CIFP César Manrique

Introducción a Arquitecturas de desarrollo Web

CFGS DAW – Desarrollo web en entorno servidor

CONTENIDOS

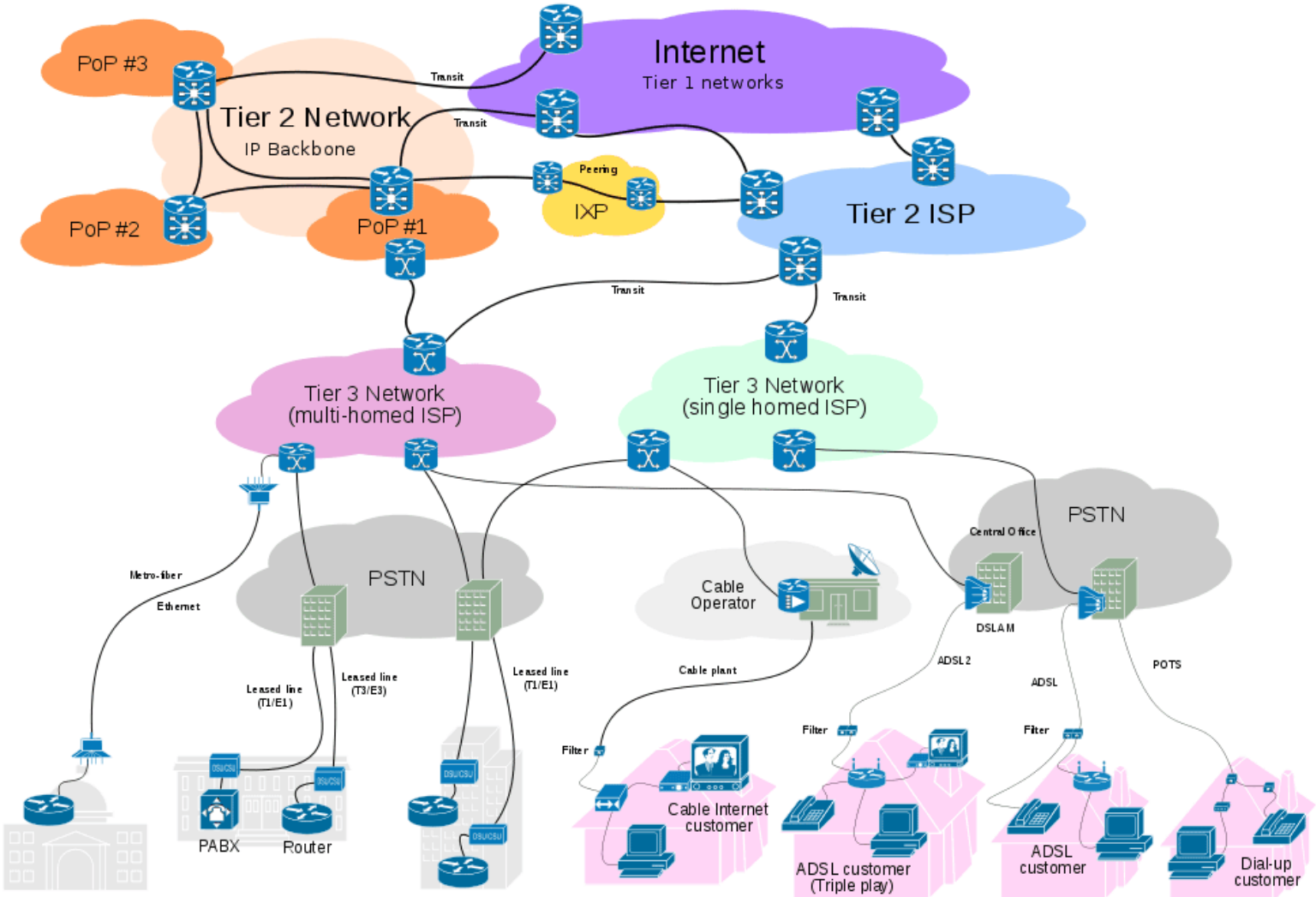
- Internet y la World Wide Web (WWW)
- Introducción al desarrollo de aplicaciones Web
- Arquitecturas hardware - software

Internet y la World Wide Web

Internet

- Internet es el sistema global de redes de computadores interconectadas, que usan la suite de protocolos TCP/IP para conectar millones de dispositivos en todo el mundo. Es la red de redes.
- Su origen está en **ARPANET**, red de computadoras creada por el Departamento de Defensa de Estados Unidos.
- Uno de los servicios que más éxito ha tenido en internet ha sido la World Wide Web (WWW o la Web), hasta tal punto que es habitual la confusión entre ambos términos.
- La Web es un sistema de distribución de documentos de hipertexto interconectados y accesibles vía Internet, a través del protocolo HTTP. La Web se nació en el CERN , a principios de los 90, gracias al trabajo de Tim Berners-Lee con la ayuda de Robert Cailliau.

