T1: Ejercicios

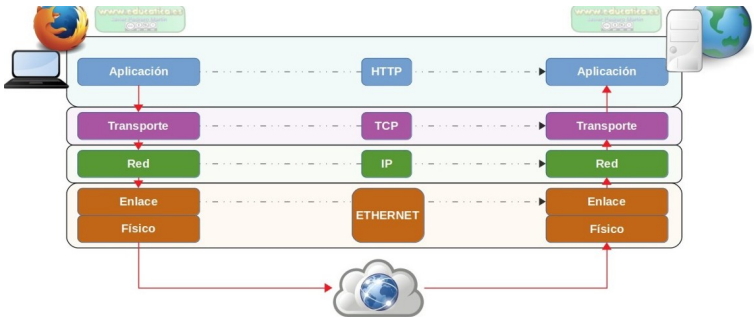
Jesús Temprano Gallego

DAW2

Contenido

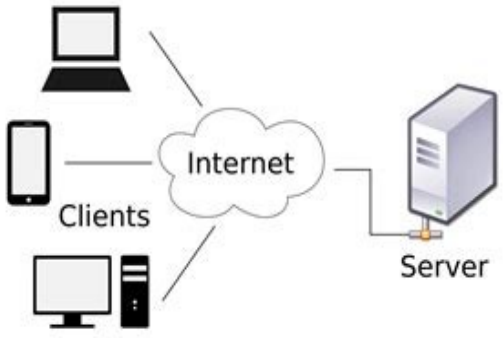
1. Protocolos de comunicaciones: IP, TCP, HTTP, HTTPS.

- **IP**:
  - IP es el protocolo que permite que los equipos se comuniquen en la red mediante direcciones únicas.
  - Cada equipo tiene una dirección única que permite enviar y recibir información correctamente. Cuando envías un mensaje o archivo, IP divide esos datos en paquetes y los envía al equipo correcto usando su dirección. Así, todos los dispositivos conectados pueden comunicarse de forma ordenada y segura.
  - https://es.wikipedia.org/wiki/Protocolo_de_internet
- **TCP**:
  - TCP asegura que los datos enviados entre equipos lleguen completos y en el orden correcto.
  - Trabaja junto con IP para enviar información entre equipos de manera fiable. Divide los datos en paquetes, los envía y comprueba que lleguen al destino sin errores y en el orden correcto. Si algún paquete se pierde o llega mal, TCP lo vuelve a enviar. Esto garantiza que la comunicación sea segura y que los archivos, mensajes o páginas web se reciban completos.
  - https://es.wikipedia.org/wiki/Protocolo_de_control_de_transmisión
- **HTTP**:
  - HTTP permite enviar y recibir datos entre clientes y servidores, principalmente para páginas web.
  - Es el que usan los navegadores para pedir páginas web a los servidores y mostrar su contenido. Funciona enviando solicitudes desde el navegador y recibiendo respuestas del servidor. HTTP no cifra la información, por lo que los datos pueden ser vistos o modificados mientras viajan por la red. Usa el puerto 80 y se apoya en los protocolos TCP e IP para garantizar la transmisión de los datos entre cliente y servidor. .
  - https://es.wikipedia.org/wiki/Protocolo_de_transferencia_de_hipertexto
- **HTTPS**:
  - HTTPS es la versión segura de HTTP que cifra los datos entre cliente y servidor.
  - Funciona igual que HTTP, pero añade cifrado para proteger la información durante la transmisión. Esto asegura que los datos enviados o recibidos, como contraseñas, información personal o datos de aplicaciones, no puedan ser interceptados ni modificados por terceros. Usa el puerto 443 y, al igual que HTTP, se apoya en los protocolos TCP e IP, pero añade una capa de seguridad (TLS/SSL) que cifra la comunicación. Es el más importante porque garantiza la seguridad y privacidad en Internet, siendo esencial en banca online, redes sociales, compras y cualquier servicio que maneje datos sensibles.
  - https://es.wikipedia.org/wiki/Protocolo_seguro_de_transferencia_de_hipertexto



2. Modelo de comunicaciones cliente – servidor y su relación con las aplicaciones web.

- En el modelo cliente-servidor, el cliente solicita recursos y el servidor los entrega.
- Es una forma de organizar la comunicación en redes donde un equipo (cliente) solicita servicios o información y otro equipo (servidor) los proporciona. En aplicaciones web, el navegador o la app actúa como cliente, enviando solicitudes al servidor que aloja la página web o la aplicación. El servidor procesa la solicitud, accede a datos si es necesario y devuelve la respuesta al cliente.
- https://es.wikipedia.org/wiki/Cliente-servidor

