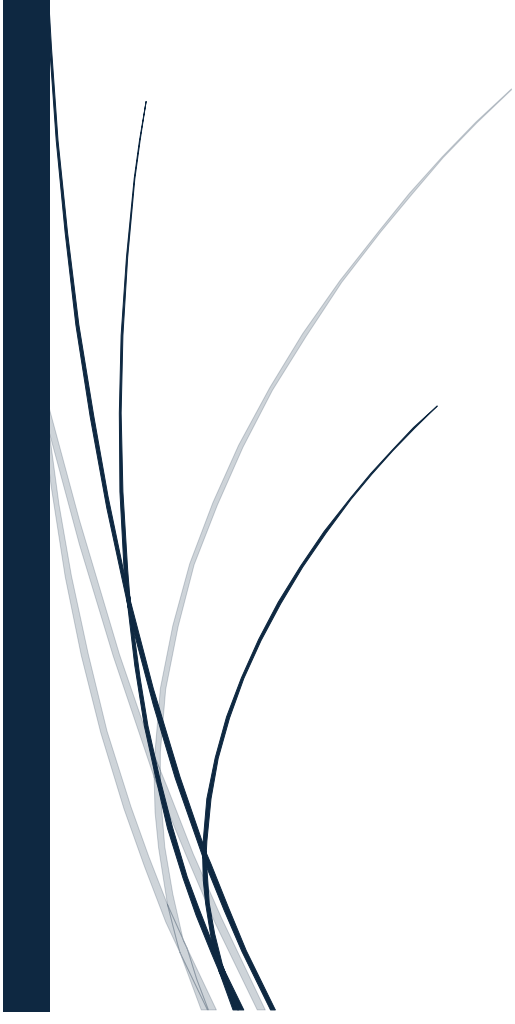
2-10-2025

Ejercicios DWES

Tema 1



Véronique Grué
DAW2 - DWES

TABLA DE CONTENIDO.

1. Protocolos de Comunicaciones: IP, TCP, HTTP, HTTPS..... 3

2. Modelo de Comunicaciones Cliente-Servidor 5

3. Métodos de Petición HTTP/HTTPS más Utilizados 6

4. Estudio sobre el concepto de URI /URL/URN. 8

5. Modelo de Desarrollo de Aplicaciones Multicapa 9

6. División Funcional Front-end / Back-end.....10

7. Página Web Estática, Dinámica, Aplicación Web y Mashup 11

8. Componentes de una Aplicación Web.....12

9. Programas del lado del cliente y del lado del servidor.....13

10. Lenguajes del Lado del Servidor de una aplicación web 15

11. Características y posibilidades de desarrollo de una plataforma XAMPP .17

12. En qué casos se instala (JVM) y JDK. 18

13. 13. IDE más utilizados (características y grado de implantación actual). ..18

14. Servidores HTTP /HTTPS más utilizados. 19

15. Apache HTTP vs Apache Tomcat.....21

16. Navegadores HTTP /HTTPS más utilizados.....22

17. Generadores de Documentación HTML (PHPDoc).....22

a. Para Código JavaScript/Front-end (Relacionado con HTML).....22

b. Para Código PHP (Usando PHPDoc) 22

18. Repositorios de Software - Sistemas de Ctrl de Versiones 23

19. Propuesta de Configuración del Entorno de Desarrollo. 23

20. Propuesta de Configuración del Entorno de Explotación. 23

21. Realizar un estudio sobre los siguientes conceptos : 24

22. Elegir y realizar un estudio y una presentación para la exposición del trabajo sobre una de las siguientes arquitecturas de desarrollo de Aplicaciones Web:.....25

23. GLOSARIO DE TÉRMINOS RELACIONADOS CON DWES 28

➢ Protocolos TCP/IP. Socket.....28

➢ Protocolo HTTP / HTTPS.....28

➢ HTML 28

➢ **XML 28**

➢ **JSON 29**

➢ **Lenguajes de programación embebidos en HTML 29**

➢ **Arquitecturas de desarrollo web 29**

➢ **Framework de desarrollo Web 30**

➢ **ERP (Enterprise Resource Planning) 30**

➢ **CMS 30**

➢ **PHP Hypertext Preprocessor) 30**

➢ **IDE 30**

➢ **Navegador 31**

➢ **Repositorio 31**

➢ **Entorno de Desarrollo 32**

➢ **Entorno de Explotación o Producción 32**

➢ **Gestión de la configuración. Control de cambios. Mantenimiento de la aplicación. 32**

➢ **Web Services 33**

➢ **AJAX 33**

➢ **Desarrollo de aplicaciones multicapa. Estrategias de diseño de aplicaciones Web. 33**

➢ **Aplicaciones basadas en microservicios 34**

➢ **Control de acceso a la aplicación web o los Web Services 35**

➢ **Validación de entrada de datos a una aplicación Web 35**

➢ **Posicionamiento de una aplicación Web 36**

➢ **Historia, situación actual y evolución del diseño de aplicaciones Web 37**

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

El protocolo de comunicación

- **HTTP (Protocolo de Transferencia de Hipertexto):** es el principal protocolo utilizado para enviar datos entre un navegador web y un sitio web. El HTTPS está encriptado para aumentar la seguridad de las transferencias de datos. Puerto por defecto: 80

- **HTTPS (Protocolo Seguro de Transferencia de Hipertexto):** Es una versión **más segura de HTTP** que cifra los datos intercambiados entre el navegador y el servidor web. Es importante para sitios donde se manejan datos sensibles, como bancos o tiendas online. Puerto por defecto: 443.

  Fuente **:** https://www.cloudflare.com/es-es/learning/ssl/what-is-https/

  Video relacionado con el protocolo HTTP/HTTPS :
  https://www.youtube.com/watch?v=nymxjNQ_qW0
  https://www.youtube.com/watch?v=60606AHuq8c

  Pasa por el cliente DNS para que convierta el nombre en IP.

  El puerto es un identificador para dirigir los datos de la capa de transporte a la capa de aplicación.

- **TCP (Protocolo de Control de Transmisión):** Este protocolo se asegura de fragmentar el paquete de datos si es más grande de lo permitido y se asegura de que todos los datos estén bien, garantiza la integridad de la información.

- **IP (Protocolo de Internet):**

  IP Privada/IP Pública:

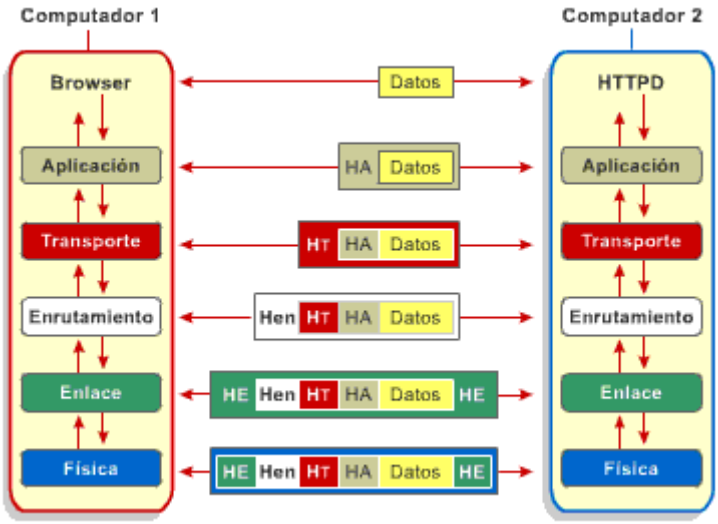
  - IP Privada: La IP privada es un número único que identifica tu dispositivo en una red privada (trabajo, casa…), asignado por el router a cada dispositivo de la red interna.
  - IP Pública: La IP publica es un numero único que es la dirección única que nos identifica ante el resto de Internet. El proveedor de servicios de internet (ISP) asigna una de su grupo de IPs disponibles. Es la que ven los servidores de las páginas web que se visitan.