Internet: La red de redes



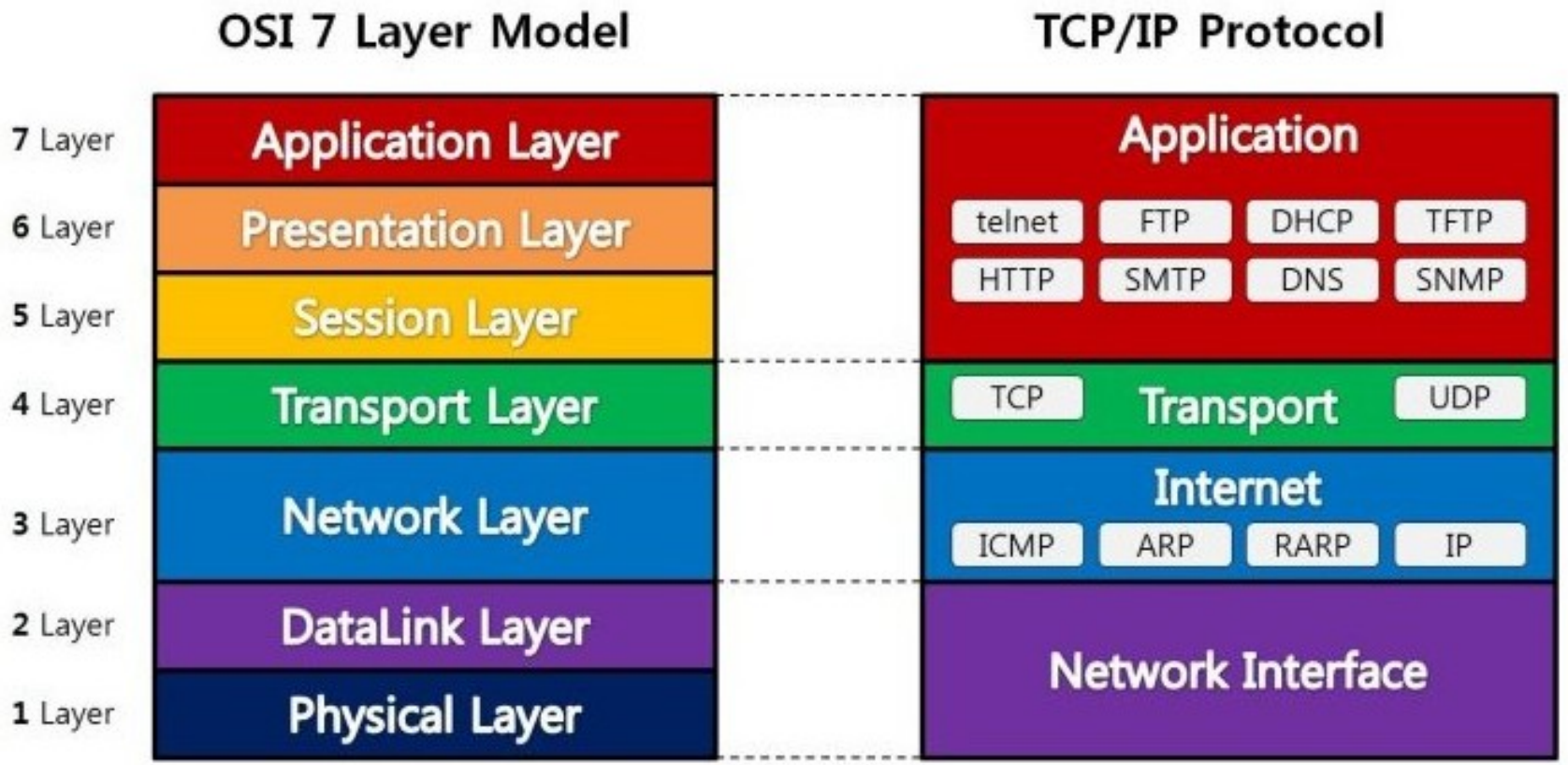
http://arstechnica.com/security/2013/04/can-a-ddos-break-the-internet-sure-just-not-all-of-it/

Protocolo HTTP

- Hypertext Transfer Protocol o **HTTP** (protocolo de transferencia de hipertexto) es el protocolo de comunicación que permite las transferencias de información en la World Wide Web. HTTP fue desarrollado en 1999 por el World Wide Web Consortium (W3C) junto con la Internet Engineering Task Force (IETF). El RFC 2616 especifica la versión 1.1 del protocolo, usada actualmente.
- HTTP sigue un modelo de comunicación de tipo petición-respuesta (**request-response**/reply) entre un cliente y un servidor.
- HTTP es un protocolo **sin estado**, lo que quiere decir que las peticiones son independientes unas de otras, no se guarda información (estado) entre peticiones.
- Las aplicaciones Web sí necesitan, por contra, vincular las diferentes peticiones de un mismo usuario. Para ello se ha creado el concepto de **sesión** en el servidor. Para "recordar" al servidor HTTPD la sesión del usuario se utilizan **cookies** o técnicas como URL rewriting.

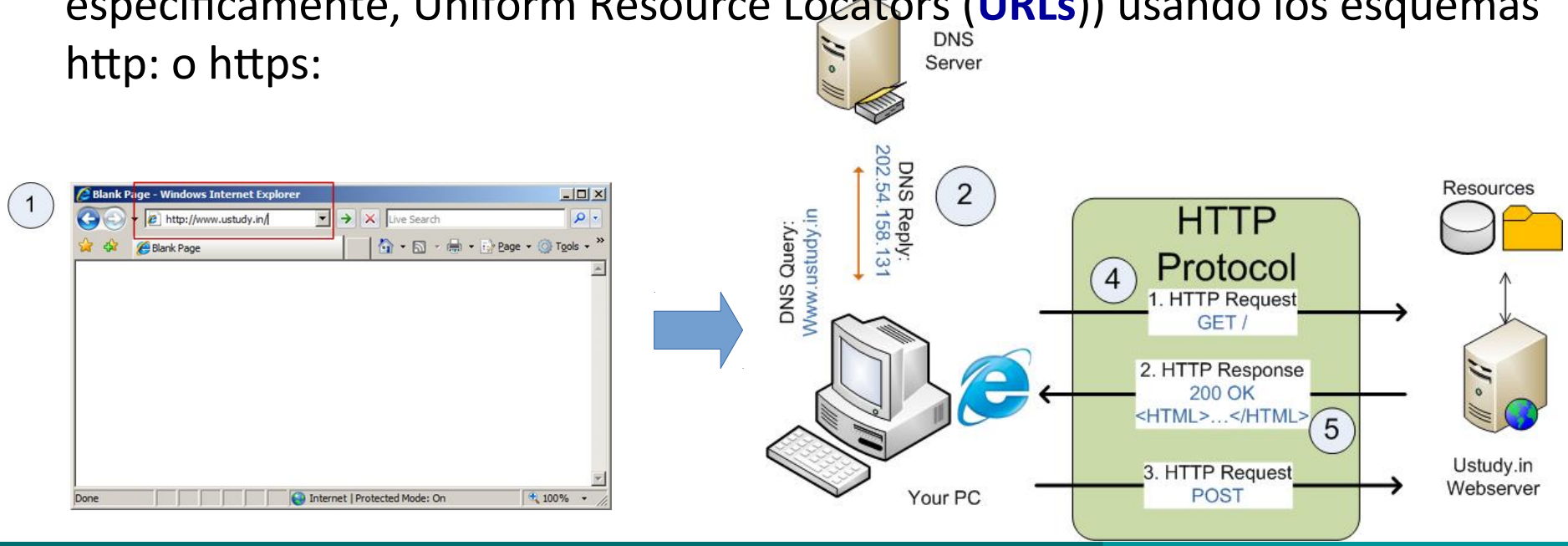
HTTP es un protocolo de nivel aplicación

- El protocolo HTTP utiliza los servicios del protocolo de nivel de transporte (TCP).



Interacción

- El cliente (o user agent) envía una petición HTTP (request) a un servidor Web (el cliente lo hace a través de un *client side socket*)
- El servidor está escuchando (***server side socket***) por peticiones en un puerto TCP, que usualmente es el 80. Cuando recibe la petición busca el recurso solicitado y lo devuelve al cliente
- Los recursos se identifican por URIs (*Uniform Resource Identifiers*) (o, más específicamente, Uniform Resource Locators (**URLs**)) usando los esquemas http: o https:



Métodos HTTP

- **OPTIONS**. Solicita opciones de comunicación disponibles
- **GET**. Solicita un recurso al servidor identificándolo por su URI
- **HEAD.** Idéntico a GET pero solo devuelve cabeceras de respuesta
- **POST.** Envío de datos al servidor (campos de formularios, etc.)
- **PUT.** Almacenar un documento en la URI especificada
- **DELETE.** Borrar el documento indicado por la URI
- **TRACE.** (ECO) Obtener del servidor copia de la petición enviada
- **CONNECT.** Reservado

https://www.w3.org/Protocols/rfc2616/rfc2616.html (derogado por otros RFC)

URI (URL, URN)

- Un identificador de recursos uniforme o URI —del inglés uniform resource identifier — es una cadena de caracteres que identifica los recursos de una red de forma unívoca. La diferencia respecto a un localizador de recursos uniforme (URL) es que estos últimos hacen referencia a recursos que, de forma general, pueden variar en el tiempo.
- Un URI consta de las siguientes partes:
  - Esquema: nombre que se refiere a una especificación para asignar los identificadores, e.g. urn:, tag:, cid:. En algunos casos también identifica el protocolo de acceso al recurso, por ejemplo http:, mailto:, ftp:, etc.
  - Dominio: elemento jerárquico que identifica el host (por ejemplo //www.example.com).
  - Ruta: Información usualmente organizada en forma jerárquica, que identifica al recurso en el ámbito del esquema URI y la autoridad de nombres (e.g. /domains/example).
  - Consulta: Información con estructura no jerárquica (usualmente pares "clave=valor") que identifica al recurso en el ámbito del esquema URI y la autoridad de nombres. El comienzo de este componente se indica mediante el carácter '?'.
  - Fragmento: Permite identificar una parte del recurso principal, o vista de una representación del mismo. El comienzo de este componente se indica mediante el carácter '#'.

```
scheme:[//[user:password@]host[:port]][/]path[?query][#fragment]
```

Más

Este es un ejemplo típico de una comunicación **cliente-servidor.**

Los pasos son los siguientes:

Tu ordenador solicita a un servidor web una página con extensión .htm, .html o .xhtml.

El servidor busca esa página en un almacén de páginas (cada una suele ser un fichero).

Si el servidor encuentra esa página, la recupera.

Y por último se la envía al navegador para que éste pueda mostrar su contenido.

¿Podemos ver una página web sin que intervenga un servidor web?

Páginas estáticas y dinámicas

Las páginas que viste en el ejemplo anterior se llaman páginas web estáticas. Estas páginas se encuentran almacenadas en su forma definitiva, tal y como se crearon, y su contenido no varía. Son útiles para mostrar una información concreta, y mostrarán esa misma información cada vez que se carguen. La única forma en que pueden cambiar es si un programador la modifica y actualiza su contenido.

En contraposición a las páginas web estáticas, como ya te imaginarás, existen las páginas web dinámicas. Estas páginas, como su nombre indica, se caracterizan porque su contenido cambia en función de diversas variables, como puede ser el navegador que estás usando, el usuario con el que te has identificado, o las acciones que has efectuado con anterioridad.

Dentro de las páginas web dinámicas, es muy importante distinguir dos tipos:

- Aquellas que incluyen código que ejecuta el navegador. En estas páginas el código ejecutable, normalmente en lenguaje JavaScript, se incluye dentro del HTML (o XHTML) y se descarga junto con la página. Cuando el navegador muestra la página en pantalla, ejecuta el código que la acompaña. Este código puede incorporar múltiples funcionalidades que pueden ir desde mostrar animaciones hasta cambiar totalmente la apariencia y el contenido de la página. En este módulo no vamos a ver JavaScript, salvo cuando éste se relaciona con la programación web del lado del servidor.
- Como ya sabes, hay muchas páginas en Internet que no tienen extensión .htm, .html o .xhtml. Muchas de estas páginas tienen extensiones como .php, .asp, .jsp, .cgi o .aspx. En éstas, el contenido que se descarga al navegador es similar al de una página web estática: HTML (o XHTML). Lo que cambia es la forma en que se obtiene ese contenido. Al contrario de lo que vimos hasta ahora, esas páginas no están almacenadas en el servidor; más concretamente, el contenido que se almacena no es el mismo que después se envía al navegador. El HTML de estas páginas se forma como resultado de la ejecución de un programa, y esa ejecución tiene lugar en el servidor web (aunque no necesariamente por ese mismo servidor).

El esquema de funcionamiento de una página web dinámica es el siguiente:

Páginas dinámicas (II)

Primero el navegador envía una petición al servidor web. Segundo el servidor web busca la página en un almacén de páginas. Tercero, si se trata de una página dinámica, ejecuta el código. Cuarto, al ejecutar el código puede necesitar información de algún repositorio. Quinto, como resultado de la ejecución se obtiene una página en formato HTML. Sexto, el servidor envía la página obtenida al navegador que la solicitó.

Pasos:

1) El cliente web (navegador) de tu ordenador solicita a un servidor web una página web.
2) El servidor busca esa página y la recupera.
3) En el caso de que se trate de una página web dinámica, es decir, que su contenido deba ejecutarse para obtener el HTML que se devolverá, el servidor web contacta con el módulo responsable de ejecutar el código y se lo envía.
4) Como parte del proceso de ejecución, puede ser necesario obtener información de algún repositorio, como por ejemplo consultar registros almacenados en una base de datos.
5) El resultado de la ejecución será una página en formato HTML, similar a cualquier otra página web no dinámica.
6) El servidor web envía el resultado obtenido al navegador, que la procesa y muestra en pantalla.

