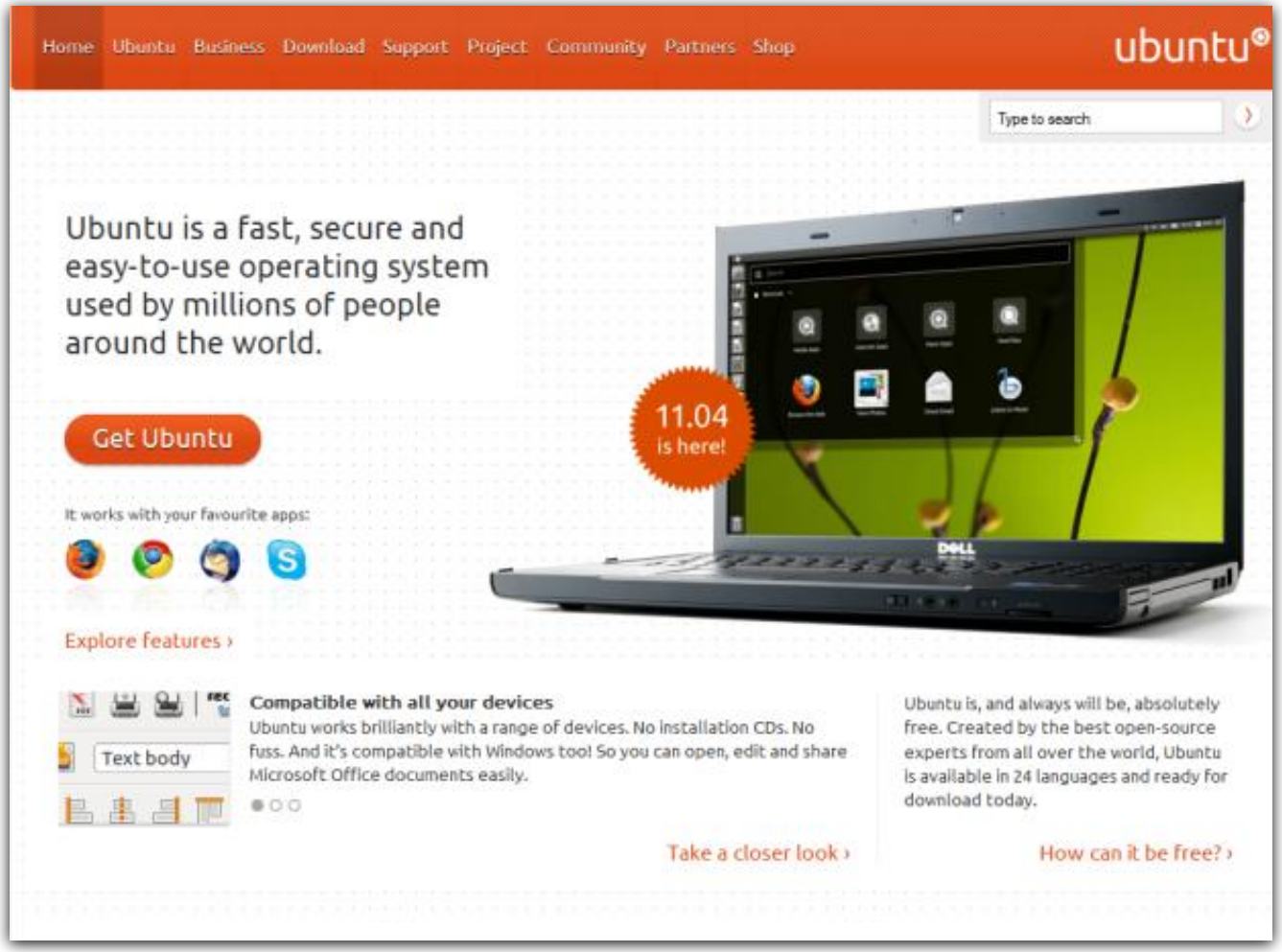
Drupal

- **Código abierto:** El código fuente de Drupal está libremente disponible bajo los términos de la licencia GNU/GPL.
- **Módulos:** La comunidad de Drupal ha contribuido con infinidad de módulos que proporcionan diversas funcionalidades.
- **Objetos de Contenido (Nodos):** El contenido creado en Drupal es, funcionalmente, un objeto (Nodo).
- **Plataforma Independiente de la base de datos:** Aunque la mayor parte de las instalaciones de Drupal utilizan MySQL.

Drupal

- **Multiplataforma:** Drupal ha sido diseñado desde el principio para ser multi-plataforma
- **Múltiples idiomas y Localización:** Drupal está pensado para una audiencia internacional y proporciona opciones para crear un portal multilingüe
- **Administración y Análisis Administración via Web:** La administración y configuración del sistema se puede realizar enteramente con un navegador

Drupal


Home
Ubuntu
Business
Download
Support
Project
Community
Partners
Shop
ubuntu
Type to search
Ubuntu is a fast, secure and easy-to-use operating system used by millions of people around the world.
Get Ubuntu
It works with your favourite apps:
11.04 is here!
Explore features ›
Text body
Compatible with all your devices
Ubuntu works brilliantly with a range of devices. No installation CDs. No fuss. And it's compatible with Windows too! So you can open, edit and share Microsoft Office documents easily.
Take a closer look ›
Ubuntu is, and always will be, absolutely free. Created by the best open-source experts from all over the world, Ubuntu is available in 24 languages and ready for download today.
How can it be free? ›

Drupal


the WHITE HOUSE
Get Email Updates
Contact Us
BLOG
PHOTOS & VIDEO
BRIEFING ROOM
ISSUES
the ADMINISTRATION
the WHITE HOUSE
our GOVERNMENT
WEST WING WEEK
This week the President talks clean energy and gas prices in Indiana, focuses on fixing our broken immigration system in Texas, and honors Top Cops here in the Rose Garden.
Watch the Video
1 2 3 4
WEST WING WEEK
FEATURED TOPICS
Joining Forces
The President, Vice President, First Lady and Dr. Jill Biden launch the national initiative to support and honor America's service members and their families.
Learn More
BACK NEXT
SEARCH the SITE
Search
YOUR FEDERAL TAXPAYER RECEIPT
THE BLOG
WHITE HOUSE SCHEDULE