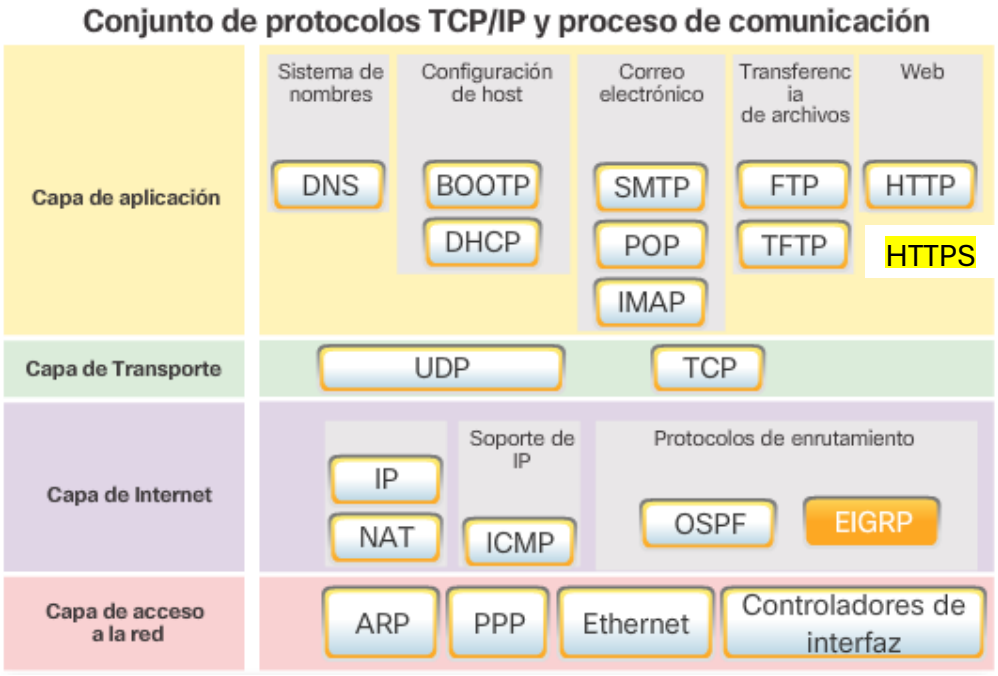
IP Dinámica/IP Estática:

- IP dinámica: Es la que se asigna automáticamente y puede cambiar a cada vez que te conectas. La mayoría de los usuarios tienen IP dinámica.
- IP estática: Es una dirección que no cambia y que se puede elegir, dentro de una red interna.

Video relacionado con los protocolos TCP/IP : https://www.youtube.com/watch?v=1pB2kan_AFk

Modelo del protocolo TCP/IP :



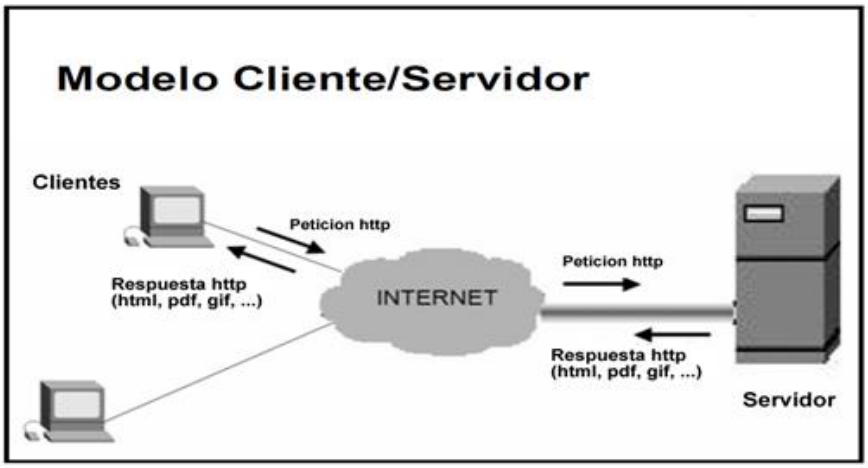


2. Modelo de Comunicaciones Cliente-Servidor

Este modelo es la base de cómo funcionan las aplicaciones web.

- **Servidor:** Ordenador o conjunto de ordenadores, que están a la espera de peticiones de solicitudes de recursos. Los servidores se pueden comunicar con muchos servidores a la vez.
- **Cliente:** Es un proceso en ejecución que realiza peticiones de recursos(navegador) hacía un servidor. Pueden ser textos, fotos, html...

Relación con aplicaciones web: Cuando se entra a una página web, el navegador (el cliente, que es sencillo) le pide una página al servidor, por medio de la URL. El servidor encuentra esa página y la envía al navegador. Es una relación de "petición y respuesta".



Video relacionado con la comunicación Cliente-Servidor:
https://www.youtube.com/watch?v=KA7Ngcgth0Q

3. Métodos de Petición HTTP/HTTPS más Utilizados

Las peticiones HTTP/HTTPS están formadas de un método, una URL, y una versión.

Estos son los métodos más utilizados:

- **GET:** Es el método más usado. Se utiliza para **obtener o pedir información**. Cuando se escribe una dirección en tu navegador, se está haciendo una petición GET para obtener la página web.
- **POST:** Se usa para **enviar información al servidor** para que la guarde o la procese. Por ejemplo, cuando se rellena un formulario de registro o escribes un comentario en un blog. Cuando se pincha en el botón de enviar de un formulario de una aplicación.
- **HEAD**: pide una respuesta idéntica a la de una petición GET, pero sin el cuerpo de la respuesta.
- **PUT**: **sube, carga o realiza una actualización de un recurso especificado**. Reemplaza todas las representaciones actuales del recurso de destino con la carga útil de la petición. Ejemplo: cambiar una foto en una aplicación.
- **DELETE**: borra un recurso en específico.
- **CONNECT**: establece un túnel hacia el servidor identificado por el recurso, para saber si se tiene acceso a un host.
- **OPTION**: es utilizado para devolver los métodos HTTP que el servidor soporta para un URL especifico.
- **TRACE**: realiza una prueba de bucle de retorno de mensaje a lo largo de la ruta al recurso de destino.
- **PATCH**: es utilizado para aplicar modificaciones parciales a un recurso.

Video relacionado con los métodos de petición HTTP/HTTPS:
https://www.youtube.com/watch?v=l2MihYAj0Iw&t=559s

De esta forma se podría incluir los métodos de petición en el código php: (página 469, libro PHP_Cookbook,_3rd_Edition(pdf)).

Next, route the request based on the HTTP method, so you can handle GETs, PUTs, POSTs, and DELETEs in different functions. For this, use `$_SERVER['REQUEST_METHOD']`:

```php
$method = strtolower($_SERVER['REQUEST_METHOD']);
switch($method) {
    case 'get':
        // handle a GET request
        get_book($request);
        break;
    case 'post':
        // handle a POST request
        post_book($request);
        break;
    case 'put':
        // handle a PUT request
        put_book($request);
        break;
    case 'delete':
        // handle a DELETE request
        delete_book($request);
        break;
    default:
        // unimplemented method
        http_response_code(405);
}
```

Como crear un enrutador con php :https://www.youtube.com/watch?v=B3SgvrFYsI0

Código con el framework Laravel. Ejemplo sacado de la IA.

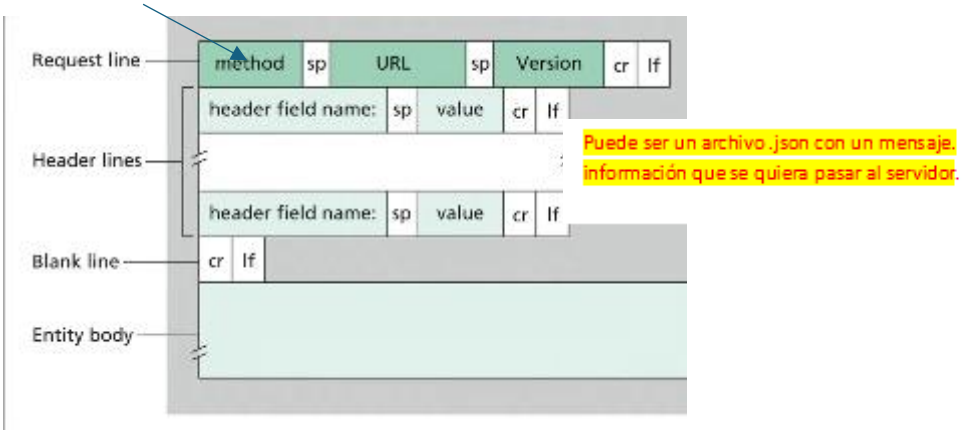
```php
use Illuminate\Support\Facades\Route;

// Ruta para obtener todos los usuarios (GET)
Route::get('/usuarios', function () {
    // Lógica para devolver la lista de usuarios
    return 'Lista de usuarios';
});

// Ruta para crear un nuevo usuario (POST)
Route::post('/usuarios', function () {
    // Lógica para guardar un nuevo usuario en la base de datos
    return 'Usuario creado';
});

// Ruta para actualizar un usuario específico (PUT)
Route::put('/usuarios/{id}', function ($id) {
    // Lógica para actualizar el usuario con el ID especificado
    return 'Usuario ' . $id . ' actualizado';
});
```

Método de petición: get, put...