Este procedimiento tiene lugar constantemente mientras consultamos páginas web. Por ejemplo, cuando consultas tu correo en GMail, HotMail, Yahoo o cualquier otro servicio de correo vía web, lo primero que tienes que hacer es introducir tu nombre de usuario y contraseña. A continuación, lo más habitual es que el servidor te muestre una pantalla con la bandeja de entrada, en la que aparecen los mensajes recibidos en tu cuenta. Esta pantalla es un claro ejemplo de una página web dinámica.

Obviamente, el navegador no envía esa misma página a todos los usuarios, sino que la genera de forma dinámica en función de quién sea el usuario que se conecte. Para generarla ejecuta un programa que obtiene los datos de tu usuario (tus contactos, la lista de mensajes recibidos) y con ellos compone la página web que recibes desde el servidor web.

Más sobre aplicaciones web y su razón de ser en EVAGD

Ejercicio 1 (I)

- **Usaremos la máquina virtual de Ubuntu. No es necesario instalarla.**
- Utilizando un cliente Telnet realizar peticiones HTTP a un servidor.
- Obtener información sobre el uso de telnet.

Ejercicio (II)

Siguiendo los siguientes ejemplos. Visualizar la web completa del diario marca en local.

Usa los siguientes ejemplos, para ayudarte.

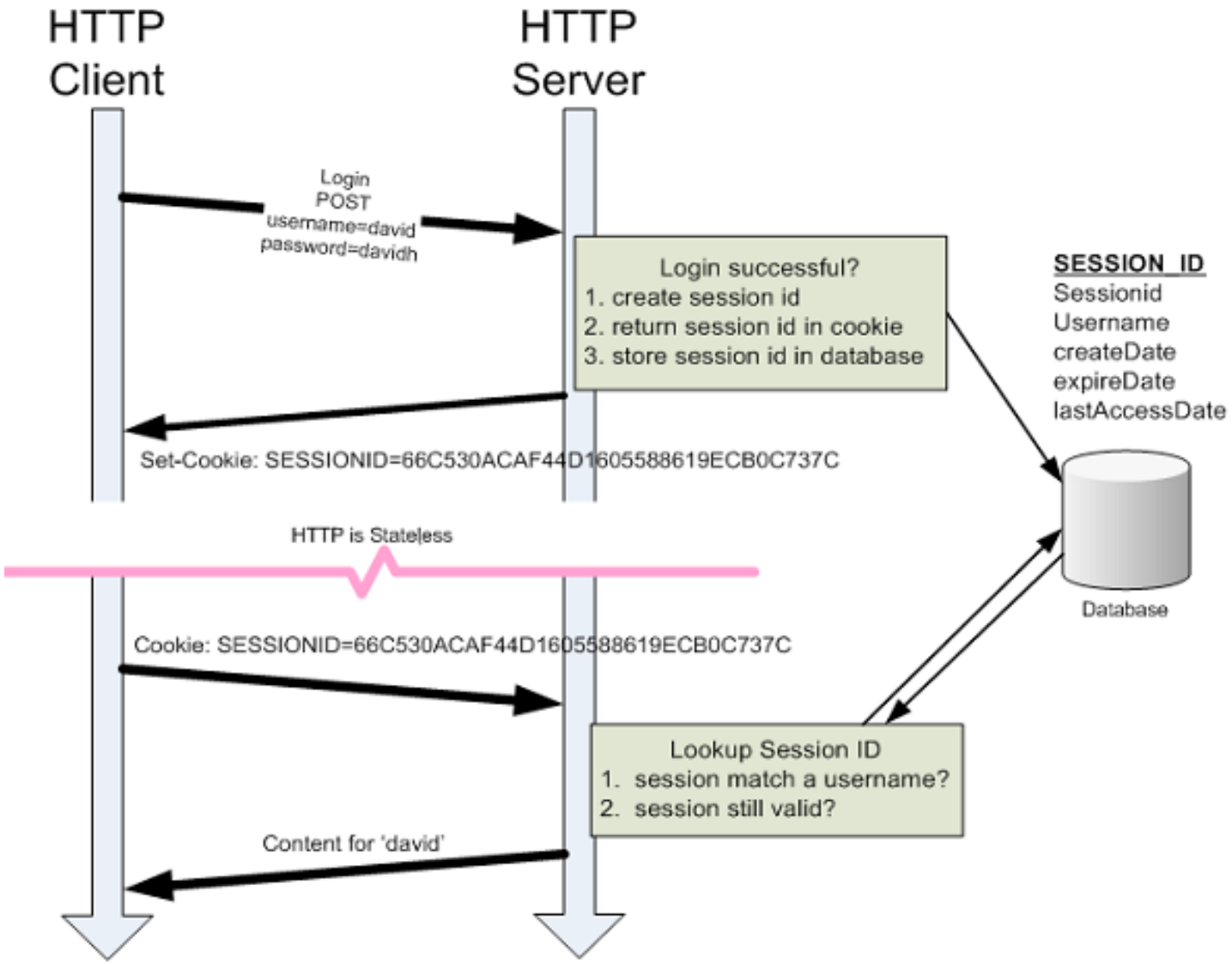
- No te olvides de los dos intros después del host!!!

```
$ telnet [SERVER] [PORT]
Trying xxx.xxx.xxx.xxx...
Connected to [SERVER].
Escape character is '^]'.
HEAD [WEB-PAGE] HTTP/1.1
HOST: [SERVER]
<Press ENTER>
```

```
$ telnet www.shellhacks.com 80
Trying xxx.xxx.xxx.xxx...
Connected to www.shellhacks.com.
Escape character is '^]'.
GET / HTTP/1.1
HOST: www.shellhacks.com
<Press ENTER>
```

