TEMA 1 – DESARROLLO WEB EN ENTORNO SERVIDOR

CONTENIDO

REALIZA UN ESTUDIO SOBRE LOS SIGUIENTES CONCEPTOS.

1. PROTOCOLOS DE COMUNICACIONES

Los **protocolos de comunicaciones** son los encargados de determinar cómo deben circular los mensajes dentro de una red. Son conjuntos de normas que permiten la comunicación entre ordenadores, estableciendo la forma de identificación de estos en la red, la forma de transmisión de los datos y la forma en que la información debe procesarse.

IP, *INTERNET PROTOCOL* O PROTOCOLO DE INTERNET

Este es un protocolo de la capa de red, cuya función es el **envío de paquetes de datos** tanto a nivel local como a través de las redes.

Es un **protocolo no orientado a conexió**n, lo que significa que el envío de los datos tratará de realizarse del mejor modo posible, pero sin garantías de que vaya a alcanzarse el destino final, siendo un protocolo de comunicación no fiable.

Para realizar el envío de paquetes de datos, este protocolo asigna una **dirección IP** a las máquinas de origen y destino. Esta dirección IP, será el identificador del dispositivo en el proceso de comunicación.

TCP, *TRANSMISSION CONTROL PROTOCOL* O PROTOCOLO DE CONTROL DE TRANSMISIÓN

Este es un **protocolo orientado a conexión**, que permite un transporte fiable y bidireccional de los datos. En la pila de protocolos TCP/IP, este actúa como capa de transporte de datos entre el protocolo de red y la aplicación.

Mediante su uso, las aplicaciones pueden comunicarse de forma segura, puesto que el protocolo TCP se encarga de que los datos que emite el cliente sean recibidos por el servidor sin errores y en el mismo orden en que fueron emitidos, a pesar de trabajar con los servicios de la capa IP, la cual no es confiable.

HTTP, *HYPERTEXT TRANSFER PROTOCOL* O PROTOCOLO DE TRANSFERENCIA DE HIPERTEXTO

Este es un protocolo de la capa de aplicación, que permite las **transferencias de información en la web**, definiendo la sintaxis y la semántica que utilizan los elementos de software de la arquitectura web para comunicarse.

Opera siguiendo el esquema petición-respuesta entre un cliente y un servidor, y es un **protocolo sin estado**, es decir, no guarda ninguna información sobre conexiones anteriores.

HTTPS, *HYPERTEXT TRANSFER PROTOCOL SECURE* O PROTOCOLO DE TRANSFERENCIA DE HIPERTEXTO SEGURO

Este es un protocolo de aplicación **basado en el protocolo HTTP**, destinado a la transferencia segura de datos de hipertexto, es decir, es la versión segura de HTTP.

El sistema HTTPS utiliza un cifrado basado en la seguridad de textos SSL/TLS para crear un canal cifrado más apropiado para el tráfico de información sensible que el protocolo HTTP.

2. MODELO DE COMUNICACIONES CLIENTE – SERVIDOR

La **arquitectura cliente-servidor** es un modelo de diseño de software con dos partes claramente diferenciadas, el cliente y el servidor, que se reparten las tareas. En este modelo de comunicaciones, un cliente realiza una petición al servidor, que la procesa y envía una respuesta.

Este tipo de arquitectura nos permite conectar a varios clientes a los servicios que provee un servidor, lo cual es imprescindible para la mayoría de aplicaciones y servicios web hoy en día.

¿CUÁL ES SU RELACIÓN CON LAS APLICACIONES WEB?

Cualquier aplicación web que queramos desarrollar va a necesitar utilizar un servidor que haga uso del protocolo http, es decir, todas las aplicaciones web que hagamos, utilizarán una arquitectura cliente-servidor.

3. MÉTODOS DE PETICIÓN HTTP/HTTPS

Los métodos de petición o *HTTP verbs*, nos permiten **indicar la acción que se desea realizar sobre un recurso** determinado. HTTP define una serie predefinida de métodos que pueden utilizarse, los más usados son los siguientes.

GET

El método GET solicita la representación de un recurso específico. Las peticiones que usan el método GET sólo deben recuperar datos.

HEAD

El método HEAD pide una respuesta idéntica a la de una petición GET, pero sin el cuerpo de la respuesta. Es útil para recuperar los metadatos de los encabezados de respuesta, sin tener que transportar todo el contenido.

POST

El método POST se encarga de enviar datos para que sean procesados por un recurso en específico. Semánticamente, está orientado a la creación de nuevos recursos.

PUT

El modo PUT también envía datos al servidor, pero en lugar de especificar el recurso que los procesará, se encarga de identificar los propios datos. Semánticamente, está más orientado a la actualización de los contenidos.