4. Estudio sobre el concepto de URI /URL/URN.

- **URI (Identificador de Recursos Uniforme):** Es una cadena de caracteres que identifica un recurso. Está identificación se puede hacer por su ubicación (URL), o su nombre (URN)
  Ejemplos de URI :
  mailto:info@example.com(URI de recurso)
  data:text/plain;base64,SGVsbG8sIFdvcmxkIQ==(URI de datos)

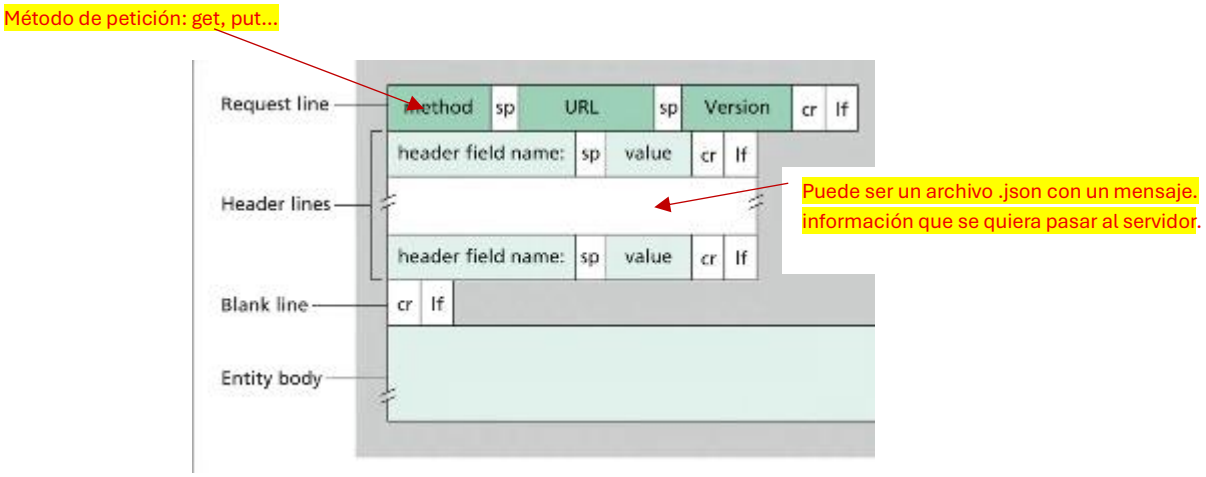
- **URL (Localizador de Recursos Uniforme):** Es una forma de URI que especifica cómo y dónde encontrar un recurso.

  Ejemplo: https://www.google.com/search.
  - **https:** es el protocolo.
  - **www.:** subdominio.
  - **Google**: nombre de dominio.
  - **.com**: extensión
  - **.es**: geolocalización
  - **/search**: ruta al recurso.

- **URN (Nombre de Recursos Uniforme):** Es una forma de URI que especifica el nombre permanente del recurso, pero no dice dónde encontrarlo.
  Ejemplo de URN : urn:isbn:0451450523

  Dentro del protocolo HTTP/HTTPS, laURL, sirve para saber la dirección del recurso solicitado en el servidor.

Esto sería un esquema de una petición : se puede ver la línea de requerimiento : donde está el método (get, put...), la url(la propia dirección), y la versión que sería una pagina en concreto de esa aplicación.



Fuente: https://www.youtube.com/watch?v=l2MihYAj0Iw&t=74s

5. Modelo de Desarrollo de Aplicaciones Multicapa

- **Capa de Presentación (Front-end):** Es lo que **se ve y con lo que el usuario interactúa** en el navegador (la interfaz gráfica, los botones, las imágenes...).
- **Capa del controlador:** Actúa como intermediario entre la capa de presentación y la lógica de negocio. Se encarga de recibir las acciones del usuario desde la interfaz (por ejemplo, al hacer clic en un botón), enviarlas a la lógica de negocio para su procesamiento y luego devolver una respuesta actualizada a la capa de presentación. En otras palabras, coordina el flujo de información entre las distintas capas para que la aplicación funcione correctamente.
- **Capa de Lógica de Negocio (Back-end):** Es el cerebro de la aplicación. Aquí se procesan las reglas del negocio. Por ejemplo, a la hora de comprar algo, esta capa calcula el precio, los impuestos y el total.
- **Capa de Acceso a Datos (Back-end):** Aquí se guarda y se accede a la información. Es la capa que se comunica con la base de datos para guardar y recuperar datos de los usuarios, productos, etc.

La **comunicación entre capas** es vital: la capa de presentación le pide datos a la capa de lógica, y esta a su vez le pide datos a la capa de datos.

Fuente: https://www.youtube.com/watch?v=UEUC3uqo9JU

6. División Funcional Front-end / Back-end

Para que sirve la aplicación en la fachada principal y la parte trasera. Del lado del Usuario normal, o del Usuario administrator.

- **Front-end:** Es la parte de la aplicación a la que los usuarios sin registrar y que se hayan registrado, pero no que sean administradores, pueden acceder. Pueden ver el contenido y rellenar formularios comprar...

  Se encarga de mostrar el **diseño**, los **gráficos**, la **estructura** y el **contenido** de forma atractiva y funcional.

- **Back-end(Lógica y Datos):** Aunque todos los usuarios **interactúan indirectamente** con él, es parte a la que los usuarios que se hayan registrado y que sean administradores puedan acceder para poder modificar el contenido de la aplicación. Procesa las peticiones del Front-end. Esto incluye:

  - Almacenar y recuperar datos de la base de datos.
  - Ejecutar la lógica de negocio (ej: calcular el precio de un envío, verificar una contraseña).
  - Gestionar la autenticación y la seguridad.
  - Generar el contenido que el Front-end mostrará.

Fuente: https://blog.hubspot.es/website/frontend-y-backend

7. Página Web Estática, Dinámica, Aplicación Web y Mashup

- **Página web estática:** El contenido no cambia a menos que el programador lo modifique. Se crea con **HTML y CSS**. Por ejemplo, un currículum online sencillo.
- **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 un paso más allá de la página dinámica. Debe tener control de acceso, no es para todo el mundo, y suele tener publicidad dirigida. (Tipos de acceso diferentes para cada usuario.)
- **Mashup(hibrida):** Es una aplicación web que **combina información de varias fuentes** para crear algo nuevo. Por ejemplo, Trivago, que solicita la información de varias páginas de viajes. (Varias tecnologías utilizadas.)

Fuente : Diferencia entre página estática y dinámica.
https://www.youtube.com/watch?v=LABbBI5O3GA

Diferencia entre página web, sitio web y aplicación web: https://www.youtube.com/watch?v=Ko1AR8-dUsE&t=143s

8. Componentes de una Aplicación Web

Una aplicación web completa generalmente tiene estos componentes:

- **Servidor Web:** Es el intermediario entre las solicitudes del cliente (navegador) y la aplicación. Procesa las peticiones HTTP entrantes y envía las respuestas adecuadas. Ejemplos populares incluyen Nginx y Apache.
- **Módulo encargado de ejecutar código:** Contiene la funcionalidad central de la aplicación, como el procesamiento de datos, la autenticación de usuarios y la implementación de la lógica de negocio. Se escribe en lenguajes de programación y backend, como (PHP, Python).
- **Sistema Gestor de Base de Datos:** Almacena y administra la información de la aplicación. Existen dos tipos principales:
    - **Bases de datos relacionales:** Como MySQL y PostgreSQL, ideales para datos estructurados.
    - **Bases de datos NoSQL:** Como MongoDB y Redis, para datos no estructurados.
- **Los ficheros escritos en esos lenguajes de programación (HTML, CSS, PHP, SQL, JS,...) y los directorios necesarios para organizarlos.**

Fuente :https://elpuig.xeill.net/Members/vcarceler/asix-m09/uf1/nf1/a1

9. Programas del lado del cliente y del lado del servidor

- **¿Qué hacen los programas del lado del servidor?**
    - **Generar contenido dinámico**: Crean páginas web que cambian según la petición del usuario o los datos de una base de datos, en lugar de entregar contenido estático. (Lenguajes que se podrían utilizar: PHP, JSP...)
    - **Interactuar con bases de datos**: Permiten almacenar, recuperar y gestionar datos, lo cual es esencial para la mayoría de las aplicaciones web modernas. (Lenguajes que se podrían utilizar: Phyton, C#...)
    - **Procesar solicitudes del cliente**: Reciben peticiones de un navegador, ejecutan la lógica necesaria y luego envían una respuesta al navegador. (Lenguajes que se podrían utilizar: Node.js, Ruby...)
    - **Ejecutar lógica empresarial compleja**: Realizan operaciones que requieren más poder de procesamiento o acceso a información sensible, como cálculos, validaciones o integraciones. (Lenguajes que se podrían utilizar: Java, Phyton, C#, Perl...)
    - **Proporcionar servicios personalizados**: Pueden encargarse de tareas como el envío de correos electrónicos, SMS o la gestión de sesiones de usuario.