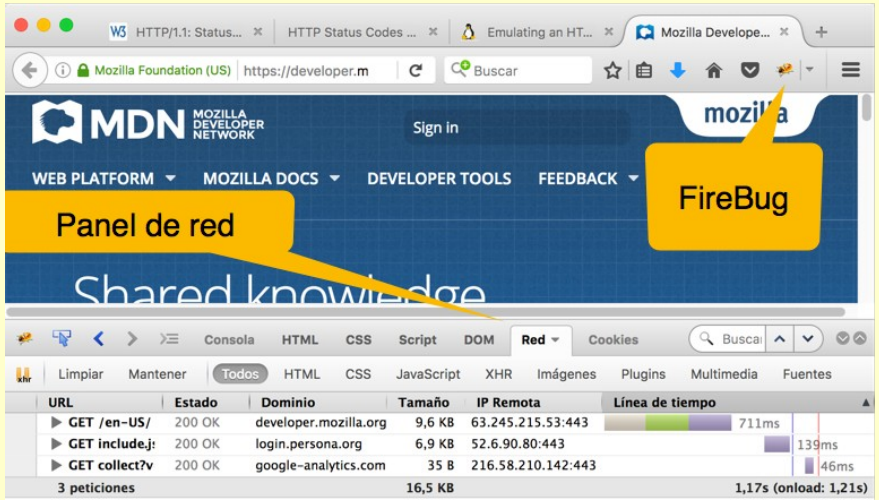
```
$ telnet [SERVER] [PORT]
Trying xxx.xxx.xxx.xxx...
Connected to [SERVER].
Escape character is '^]'.
GET [WEB-PAGE] HTTP/1.1
HOST: [SERVER]
<Press ENTER>
```

Sesiones HTTP y cookies



Ejercicio 2

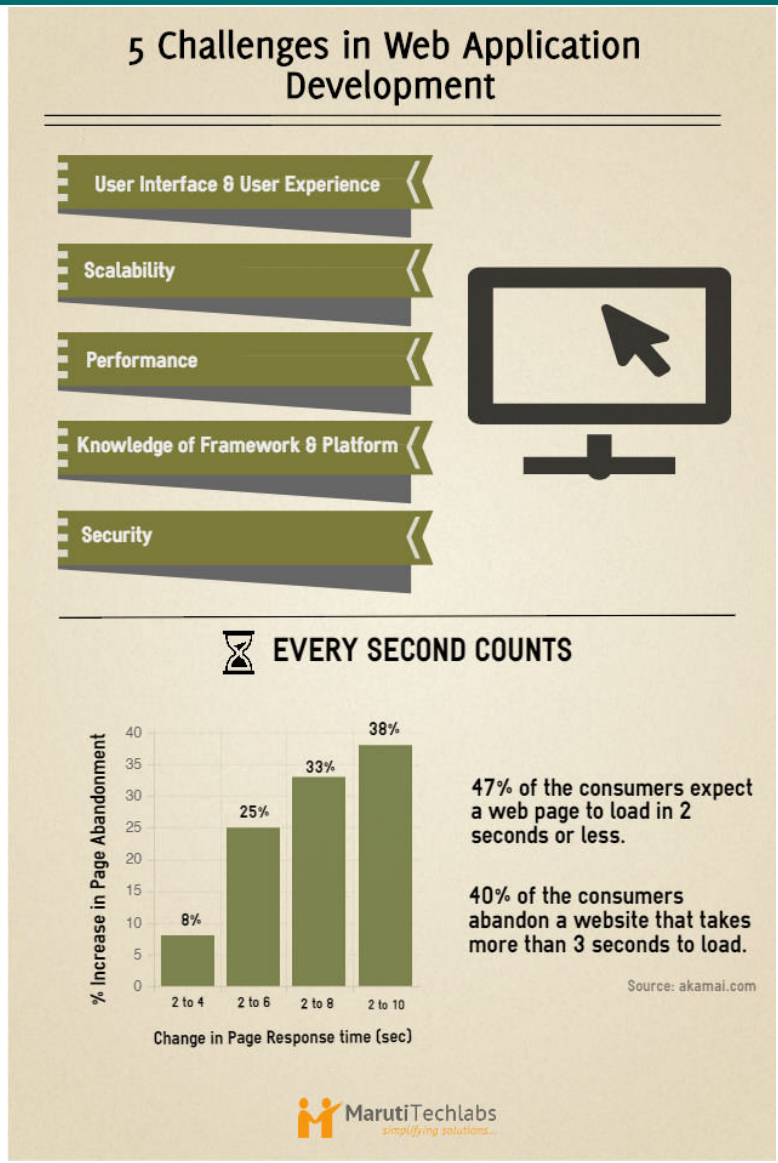
- Que es FireBug?
- Instalalo en tu firefox. U otro complemento similar.
- Habilitar el complemento para todas las Web y el panel de red
- Habilitar y ver las cookies en una petición a http://www.google.com
- Lanzar una petición a http://developer.mozilla.org y localizar la cabecera de la solicitud que indica el idioma preferido
- Cambiar el idioma en Firefox y ver si cambia la cabecera



Introducción al desarrollo de aplicaciones Web

Desafíos a afrontar en el desarrollo Web

- **Rendimiento**
  - Ofrecer alto throughput (n.º transacciones por segundo que puede atender el sistema
  - Bajo tiempo de respuesta del servidor
  - Asegurar la disponibilidad del servicio
- **Escalabilidad**
  - Capacidad de ampliar el sistema cuando aumenta la carga de trabajo
- **Seguridad**
  - Evitar: suplantación, modificación de recursos sin autorización (*tampering*), alterar evidencias, revelación de información, denegación de servicio (DoS), elevación de privilegios



http://www.marutitech.com/5-challenges-in-web-application-development/
http://www.drdobbs.com/web-development/building-scalable-web-architecture-and-d/240142422

Desafíos a afrontar en el desarrollo Web (II)

- Accesibilidad, usabilidad y diseño inclusivo
  - **Accesibilidad**. Busca que las personas con discapacidad puedan percibir, entender, navegar e interactuar con sitios web y herramientas eliminando las barreras de acceso al contenido (ej: discapacidad visual). **WACAG** (Web Content Accessibility Guidelines) define reglas de accesibilidad tales como: proporcionar alternativas textuales a contenido multimedia, crear contenido compatible con tecnologías de asistencia (lectores de contenido), permitir la interacción con el sitio Web eliminando la obligatoriedad del uso del ratón, etc.
  - **Usabilidad y experiencia de usuario.** Enfoque de diseño del software para ser lo más eficaz, eficiente y satisfactorio. ISO 9241-11 define usabilidad como: "grado en que un producto puede ser usado por determinados usuarios para conseguir objetivos específicos con efectividad, eficiencia y con satisfacción en un contexto de uso". Aplicaciones **RIA** (*Rich Internet Applications*)
  - **Inclusión**. Diseño inclusivo, universal y para todos implica el diseño de sitios web, de forma que lleguen a todo tipo de usuarios. Un área importante es el **Responsive Web Design (RWD)**, que permite que el sitio Web se adapte al dispositivo desde el que se accede (ejemplo: móvil). Inclusión abarca, además, el acceso y la calidad del hardware, software y conectividad a Internet; conocimientos de los usuarios; ubicación geográfica y lenguaje, así como la edad y la discapacidad (aunque ésta se trata más específicamente en Accesibilidad)

