Estudio Tema 1

DWES

ÁLVARO ALLÉN PERLINES

Contenido

Ejercicio 1: IP, TCP, HTTP, HTTPS

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

Estos cuatro protocolos son los responsables del funcionamiento y de la comunicación entre el cliente y el servidor de una aplicación web.

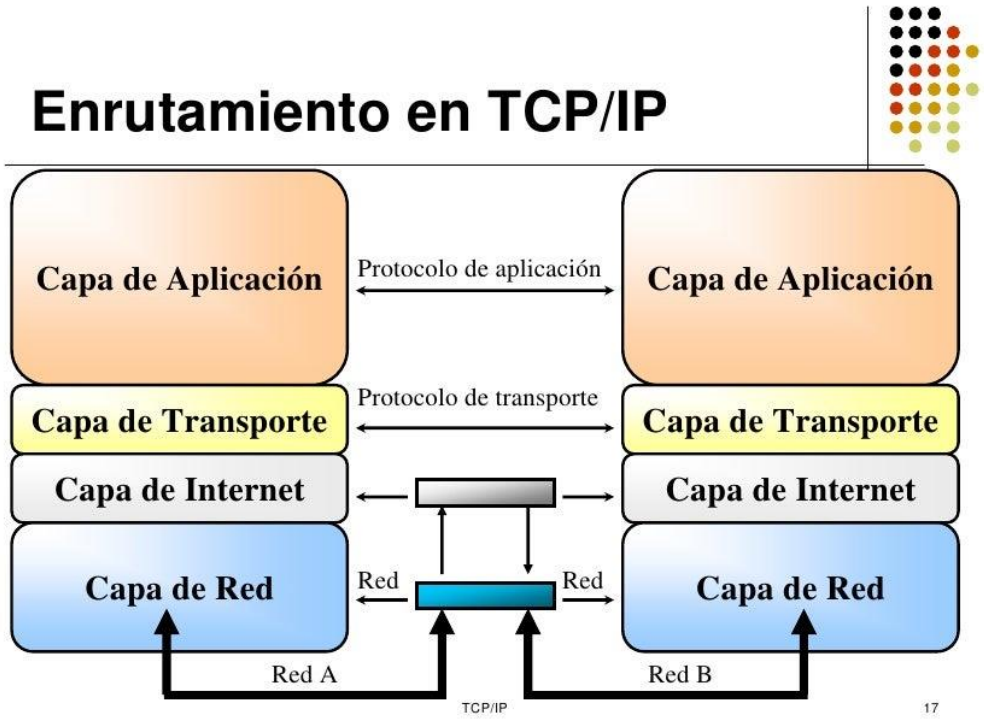
IP: es el protocolo de Internet que dirige los paquetes de datos entre dispositivos en una red, identificando su origen y destino.

TCP: garantiza que los datos lleguen completos y en orden, estableciendo una conexión fiable.

HTTP: permite la comunicación entre navegadores y servidores para acceder a páginas web.

HTTPS: es la versión segura de HTTP, que cifra la comunicación mediante SSL/TLS para proteger la información.

Aunque en el enunciado haya cuatro protocolos, realmente son tres. Eso lo explicaré más adelante. Los dos primeros, el protocolo IP y el protocolo TCP pertenecen al modelo TCP/IP. **El protocolo IP** (Internet Protocol), **perteneciente a la capa de red**, es él que proporciona un servicio de transmisión de datos. Los datos se transmiten enlace por enlace. **El protocolo TCP** (Transmision Control Protocol), **situado en la capa de transporte**, proporciona un servicio de transmisión de datos fiable, dúplex y orientado a conexiones. **El protocolo HTTP** (HyperText Transfer Protocol), **también perteneciente a la capa de aplicación del modelo TCP/IP**, se usa entre cliente y servidor. Por otro lado, **el protocolo HTTPS** es la versión segura de HTTP que, como característica principal, cifra la comunicación entre cliente y servidor para que sea ilegible para el resto.



[Explicación Modelo TCP/IP-Contando Bits]

