Arquitecturas Distribuidas

Tema 4. VI- La Web en la actualidad

VI. La Web en la actualidad

1. Aplicaciones Web
2. Arquitectura en 3 capas de las aplicaciones web
3. Pero, ¿qué es una aplicación web?
4. Web 2.0

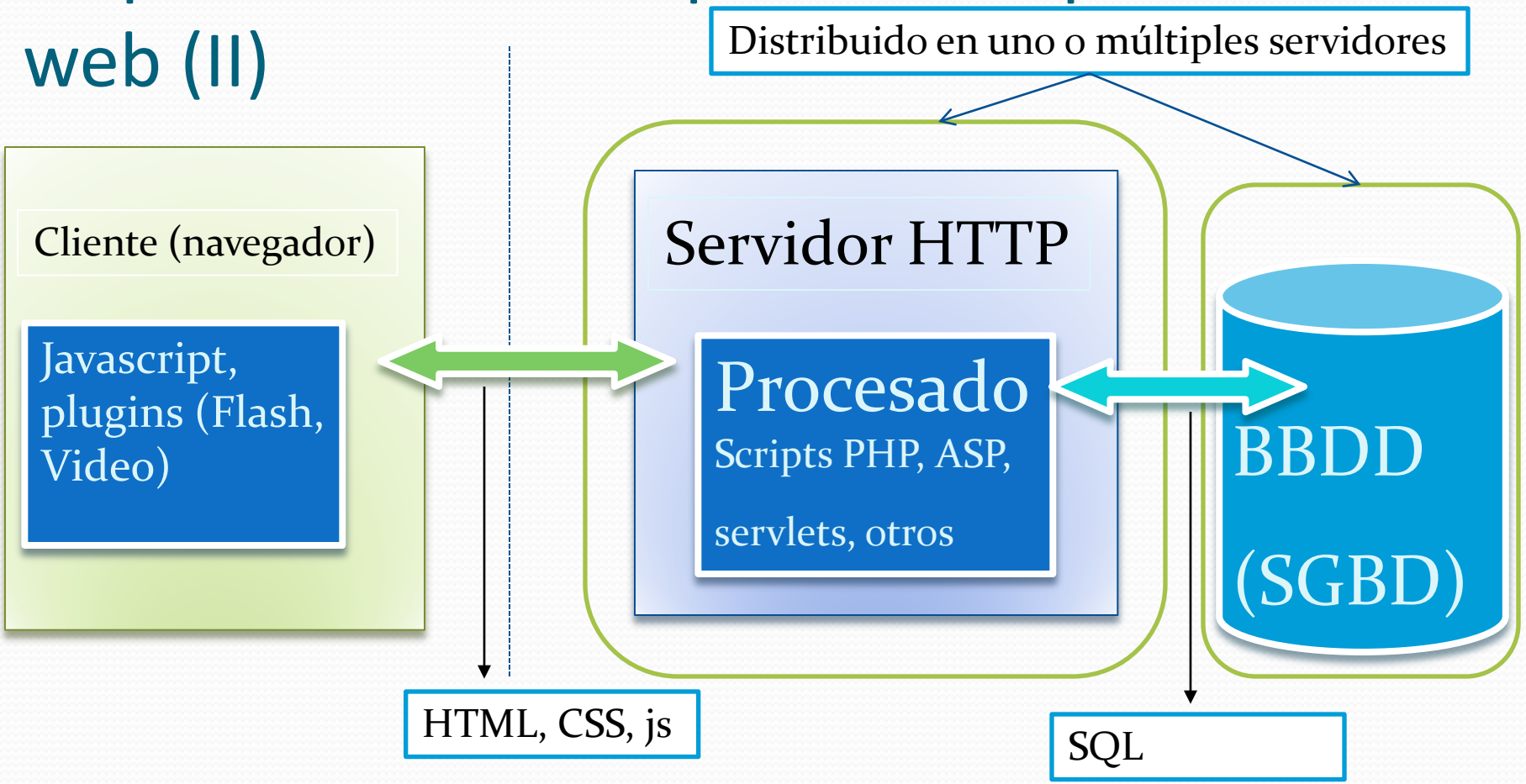
Aplicaciones Web

- Una aplicación típica de La Web en la actualidad utiliza múltiples tecnologías.
- El modelo más común organiza la arquitectura de una aplicación en tres capas diferenciadas:
  - Capa de datos: fuentes de datos utilizadas para generar el contenido: bases de datos, imágenes, datos espaciales (GIS), video, animaciones. **SGBD y datos en bruto**.
  - Capa de lógica o de negocio: aplicaciones de tratamiento y generación del contenido. **Lenguajes de script o de propósito general**: PHP, Java, C, C#, etc.
  - Capa de presentación: aplicaciones de mejora de la interactividad con el usuario. Solicitud de datos en segundo plano. Actualización dinámica de los contenidos. Diseño y estilo**. Javascript (con acceso DOM) o AJAX. CSS. Plugins.**

Arquitectura en 3 capas de las aplicaciones Web (I)

- Capa de datos: los datos que se manejan se almacenan en BBDD normalmente, se encuentran en el **servidor**.
- Capa de lógica o negocio: se **ejecuta** en el **servidor**. Son el conjunto de aplicaciones que atienden las peticiones del cliente, consultan las BBDD y generan una respuesta para el cliente.
- Capa de presentación: se **ejecuta** en el **cliente**. Normalmente consiste en código en el lado del cliente que gestiona la presentación y la interfaz de usuario.

Arquitectura en 3 capas de las aplicaciones web (II)



Arquitectura en 3 capas de las aplicaciones Web (III)

- Si un programador decidiera implementar una aplicación web medianamente compleja tendría que:
  1. Diseñar la estructura de la BBDD que va a utilizar: tablas, relaciones, etc.
  2. Implementar dicha estructura en un SGBD.
  3. Implementar la interfaz de acceso a los datos: código SQL.
  4. Implementar la lógica de negocio: cómo se atienden y procesan las peticiones de los clientes. Integración del código SQL. Código HTML común que se genera.
  5. Diseñar la hoja de estilo (CSS) correspondiente. Imágenes, logos, animaciones flash, vídeo, audio.
  6. Implementar el código javascript que se ejecutará en el cliente.