http://www.w3.org/WAI/users/Overview.html

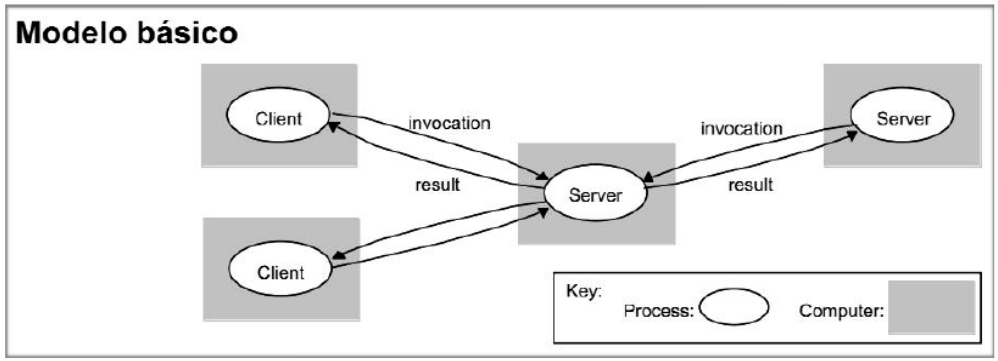
Desafíos a afrontar en el desarrollo Web (III)

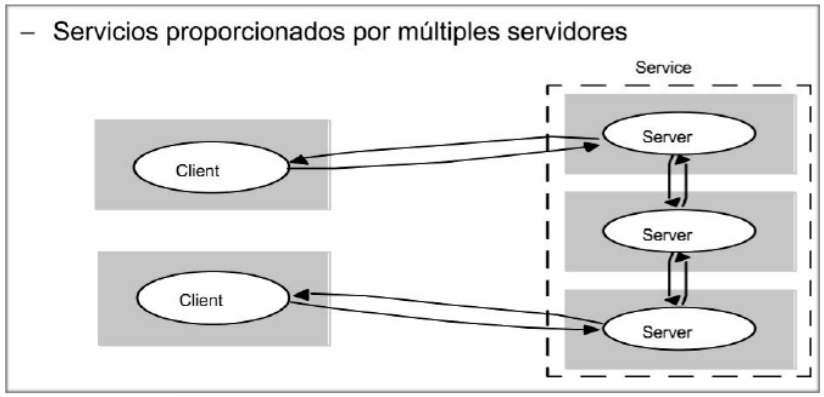
- **Integración con otras aplicaciones y el backend**
  - Nuestra aplicación ofrece un ***front-end*** (en nuestro caso un cliente web, denominado cliente ligero), con el que los usuarios pueden trabajar.
  - Además desarrolla una lógica de negocio, proporcionando servicios a los usuarios.
  - Para desarrollar esta lógica de negocio la aplicación precisa acceder a los datos del ***back-end***; ya sea de forma directa (conectando directamente con bases de datos), o empleando servicios de terceros (otras aplicaciones).
  - En este segundo caso aparece el concepto de ***Middleware,*** que es una capa de abstracción de software distribuida, que se sitúa entre las capas de aplicaciones y las capas inferiores (sistema operativo y red) para facilitar el intercambio de datos entre aplicaciones, proporcionando una API para la programación y manejo de aplicaciones distribuidas.
  - Sea cual sea el enfoque (normalmente se combinan ambos), nuestras aplicaciones deberán utilizar diversos protocolos, conectores, frameworks para integrar datos procedentes de diversas fuentes, consolidarla y generar información en base a esos datos.

Arquitecturas
Hardware
Software

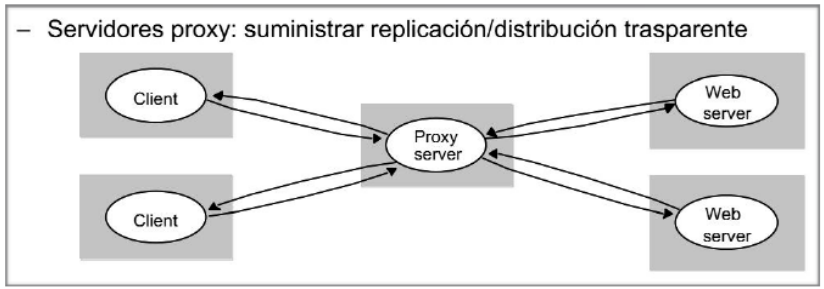
Ejemplo: Configuración de servidores

Los Sistemas Distribuidos son los sistemas de software más complejos, la corporación Nortel Networks por ejemplo, crea switches los cuales pueden contener entre 25-30 millones de líneas de código, interviniendo 3000 desarrolladores de software, y con un ciclo de vida de 20 años para actualizar.





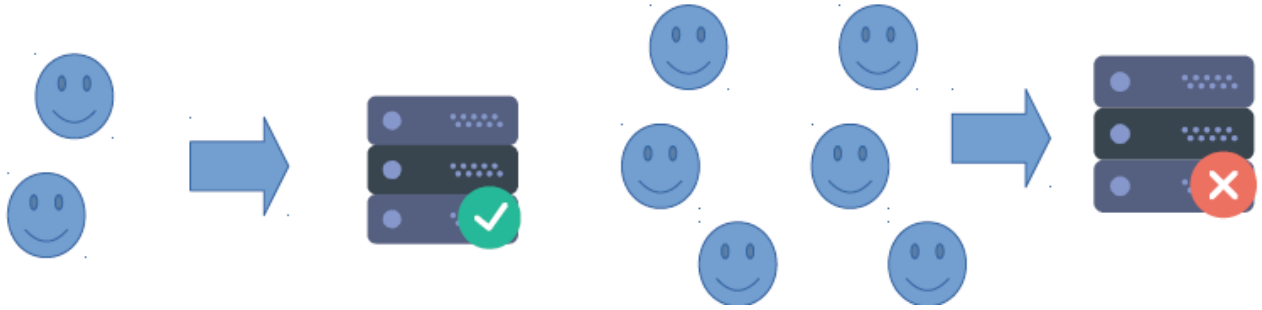
(...) Muchos servicios de comercio Web están implementados en diferentes servidores. son muy confiables gracias a que mantienen bases de datos replicadas o distribuidas.