DELETE

El método DELETE borra un recurso en específico.

CONNECT

El método CONNECT se utiliza para saber si se tiene acceso al servidor especificado.

OPTIONS

El método OPTIONS devuelve los métodos HTTP que soporta un servidor específico.

TRACE

El método TRACE solicita al servidor que introduzca en la respuesta todos los datos que reciba en el mensaje de petición. Se utiliza con fines de depuración y diagnóstico ya que el cliente puede ver lo que llega al servidor y de esta forma ver todo lo que añaden al mensaje los servidores intermedios.

PATCH

El método PATCH se utiliza para actualizar de manera parcial uno o varios recursos, sobrescribiéndolos.

Ejemplos: https://yosoy.dev/peticiones-http-get-post-put-delete-etc/

Añadir construcción de url.

4. MODELO DE DESARROLLO DE APLICACIONES MULTICAPA

La **arquitectura multicapa** es un modelo de desarrollo de software que estructura el código de las aplicaciones en distintas capas o niveles.

El objetivo de dividir en capas el diseño de una aplicación es puedan separarse las diferentes funciones lógicas de la misma, de forma que podamos distribuir las tareas de creación de la aplicación por niveles, formando grupos de trabajo independientes.

En una aplicación que utiliza este tipo de arquitectura, podremos distinguir, de forma general, tres capas o niveles:

La **capa de presentación**, que se encarga de dar formato a los datos para presentárselos al usuario final. Es la interfaz gráfica de la aplicación. Esta capa se comunica únicamente con la capa de negocio.

La **capa de negocio**, que es donde se reciben las peticiones del usuario, se ejecutan los programas y se envían las respuestas tras el proceso. Aquí es donde se establecen todas las reglas que deben cumplirse. Esta capa, se comunica con la capa de presentación para recibir las peticiones, y con la capa de acceso a datos para solicitar al gestor de la base de datos que almacene o recupere información.

La **capa de acceso a datos**, que es la encargada de almacenar la información de la aplicación y recuperarla cuando sea necesario. Esta capa está formada por uno o más gestores de bases de datos que reciben peticiones desde la capa de negocio.

5. MODELO DE DIVISIÓN FUNCIONAL FRONT-END Y BACK-END

A la hora de desarrollar una aplicación web, necesitaremos diferenciar entre el front-end y el back-end.

FRONT END

El front-end es la parte de la aplicación que pensamos y programamos para la interacción con los usuarios normales. Es la interfaz de nuestra aplicación, que contendrá toda la información

gráfica: estilos, colores, fondos, tamaños, animaciones… y deberá ser lo más atractiva posible, puesto que está orientada a la interacción con los usuarios.

Además, será la parte responsable de recolectar los datos de entrada del usuario y transformarlos ajustándolos a las especificaciones que demanda el back-end para poder procesarlos.

BACK END

El **back-end** es la parte de la aplicación que programamos para que sea utilizada por los usuarios especiales, es decir, aquellos que tendrán permisos para administrar web, o añadir contenido. Es lo que conocemos como el lado del servidor.

Cuando hablamos de back-end, estamos hablando de la parte del desarrollo de nuestra web que se encargará de que todo funcione, es decir, será el encargado de llevar a cabo todas las funciones que simplifiquen el proceso de desarrollo, de las conexiones con las bases de datos, de la gestión de las librerías, de la optimización de los recursos y de la seguridad de nuestra aplicación.

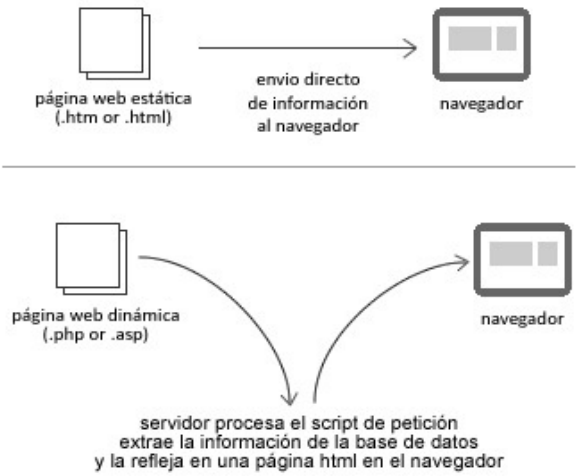
Tanto el front-end como el back-end **son esenciales para la construcción de una aplicación web**. Abarcan distintos aspectos del desarrollo de la misma, pero deben trabajar juntos, de forma coordinada, para que esta se entienda y funcione correctamente.