- Los puntos 4, 5 y 6 suelen estar imbricados: HTML, CSS y javascript se suele generar dinámicamente mediante el lenguaje de programación del servidor y con los datos de la BBDD.

Ejemplos

- Servicios de Gestión de Contenidos (CMS, Content Management System):
  - Facilitan la creación y gestión de contenido web, gestión de usuarios, etc.
  - Periódicos, revistas, web corporativas, universidades, asociaciones, etc.
- Tiendas online y e-comercio
  - Se centran en la exhibición de un catálogo y la gestión relacionada con la venta.
- Foros, chats y sitios de discusión
  - Permiten la inserción de mensajes de los usuarios.
- Los CMS incorporan a los demás en muchas ocasiones.
- Hay muchos paquetes disponibles que facilitan la implementación: joomla, drupal, phpbb, magento, etc.

Pero, ¿qué es una aplicación web?

- Algunos de ejemplos anteriores en cierto modo son "autocontenidos", en el sentido de que se centran en ofertar "un producto", p.e. "un periódico".
- Otras aplicaciones son "servicios" que aprovechan la **infraestructura de transporte y almacenamiento de datos de Internet.**
- **Internet es una plataforma de distribución de contenidos y servicios**. Los servicios se construyen utilizando su infraestructura básica (tecnologías, redes y protocolos).
- Pero, ¿el modelo anterior describe la forma/organización de todos los servicios que encontramos en la web? No siempre. Por ejemplo:
  - **Agregadores de contenidos**: portales de búsqueda de viajes y hoteles, buscadores temáticos, etc. Extraen información de otros servidores .
  - Su servicio funciona porque "**interrogan" diferentes servidores** y muestran organizadamente los resultados. La arquitectura anterior no es completa.
  - No existe intervención de un usuario intermedio ¿Cómo puede un servidor interrogar a otro servidor? Es decir, ¿existen procedimientos estándar para que los servidores se intercambien datos entre sí utilizando la plataforma de Internet? Sí: llamadas a procedimiento remoto, lo estudiaremos en prox. tema.
- Aún así, hay otros servicios que no encajan en una categorías: en los ejemplos anteriores el contenido lo genera una entidad identificable: compañía aérea, editor, etc. pero ¿y los foros o redes sociales o blogs?