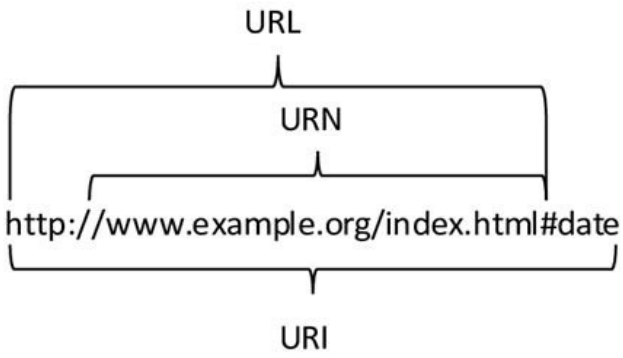


3. Estudio sobre los métodos de petición HTTP / HTTPS más utilizados.

- Los métodos HTTP/HTTPS más usados son GET, POST, PUT y DELETE
- En HTTP/HTTPS, los métodos de petición indican qué acción quiere realizar el cliente sobre un recurso del servidor:
  - **GET**: solicita información o datos al servidor sin modificarlos. Es el método más usado y el principal para la mayoría de webs.
  - **POST**: envía datos al servidor para crear o procesar información.
  - **PUT**: crea, actualiza o reemplaza un recurso existente con nuevos datos.
  - **DELETE**: elimina un recurso en el servidor.
- https://developer.mozilla.org/es/docs/Web/HTTP/Reference/Methods

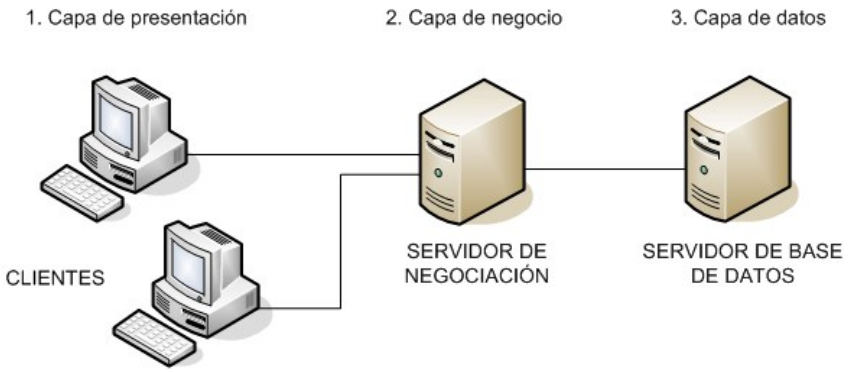
4. Estudio sobre el concepto de URI (Identificador de Recursos Uniforme)/URL/URN, estructura, utilidad y relación con el protocolo HTTP/HTTPS.

- **URI** (Identificador de recursos uniforme)
  - URI es un identificador que señala cualquier recurso en la red.
  - Es un conjunto de caracteres que permite identificar un recurso en Internet de manera única, sin indicar necesariamente dónde se encuentra. Sirve como base para URL y URN y facilita la comunicación entre clientes y servidores.
  - https://es.wikipedia.org/wiki/Identificador_de_recursos_uniforme
- **URL** (Localizador de recursos uniforme)
  - URL indica la ubicación exacta de un recurso y cómo acceder a él.
  - Es un tipo de URI que especifica dónde se encuentra un recurso en la red y qué protocolo usar para acceder a él. Su estructura incluye protocolo, servidor, puerto (*opcional*), ruta, parámetros y fragmento.
  - https://es.wikipedia.org/wiki/Localizador_de_recursos_uniforme
- **URN** (Nombre de Recurso Uniforme)
  - URN identifica un recurso por su nombre dentro de un espacio de nombres, sin indicar su ubicación.
  - Es un tipo de URI que identifica un recurso únicamente por su nombre dentro de un espacio de nombres definido, sin especificar dónde se encuentra. Son útiles para referirse a recursos incluso si cambian de ubicación.
  - https://en.wikipedia.org/wiki/Uniform_Resource_Name



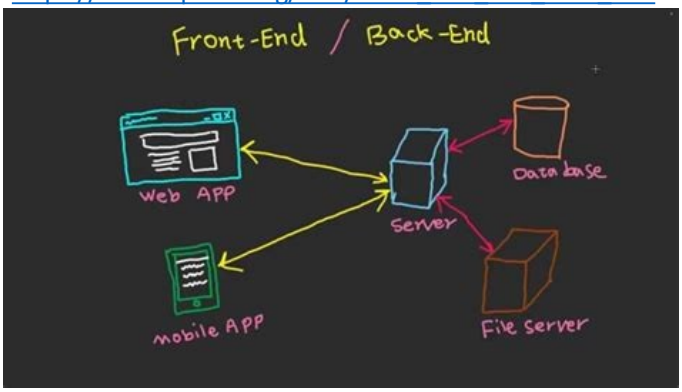
5. Modelo de desarrollo de aplicaciones multicapa – comunicación entre capas – componentes – funcionalidad de cada capa.