6. PÁGINAS WEB ESTÁTICAS Y DINÁMICAS, APLICACIONES WEB Y MASHUP

PÁGINA WEB

Una **página web** es un documento electrónico, capaz de contener texto, imágenes, sonido, vídeo, enlaces, programas…. al que se puede acceder a través de un navegador. Generalmente, se encuentran en formato HTML o XHTML y pueden estar almacenadas en un ordenador, o en un servidor web. Además, cuando hablamos de páginas web, podemos diferenciar entre páginas web estáticas y páginas web dinámicas.

Una **página web estática** es aquella enfocada principalmente a mostrar una información permanente. Son funcionalmente básicas, con poca programación, principalmente en lenguajes HTML, CSS o JavaScript.



Una **página web dinámica** ofrece una mayor interactividad con los usuarios que la visitan y permite la creación de aplicaciones dentro de la página web. Su creación es más compleja ya que utiliza lenguajes de programación y programas de gestión de bases de datos.

Dentro de las páginas web dinámicas, podemos distinguir **dos tipos**: aquellas que están escritas en HTML, pero contienen código que debe ser ejecutado por el navegador

(normalmente en JavaScript); y otras que, en lugar de almacenar en el servidor las páginas que se van a mostrar en el navegador, son generadas por la ejecución de un programa.

Características: página web estática vs página web dinámica

Realizadas en HTML o XHTML	Utilizan varias técnicas de programación
Proceso de actualización lento y manual	Proceso de actualización sencillo
Ausencia de movimiento y funcionalidades	Múltiples funcionalidades: bases de datos, foros…
Es necesario acceder al servidor para realizar cambios	Pueden concedérsele derechos de administrador a los usuarios, para que puedan alterar el diseño o los contenidos

APLICACIÓN WEB

Una **aplicación web** es una página web dinámica que ejecutamos en un servidor web y mostramos en un navegador. Las aplicaciones web contienen elementos que permiten que el usuario acceda a los datos de modo interactivo, gracias a que la página responderá a cada una de sus acciones.

Las aplicaciones web tienen grandes ventajas, como por ejemplo que no necesitan ningún tipo de instalación, que son multiplataforma y multidispositivo, que no necesitamos tener un ordenador muy potente para utilizarlas, puesto que la mayor parte del peso de estas lo soporta el servidor en el que están alojadas, y que son adaptables, intuitivas y muy fáciles de actualizar.

MASHUP

En desarrollo web, una **mashup** es una forma de integración y reutilización. Los mashup son aplicaciones web que utilizan los servicios que ofrecen otras aplicaciones web, con el fin de reutilizar su contenido o funcionalidad.

Un ejemplo de mashup sería, por ejemplo, la página web de un hotel, que utilice el servicio de Google Maps para integrar un mapa con la ubicación del mismo.

7. COMPONENTES DE UNA APLICACIÓN WEB.

Para poder

Servidor web

Módulo encargado de ejecutar el código

Aplicación de base de datos

Lenguaje de programación

8. PROGRAMAS EJECUTADOS EN EL LADO DEL CLIENTE Y PROGRAMAS EJECUTADOS EN EL LADO DEL SERVIDOR – LENGUAJES DE PROGRAMACIÓN EN CADA CASO.

Por qué separamos lado del cliente de lado del servidor.

9. LENGUAJES DE PROGRAMACIÓN UTILIZADOS EN EL LADO SERVIDOR DE UNA APLICACIÓN WEB (CARACTERÍSTICAS Y GRADO DE IMPLANTACIÓN ACTUAL).

Profundizamos en las tecnologías que se utilizan del lado del servidor: Spring, XAMPP, .NET, MEAN.

10. CARACTERÍSTICAS Y POSIBILIDADES DE DESARROLLO DE UNA PLATAFORMA XAMPP.

Es la forma que tengo yo de instalar rápidamente un servidor web en mi ordenador, con todos sus componentes.

11. EN QUÉ CASOS ES NECESARIA LA INSTALACIÓN DE LA MÁQUINA VIRTUAL JAVA (JVM) Y EL SOFTWARE JDK EN EL ENTORNO DE DESARROLLO Y EN EL ENTORNO DE EXPLOTACIÓN.

12. IDE MÁS UTILIZADOS (CARACTERÍSTICAS Y GRADO DE IMPLANTACIÓN ACTUAL).

NetBeans, Eclipse, VisualStudio, PHPStrom.

13. SERVIDORES HTTP/HTTPS MÁS UTILIZADOS (CARACTERÍSTICAS Y GRADO DE IMPLANTACIÓN ACTUAL).

Apache y Apache Tomcat. Tabla de servidores web y de quien es cada uno de ellos. Mención a servidores web portables, para llevar el un pincho

14. APACHE HTTP VS APACHE TOMCAT

Apache orientado a aplicaciones web

Tomcat orientado a java

15. NAVEGADORES HTTP /HTTPS MÁS UTILIZADOS (CARACTERÍSTICAS Y GRADO DE IMPLANTACIÓN ACTUAL).

Hay que saber jugar con todo lo que nos ofrecen los distintos navegadores.