Ejercicio 2: cliente-servidor

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

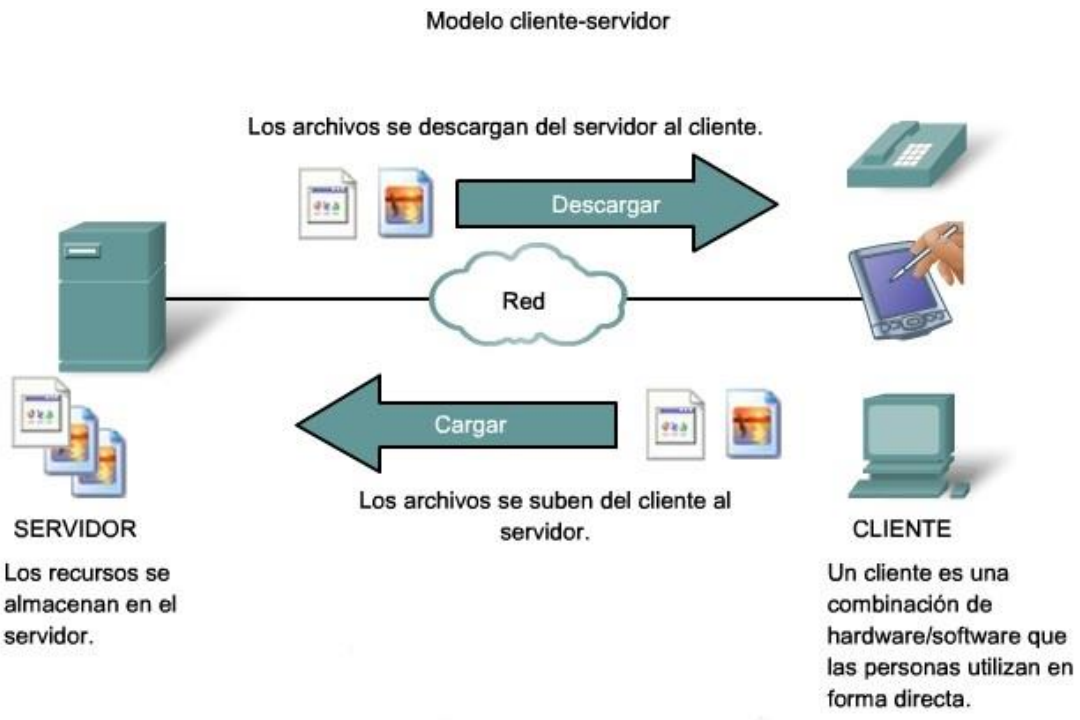
Es una arquitectura de software que permite la interacción entre un cliente y un servidor.

En las aplicaciones web esta comunicación es fundamental para su funcionamiento. Cuando un usuario interactúa con una página web, el navegador actúa como cliente, enviando solicitudes al servidor web. La forma de comunicación entre el cliente y el servidor puede variar de dos formas distintas:

- **Comunicación síncrona:** el cliente envía una petición al servidor y espera a la respuesta antes de continuar. Los protocolos más usados son HTTP o HTTPS.
- **Comunicación asíncrona:** el cliente envía solicitudes sin necesidad de recargar toda la página y el servidor responde, normalmente, con datos como JSON y XML. Esto le da más dinamismo a la página.

A mayores podemos hablar de las páginas con **comunicación en tiempo real** en las que el cliente y el servidor pueden enviar y recibir datos sin esperar solicitudes nuevas.

Es por esto que **el modelo cliente-servidor** es la base de las aplicaciones web, ya que permite la comunicación entre el navegador del usuario y el servidor que procesa las solicitudes y envía las respuestas con la información o los servicios solicitados.



Bibliografía:

- Arquitectura cliente servidor: qué es, tipos y ejemplos | Blog de Arsys | Blog de Arsys
- Arquitectura cliente servidor explicación con ejemplos【Guía】

Ejercicio 3: métodos HTTP/HTTPS

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

- **GET:** recupera la información de un servidor sin modificarla. Sirve para obtener datos de una página web o una API.
- **POSTS:** se emplea para enviar datos al servidor, como formularios o archivos. Este método puede modificar el estado del servidor, como crear nuevos registros o actualizar información existente.
- **PUT:** se utiliza para actualizar un recurso existente o crearlo si no existe Es común en aplicaciones que gestionan bases de datos.
- **DELETE:** elimina el recurso específico en el servidor.
- **HEAD:** es similar a GET pero solo recupera los encabezados de la respuesta, sin incluir el cuerpo, es decir, no descarga el contenido del recurso. Se usa para comprobar si un archivo existe, su tamaño, tipo o fecha de modificación.

Bibliografía:

- Métodos de petición HTTP - HTTP | MDN

Ejercicio 4: URI/URL/URN y relación HTTP/HTTPS

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

URI: es una cadena de caracteres que identifica los recursos (físicos o abstractos) de una red de forma unívoca. La diferencia entre esta URI y una URL es que ésta última hace referencia recursos que pueden variar en el tiempo. Al final, una URI puede estar compuesta por una URL o una URN.

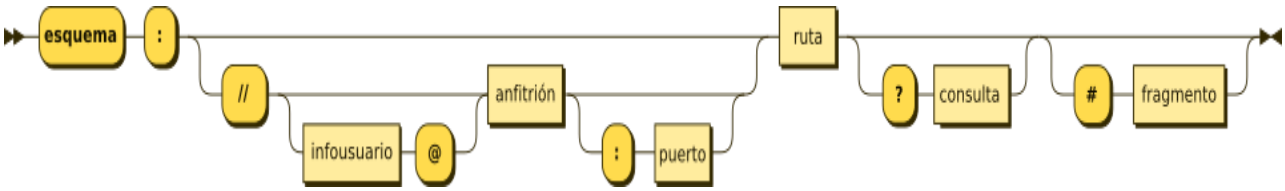
URL: es una secuencia específica de caracteres que identifica y permite localizar y recuperar una información determinada en internet.

URN: nombre de un recurso, único en el mundo e independiente de su localización. Un ejemplo conocido es el código ISBN.