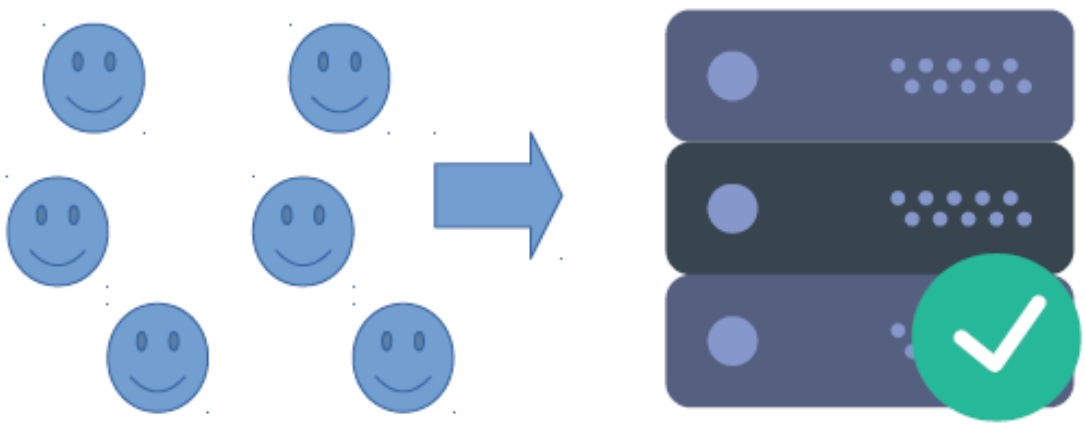
(...) Uso de Caching. Los servidores proxy mantienen caches, como almacenes de recursos solicitados recientemente. Son redes Utilizados frecuentemente en motores de búsqueda como google,etc.

Problema: Aplicación Web → alto número de potenciales clientes

- La aplicación responde bien hasta un número determinado de usuarios, pero cuando se sobrepasa la capacidad del sistema éste se satura y deja de dar servicio. El cliente recibe **Errores HTTP 5xx**



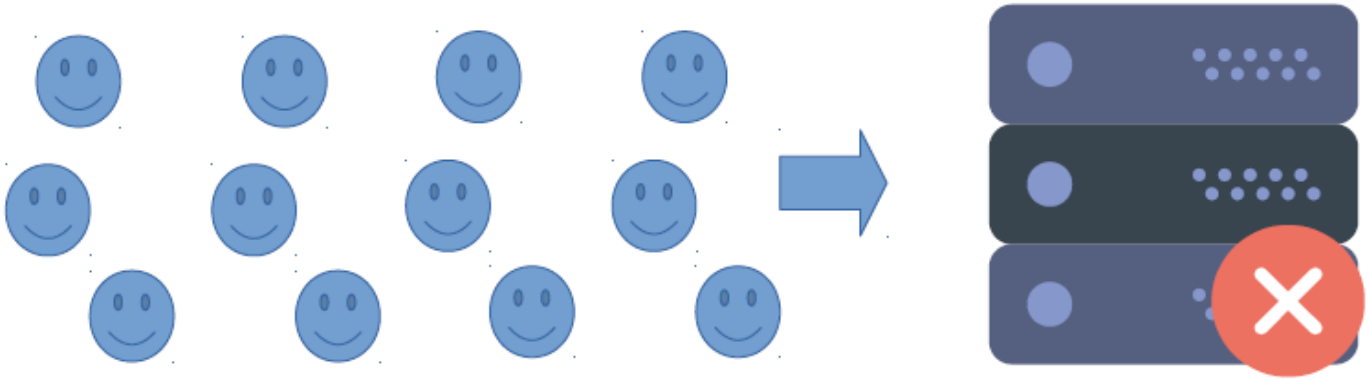
- Una solución r uno más grande)



Iconos creados por MadeByOliver

Problemas para aumentar la capacidad del sistema (escalabilidad)

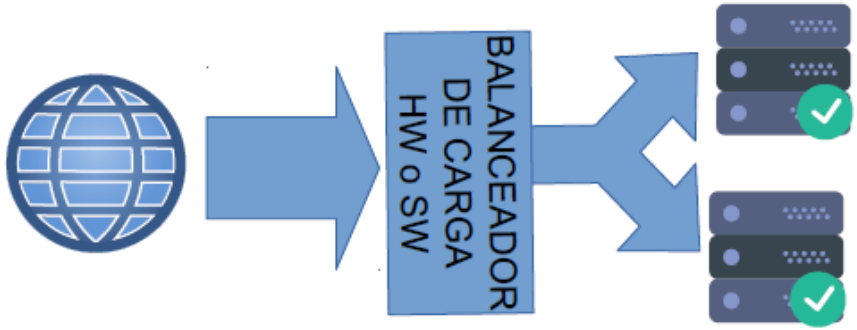
- El problema es que nuevos aumentos en la carga del sistema hacen que necesitemos cada vez máquinas “más gordas”



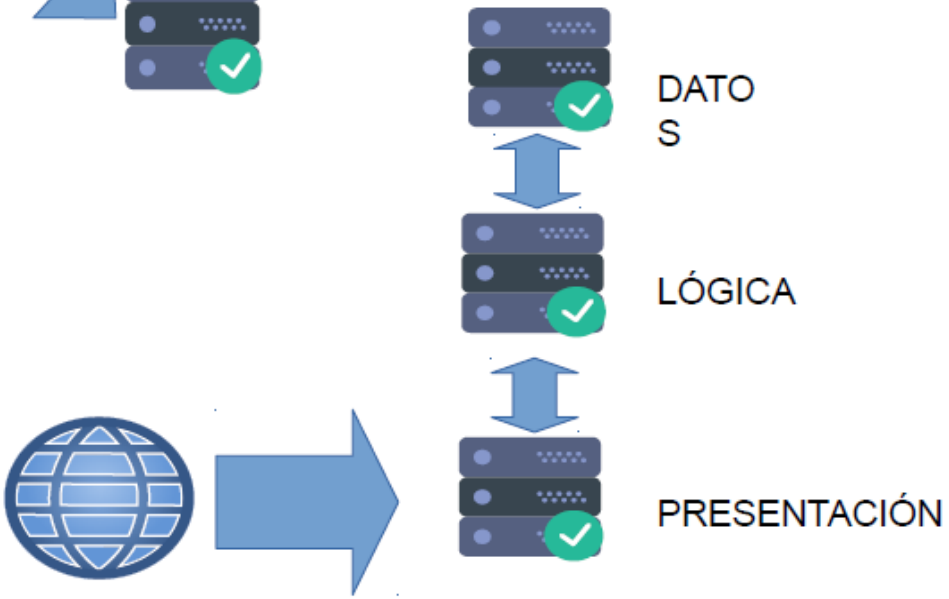
- Llega un punto que no podemos ampliar más la máquina para acomodar cargas mayores. Hay limitaciones físicas que nos impiden ampliar continuamente el servidor.