- **Modelo multicapa: divide las aplicaciones en capas que interactúan entre sí, cada una con funciones específicas.**
  Las mas comunes son:
  - **Capa de presentación**: interactúa con el usuario, mostrando información y recibiendo entradas.
  - **Capa de negocio**: procesa las peticiones del usuario, ejecuta la lógica de la aplicación y es la encargada de gestionar cómo se manejan los datos entre la capa de presentación y la de datos.
  - **Capa de datos**: almacena y gestiona la información, normalmente conectándose a bases de datos.
- **Comunicación entre capas**: Cada capa se comunica únicamente con la capa inmediatamente superior o inferior, usando interfaces o servicios que permiten intercambiar información de manera controlada.
- https://es.wikipedia.org/wiki/Arquitectura_multicapa



6. Modelo de división funcional front-end / back-end para aplicaciones web.

- Este modelo divide la aplicación web en front-end (interfaz de usuario) y back-end (lógica y datos).
  - **Front-end:** es la parte que interactúa directamente con el usuario. Incluye la interfaz visual, formularios, botones y otros elementos interactivos, desarrollados principalmente con HTML, CSS y JavaScript. Su función principal es mostrar la información y recoger las acciones del usuario.
  - **Back-end:** es la parte que gestiona la lógica de negocio y el acceso a los datos. Procesa las solicitudes del front-end, aplica reglas de la aplicación y se comunica con la base de datos. Se desarrollan con lenguajes y frameworks como Node.js, PHP, Python, Java o .NET.
- https://en.wikipedia.org/wiki/Front_end_and_back_end



7. Página web estática – página web dinámica – aplicación web – mashup.

- **Página web estática**: muestra contenido fijo que no cambia según el usuario. Se construye con HTML y CSS, y cada visitante ve la misma información.
  - https://es.wikipedia.org/wiki/Página_web_estática
- **Página web dinámica**: genera contenido diferente según la interacción del usuario o datos de un servidor. Utiliza lenguajes como PHP, JavaScript o frameworks para actualizar la información en tiempo real. El contenido puede variar según la interacción del usuario o la información obtenida del servidor, como resultados de búsqueda, formularios o noticias actualizadas. Esto permite personalizar la experiencia de cada visitante.
  - https://es.wikipedia.org/wiki/Página_web_dinámica
- **Aplicación web**: es un software completo accesible desde un navegador que permite al usuario interactuar, realizar procesos y gestionar información, como Google Docs o plataformas de banca online.
  - https://es.wikipedia.org/wiki/Aplicación_web
- **Mashup**: Es una web que integra datos, contenidos o servicios de diferentes aplicaciones en una sola. Por ejemplo, un mapa que muestra restaurantes y tráfico en tiempo real combinando varias APIs. Permite acceder a información de manera fácil y ofrecer nuevas funcionalidades al usuario sin crear los datos desde cero.
  - https://es.wikipedia.org/wiki/Mashup_(aplicación_web_híbrida)

8. Componentes de una aplicación web.

- **Cliente**:
  - Muestra la interfaz al usuario.
  - Recoger entradas (formularios, botones, etc.).
  - Se comunica con el servidor mediante peticiones HTTP/HTTPS.
- **Servidor**:
  - Procesa la lógica de la aplicación.
  - Valida los datos enviados por el cliente.
  - Accede a la base de datos.
  - Devuelve las respuestas al cliente.
- **Base de datos**:
  - Hace consultas, inserciones, actualizaciones y eliminaciones.
  - Almacena información (usuarios, artículos, pedidos, etc.).
- **Servidor web (Apache,** Nginx, ...**)**:
  - Recibe solicitudes del navegador.
  - Las redirige al back-end.
  - Sirve archivos estáticos

9. Programas ejecutados en el lado del cliente y programas ejecutados en el lado del servidor - lenguajes de programación utilizados en cada caso.

-

10. Lenguajes de programación utilizados en el lado servidor de una aplicación web (características y grado de implantación actual).
11. Características y posibilidades de desarrollo de una plataforma XAMPP.
12. En que casos es necesaria la instalación de la máquina virtual Java (JVM) y el software JDK en el entorno de desarrollo y en el entrono de explotación.
13. IDE más utilizados (características y grado de implantación actual).
14. Servidores HTTP /HTTPS más utilizados (características y grado de implantación actual).
15. Apache HTTP vs Apache Tomcat
16. Navegadores HTTP /HTTPS más utilizados (características y grado de implantación actual).
17. Generadores de documentación HTML (PHPDoc): PHPDocumentor, ApiGen, …
18. Repositorios de software – sistemas de control de versiones: GIT , CVS, Subversion, ...
19. Propuesta de configuración del entorno de desarrollo para la asignatura de Desarrollo web del lado servidor en este curso (incluyendo las versiones): xxx-USED y xxx-WXED.
20. Propuesta de configuración del entorno de explotación para la asignatura de Desarrollo web del lado servidor en este curso (incluyendo las versiones): xxx-USEE
21. Realizar un estudio sobre los siguientes conceptos y su relación con el desarrollo de aplicaciones web:
22. CMS – Sistema de gestión de contenidos:
23. ERP – Sistema de planificación de los recursos empresariales: