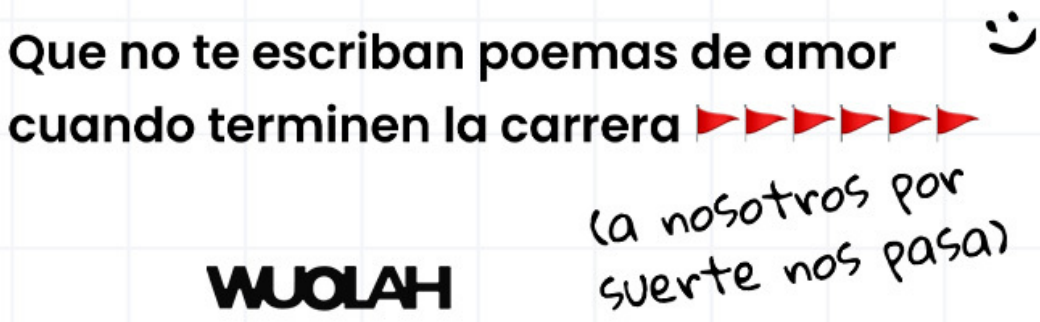
Resumen-SI-Temas-1-y-2-wuolah.pdf

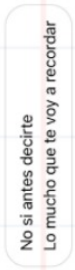
EPS_UAM_4o

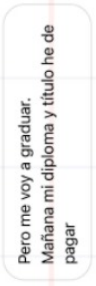
Sistemas Informaticos I

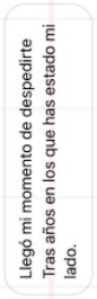
3º Grado en Ingeniería Informática

Escuela Politécnica Superior
Universidad Autónoma de Madrid









ÍNDICE

Tema 1: Introducción a los Sistemas Distribuidos

- Coordinación hace referencia a cómo se comunican y coordinan los nodos:
  - Acoplamiento temporal
  - Acoplamiento referencial (espacial)

- Coordinación directa: acoplamiento referencial y temporal

- Distintos modelos arquitectónicos de sistemas distribuidos consiguen flexibilidad mediante modelos de coordinación indirecta:
  - Desacoplamiento referencial: los nodos no se conocen de forma explícita
  - Desacoplamiento temporal: los nodos no tienen que estar necesariamente activos de forma simultánea para coordinarse

-> Tipos de SD

Atendiendo a su grado de acoplamiento (HW):

- **Fuertemente acoplados**: Procesadores que comparten memoria o buses de entrada/salida. Aplicaciones multiprocesador
- **Débilmente acoplados**: Procesadores autónomos interconectados por sistemas de comunicaciones

Atendiendo a su arquitectura software típicamente distinguimos dos tipos:

- **Igual a igual** (peer to peer, p2p):
  - Sistema simétrico
  - Todos los procesos desempeñan tareas semejantes
  - Interactúan para realizar una actividad distribuida
  - Interacción N-N
- **Cliente-servidor**:
  - Sistema asimétrico
  - Procesos clientes solicitan servicios
  - Procesos servidores los ejecutan y devuelven los resultados
  - Interacción N-1

-> Arquitectura SW

Componentes:

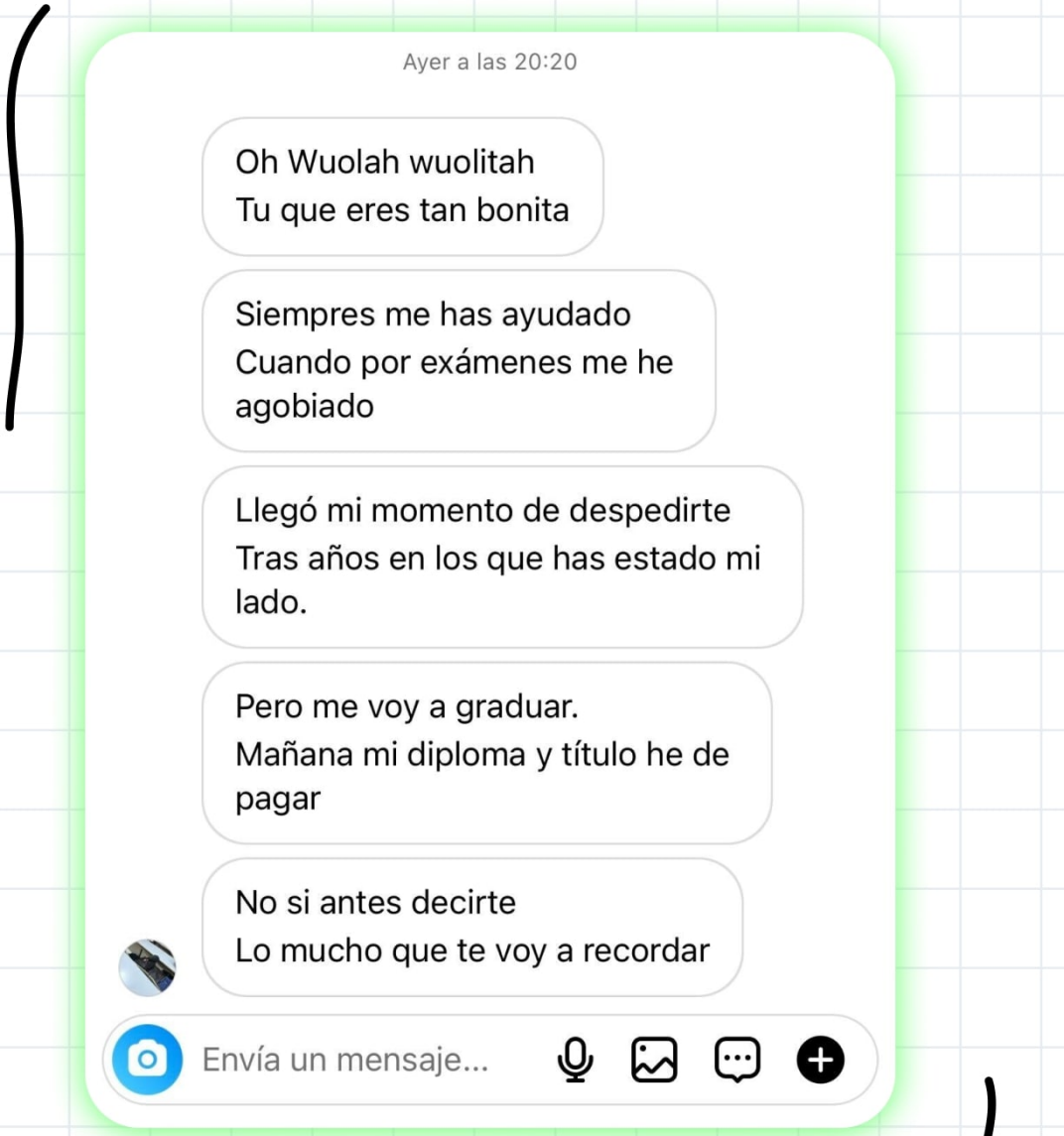
- clientes
- servidores
- bases de datos
- ...

Interacciones:

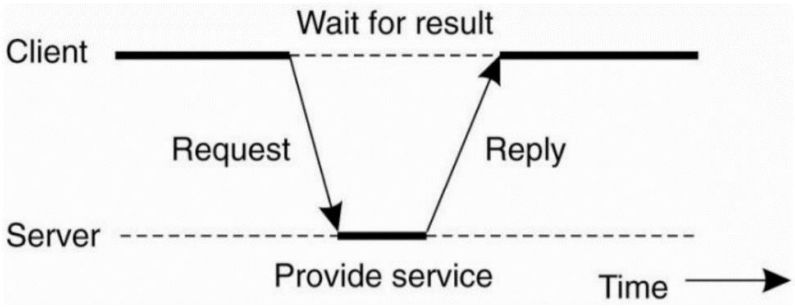
- llamadas a procedimientos (p.ej., RPC o RMI)
- mensajes
- compartición de variables
- protocolos cliente/servidor
- protocolos de acceso a BB.DD.
- streaming
- ...

Que no te escriban poemas de amor cuando terminen la carrera 🚩🚩🚩🚩🚩🚩

(a nosotros por suerte nos pasa)



WUOLAH



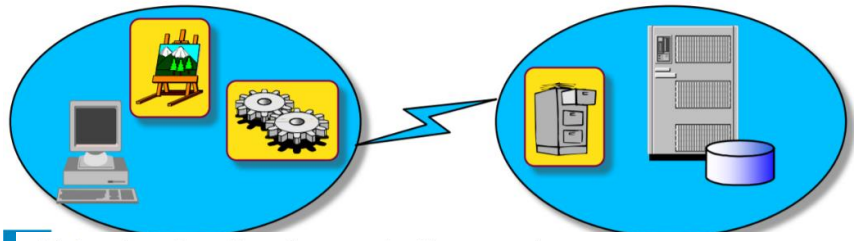
- Capa: se refiere a la arquitectura SW del sistema (capa lógica, *layer*)
- Nivel: se refiere a la arquitectura HW del sistema (nivel físico, *tier)*

El SI consta de tres elementos principales:

- **Capa de interfaz de usuario-aplicación** (normalmente, gráfica)
- **Capa de procesamiento:** lógica de aplicación
- **Capa de datos:** datos que el cliente quiere manipular a través de los componentes de la aplicación

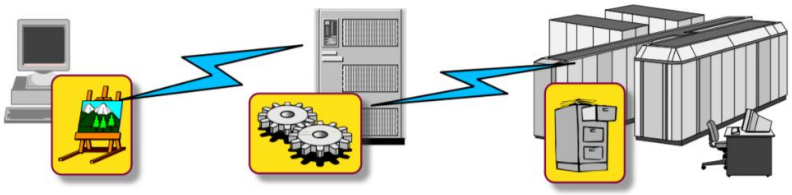
- La lógica de la aplicación se ejecuta junto con la lógica de presentación (modelo cliente pesado)
- El servidor realiza acceso a datos
- Implementación sencilla para aplicaciones pequeñas
- Ampliación de la aplicación y migración a otros entornos compleja

2 Niveles

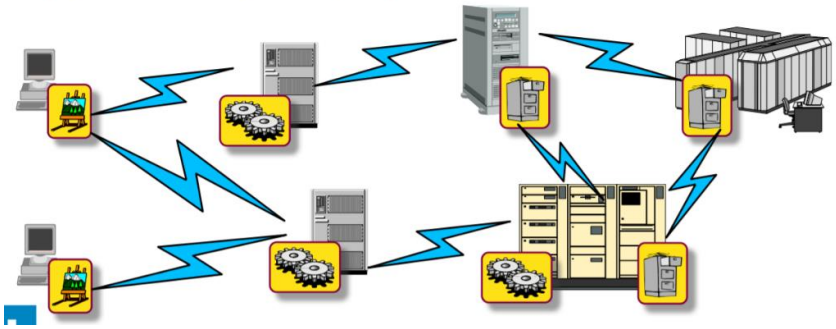


- Lógica de aplicación y de presentación separadas
- La lógica de la aplicación reside en cualquiera de los elementos de la red
- Aplicaciones más robustas y fácilmente escalables

3 Niveles



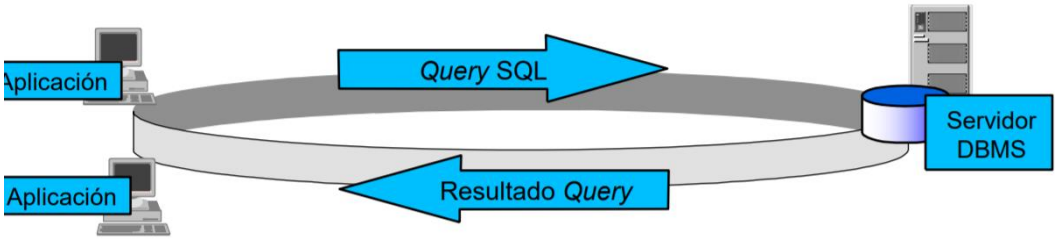
- Múltiples niveles de clientes y servidores
- Representan nuevos modelos de arquitecturas de sistemas distribuidos basados en la arquitectura cliente-servidor



N Niveles

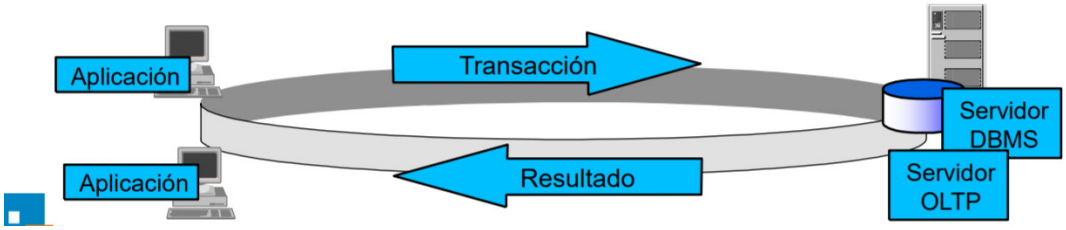
-> Servidores de BBDD

- El cliente pasa una petición (*query*) SQL en un mensajes al servidor (*Data Base Management Server, DBMS*)
- Los resultados de cada consulta SQL se devuelven por la red
- El código que ejecuta la petición SQL se encuentra en el mismo ordenador que los datos



-> Proceso de transacciones

- El cliente solicita la ejecución de un procedimiento remoto en el servidor
- El procedimiento ejecuta un grupo de peticiones SQL o de otro tipo (**transacción**)
- Todas las peticiones se ejecutan o fallan como una unidad
- El resultado final se devuelve al cliente
- La aplicación se desarrolla escribiendo el cliente y el código de las transacciones en el servidor
- *Online Transaction Processing, OLTP*



-> SSDD basados en la WWW

- En origen, sistemas distribuidos basados en la extensión del modelo de cliente ligero bajo protocolo HTTP para el intercambio de información
- Clientes:
  - Universales: navegadores web (Web Browsers)
  - Específicos: programas ejecutados en el cliente
- Servidores Web + Servidor de Aplicaciones:
  - Repositorios de documentos
  - Entorno de ejecución de aplicaciones
- Servidores de *Back-End*:
  - Enlace con programas ya existentes o desarrollos antiguos (*legacy systems*).
  - Programas en el servidor Web les realizan consultas a través de algún tipo de middleware más o menos elaborado
  - Los casos más comunes son:
    - Servidores de bases de datos
    - Servidores de proceso de transacciones.

-> Características computación en la nube

1. **Servicios bajo demanda**. Las capacidades (procesamiento y recursos físicos o virtuales) se adquieren bajo demanda en función de las necesidades
2. **Amplio acceso de red**. Los servicios se encuentran disponibles en una red que puede ser privada, compartida o pública, siendo accesibles a través de mecanismos estándar que permiten su uso por clientes ligeros heterogéneos
3. **Pooling de recursos**. Las capacidades se asignan y reasignan dinámicamente entre los distintos "clientes" en función de la demanda
4. **Rápida elasticidad**. El escalado horizontal y vertical de capacidades debe producirse rápidamente (en algunos casos incluso de forma automática)
5. **Medición de servicios**. El uso de recursos debe monitorizarse y reportarse de manera automática de cara a su control y optimización

-> Modelos de despliegue comp. Nube

1. **Nube privada**. Sólo una organización tiene acceso a la infraestructura
2. **Nube pública**. El acceso a la infraestructura es abierto
3. **Nube comunitaria**. La infraestructura es compartida por varias organizaciones
4. **Nube híbrida**. La infraestructura general se compone de dos o más infraestructuras con distintos modelos de despliegue

-> Modelos de servicio comp. Nube

1. **Infraestructura (*IaaS*)**: Se proporciona capacidad de procesamiento, almacenamiento, red y otros recursos computacionales fundamentales (servidores, sistemas operativos, **virtualización**…). Esto permite desplegar y ejecutar cualquier software arbitrario. El proveedor del servicio es el dueño del equipamiento y el responsable del housing y el mantenimiento. El "cliente" controla y administra el sistema operativo, las aplicaciones, los datos, y ciertos componentes de la red (p.ej., firewalls)
2. **Plataforma (*PaaS*)**: Este tipo de arquitectura está orientada principalmente a desarrolladores. Ofrece un entorno preconfigurado de desarrollo/ejecución usando los lenguajes de programación, librerías, servicios y herramientas soportados por el proveedor. El "cliente" no gestiona, ni controla la infraestructura
3. **Software (*SaaS*)**: Se da acceso a las aplicaciones que el proveedor ejecuta en su infraestructura, sin tener ningún control sobre ésta

Tema 2: SSDD basados en la WWW

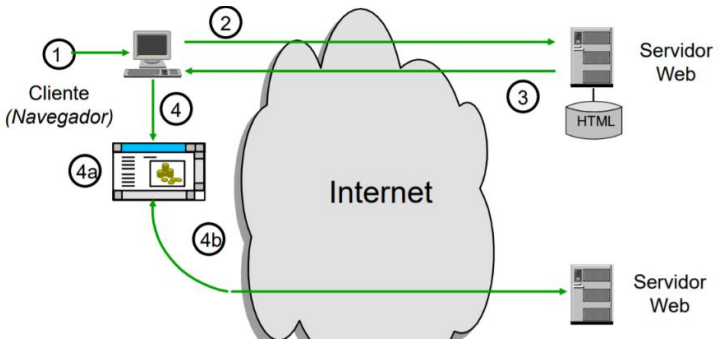
En origen, **sistemas distribuidos** basados en la extensión del modelo de **cliente ligero** bajo **protocolo HTTP** para el intercambio de información

- El **cliente Web** se convierte en la interfaz de uso genérico para la interacción de los usuarios con un sistema distribuido
  - En el modelo actual, las carencias del modelo de cliente ligero universal se suplen con elementos de programación complementarios
- El **servidor Web** pasa a ser el núcleo básico donde se gestiona la ejecución de programas

El cliente accede a los documentos a través de un visualizador (*browser*) o navegador → cliente ligero

-> Funcionamiento web hipertexto

1. El usuario solicita un recurso mediante su URL en un navegador (cliente)
2. El navegador genera una petición HTTP y la envía al servidor Web
3. El servidor Web recibe la petición y envía el recurso solicitado
   - Una sesión TCP por cada solicitud
   - Los documentos se codifican utilizando HTML
4. El cliente interpreta y muestra el documento recibido
   - Puede tener asociados nuevos elementos en su interior
   - Nueva petición HTTP a los servidores que los contienen para su recuperación
   - Originalmente nueva sesión TCP, el protocolo HTTP actual permite reutilizar sesiones



si lees esto me debes un besito

-> Protocolo HTTP

Intercambio de mensajes ASCII entre ambos:

- Cliente realiza una petición

```
GET /hypertext/www/TheProject.html HTTP/1.0
```

- El servidor responde con un mensaje MIME (*Multipurpose Internet Mail Extensions*):

```
HTTP:/1.0 200 Document follows
MIME-Version: 1.0
Server:CERN/3.0
Content-Type: text/html
Content-Length: 8247

<HEAD><TITLE>The World Wide Web Consortium (W3C)</TITLE></HEAD>
<BODY>
<H1><IMB ALIGN=MIDDLE ALT="W3C" SRC="icons/WWW(w3c_96x67.gif>
The World Wide Web Consortium</H1><P>
```

Formato de las peticiones: [método] URI [protocolo]

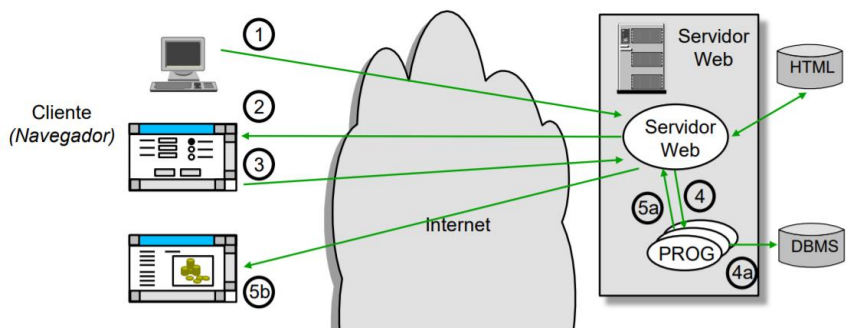
método: PUT, DELETE, GET, POST protocolo: http/1.0 o http/1.1

-> Web Interactiva

El modelo Web hipertexto no permite más interacción del usuario que seguir hiperenlaces para obtener **contenido estático** en una lectura no secuencial de documentos

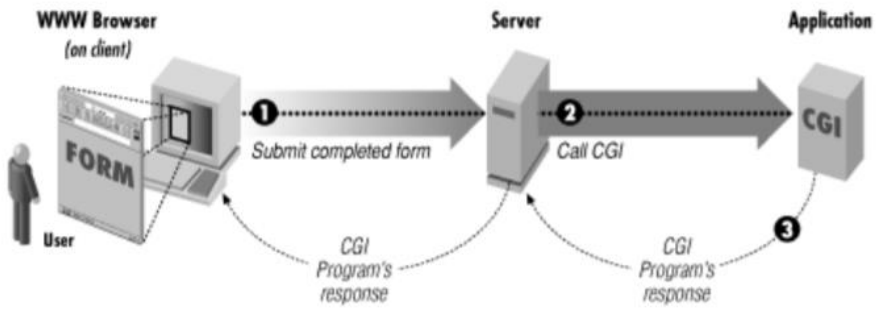
Necesario establecer comunicación entre programas que se ejecutan en el "servidor" con los datos que proporciona el usuario (del lado del cliente)

1. El cliente solicita una página al servidor web
2. El servidor web envía una página que contiene un formulario
3. El cliente envía el formulario al servidor con los datos "suministrados" por el usuario de la aplicación
4. *El servidor web:*
   a. *Recibe y procesa la petición HTTP/S*
   b. *Identifica el recurso o servicio que debe procesar el formulario por su URL*
   c. *Extrae los parámetros de la llamada*
   d. *Invoca a la funcionalidad correspondiente pasándole los datos recibidos del cliente*
5. El programa genera una página con los resultados de su ejecución y la devuelve al cliente a través del servidor web



si lees esto me debes un besito

- CGI define un método "estándar" para que un servidor WWW pueda ejecutar programas externos y recoger información de ellos
- El programa externo recibe del servidor web información:
  - Asociada a la transmisión: Origen, URL, protocolo utilizado...
  - Introducida por el usuario, en un formulario de entrada.



- Asociado al concepto de sesión, se habla de la sesión como un contexto persistente en el que almacenar/recuperar datos mientras la relación entre cliente y servidor se mantenga activa
  - El típico ejemplo es un "carrito de la compra"
  - El servidor debe mantener información asociada al usuario de la aplicación a lo largo de todo el proceso
- ... pero HTTP es un protocolo "sin memoria" → cookies HTTP
  - Pequeño fragmento de información que el servidor (con el permiso del navegador) almacena en el cliente
  - Cada vez que el navegador solicite una nueva página al servidor envía también la cookie
  - Aunque en origen se crearon para la comunicación entre el cliente y el servidor, en la actualidad, la funcionalidad en el cliente también puede escribir y leer en las cookies
  - Sobre el papel un gran invento. En la práctica, presentan serios problemas de seguridad

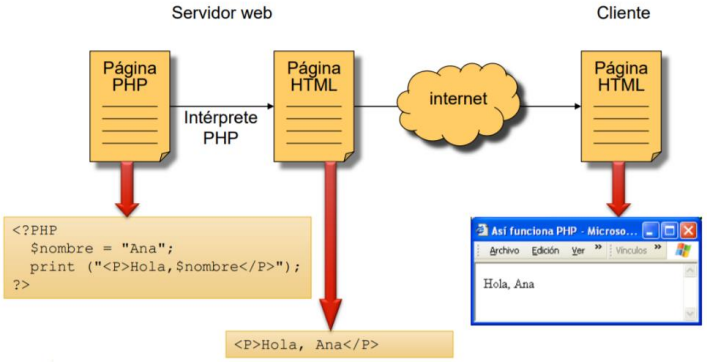
-> Web API

- Surgen para tratar de evitar los problemas de bajo rendimiento de la interfaz CGI:
  - Los nuevos programas se enlazan junto con el servidor en una librería dinámica
  - El servidor llama a las funciones de librería como tareas dentro del propio proceso servidor
  - El proceso servidor no finaliza: se mantienen ficheros abiertos, conexiones a bases de datos, etc. entre llamadas a funciones
  - Se proporciona una API de acceso a los datos y estado del servidor

- Interfaces híbridas entre CGI y API: diapo 4 (2.3.2)

->PHP

- En el contexto web utilizado para generar contenido dinámico del lado del servidor.
  - Los scripts PHP están incrustados en los documentos HTML y el servidor los interpreta antes de servir las páginas al cliente
  - El cliente no ve el código PHP, sino los resultados que produce
  - Software abierto y gratuito
  - Integra acceso a formularios, archivos, bases de datos...



-> Objetivos JavaScript, Ajax, JQuery

- Aumentar la capacidad de interacción del usuario con la aplicación
- Disminuir el trasiego de datos entre el cliente y el servidor
  - Capturar los eventos generados por el usuario y responder a ellos sin depender del servidor
  - Realizar cálculos sencillos sin necesidad de comunicación con el servidor
  - Validar ciertos campos de formulario antes de enviarlos al servidor

-> JavaScript

- Interpretado por el motor *Javascript* del navegador (cliente)
  - No se compila
  - No se ejecuta de forma independiente como aplicación autónoma
  - Se inserta en el código HTML y es interpretado por el navegador junto al resto del documento

-> Servlets

- Aplicación Java que se ejecuta en el servidor gestionando y procesando peticiones HTTP
  - Se ejecutan totalmente en el servidor bajo petición de un cliente (vs. applets)
  - Se invocan a través de una URL que identifica el programa
  - Reemplaza a los programas de interfaz CGI. Sintaxis más sencilla

¿Te está sirviendo de ayuda este resumen?

Este resumen es una demostración del documento completo, que contiene **30** páginas en su totalidad, si quieres verlas todas escribe a juanlu.sc56@gmail.com, respondo en poco tiempo 😊

Mi nombre es Juanlu, soy un estudiante que ya ha superado todos los créditos de las asignaturas del grado de Ingeniería Informática en la EPS UAM.

Ofrezco apuntes, clases y apoyo en prácticas (tengo todas las prácticas hechas), contacta conmigo en juanlu.sc56@gmail.com.

si lees esto me debes un besito