La estructura de un URI está compuesta por:

- **Esquema:** nombre que se refiere a una especificación para asignar identificadores. También puede identificar el protocolo de acceso al recurso.
- **Autoridad:** elemento jerárquico que identifica la autoridad de nombres.
- **Ruta:** información organizada en forma jerárquica, que identifica al recurso en el ámbito del esquema URI y la autoridad de nombres.
- **Consulta:** información con estructura no jerárquica que identifica el recurso en el ámbito del esquema URI y la autoridad de nombres.
- **Fragmento:** permite identificar una parte del recurso principal, o vista de una representación del mismo.

El diagrama que explica su estructura es el siguiente:



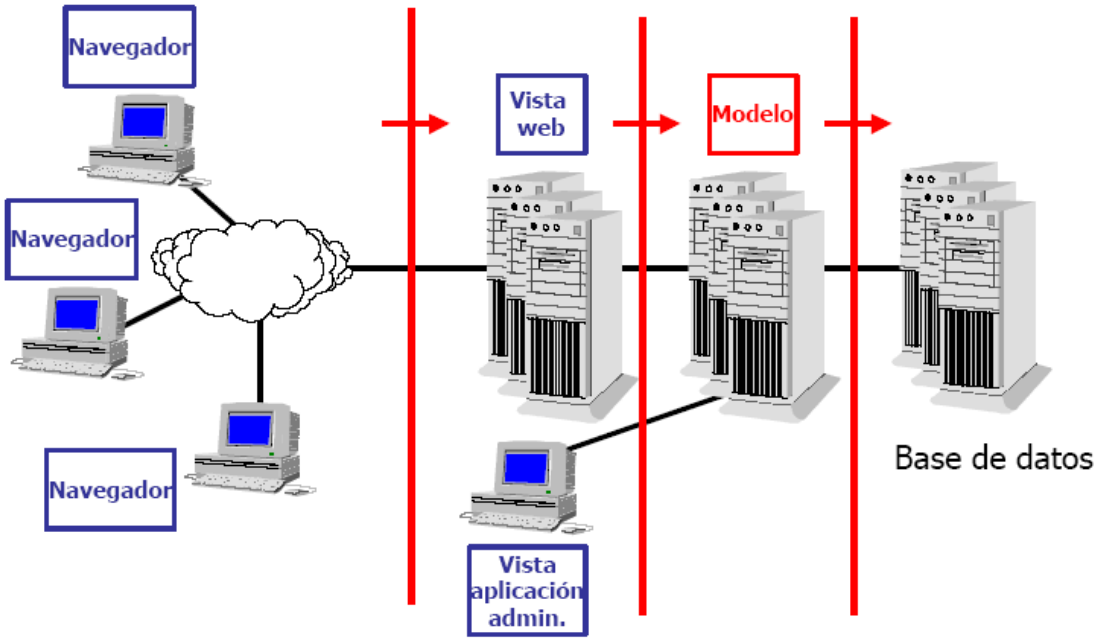
La relación que guarda el URI con los protocolos HTTP y HTTPS es que las URLs son un tipo dentro de las URIs que abarcan muchos más tipos de identificadores.

Bibliografía:

- Identificador de recursos uniforme - Wikipedia, la enciclopedia libre
- URL - Concepto, usos, partes y características
- Definición de URI: URL + URN + URC – idearius

Ejercicio 5: modelo de desarrollo, capas y funcionalidad.

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



Una aplicación multicapa tiene como enfoque un diseño de software que organiza una aplicación en capas independientes, es decir, cada capa tiene una responsabilidad específica. Estas capas son:

- **Capa de presentación:** interfaz de usuario que presenta la información al usuario.
- **Capa de lógica de negocio:** contiene las reglas y operaciones que definen cómo funciona el sistema.
- **Capa de datos:** maneja la consulta de la información comunicándose con la base de datos. Ejecuta consultas, inserciones y actualizaciones.
- **Capa de persistencia:** almacena y gestiona los datos de forma persistente.

Este diseño tiene como características principales la mejora de la modularidad y la facilidad del mantenimiento y de la escalabilidad de la aplicación.

La comunicación entre las capas es unidireccional y jerárquica, es decir, cada capa solo se comunica con al capa inmediatamente inferior o superior.

Bibliografía:

- Arquitectura multicapa - Wikipedia, la enciclopedia libre

Ejercicio 6: front-end/back-end

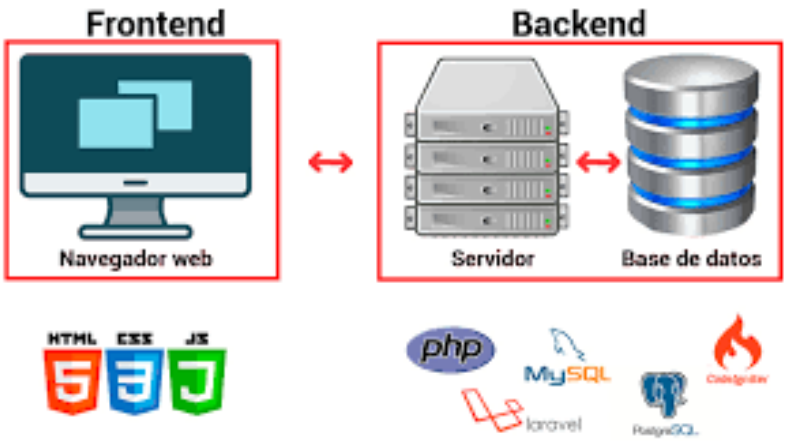
Modelo de división funcional front-end/back-end para aplicaciones web.