Soluciones para la escalabilidad del sistema

- Escalabilidad horizontal
  - Creamos múltiples réplicas de la máquina y se distribuye la carga entre ellas. El balanceador hardware (o software) reparte la carga entre los clones

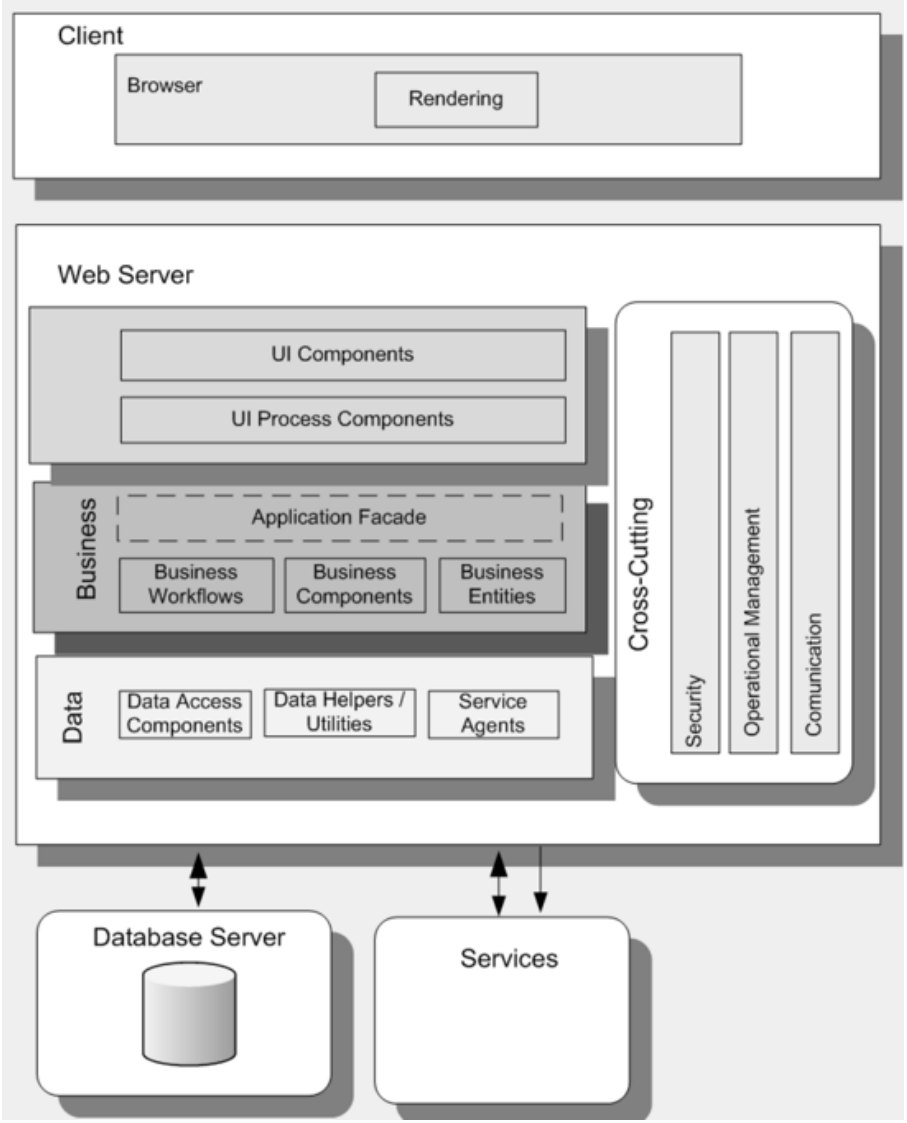


- Escalabilidad vertical
  - En este caso "partimos" el sistema entre varias máquinas. Cada una de ellas será responsable de una parte del mismo, de forma que la carga se distribuye



Escalabilidad vertical: arquitectura software de división en capas

- **Capa de presentación**
  - Subcapa componentes de presentación
  - Subcapa de lógica de presentación
- **Capa de lógica de negocio**
  - Fachada (*facade*) de servicios
  - Componentes, lógica de negocio
- **Capa de acceso a datos o persistencia**
- **Capa de infraestructura (transversal)**
  - Seguridad
  - Gestión de servicios, etc.
- Helpers
  - Clases de transferencia entre capas



Resumen

Para poder abordar la creación de aplicaciones web es necesario conocer antes el contexto en el que se desarrollan estas aplicaciones. Así, podemos ver que el modelo arquitectónico en el que se basan es el denominado “cliente/servidor”, donde un cliente, a través de un navegador, realiza peticiones a un servidor, que será el encargado de gestionar su petición y devolver una respuesta adecuada. Las aplicaciones desarrolladas en este contexto pueden ser muy sencillas (aplicaciones web estáticas) o muy complejas (aplicaciones web dinámicas e interactivas). La selección de una u otra tecnología, o lenguaje, dependerá principalmente del modelo de programación escogido, es decir, del reparto de funcionalidades entre el cliente y el servidor.

Los lenguajes utilizados en el entorno del servidor se agrupan en lenguajes de *scripting* (interpretados como PHP, AS P, Perl o Python), lenguajes para aplicaciones ejecutadas de forma independiente del servidor web (CGI y derivados) o lenguajes que optan por una programación “híbrida” (ASP.Net). Finalmente, para el desarrollo óptimo de aplicaciones web se suelen utilizar herramientas específicas que ayudan al desarrollador en su labor.

PREGUNTAS