16. GENERADORES DE DOCUMENTACIÓN HTML (PHPDOC): PHPDOCUMENTOR, APIGEN, …

17. REPOSITORIOS DE SOFTWARE – SISTEMAS DE CONTROL DE VERSIONES: GIT, CVS, SUBVERSION, ...

Git, GitHub, GitLab

18. 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-W10ED.

Sistema Operativo: Windows 10.

- **Servidor administración remota: SSH.**
- **Servidor de transferencia de ficheros: SFTP (SSH) Puerto 22.**
- **Repositorio: GitHub.**
- **Servidor web: Apache 2.4 HTTP**
- **Gestor de Bases de Datos: MySQL 8.0.**
- **Navegador: Mozilla Firefox 81.0.**
- **IDE: NetBeans 12.0.**
- **Ofimática, multimedia, generador html: LibreOffice 6.4.**
- **Frameworks PHP: Pendiente de actualizar.**
- **Cliente SSH: Filezilla 3.50.0.**

19. PROPUESTA DE CONFIGURACIÓN DEL ENTORNO DE EXPLOTACIÓN PARA LA ASIGNATURA DE DESARROLLO WEB DEL LADO SERVIDOR EN ESTE CURSO (INCLUYENDO LAS VERSIONES): XXX-USEE

Sistema Operativo: Ubuntu Server 20.04.

- **Servidor administración remota: SSH.**
- **Servidor de transferencia de ficheros: SFTP (SSH) Puerto 22.**
- **Repositorio: GitHub.**
- **Servidor Web: Apache 2.4 HTTP**
- **Gestor de Bases de Datos: MySQL 8.0.**
- **Cliente SSH: NetBeans 12.0 / Filezilla client 3.50.0.**
- **Navegador: Mozilla Firefox 81.0.**

21.

Año, versión, desarrolladores, características, componentes, funcionalidad

BIBLIOGRAFÍA

PROTOCOLOS DE COMUNICACIONES:

https://es.wikipedia.org/wiki/Protocolo_de_comunicaciones

https://definicion.de/protocolo-de-comunicacion/

https://es.wikipedia.org/wiki/Protocolo_de_internet

https://es.wikipedia.org/wiki/Protocolo_de_control_de_transmisi%C3%B3n

https://es.wikipedia.org/wiki/Protocolo_de_transferencia_de_hipertexto

https://es.wikipedia.org/wiki/Protocolo_seguro_de_transferencia_de_hipertexto

MODELO DE COMUNICACIONES CLIENTE-SERVIDOR

https://es.wikipedia.org/wiki/Cliente-servidor

p.8-9 Teoría **Desarrollo Web Entorno Servidor**, por *José Luis Comesaña*.

MÉTODOS DE PETICIÓN

https://es.wikipedia.org/wiki/Protocolo_de_transferencia_de_hipertexto

https://developer.mozilla.org/es/docs/Web/HTTP/Methods

https://diego.com.es/metodos-http

MODELO DE DESARROLLO DE APLICACIONES MULTICAPA

p.8-9 Teoría **Desarrollo Web Entorno Servidor**, por *José Luis Comesaña*.

https://es.wikipedia.org/wiki/Programaci%C3%B3n_por_capas

MODELO DE DIVISIÓN FUNCIONAL FRONT-END Y BACK-END

https://es.wikipedia.org/wiki/Front_end_y_back_end

https://nestrategia.com/desarrollo-web-back-end-front-end/

https://descubrecomunicacion.com/que-es-backend-y-frontend/

PÁGINAS WEB ESTÁTICAS Y DINÁMICAS, APLICACIONES WEB Y MASHUP

https://es.wikipedia.org/wiki/P%C3%A1gina_web

p.3 Teoría **Desarrollo Web Entorno Servidor**, por *José Luis Comesaña*.

https://brandmedia.es/diferencias-pagina-web-estatica-dinamica-mejor/

https://www.antevenio.com/blog/2017/11/web-estatica-vs-dinamica-que-es-mejor-para-seo/

p.5 Teoría **Desarrollo Web Entorno Servidor**, por *José Luis Comesaña*.

https://www.neosoft.es/blog/que-es-una-aplicacion-web/

https://es.wikipedia.org/wiki/Mashup_(aplicaci%C3%B3n_web_h%C3%ADbrida)

https://sg.com.mx/content/view/256

REALIZADO EN COLABORACIÓN CON SONIA ANTÓN LLANES