T1: Ejercicios

Jesús Temprano Gallego

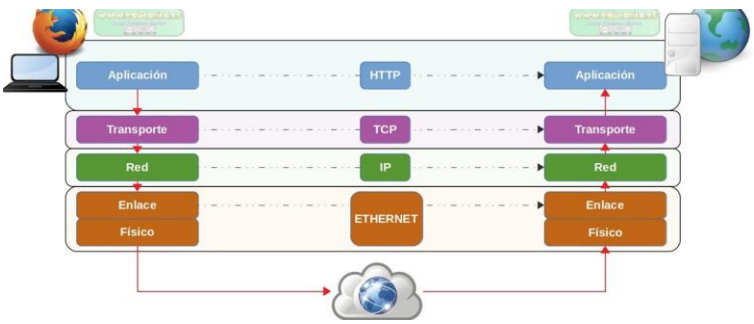
DAW2

Ultima Revisión: 22/10/2025

Contenido

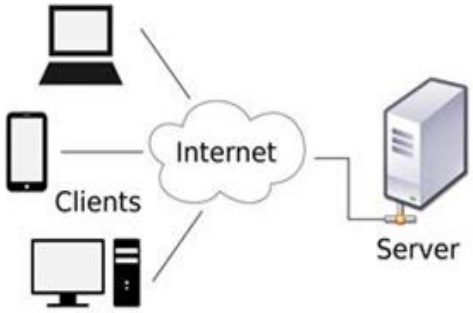
1. Protocolos de comunicaciones: IP, TCP, HTTP, HTTPS.

- **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 la base del funcionamiento de las aplicaciones web.
- 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**: Es el más usado. Se utiliza para solicitar información o datos sin modificarlos. Se usa cuando pones una dirección en el navegador, la pagina carga una imagen, …
  - **POST**: envía datos al servidor para crear o procesar información. Como cuando rellenas un formulario, escribes un comentario.
  - **PUT**: actualiza o reemplaza un recurso existente con nuevos datos. Como cuando subes una imagen a una red social, cambias la información de tu perfil en una app.
  - **DELETE**: elimina un recurso del servidor.
  - **HEAD**: Solicita información de un recurso sin descargar el contenido. Se usa para comprobar si un archivo existe, su tamaño, tipo o fecha de modificación, sin transferir toda la información.
- https://developer.mozilla.org/es/docs/Web/HTTP/Reference/Methods

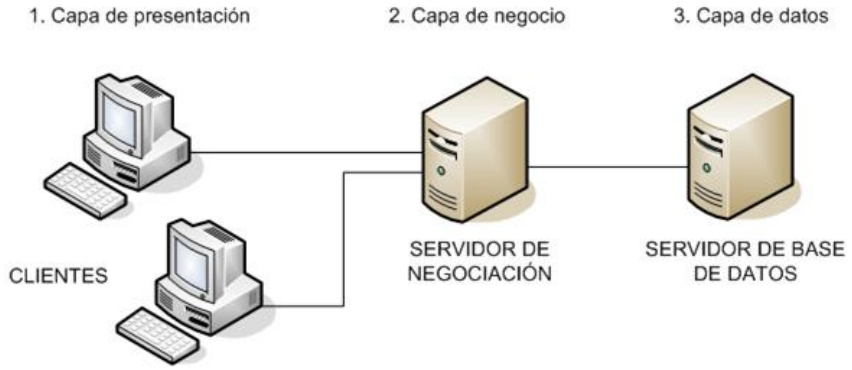
4. Estudio sobre el concepto de URI (Identificador de Recursos Uniforme)/URL/URN, estructura, utilidad y relación con el protocolo HTTP/HTTPS.

- **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 como y donde esta un recurso para 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/dominio**, **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, pero sin indicar su ubicación. *Son útiles para referirse a recursos incluso si cambian de ubicación.*
  - https://en.wikipedia.org/wiki/Uniform_Resource_Name

En qué se diferencia la URL, URI y URN

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 más 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.

- El modelo front-end/back-end separa lo que ve el usuario de lo que gestiona el administrador.
  - **Front-end** *(lo que ve el usuario)*: incluye todo lo que el usuario final utiliza directamente, como formularios, menús, visualización de información. Permite consultar, navegar y realizar acciones sin acceder a la gestión interna.
  - **Back-end** *(lo que ve el administrador)*: es la parte que gestiona y controla la aplicación, accesible solo para administradores o personal autorizado. Permite gestionar usuarios, contenido, configuraciones, bases de datos y procesos internos que no se muestran al usuario final.