- **Lado del Cliente:** El código se ejecuta en tu navegador. Se encarga de la interfaz y la interacción directa con el usuario.
    - **Construcción de la interfaz:** Usan **HTML** para la estructura del contenido (texto, imágenes) y **CSS** para el estilo y la apariencia de la página. Ajax son las funcionalidades asíncronas de JavaScript.

- **Interactividad:** Mediante JavaScript, permiten crear funcionalidades dinámicas como menús, formularios, animaciones y la respuesta a las acciones del usuario, haciendo que las páginas sean más receptivas. **Ajax** son las funcionalidades asíncronas de JavaScript.
- **Validación de datos:** Pueden verificar la información introducida por el usuario en los formularios antes de enviarla al servidor, proporcionando una respuesta inmediata si hay errores.
- **Mejora de la experiencia de usuario:** Al gestionar la interacción en el propio dispositivo, se reduce la necesidad de recargar la página o enviar constantes solicitudes al servidor, resultando en una navegación más rápida y fluida.
- **Almacenamiento local:** Permiten guardar datos en el navegador del usuario, como las preferencias de personalización o el contenido de un carrito de compras, para que la web sea más rápida y pueda funcionar parcialmente sin conexión a internet. (cookies)

Fuente: https://www.youtube.com/watch?v=FhVeO1sDLHk

10. Lenguajes del Lado del Servidor de una aplicación web

- **Lado del Servidor:**
  - **Lenguajes de scripting – Lenguajes de guiones:**
    - **PHP:** Muy popular para la web, es fácil de integrar con HTML y se utiliza en muchos gestores de contenido como WordPress. Es un lenguaje que se interpreta
    - **Python:** Ampliamente usado, especialmente para startups y para el desarrollo de aplicaciones back-end de manera general, con frameworks como Django.
    - **JSP (JavaServer Pages)** es una tecnología de Java que permite integrar código Java dentro de documentos HTML para generar páginas web dinámicas. Permite mezclar contenido estático con lógica de programación para crear aplicaciones web que responden a las peticiones de los usuarios, siendo un componente fundamental en el desarrollo web con Java.
    - PERL(Practical Extraction and Report Language) (Lenguaje Práctico de Extracción e Informes): permite trabajar con texto y generar informes de manera práctica.
    - **Ruby:** Conocido por su facilidad de uso y el framework Ruby on Rails.
    - **JavaScript (Node.js):** Permite usar JavaScript tanto en el cliente como en el servidor, ofreciendo gran flexibilidad.
    - **C# (ASP.NET):** Un lenguaje de Microsoft que se utiliza para construir aplicaciones web robustas.
  - **Lenguajes compilados a código nativo:**

    Aplicaciones CGI(Common Gateway Interface): son programas que permiten a un servidor web ejecutar aplicaciones externas para generar contenido dinámico en las páginas web.(C, C++...)

- **Lenguajes compilados a código intermedio:**

  También conocido como bytecode, son Java, C# y otros lenguajes que se ejecutan sobre máquinas virtuales como la Máquina Virtual de Java (JVM) o el Common Language Runtime (CLR) de .NET. Este código intermedio es una representación más abstracta del código máquina que permite una mayor portabilidad, ya que el código se traduce una sola vez y luego es interpretado y ejecutado en cualquier plataforma que tenga el entorno de ejecución adecuado.

Programas más populares del lado del servidor en agosto 2025:

Most popular server-side programming languages

© W3Techs.com	usage	change since 1 August 2025
1. PHP	73.4%	-0.3%
2. Ruby	6.4%	+0.1%
3. Java	5.3%	
4. JavaScript	5.0%	+0.3%
5. ASP.NET	4.8%	-0.1%

percentages of sites

Fuente: https://w3techs.com/

11. Características y posibilidades de desarrollo de una plataforma XAMPP

XAMPP es una plataforma de desarrollo local. Es como un paquete todo en uno que te permite crear y probar aplicaciones web en tu propio ordenador, sin tener que subir nada a un servidor real.

- **Características:** Es muy fácil de instalar y usar. Incluye los componentes más comunes:
    - **X:** Multiplataforma (funciona en Windows, macOS, Linux).
    - **A: Apache** (Un servidor web que se encarga de recibir las peticiones de los navegadores y enviarles las páginas web.).
    - **M: MariaDB** (base de datos, similar a MySQL).
    - **P: PHP** (Lenguaje de programación del lado del servidor que permite crear sitios web dinámicos e interactivos).
    - **P: Perl**.(Otro lenguaje de programación que se incluye en el paquete para el desarrollo web.)
- **Utilidad :**
    - **Desarrollo y prueba local:** Permite a los desarrolladores crear y probar sitios web y aplicaciones en sus propios ordenadores sin necesidad de un servidor en línea, lo que les da un control total sobre su entorno de trabajo.
    - **Facilidad de configuración:** XAMPP es conocido por su facilidad de instalación y uso, ya que agrupa todos los componentes necesarios en un solo paquete, simplificando la configuración de un servidor web.
    - **Ideal para principiantes:** Es una herramienta excelente para quienes se inician en el desarrollo web, ya que les permite comenzar a experimentar con tecnologías web sin tener que configurar cada componente por separado.
    - **Entorno de desarrollo controlado:** Ofrece un entorno seguro y controlado para trabajar en un proyecto antes de llevarlo a un entorno de producción o publicarlo en un servidor en línea.

Fuente: https://www.dongee.com/tutoriales/que-es-xampp/

12. En qué casos se instala (JVM) y JDK.

- **Máquina Virtual Java (JVM):** Es como un intérprete especial que permite que el código Java se ejecute en cualquier computadora, sin importar el sistema operativo. Por eso se dice que Java es "escribe una vez, ejecuta en cualquier lugar". Es necesaria en el entorno de explotación para que la aplicación funcione.
- **Java Development Kit (JDK):** Es un conjunto de herramientas para desarrollar aplicaciones Java. Incluye la JVM y otras herramientas como un compilador. Es necesaria en el entorno de desarrollo para poder escribir y compilar el código.

En resumen: Necesitas el **JDK para programar** en Java y la **JVM para ejecutar** los programas Java.

Fuente: https://www.q2bstudio.com/nuestro-blog/20677/jdk-jre-y-jvm-diferencias-para-desarrolladores

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

Un **IDE (Entorno de Desarrollo Integrado)** es un programa que facilita la escritura de código, como un editor de texto súper avanzado para programadores.

- **Visual Studio Code (VS Code):** Es un editor de código muy popular y ligero. Es gratuito y extensible con miles de plugins, lo que lo hace ideal para casi cualquier lenguaje de programación. Su implantación es altísima.
- **IntelliJ IDEA:** Es un IDE muy potente, ideal para proyectos Java. Ofrece herramientas avanzadas que facilitan el desarrollo. Su versión gratuita (Community Edition) es muy usada.
- **Eclipse:** Un IDE gratuito y de código abierto, muy popular en su momento para Java y otros lenguajes. Aunque su interfaz es un poco más compleja, sigue siendo muy usado.
- **Android Studio:** Basado en IntelliJ, es el IDE oficial para desarrollar aplicaciones de Android.
- **PyCharm :** Si se trabaja con Python, PyCharm es una de las mejores opciones, ya que incluye potentes herramientas de refactorización y análisis de código.

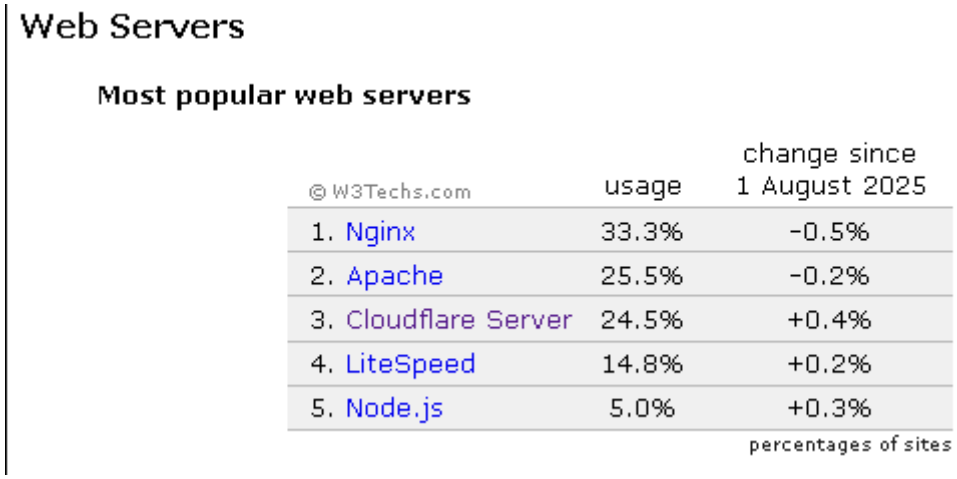
- **Netbeans:** Si se necesita programar aplicaciones en Java, HTML 5, C/C++, CSS, Javascript y PHP entre otros, una de las mejores y más veteranas IDE que tenemos a nuestra disposición es NetBeans, una aplicación que incluye un editor de texto que analiza el código tanto sintáctica como semánticamente, lo que permite evitar errores básicos de programación que nos puede hacer perder muchas horas. Es multiplataforma, con aplicaciones disponibles para todos los sistemas operativos compatibles con Java como Windows, Mac, Linux, OSX y BSD.

  Fuente: https://www.softzone.es/programas/lenguajes/mejores-ide-programar/

14. Servidores HTTP /HTTPS más utilizados.

- **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.
- **Microsoft IIS:** Es el servidor web de Microsoft, integrado en los sistemas operativos Windows Server. Muy utilizado en entornos de empresas que ya usan tecnologías de Microsoft.

Servidores más populares en agosto 2025.



Web Servers

Most popular web servers

© W3Techs.com	usage	change since 1 August 2025
1. Nginx	33.3%	-0.5%
2. Apache	25.5%	-0.2%
3. Cloudflare Server	24.5%	+0.4%
4. LiteSpeed	14.8%	+0.2%
5. Node.js	5.0%	+0.3%

percentages of sites

Fuente: https://w3techs.com/

15. Apache HTTP vs Apache Tomcat

- **Apache HTTP:** es un software que se ejecuta en un servidor(Hardware+sistema operativo). Su trabajo es establecer una conexión entre un servidor y los navegadores de los visitantes del sitio web (Firefox, Google Chrome, Safari, etc.) mientras envían archivos entre ellos (estructura cliente-servidor). Apache es un software multiplataforma, por lo cual funciona tanto en servidores Unix como en Windows.
- **Apache Tomcat:** es un contenedor web, que tiene como función ejecutar aplicaciones Java en lugar de sitios web estáticos (es un contenedor de servlets (Programas en java que generan contenido dinámico)). Tomcat puede ejecutar varias especificaciones diferentes de Java, como Java Servlet, JavaServer Pages (JSP), Java EL y WebSocket.

  Relación con Apache HTTP: A menudo, en entornos de alta producción, el Apache HTTP Server se coloca delante de Tomcat. Apache HTTP gestiona el tráfico y sirve el contenido estático, delegando solo las peticiones de aplicaciones Java a Tomcat, mejorando así el rendimiento y la seguridad.

Característica	Apache HTTP Server	Apache Tomcat
Rol Primario	Servidor Web (HTTP/S)	Contenedor de Servlets/Web Java
Contenido Principal	Estático (HTML, imágenes)	Dinámico (Servlets, JSPs, Java Apps)
Tecnología Clave	Módulos (mod_php, mod_rewrite)	Java, Servlets, JSP
¿Necesita otro Server?	No (Puede ir solo)	Incluye su propio HTTP Server (Coyote)

Fuente: https://www.hostinger.com/es/tutoriales/que-es-apache

16. Navegadores HTTP /HTTPS más utilizados.

- **Google 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. **69,23%**
- **Safari:** El navegador de Apple, usado en dispositivos macOS y iOS. Es conocido por su eficiencia energética. **14,98%**
- **Microsoft Edge:** El navegador de Microsoft, basado en la misma tecnología que Chrome. Es rápido y está bien integrado con Windows.**5,03%**
- **Mozilla Firefox:** Es un navegador de código abierto que se enfoca en la privacidad y la seguridad. Es una excelente alternativa a Chrome.**2,26%**

Fuente: https://gs.statcounter.com/

17. Generadores de Documentación HTML (PHPDoc)

a. Para Código JavaScript/Front-end (Relacionado con HTML)

El estándar más popular en este caso es JSDoc:

- JSDoc (Estándar para JavaScript): Es el formato de comentario y herramienta de generación de documentación más común para JavaScript. Genera un sitio web HTML que explica las clases, métodos y propiedades de tu código.
- TypeDoc (Para TypeScript): Similar a JSDoc, pero diseñado específicamente para leer anotaciones de tipos de TypeScript y generar una documentación HTML clara.

b. Para Código PHP (Usando PHPDoc)

Estos son los programas que leen los comentarios PHPDoc en tu código PHP y generan la documentación:

- phpDocumentor (El estándar de la comunidad PHP): Es el generador más establecido y respeta la especificación PHPDoc oficial. Genera documentación de referencia completa para bibliotecas y proyectos PHP.
- Doxygen (Multi-lenguaje (soporta PHP)): Es muy potente y versátil. Aunque se usa para muchos lenguajes (C++, Java, Python), es una opción popular para generar documentación a partir de código PHP con sintaxis PHPDoc.

18. Repositorios de Software - Sistemas de Ctrl de Versiones

Un sistema de control de versiones (VCS) es un historial de cambios para el código. Permite guardar diferentes versiones del proyecto, volver a versiones anteriores, y trabajar en equipo sin pisar el código de los demás.

- **GIT:** Es el VCS más popular y moderno. Es distribuido, lo que significa que cada programador tiene una copia completa del historial del proyecto. Es el estándar de la industria. GitHub y GitLab son plataformas que usan Git.

19. Propuesta de Configuración del Entorno de Desarrollo.

- **Servidor Web: XAMPP (con Apache HTTP Server)**. Es el más sencillo de instalar y usar para un entorno de aprendizaje. Nosotros haremos un servidor Apache con Ubuntu
- **Lenguaje de Programación: PHP**. Es el lenguaje que viene por defecto en XAMPP y es ideal para el desarrollo web del lado del servidor. Es el Lenguaje que se utilizará
- **IDE:** NetBeans, **Visual Studio Code (VS Code)**. Es gratuito, ligero y cuenta con excelentes extensiones para PHP.
- **Control de Versiones: Git**.
- **Entorno de Pruebas: XAMPP**.

20. Propuesta de Configuración del Entorno de Explotación.

- **Servidor Web: Apache HTTP Server** o **Nginx**.
- **Base de Datos: MySQL o MariaDB** (la que viene con XAMPP es ideal).
- **Lenguaje de Programación: PHP (versión 8.x)**.
- **Herramientas:** phpMyAdmin, que viene con XAMPP).

21. Realizar un estudio sobre los siguientes conceptos :

- **CMS (Sistema de Gestión de Contenidos):** Es un programa que permite crear, editar y gestionar el contenido de una página web sin necesidad de saber programar. Es ideal para blogs, tiendas online o sitios corporativos. Ejemplos: WordPress, Joomla, Drupal. Su relación con el desarrollo web es que son herramientas que permiten crear sitios complejos de manera muy rápida.

- **ERP (Sistema de Planificación de los Recursos Empresariales**): Es un software que integra todos los procesos de una empresa (ventas, finanzas, recursos humanos, inventario) en un solo sistema. Es útil para la gestión interna de grandes empresas. Su relación con el desarrollo web es que a menudo tienen interfaces web para que los empleados accedan a ellos desde cualquier lugar. Ejemplos: SAP, Oracle ERP cloud, Microsoft Dynamics 365, NetSuite.

22. Elegir y realizar un estudio y una presentación para la exposición del trabajo sobre una de las siguientes arquitecturas de desarrollo de Aplicaciones Web:

es un esquema de cómo interactúan entre sí los distintos componentes de tu aplicación web.

- **MEAN**: Es una arquitectura de JavaScript de punta a punta. Se usa para crear aplicaciones web dinámicas y escalables.
    - MongoDB (base de datos)
    - Express.js (framework del back-end)
    - Angular (framework del front-end)
    - Node.js (entorno de ejecución del back-end)
- **Java EE vs Spring:** es un conjunto de especificaciones para aplicaciones empresariales de gran escala en Java. Spring es un framework que simplifica y mejora el desarrollo en Java, siendo más flexible y ligero.
- **Laravel**: Es un framework de PHP muy popular que facilita la creación de aplicaciones web. Destaca por su sintaxis elegante y sus herramientas para tareas comunes. Es una excelente opción si eliges aprender PHP.
- **Angular:** Es un framework de JavaScript desarrollado por Google para crear interfaces de usuario de aplicaciones de una sola página (SPA). Es muy potente y estructurado.
- **Microsoft .**NET: es una plataforma de desarrollo de software gratuita, multiplataforma y de código abierto que permite crear diversos tipos de aplicaciones, como aplicaciones web, móviles, de escritorio y de servicios. Se compone de herramientas, lenguajes como C#, F# y VB, y un conjunto de bibliotecas y un entorno de ejecución (CLR) que proporcionan una base para el desarrollo moderno, escalable y de alto rendimiento en Windows, Linux, macOS y otras plataforma.
- **Symfony:** es un framework de código abierto en PHP que proporciona herramientas y componentes reutilizables para el desarrollo eficiente de aplicaciones web y servicios. Basado en el patrón Modelo-Vista-Controlador (MVC), ayuda a los desarrolladores a organizar su código, separar la lógica de negocio de la interfaz de usuario y acelerar la creación de proyectos de cualquier tamaño, desde pequeñas aplicaciones hasta sistemas complejos.

- **CakePHP**: es un *framework* PHP de código abierto que permite a los desarrolladores crear aplicaciones web de manera rápida y estructurada utilizando la arquitectura Modelo-Vista-Controlador (MVC). Se inspira en Ruby on Rails y proporciona una estructura y herramientas predefinidas, como la generación de código y la gestión de bases de datos, para facilitar el desarrollo de aplicaciones web robustas y escalables.
- **CodeIgniter**: es un framework PHP de código abierto, ligero y rápido que proporciona herramientas y una estructura para desarrollar aplicaciones web de manera más eficiente y organizada. Sigue el patrón de diseño MVC (Modelo-Vista-Controlador), lo que ayuda a separar la lógica de la interfaz y facilita la creación de sitios web dinámicos. Es conocido por su rendimiento, fácil instalación y una buena documentación, lo que lo hace ideal tanto para principiantes como para proyectos complejos.

CMS (Content Management System)

- **Propósito:** Administrar el contenido de un sitio web o aplicación digital, como artículos de blog, imágenes y videos.
- **Funciones:** Permite a los usuarios crear, editar, organizar y publicar contenido en una plataforma digital sin necesidad de codificación compleja.
- **Ejemplo:** Un CMS como WordPress se utiliza para gestionar el contenido de un blog, mientras que un CMS web como Drupal se usa para sitios públicos.

Video donde ver las diferencias entre un CMS y un CRM:
https://youtube.com/shorts/6TKuI4qoOjg?si=jW_3IL_STsiW4iZo

CRM (Customer Relationship Management)

- **Propósito:** Gestionar y analizar las interacciones con los clientes a lo largo de su ciclo de vida.
- **Funciones:** Controla datos de clientes, historial de compras, actividades de marketing y ventas, y atención al cliente para mejorar la experiencia del cliente.
- **Enfoque:** Se centra en el "front office" (la interacción directa con el cliente) para fortalecer las relaciones y aumentar la satisfacción del cliente.

ERP (Enterprise Resource Planning)

- **Propósito:** Integrar y gestionar todos los procesos centrales de un negocio en un único sistema.
- **Funciones:** Abarca áreas como contabilidad, finanzas, inventario, compras, producción, recursos humanos y ventas, para una visión integral de las operaciones.
- **Ventajas:** Optimiza la eficiencia, ahorra costos, mejora la toma de decisiones y facilita la colaboración entre departamentos.

Video para entender mejor la diferencia entre ERP y CRM:
https://www.youtube.com/watch?v=tRGCq5upKeI

Diferencias clave

- **Enfoque:** CMS es para contenido digital, CRM es para clientes, y ERP es para procesos empresariales internos.
- **Uso:** CMS es para publicar y administrar un sitio web, CRM es para la gestión de ventas y atención al cliente, y ERP es para la administración general del negocio.
- **Datos gestionados:** CMS maneja activos digitales; CRM gestiona datos del cliente; ERP maneja datos financieros, de inventario y de producción.

23. GLOSARIO DE TÉRMINOS RELACIONADOS CON DWES

Estudios propuestos para el módulo Desarrollo Web en Entorno Servidor

➢ **Contenidos y la diferencia entre los módulos que tienes en este curso.**

➢ **Protocolos TCP/IP. Socket.**

El protocolo TCP/IP (Protocolo de Control de Transmisión/Protocolo de Internet) es el conjunto de reglas fundamentales que permiten a los dispositivos comunicarse y transferir datos en redes. Funciona dividiendo los datos en paquetes y dirigiéndolos a su destino de forma fiable (TCP) y enrutándolos por la red (IP).

Fuente: https://www.cloudflare.com/es-es/learning/ssl/what-is-https/

Los sockets son canales de comunicación que permiten que procesos no relacionados intercambien datos localmente y entre redes. Un único socket es un punto final de un canal de comunicación bidireccional.

Fuente: https://www.ibm.com/docs/es/aix/7.2.0?topic=concepts-sockets

➢ **Protocolo HTTP / HTTPS**

El protocolo de transferencia de hipertexto seguro (HTTPS) es la versión segura de HTTP, que es el principal protocolo utilizado para enviar datos entre un navegador web y un sitio web. El HTTPS está encriptado para aumentar la seguridad de las transferencias de datos.

➢ **HTML**

El Lenguaje de Marcado de Hipertexto (HTML) es el código que se utiliza para estructurar y desplegar una página web y sus contenidos. Es un lenguaje de marcado interpretado.

Fuente:
https://developer.mozilla.org/es/docs/Learn_web_development/Getting_started/Your_first_website/Creating_the_content

➢ **XML**

El lenguaje de marcado extensible (XML) permite definir y almacenar datos de forma compartible. XML admite el intercambio de información entre sistemas de computación, como sitios web, bases de datos y aplicaciones de terceros. Las reglas predefinidas facilitan la transmisión de datos como archivos XML a través de cualquier red, ya que el destinatario puede usar esas reglas para leer los datos de forma precisa y eficiente.

Fuente: https://aws.amazon.com/es/what-is/xml/

➢ JSON

Un archivo JSON (JavaScript Object Notation) es un formato de almacenamiento e intercambio de datos que utiliza una estructura de clave-valor para representar información de manera legible tanto para humanos como para máquinas. Se basa en la sintaxis de los objetos de JavaScript, aunque es compatible con muchos otros lenguajes de programación.