Front-end: se refiere a la parte visible y accesible de un sitio web o aplicación, con la que los usuarios interactúan. Incluye el diseño, la estructura y la funcionalidad de la interfaz de usuario. Sus características principales son:

- Se utilizan lenguajes de programación como HTML, CSS o JavaScript para crear y dar estilos.
- Es responsable de la optimización de la experiencia del usuario, asegurando que la navegación sea intuitiva y atractiva.
- Siempre está en constante evolución, adaptándose a las nuevas tendencias.

Back-end: se refiere a la parte de un sistema o aplicación que se encarga de procesar y almacenar los datos. Es la parte invisible para el usuario final, pero es esencial para el funcionamiento de cualquier plataforma digital. Éste se encarga de gestionar la lógica de negocio, la base de datos y la comunicación con el front-end. Sus características principales son:

- Utiliza lenguajes de programación como PHP, Python, Ruby o Java para desarrollar la lógica del sitio.
- Es responsable de la seguridad, la eficacia y la escalabilidad de la aplicación.



Bibliografía:

- Frontend: Qué es, definición, significado y ejemplos ⋆ Sensacionweb.com
- Backend - Qué es, definición, significado y ejemplos ⋆ Sensacionweb.com

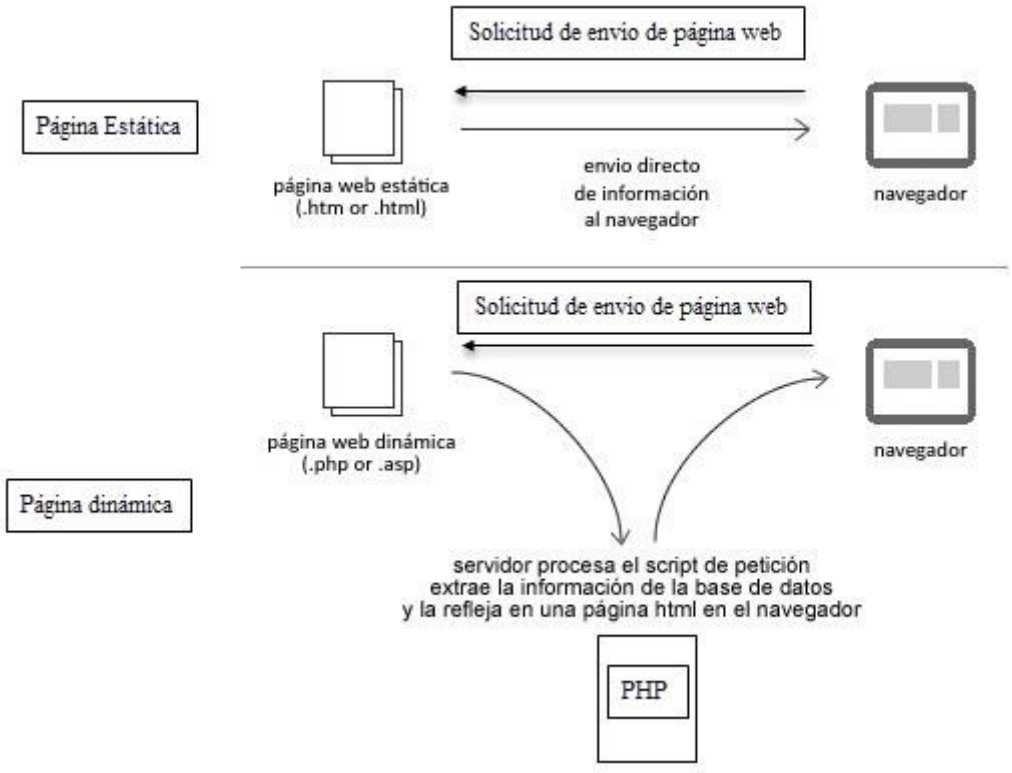
Ejercicio 7: páginas web estáticas/dinámicas/aplicaciones/mashup

Página web estática – página web dinámica – aplicaciones web – mashup.

Página web estática: es un conjunto de páginas web cuyo contenido es el mimo cada vez que los usuarios acceden a ella. Estas páginas se pueden construir con HTML, CSS, JavaScript, pero sin necesidad de scripts ni de lado servidor dado que no hace falta bases de datos.

Página web dinámica: es un documento en línea que cambia su contenido en función de la interacción del usuario, la ubicación, y otros factores. Éstas utilizan lenguajes como PHP o JavaScript junto a bases de datos para generar información en tiempo real y ofrecer experiencias personalizadas.

Aplicación web: es un software que se ejecuta directamente en un navegador web, es decir, es interpretado por él, permitiendo a los usuarios acceder a funcionalidades complejas sin necesidad de instalar programas adicionales. Son utilizadas por empresas para intercambiar información, ofrecer servicios de forma remota y comunicarse con los clientes de manera segura. Gracias a su fácil accesibilidad están a la orden del día.



Bibliografía:

- Qué es una página web estática y cómo crearla + ejemplos
- ¿Qué es una aplicación web? - Explicación de las aplicaciones web - AWS

Ejercicio 8: componentes aplicación web

Componentes de una aplicación web.

Los componentes de una aplicación web se clasifican de dos formas:

- **Componentes del lado servidor.**
  - **Servidor web:** es el responsable de procesar las peticiones HTTP de los usuarios. Éstos son utilizados por los clientes HTTP, que también pueden ser un navegador, para encontrar y enviar solicitudes a los componentes de tu aplicación.
  - **Almacenamiento de datos:** es la parte que se dedica a la persistencia de datos.
    - **Bases de datos:** se utilizan para almacenar datos para un acceso rápido. Normalmente, admiten el almacenamiento de una pequeña cantidad de datos a los que accede regularmente tu aplicación.
  - **Almacenamiento en caché:** es una característica opcional que suele implementarse en las arquitecturas de las aplicaciones web para servir el contenido más rápidamente a los usuarios.
    - **Almacenamiento de datos en caché:** introduce una forma de que tu aplicación acceda fácil y rápidamente a los datos utilizados regularmente y que no cambian con frecuencia.
    - **Almacenamiento en caché de páginas web:** almacena en caché las páginas web. Para las aplicaciones web renderizadas en el lado servidor, el almacenamiento en caché no sirve de mucho, ya que se supone que su contenido es muy dinámico.
- **Componentes del lado cliente.**
  - **Interfaz de usuario:** es el aspecto visual de tu aplicación. Es lo que tus usuarios ven y con lo que interactúan para acceder a tus servicios. Esta interfaz se construye principalmente con tres tecnologías: HTML, CSS y JavaScript. La interfaz de usuario puede ser una aplicación en sí misma con su propio ciclo de vida de desarrollo de software.

Bibliografía:

- ▷ ¿Cuáles son los componentes de una aplicación web? | Actualizado noviembre 2025
- Arquitectura de las aplicaciones web-Wordpress

Ejercicio 9: programas ejecutados y lenguajes usados en lado cliente y lado servidor

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

En el lado cliente, todos los archivos HTML y CSS son interpretados por el navegador. Los procesos usados desde el lado cliente se escriben en JavaScript. Los programas usados desde este lado son los navegadores. Con tener instalado uno ya te sirve (salvo alguna incompatibilidad puntual) para ejecutar cualquier aplicación web.

Programas usados en el lado cliente:

- JavaScript: da interactividad y lógica a la página.
- HTML: es la estructura y el contenido básico de una página web.
- CSS: son los estilos y apariencias de una página web.
- Frameworks JS: para apps dinámicas.
- WebAssembly: ejecuta código de otros lenguajes en el navegador.

En el lado servidor tenemos como lenguajes principales tenemos PHP para realizar peticiones al servidor como autenticación de usuarios, lógica empresarial etc…, y SQL para realizar peticiones a la base de datos, modificarla, etc... Los programas que se ejecutan en el lado servidor son los intérpretes de los lenguajes.

Programas usados en el lado servidor:

- Apache: el servidor web más usado en la actualidad, estable y configurable.
- Nginx: servidor eficiente, ideal para alto tráfico.
- Laravel: framework de PHP.
- Django: framework de Python.
- Spring Boot: framework de Java.
- MySQL/PostgreSQL: para gestión y consulta de base de datos.

Bibliografía:

- Arquitectura cliente servidor: qué es, tipos y ejemplos | Blog de Arsys | Blog de Arsys
- Video del uso de PHP en el lado servidor usando XAMPP
- Para quien tenga tiempo y ganas de aprender HTML y CSS
- Si quieres aprender aun más del lado cliente aquí otro curso de JavaScript
- Curso para iniciarse en Spring Boot (framework de Java)
- Curso para iniciarse en bases de datos (MySQL)

Ejercicio 10: características de lenguajes en lado servidor

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

Los lenguajes de programación más usados en el lado servidor son los siguientes:

- **PHP:** es un lenguaje de código abierto, es decir, cualquiera puede acceder a el y modificarlo, es interpretado y está orientado a objetos. Es débilmente tipado, al declarar las variables no es necesario especificar el tipo, permite la integración de módulos externos para mejorar la aplicación web.
- **Ruby:** bastante parecido a PHP, es un lenguaje orientado a objetos, dinámico y flexible pudiendo modificar la estructura durante la ejecución, es interpretado y contiene una extensa gama de bibliotecas y gemas (paquetes de código reutilizable).
- **Java:** es un lenguaje orientado a objetos, altamente tipado y compilado mediante el JVM.
- **JavaScript:** es un lenguaje interpretado, débilmente tipado y orientado a objetos basado en prototipos. A mayores es versátil, es decir, sirve para lado servidor y para lado cliente, y es asíncrono y basado en eventos.

© W3Techs.com	usage	change since 1 September 2025
1. PHP	73.3%	-0.3%
2. Ruby	6.4%	+0.1%
3. Java	5.4%	+0.1%
4. JavaScript	5.1%	+0.2%
5. ASP.NET	4.8%	

percentages of sites

1. Fecha de captura: 14/10/2025

Bibliografía:

- W3Techs - extensive and reliable web technology surveys

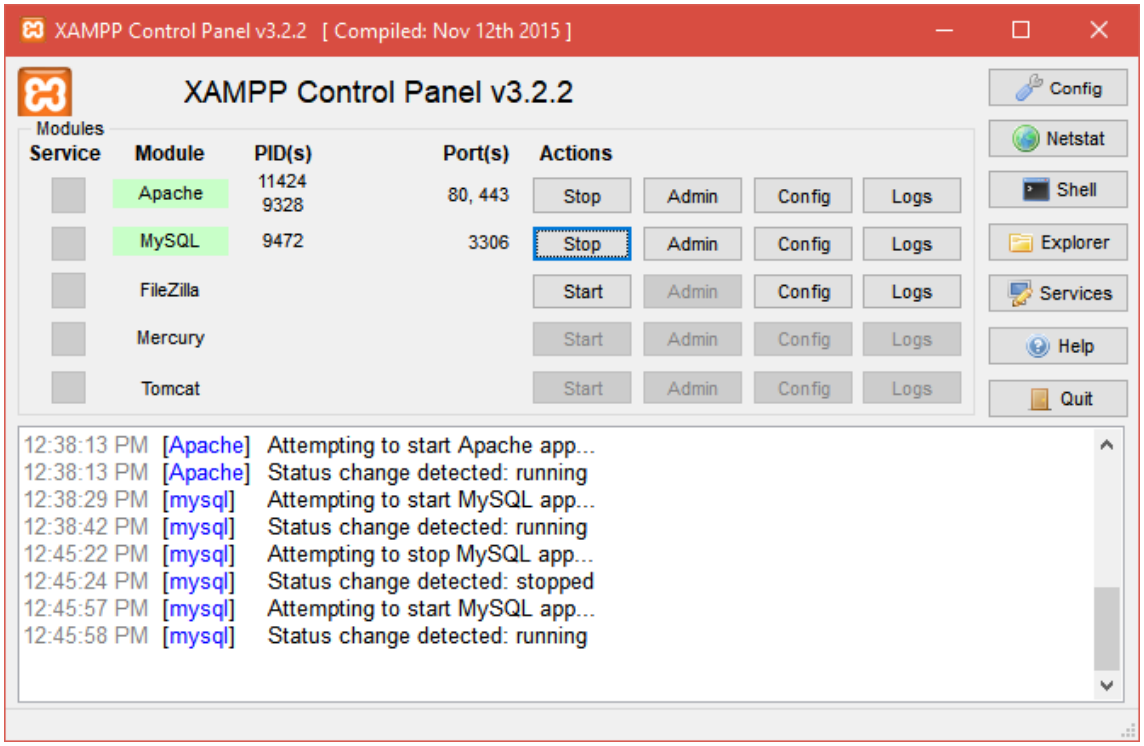
Ejercicio 11: XAMPP

Características y posibilidades de desarrollo de una plataforma XAMPP.

Es un paquete de software libre que incluye Apache, MySQL, PHP y Perl, diseñado para crear un entorno de desarrollo web local. Su principal objetivo es facilitar la instalación de un servidor local en tu equipo, permitiendo desarrollar, probar y administrar aplicaciones web de manera sencilla.

Sus características principales son:

- **Paquete de software completo:** incluye servicio de servidor, bases de datos, y PHP para poder desarrollar una aplicación web.
- **Instalador simplificado.**
- **Funcionalidad offline:** no requiere de acceso a Internet para poder probar las aplicaciones.
- **Multiplataforma:** compatible con varios sistemas operativos.
- **Código abierto:** desarrollado por Apache Friends, su código puede ser revisado y modificado por la comunidad.



Bibliografía:

- ¿Qué es XAMPP? Funciones, instalación y uso detallado en 2024
- Enlace oficial de descarga de XAMPP

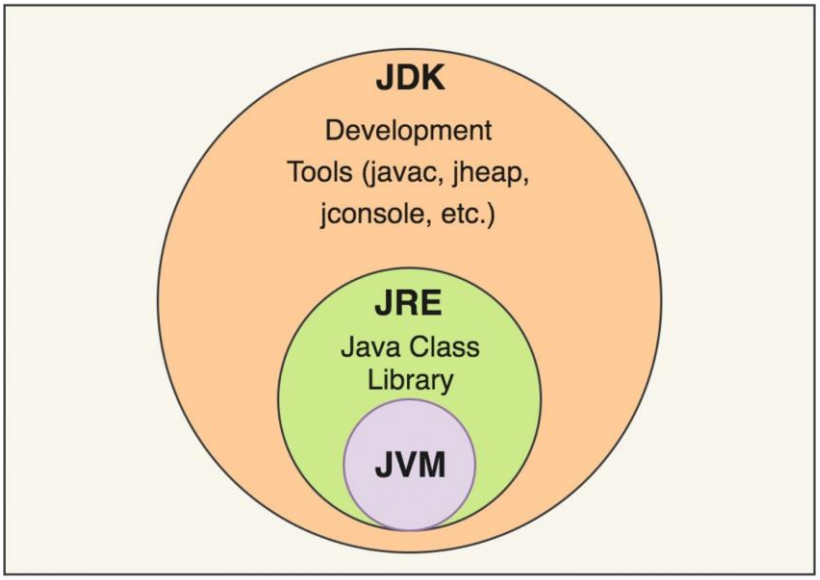
Ejercicio 12: uso del JVM y JDK

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.

Cuando estamos hablando de entornos de desarrollo es fundamental instalar el JDK (Java Development Kit). Sin este paquete no tendríamos posibilidad de crear programas Java. Es decir, si podríamos crearlos ya que solo es necesario un editor de texto plano y los conocimientos básicos del lenguaje. El problema está en que no podríamos ejecutar el programa de ninguna manera ya que este kit de desarrollo contiene el compilador de java, javac. Este compilador transforma el código fuente de archivos escritos en extensión .java a archivos bytecode con extensión .class. En el caso de la máquina virtual de Java (JVM) no es necesaria su instalación porque ya viene incluida en el kit de desarrollo.

En el entorno de explotación, que es donde la aplicación ya está terminada y solo necesita ejecutarse, no se requiere de JDK. Solo bastaría con tener la JVM o el JRE que es el Java Runtime Environment que incluye a mayores bibliotecas necesarias para ejecutar los programas Java.

En resumen, si estás en un entorno de desarrollo necesitas el JDK para poder probar tus programas y ejecutar en caso de tener la versión final y querer probarla. En cambio, para entornos de explotación con tener el JVM o el JRE para ejecutarla es suficiente.



Bibliografía:

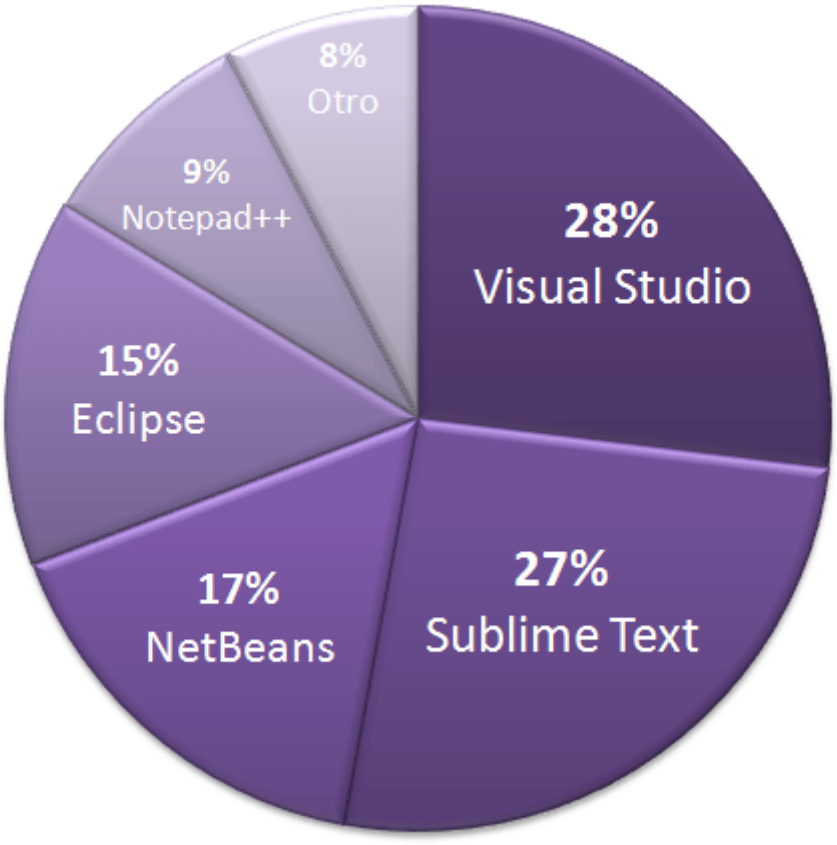
- Entendiendo JVM, JDK y JRE: Componentes Clave de Java

Ejercicio 13: IDEs más usados

IDEs más utilizados (características y grado de implementación actual).

Los IDEs más usados actualmente son:

- **Visual Studio Code:** compatible con muchos lenguajes de programación, con un gran catálogo de extensiones para resolver cualquier tipo de incompatibilidad e implementación de herramientas, muy ligero y modular, gratuito (aunque con versión de pago más completa).
- **Intellij IDEA:** muy potente para Java/Kotlin y ecosistema JVM, autocompletado, muy usado en empresas y proyectos grandes. Igual que VSC tiene versión gratuita y de pago.
- **Eclipse:** entorno de Java, muy usada en universidades para la introducción a la programación y en empresas con software legado, flexible y extensible.
- **NetBeans:** perfecto para el desarrollo web dado su soporte con Java, PHP y C/C++, integración con Maven y JavaFX aunque en comparación con el resto de IDEs ha quedado atrás en innovación.