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.
- **Página web dinámica**: Es un documento que cambia a cada vez que entras sin que el programador lo tenga que modificar. Su contenido se genera en el servidor cada vez que alguien la visita. Por ejemplo, la página de noticias de un periódico online, donde el contenido se actualiza constantemente.
- **Aplicación Web**: es una página web dinámica, pero con control de acceso donde cada usuario puede ver cosas diferentes.
- **Mashup**: Es una aplicación web que combina información de varias apps para crear algo nuevo. Por ejemplo, un mapa que muestra restaurantes y tráfico en tiempo real combinando varias APIs.

8. Componentes de una aplicación web.

- **Servidor Web**: Es el intermediario entre las solicitudes del cliente (navegador) y la aplicación. Procesa las peticiones HTTP entrantes y envía las respuestas adecuadas.
- **Base de datos**: Almacena y organiza la información de la aplicación web.
- **Servidor**: Procesa la lógica de la aplicación, valida los datos enviados por el cliente, accede a la base de datos y le devuelve las respuestas al cliente.
- **Cliente**: Muestra la interfaz al usuario, recoge las entradas y se comunica con el servidor mediante las peticiones HTTP/S.

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

- **Lado del cliente:** Son programas que se ejecutan en el navegador del usuario y se encargan de la interacción, la presentación de datos y la experiencia visual. Los lenguajes más usados son **JavaScript, HTML, CSS** y frameworks como React, Angular o Vue.js. Permiten validar formularios, animar interfaces y mostrar información dinámica sin necesidad de recargar la página.
- **Lado del servidor:** Son programas que se ejecutan en el servidor y se encargan de la lógica de negocio, la gestión de datos y el procesamiento de solicitudes del cliente. Los lenguajes más utilizados incluyen **PHP, Python, Java, C#, Node.js** y frameworks como Django, Spring o Express. Gestionan bases de datos, autenticación, cálculos y envían las respuestas adecuadas al cliente.

10. Lenguajes de programación utilizados en el lado servidor de una aplicación web (características y grado de implantación actual).

- **PHP:** Fácil de aprender, ampliamente usado en desarrollo web y soporte para la mayoría de servidores. Muy implantado en CMS como WordPress y PrestaShop, lo que lo convierte en el lenguaje de servidor más implementado a nivel mundial.
- **Java**: Robusto y orientado a objetos, usado en aplicaciones grandes y sistemas empresariales; frameworks como Spring permiten construir aplicaciones escalables. Mantiene un alto grado de implantación en grandes corporaciones y entornos críticos.
- **C# (ASP.NET):** Muy utilizado en entornos Microsoft; permite crear aplicaciones web con gran integración en servidores Windows. Su implementación es fuerte en empresas que usan tecnologías Microsoft y soluciones corporativas.
- **Node.js (JavaScript):** Permite usar JavaScript en el servidor, ideal para aplicaciones en tiempo real; su uso ha crecido enormemente por su rendimiento y escalabilidad, con una implantación creciente en startups y desarrollos modernos de aplicaciones web.

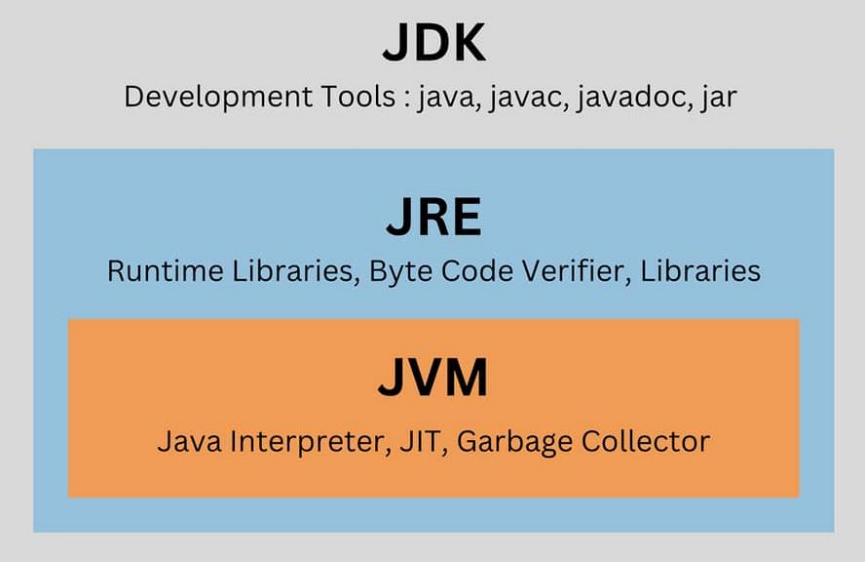
https://w3techs.com/technologies/overview/programming_language

11. Características y posibilidades de desarrollo de una plataforma XAMPP.

- XAMPP es una plataforma que instala Apache, MySQL, PHP y Perl para desarrollar y probar aplicaciones web localmente. Es multiplataforma y permite un desarrollo sencillo para desarrollo y testing rápido.
- https://www.godaddy.com/resources/es/crearweb/xampp-que-es

12. 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.

- **Entorno de desarrollo**: Es necesario instalar el JDK (Java Development Kit) porque incluye compilador, bibliotecas y herramientas necesarias para crear, compilar y probar aplicaciones Java. Sin el JDK, no se puede desarrollar ni generar archivos ejecutables (.class o .jar) desde el código fuente. La JVM (Java Virtual Machine) también se instala, ya que permite ejecutar las pruebas de las aplicaciones durante el desarrollo.
- **Entorno de explotación**: Solo es necesaria la JVM, ya que las aplicaciones Java ya están compiladas. La JVM interpreta y ejecuta los programas en cualquier sistema operativo compatible, garantizando portabilidad. No se requiere el JDK en producción, salvo que se necesite recompilar código en el servidor.

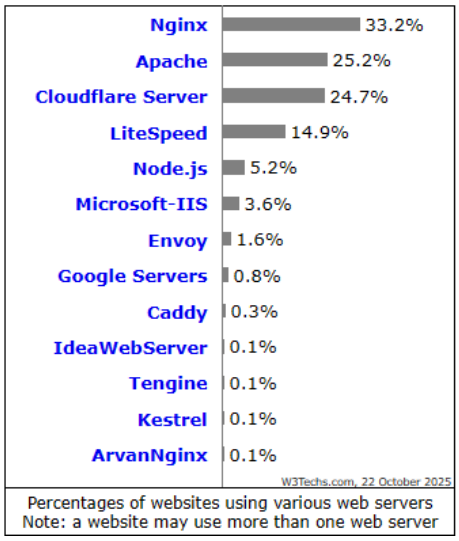


13. IDE más utilizados (características y grado de implantación actual).

- **Visual Studio Code (VS Code)**: editor ligero pero extensible que, con plugins, funciona como un IDE completo. Muy flexible, se puede usar para todo tipo de programas y lenguajes, gran ecosistema de extensiones y mucha comunidad. Es, **con diferencia**, el entorno más usado por desarrolladores en encuestas recientes.
- **IntelliJ IDEA (JetBrains)**: IDE potente orientado a Java y lenguajes JVM, con autocompletado inteligente, refactorizaciones avanzadas y herramientas integradas (testing, depuración, gestión de dependencias). Alto grado de implantación en proyectos empresariales Java y muy valorado entre desarrolladores backend.
- **Visual Studio (Microsoft)**: IDE completo para .NET, C#, C++ y desarrollo empresarial en Windows. Muy utilizado en empresas que usan tecnologías Microsoft; implantación especialmente alta en entornos corporativos y de aplicaciones de escritorio/servidor.
- **PyCharm (JetBrains)**: IDE especializado en Python, con buenas herramientas para testing, depuración y frameworks web (Django, Flask). Muy adoptado en proyectos Python profesionales, data science y backend Python.
- **Eclipse**: IDE histórico para Java (y otros lenguajes mediante plugins). Sigue presente en proyectos legacy y corporativos, aunque su uso ha decrecido frente a IntelliJ y VS Code; implantación moderada en entornos empresariales que mantienen infraestructuras antiguas.
- **Android Studio**: IDE oficial para desarrollo Android (basado en IntelliJ). Estándar en desarrollo móvil Android; prácticamente obligatorio si trabajas con apps nativas Android. Alto grado de implantación en desarrollo móvil.
- **NetBeans / Otros**: NetBeans sigue en uso en ciertos entornos educativos y proyectos Java; editores como Sublime o similares se usan por desarrolladores que prefieren ligereza. Su implantación es mucho menor comparada con VS Code o IntelliJ.
- https://survey.stackoverflow.co/2024/technology#1-integrated-development-environment

14. Servidores HTTP /HTTPS más utilizados (características y grado de implantación actual).

- **Nginx**: Es más moderno y muy rápido. Es ideal para sitios web con mucho tráfico, ya que gestiona las conexiones de manera más eficiente. Su popularidad ha crecido mucho en los últimos años.
- **Apache HTTP Server**: Es el servidor web más tradicional y popular. Es gratuito, robusto y tiene una gran comunidad. Funciona muy bien en la mayoría de los sistemas operativos. Su implantación es enorme.
- **Cloudflare Server**: se refiere a cualquiera de los miles de servidores proxy distribuidos globalmente por la empresa Cloudflare.
- **LiteSpeed**: Un servidor optimizado para el hosting compartido, que también busca la eficiencia en el consumo de recursos.
- **NodeJS**: Permite crear servidores web usando JavaScript. Es rápido y adecuado para aplicaciones que necesitan respuestas en tiempo real. Su uso ha aumentado mucho en proyectos modernos.



En Octubre de 2025

https://w3techs.com/technologies/overview/web_server

15. Apache HTTP vs Apache Tomcat

- **Apache HTTP Server**: Es un servidor web diseñado para entregar contenido estático o dinámico (como HTML, imágenes o archivos) a través de HTTP/HTTPS. Puede trabajar con diferentes lenguajes mediante módulos (como PHP). Es uno de los servidores más usados en el mundo por su estabilidad y compatibilidad.
- **Apache Tomcat**: Es un servidor especializado para ejecutar aplicaciones web basadas en Java (como servlets o JSP). Funciona como un contenedor de aplicaciones Java y está orientado a proyectos que requieren lógica de negocio con lenguaje Java.

  https://www.geeksforgeeks.org/devops/difference-between-apache-tomcat-server-and-apache-web-server/

16. Navegadores HTTP /HTTPS más utilizados (características y grado de implantación actual).

- **Chrome**: Es el navegador más usado del mundo. Destaca por su velocidad, su simplicidad y su gran cantidad de extensiones. Su implantación es la más alta.
- **Safari**: El navegador de Apple, usado en dispositivos macOS y iOS. Es conocido por su eficiencia energética.
- **Firefox**: Es un navegador de código abierto que se enfoca en la privacidad y la seguridad. Es una excelente alternativa a Chrome.

  https://radar.cloudflare.com/reports/browser-market-share-2025-q1

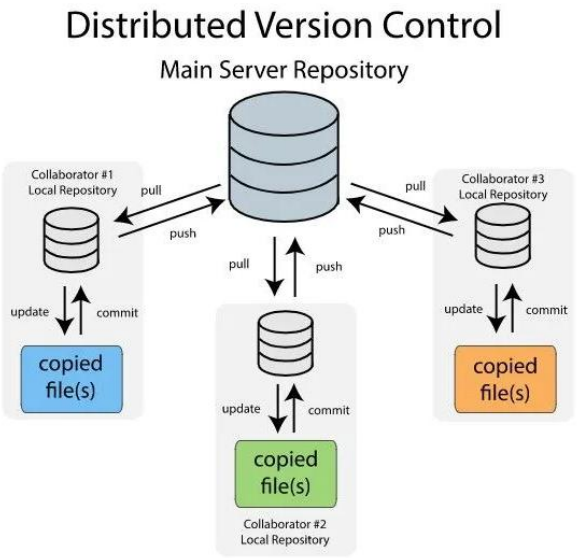
17. Generadores de documentación HTML (PHPDoc): PHPDocumentor, ApiGen, …

- Los generadores de documentación HTML crean automáticamente documentación de proyectos PHP a partir de los comentarios en el código (estándar PHPDoc). Muestran clases, funciones, métodos y variables en HTML, lo que facilita entender y mantener el proyecto.
  - **PHPDocumentor:** Es la herramienta más conocida. Genera documentación completa y estructurada, incluyendo diagramas, jerarquías de clases y ejemplos de uso. Soporta diferentes versiones de PHP y permite personalizar plantillas para adaptar la presentación de la documentación.
  - **ApiGen:** Especializado en generar documentación de APIs, muy útil cuando se desarrollan librerías o servicios que serán consumidos por otros desarrolladores. Ofrece un diseño limpio, navegación sencilla y enlaces directos a la definición de cada elemento del código.
  - **Otras alternativas:** Existen otras herramientas similares como Doxygen (*para múltiples lenguajes, incluido PHP*) o Sami (*inspirado en ApiGen*), que también generan documentación automática basada en comentarios.

  https://zonaphp.com/generadores-de-documentacion/

18. Repositorios de software – sistemas de control de versiones: GIT, CVS, Subversion, ...

- Un **sistema de control de versiones (VCS, Version Control System)** es una herramienta que registra los cambios realizados en los archivos de un proyecto a lo largo del tiempo. Permite a los desarrolladores:
  - Trabajar en equipo sin sobrescribir el trabajo de otros.
  - Revisar el historial de cambios y revertir a versiones anteriores si es necesario.
  - Gestionar ramas para experimentar con nuevas funcionalidades sin afectar la versión estable.
- Algunos de los sistemas más conocidos:
  - **GIT:** Sistema distribuido muy popular. Cada desarrollador tiene una copia completa del repositorio, lo que permite trabajar offline y fusionar cambios fácilmente. Usado por GitHub, GitLab y Bitbucket. Gran adopción en proyectos de código abierto y empresas.
  - **CVS (Concurrent Versions System):** Sistema más antiguo y centralizado. Permite mantener versiones históricas y colaborar, pero tiene limitaciones frente a sistemas modernos, como la gestión de ramas y trabajo distribuido. Su uso ha disminuido.
  - **Subversion (SVN):** Evolución de CVS, centralizado, pero más potente. Permite controlar versiones de archivos y carpetas, gestionar permisos y mantener un historial completo. Todavía se usa en empresas que mantienen proyectos legacy.
  - **Otros:** Mercurial, Perforce o Bazaar son también sistemas de control de versiones con distintas características, pero GIT domina la mayoría de proyectos actuales.



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.

- **Servidor web y PHP:** XAMPP, que incluye Apache HTTP Server, PHP 8.3 y BBDD. Esto permite desarrollar, probar y ejecutar aplicaciones web locales y dinámicas.
- **Bases de datos**: MySQL o MariaDB, integradas en XAMPP, para almacenar y gestionar la información de las aplicaciones web.
- **Lenguajes del cliente**: HTML, CSS y JavaScript, para desarrollar la interfaz y la interacción con el usuario desde el navegador.
- **Control de versiones**: Git, para gestionar el historial del código y facilitar la colaboración en proyectos de desarrollo web.
- **IDE/Editor de código**: VS Code, con extensiones para PHP, HTML, CSS y JavaScript, que permite depuración, autocompletado y pruebas locales.
- **Documentación de código**: Herramientas como PHPDocumentor para generar documentación HTML automáticamente a partir de los comentarios en el código.

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

- **Servidor web:** Apache HTTP Server, para ejecutar aplicaciones web dinámicas.
- **Bases de datos:** MySQL o MariaDB, gestionando la información de las aplicaciones.
- **Lenguajes del servidor:** PHP,
- **Lenguajes del cliente:** HTML, CSS y JavaScript, interpretados por el navegador del usuario final.
- **Acceso y despliegue**: Las aplicaciones se alojan en el servidor y se accede a ellas a través de navegadores web desde cualquier estación de trabajo de la red.
- **Control de versiones (opcional):** Git puede estar instalado para actualizar y traer los cambios de la rama master del repositorio principal de la aplicacion.

21. Realizar un estudio sobre los siguientes conceptos y su relación con el desarrollo de aplicaciones web:

- **CMS (Sistema de gestión de contenidos)**: es una aplicación que permite *crear*, *editar* y *gestionar contenidos* en un sitio web de forma sencilla, sin necesidad de programar directamente. Facilita tareas como publicar artículos, gestionar imágenes, administrar usuarios y aplicar plantillas de diseño.

  Los CMS permiten a los desarrolladores y administradores centrarse en la **f**uncionalidad y personalización mientras se simplifica la gestión del contenido. Suelen integrar bases de datos (MySQL/MariaDB) y funcionan sobre servidores web con PHP.

- **ERP (Sistema de planificación de los recursos empresariales)**: es un sistema que permite *gestionar* y *automatizar los procesos internos* de una empresa mediante módulos especializados, como ventas, contabilidad, inventario, recursos humanos o logística.

  En el desarrollo de aplicaciones web, los ERP pueden funcionar como aplicaciones web empresariales, accesibles desde el navegador y conectadas a bases de datos centrales. Su desarrollo requiere integrar la división funcional front-end y back-end, gestionar roles de usuario, seguridad, datos en tiempo real y APIs para comunicación con otros sistemas.