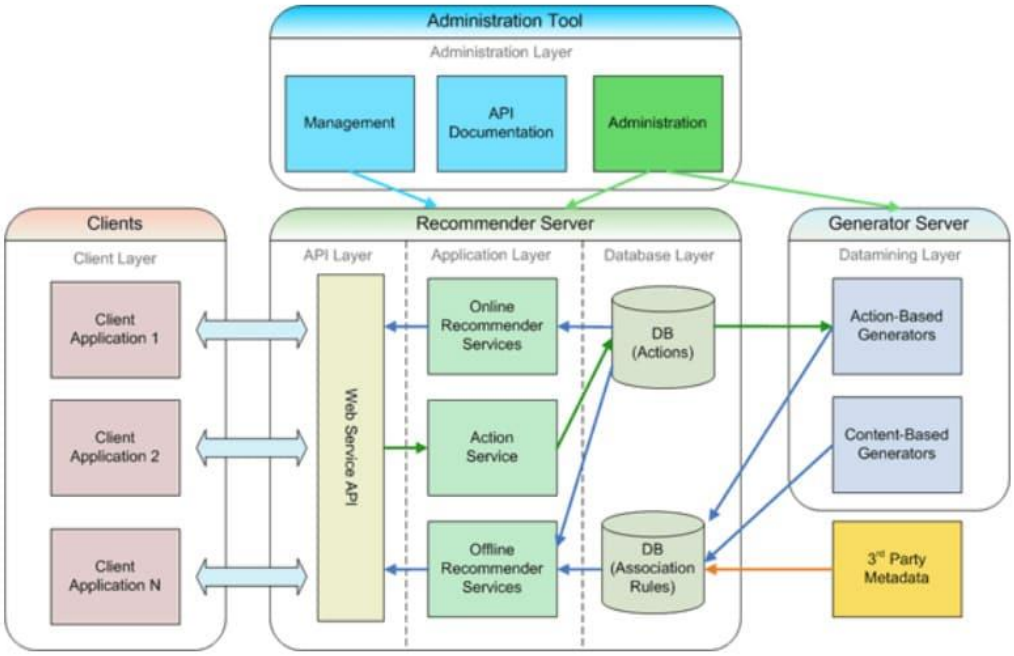
Fuente: https://www.arsys.es/blog/archivo-json-que-es-y-para-que-sirve

➢ Lenguajes de programación embebidos en HTML

Los lenguajes embebidos en HTML son aquellos que se insertan dentro del código HTML para dar más funcionalidad e interactividad a una página web. El lenguaje principal para esto es JavaScript, en el lado cliente y PHP en el lado servidor, que permiten ejecutar código en el navegador y crear elementos dinámicos. Otros lenguajes, como CSS, también se integran para estilizar los elementos HTML, aunque son lenguajes de marcado y diseño, no de programación.

➢ Arquitecturas de desarrollo web

es un esquema de cómo interactúan entre sí los distintos componentes de tu aplicación web.



Fuente: https://kinsta.com/es/blog/arquitectura-aplicaciones-web/

➢ Framework de desarrollo Web

es un conjunto de herramientas, librerías y estructuras predefinidas que proporcionan una base para construir aplicaciones web de forma más rápida, organizada y eficiente.

Fuente: https://www.youtube.com/watch?v=1gbdo1Nq-d8

➢ ERP (Enterprise Resource Planning)

Es sistema de planificación de recursos empresariales en español, es un software que permite controlar y automatizar todos los flujos clave de una empresa. Sirve para centralizar los flujos de información y trabajo, ofreciendo una visión unificada de la empresa, para mejorar la eficiencia, toma de decisiones y productividad.

Fuente: https://www.holded.com/es/blog/que-es-erp-y-para-que-sirve?

➢ CMS

Es un software que ayuda a los usuarios a crear, administrar y modificar contenido en un sitio web sin la necesidad de conocimientos técnicos especializados.

En un lenguaje más simple, un sistema de gestión de contenido es una herramienta que le ayuda a construir un sitio web sin necesidad de escribir todo el código desde cero

Fuente: https://kinsta.com/es/base-de-conocimiento/sitema-de-gestion-de-contenido/

➢ PHP Hypertext Preprocessor)

es un lenguaje de programación de código abierto ampliamente utilizado en el desarrollo web. En otras palabras, es la tecnología que permite crear sitios web dinámicos, donde el contenido puede cambiar según la interacción del usuario.

➢ IDE

Un entorno de desarrollo integrado (IDE) es un sistema de software para el diseño de aplicaciones que combina herramientas del desarrollador comunes en una sola interfaz gráfica de usuario (GUI). Generalmente, un IDE cuenta con las siguientes características:

- Editor de código fuente: editor de texto que ayuda a escribir el código de software con funciones como el resaltado de la sintaxis con indicaciones visuales, el relleno automático específico para el lenguaje y la comprobación de errores a medida que se escribe el código.

- Automatización de las compilaciones locales: herramientas que automatizan las tareas sencillas y repetitivas como parte de la creación de una compilación local del software para que use el desarrollador, como la compilación del código fuente de la computadora en código binario, el empaquetado de ese código y la ejecución de pruebas automatizadas.
- Depurador: programa que sirve para probar otros programas y mostrar la ubicación de un error en el código original de forma gráfica.

Fuente:https://www.redhat.com/es/topics/platform-engineering/what-is-ide

➢ Navegador

Un navegador web o explorador web (web browser, en inglés) es un software de aplicación que permite al usuario ingresar a diferentes páginas web en internet, ya sea directamente a través de una dirección URL o bien desde un hipervínculo o link.

Fuente:https://concepto.de/navegador

➢ Repositorio

Un repositorio, o repo, es un tipo de almacenamiento digital centralizado que los desarrolladores utilizan para realizar y administrar cambios en el código fuente de una aplicación. Los desarrolladores tienen que almacenar y compartir carpetas, archivos de texto y otros tipos de documentos al desarrollar software. Un repositorio cuenta con características que permiten a los desarrolladores rastrear con facilidad cambios en el código, editar archivos de manera simultánea y colaborar de forma eficiente en el mismo proyecto desde cualquier ubicación.

Fuente: https://aws.amazon.com/es/what-is/repo/

➢ Entorno de Desarrollo

El entorno de desarrollo es el lugar donde los desarrolladores de software crean y prueban su código antes de que esté listo para ser lanzado al mundo. Es como un taller de artesanía donde se construye y perfecciona la obra maestra antes de exhibirla en una galería. En este entorno, los desarrolladores tienen la libertad de experimentar, cometer errores y aprender sin temor a afectar a los usuarios finales.

En un entorno de desarrollo, los programadores utilizan herramientas como Visual Studio o Eclipse para escribir y depurar su código. También pueden interactuar con una base de datos de prueba para simular el comportamiento del software en un entorno controlado.

Fuente: https://keepcoding.io/blog/entornos-de-desarrollo-entornos-de-produccion/

➢ Entorno de Explotación o Producción

El entorno de producción, por otro lado, es el lugar donde el software ya terminado se ejecuta y atiende a los usuarios reales. Este es el entorno en el que los usuarios experimentan la aplicación tal como fue diseñada para funcionar. Aquí, la estabilidad y el rendimiento son críticos, ya que cualquier problema puede tener un impacto directo en los usuarios finales.

En el entorno de producción, el software se ejecuta en un servidor de producción, que suele ser más robusto y seguro que el servidor de desarrollo. Además, se utiliza una base de datos de producción, que contiene datos reales y actualizados que los usuarios necesitan.

Fuente: https://keepcoding.io/blog/entornos-de-desarrollo-entornos-de-produccion/

➢ Gestión de la configuración. Control de cambios. Mantenimiento de la aplicación.

La gestión de la configuración es un proceso de ingeniería de sistemas que ayuda a las empresas a mantener la calidad del rendimiento y la funcionalidad de un determinado producto, sistema u otro activo informático a lo largo de su ciclo de vida.

Fuente: https://www.ibm.com/es-es/think/topics/configuration-management#:~:text=La%20gesti%C3%B3n%20de%20la%20configuraci%C3%B3n%20es%20un%20proceso%20de%20ingenier%C3%ADa,de%20su%20ciclo%20de%20vida.

➢ Web Services

Un servicio web (o web service) **es una vía de intercomunicación e interoperabilidad entre máquinas conectadas en Red**. En el mundo de Internet se han popularizado enormemente, ya se trate de web services públicos o privados. Generalmente, la interacción se basa en el envío de solicitudes y respuestas entre un cliente y un servidor, que incluyen datos.

Fuente: https://www.arsys.es/blog/web-services-desarrollo

➢ AJAX

AJAX significa JavaScript asíncrono y XML (Asynchronous JavaScript and XML). Es un conjunto de técnicas de desarrollo web que permiten que las aplicaciones web funcionen de forma asíncrona, procesando cualquier solicitud al servidor en segundo plano. Espera, ¿qué es AJAX de nuevo? Vamos a revisar cada término por separado.

Fuente: https://www.hostinger.com/es/tutoriales/que-es-ajax

➢ Desarrollo de aplicaciones multicapa. Estrategias de diseño de aplicaciones Web.

La arquitectura de tres niveles es una arquitectura de aplicación de software bien establecida, que organiza las aplicaciones en tres niveles informáticos lógicos y físicos: el nivel de presentación o interfaz de usuario, el nivel de aplicación, donde se procesan los datos, y el nivel de datos, donde se almacenan y gestionan los datos asociados con la aplicación.

Fuente: https://www.ibm.com/mx-es/think/topics/three-tier-architecture

➢ Aplicaciones basadas en microservicios

Las **aplicaciones basadas en microservicios** son un enfoque de arquitectura de software en el que una aplicación grande y compleja se construye como una colección de **servicios pequeños e independientes** que se comunican entre sí. Es el modelo opuesto a la arquitectura monolítica tradicional.

Características:

- **Independencia:** Cada microservicio se desarrolla, se despliega (instala) y se ejecuta de forma totalmente **autónoma**. Si un servicio falla (por ejemplo, el de pagos), el resto de la aplicación (por ejemplo, el catálogo de productos) puede seguir funcionando.

- **Comunicación:** Los microservicios se comunican entre sí a través de interfaces ligeras, generalmente usando protocolos HTTP/HTTPS con **APIs REST** o sistemas de mensajería (colas).

- **Tecnología Heterogénea:** Cada servicio puede utilizar el **lenguaje de programación** (Java, Python, Node.js) y su propia **base de datos** que mejor se adapte a su función, sin estar limitado por la tecnología del resto de la aplicación.

- **Centrado en el Negocio:** Cada servicio se organiza en torno a una **capacidad empresarial** clara. Por ejemplo, en una tienda online, tendrías un microservicio para "Inventario", otro para "Pedidos", y otro para "Usuarios".

Fuente:
https://www.atlassian.com/es/microservices/microservices-architecture
https://cloud.google.com/learn/what-is-microservices-architecture?hl=es
https://www.redhat.com/es/topics/microservices

➢ **SaaS: Sofware as a Service**

El software como servicio (SaaS) se considera tradicionalmente un modelo de software basado en la nube, el cual ofrece aplicaciones a los usuarios finales a través de un navegador de Internet. Los proveedores de SaaS alojan servicios y aplicaciones para que los clientes puedan acceder a ellos bajo demanda. Con una oferta SaaS, no hay que pensar en cómo se mantiene el servicio o cómo se administra la infraestructura subyacente; solo hay que pensar en cómo se va a utilizar el software. Otro aspecto común del modelo SaaS es el modelo de pago por suscripción o por pago por uso, el cual se diferencia de la compra masiva de todas las funcionalidades. Un ejemplo común de una aplicación SaaS es una aplicación externa de correo electrónico basada en la Web, la cual permite enviar y recibir mensajes sin tener que administrar la incorporación de características ni mantener los servidores y los sistemas operativos en los que se ejecuta la aplicación.

Fuente: https://aws.amazon.com/es/what-is/saas/

➢ **Control de acceso a la aplicación web o los Web Services**

Un servicio web (o web service) es una vía de intercomunicación e interoperabilidad entre máquinas conectadas en Red. En el mundo de Internet se han popularizado enormemente, ya se trate de web services públicos o privados. Generalmente, la interacción se basa en el envío de solicitudes y respuestas entre un cliente y un servidor, que incluyen datos.

Fuente: https://www.arsys.es/blog/web-services-desarrollo

➢ **Validación de entrada de datos a una aplicación Web**

La validación de entradas en programación es un componente esencial para garantizar que una aplicación web funcione de manera segura y sin problemas. Cuando se trabaja con entradas de datos, ya sea desde formularios en una página web o interacciones con una base de datos, la validación adecuada de estos datos es fundamental. En este artículo, explorarás las técnicas de validación de entradas en programación y cómo pueden contribuir a la seguridad y la integridad de una aplicación web.

Fuente: https://keepcoding.io/blog/validacion-de-entradas-en-programacion/

➢ Posicionamiento de una aplicación Web

Posicionamiento Web o SEO: Consiste en aplicar diversas técnicas orientadas a lograr que los buscadores de Internet sitúen determinada página web en una posición y categoría alta (primeras posiciones) dentro de su página de resultados para determinados términos y frases clave de búsqueda.

El posicionamiento web orgánico o natural en Google y demás buscadores es una de las estrategias más efectivas a medio y largo plazo para captar potenciales clientes desde Internet.

Actualmente, los buscadores, son el principal recurso que emplean los usuarios de internet a la hora de obtener información sobre cualquier un producto o servicio en Internet, por lo que representa una importante fuente potencial de tráfico para tu web.

Fuente: https://www.esepestudio.com/noticias/que-es-y-como-se-hace-el-posicionamiento-web-en-buscadores

➢ Historia, situación actual y evolución del diseño de aplicaciones Web

1. Historia y Evolución del Diseño Web (Décadas)

La evolución del diseño web se puede segmentar en distintas etapas marcadas por las limitaciones tecnológicas y el cambio en las prioridades de contenido:

Década	Nombre de la Era	Tecnologías Clave	Características del Diseño
1990 - 1995	Web Estática y de Documentos	HTML básico	Solo texto, fondo blanco, enlaces azules. El objetivo era solo informar (como un folleto digital).
1995 - 2000	Diseño con Tablas y Gráficos	HTML con <tables>, aparición de JavaScript y CSS (aunque con baja adopción).	Se usan tablas invisibles para estructurar el contenido y posicionar imágenes. Diseños a menudo caóticos con GIFs animados y colores brillantes (la "época oscura").
2000 - 2010	Web 2.0 y Flash	CSS (auge), Flash, PHP.	El diseño se separa del contenido. Los sitios se vuelven más dinámicos y visuales (gracias a Flash), pero a menudo lentos. Se populariza la Usabilidad y el Comercio Electrónico.
2010 - 2020	Diseño Responsive (Móvil)	HTML5, CSS3, Media Queries, Frameworks (React, Angular, Vue).	Revolución del Diseño Responsive (adaptable a todos los dispositivos) debido a los smartphones. Predominan el Diseño Plano (Flat Design) y el Minimalismo para optimizar la carga y la UX en móviles.

2. Situación Actual del Diseño de Aplicaciones Web (Hoy)

Actualmente, el diseño de aplicaciones web está regido por la sofisticación técnica y la mentalidad centrada en el usuario y el rendimiento:

- **Prioridad Absoluta: Mobile First (Móvil Primero)**
    - El diseño se conceptualiza primero para pantallas pequeñas y luego se escala a escritorios. Esto es crucial no solo para la UX, sino también para el **SEO** (la indexación de Google prioriza la versión móvil).
- **Enfoque de Desarrollo: SPAs y PWAs**
    - **Single Page Applications (SPA):** Aplicaciones que cargan una sola vez y solo actualizan el contenido necesario (como Gmail o Twitter), ofreciendo una experiencia rápida, similar a una aplicación de escritorio.
    - **Progressive Web Apps (PWA):** Aplicaciones que combinan lo mejor de las apps nativas (funcionamiento *offline*, notificaciones) con la web, mejorando la usabilidad y la velocidad.
- **Filosofía de Diseño: UX/UI**
    - El **Diseño UX (Experiencia de Usuario)** y el **Diseño UI (Interfaz de Usuario)** son disciplinas centrales. El diseño debe ser intuitivo, accesible y estar basado en la investigación del comportamiento del usuario.

3. Tendencias de Evolución y Futuro (2025 en adelante)

Tendencia	**Enfoque y Tecnología**
Inteligencia Artificial (IA) en UX	Uso de IA para la personalización en tiempo real (adaptación del contenido/diseño al usuario), la automatización de testing y la generación de prototipos y código.
Accesibilidad y Diseño Inclusivo	Diseño que garantiza que personas con discapacidades (visuales, motoras, etc.) puedan usar la aplicación. Se prioriza el contraste adecuado, la navegación con teclado y la compatibilidad con lectores de pantalla.
Sostenibilidad Digital (Ecodiseño)	Reducción de la huella de carbono de la web mediante la optimización de código, la minimización de recursos y el uso de hosting ecológico (servidores con energía verde).
Elementos Interactivos y 3D	Integración de Microanimaciones, efectos Parallax, y elementos en Realidad Aumentada (AR) o 3D para ofrecer experiencias más inmersivas y visualmente ricas sin sacrificar la velocidad.
Estilos Visuales	Se consolidan el minimalismo funcional y el uso de tipografías expresivas que comunican personalidad. También hay un resurgimiento del diseño retro (dial-up, 80's) aplicado con técnicas modernas.

Fuentes:

- *https://mimedu.es/blog/la-evolucion-del-diseno-web-de-los-anos-90-a-la-actualidad/*
- *https://bowwe.com/es/blog/historia-del-diseno-web*
- *https://es.wix.com/blog/tendencias-diseno-web*