Web 2.0

- El término se aplica en diversos contextos y tecnologías.
- **Internet es una plataforma de distribución de contenidos y servicios**. ¿Cómo desarrollamos u organizamos esos servicios aprovechando al máximo esta infraestructura?
- El concepto de **Web 2.0 es el resultado de aplicar cierta filosofía en el desarrollo de aplicaciones y servicios en Internet** que se basa en:
  - Los servicios no se "empaquetan y se venden". El **servicio** se pone a disposición del usuario (gratis o no) y está **en continua evolución**. E.g.: enciclopedia en CD vs Wikipedia, MS Office vs Google Docs.
  - Arquitectura de **participación**.
  - Datos y servicios **remezclables y reutilizables**. Mash-ups

Web 2.0: participación

- El contenido lo generan los propios usuarios: blogs, redes sociales, wikis. Incluso lo programan los propios usuarios.
- Los servicios mejoran gracias a la propia respuesta de los usuarios: la actividad y decisión individual se agrega de manera que el resultado es mejor, más interesante o más fiable que dichas actividades aisladas. "**Inteligencia colectiva**". E.g: Amazon recomienda libros en función de lo que los usuarios compran o visitan más. Es más fiable que las recomendaciones de la editorial, p.e.
- Permite que los servicios escalen y evoluciones de acuerdo con la adopción por parte del usuario y no fija un comportamiento del mismo predefinido.
- Por tanto, la arquitectura de muchos servicios se diseña teniendo en cuenta la participación del usuario.

Web 2.0: mezclas y mashups

- Los servicios se diseñan con componentes pequeños y con una funcionalidad bien definida que se puede componer.
- Estos pequeños componentes pueden ser transformados, compuestos, remezclados, alterados para crear nuevos servicios (mashups). E.g.: panoramio, wikitude, aplicaciones facebook, etc.
- La web es programable: los usuarios programan nuevos servicios de manera que se generan relaciones dinámicas entre los servicios y sus usuarios.
  - Google, Facebook, Amazon, etc. publican sus API para trabajar con sus servicios.
  - Hay mayor número de fuentes de datos: infraestructura pública de datos espaciales (IPDE), GPS, sensores, cámaras, etc.
  - La web proporciona la infraestructura en forma de redes y estándares abiertos y fuentes de datos disponibles.

Ejemplo: aplicación de geolocalización de fotos (I)

Ejemplo: Aplicación de geolocalización de fotos (II)

- La fotografía se sube al servidor: bien mediante el móvil o bien el propio usuario. Para ello el usuario se registra previamente en el servicio (usa un formulario HTML) => Las aplicaciones del servidor guardan esa información en el SGBD.
- Mediante el API javascript de Google Maps se pueden situar las fotos sobre el mapa, junto con comentarios.
  - Las aplicaciones del servidor generan dinámicamente el javascript+HTML+CSS necesario para mostrar las fotos de un determinado usuario.
  - Al generar el javascript las funciones del API Google Maps se combinan (invocan) con los datos de la BBDD (coordenadas y URL de la foto).
  - El navegador, al cargar la página ejecuta el javascript, que internamente y en segundo plano descarga las fotos de los mapas, mediante AJAX.
  - De hecho, la implementación de la funcionalidad básica javascript se descarga de los servidores de Google, el resto consiste en invocaciones del usuario de las funciones declaradas en el API.
  - Finalmente, las fotos del usuario se descargan del servidor dónde se aloja la aplicación.

Referencias y bibliografía

- I. Taylor y A. Harrison, “From P2P and Grids to Services on the Web”, 2º Ed., Springer
- La web en general
- Algunas API de desarrollo:
  - Google Maps: http://code.google.com/intl/es-ES/apis/maps/
  - Code playground: http://code.google.com/apis/ajax/playground/
  - Facebook: http://developers.facebook.com/
- Y muchas de ellas en un solo enlace: http://www.programmableweb.com/apis