Bibliografía:

- 22 Entornos de Desarrollo Integrado más Populares 2025

Ejercicio 14: servidores HTTP/HTTPS más usados

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

Los servidores HTTP/HTTPS más usados son:

- **Nginx:** muy ligero y orientado a alto rendimiento. Ideal como reverse proxy, balanceador de carga, microservicios, etc… Configuración basada en bloques y directivas.
- **Apache HTTP server:** arquitectura modular, flexible pero más pesado que Nginx. Soporta casi cualquier modulo, muy robusto y estable. Sigue muy presente en hosting compartido y entornos empresariales tradicionales.
- **Caddy:** orientado a configuración mínima y seguridad por defecto. Sintaxis de configuración muy simple, estilo declarativo.
- **Microsoft IIS:** integrado en Windows Server muy usado en entornos corporativos. Gestión visual vía GUI, muy alineado con el ecosistema Microsoft.

Por lo general, Nginx y Apache lideran el mercado de servidores HTTP. El primero se usa en sitios con más tráfico con respecto al segundo.

Bibliografía:

- Palentino Blog - Servidores web más usados actualmente, características y comparativa.

Ejercicio 15: Apache HTTP vs Tomcat

Apache HTTP vs Apache Tomcat.

El servidor **Apache HTTP** es un servidor web tradicional diseñado para servir páginas web estáticas. Esto incluye archivos como documentos HTML, imágenes, hojas de estilo, scripts y otros recursos que no requieren de procesamiento del lado servidor. También puede manejar un gran volumen de peticiones y es altamente configurable mediante módulos, lo que le da un mayor nivel de seguridad y balanceo de carga. Por lo contrario, como parte negativa está que no puede ejecutar aplicaciones Java por si solo, necesita de una JVM la cual no tiene de serie.

Apache Tomcat, por otro lado, es un servidor de aplicaciones Java. Está diseñado para ejecutar aplicaciones basadas en Java, como servlets y páginas JSP, que requieren ser procesadas por una máquina virtual Java. Aunque puede servir para contenido estático, no es tan eficiente como Apache para desempeñar dicha acción.

Tabla resumen de las diferencias entre Apache HTTP y Apache Tomcat

Característica	Apache HTTP Server	Apache Tomcat
Tipo de servidor	Servidor web	Servidor de aplicaciones Java
Contenido soportado	Estático (HTML, CSS, JS)	Dinámico (Servlets, JSP, JSF)
Lenguaje principal	Cualquiera (no procesa Java)	Java
Uso común	Sitios, archivos estáticos	Aplicaciones web en Java
Rendimiento para contenido ligero	Excelente	Bueno, enfocado en lógica de aplicación
Integración con Java EE	No	Sí
Pueden usarse juntos	Sí	Sí

Bibliografía:

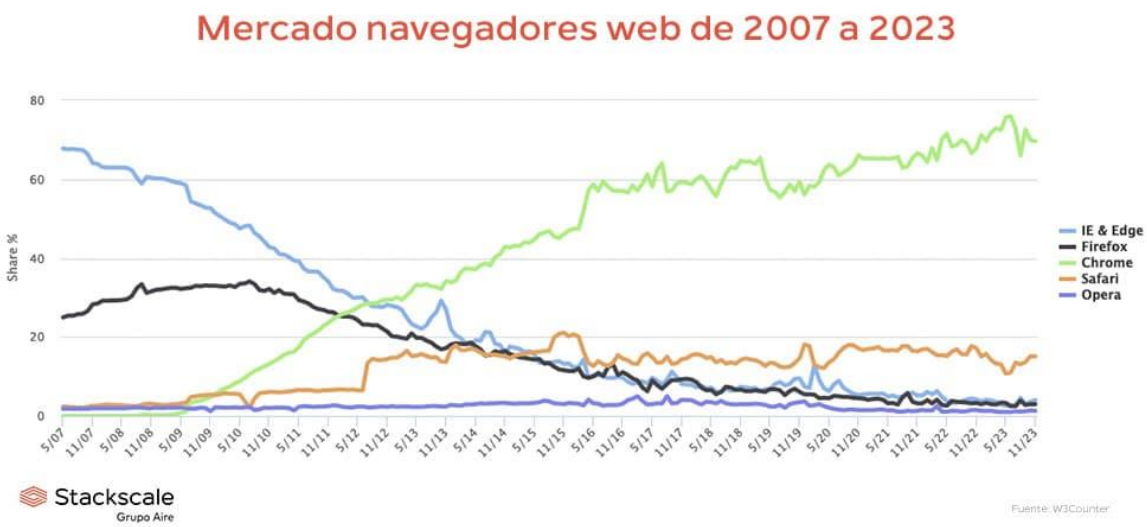
- ▷ Diferencia entre Apache y Tomcat. Ejemplos y Descripción
- Apache Tomcat: qué es, para que sirve y cómo usarlo en la práctica

Ejercicio 16: navegadores más usados

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

Los navegadores HTTP/HTTPS más usados son:

- **Google Chrome:** el navegador rey por antonomasia desde hace dos décadas con un motor Blink, orientado a integración de servicios de Google, rápido, estándar de facto para desarrollo web.
- **Microsoft Edge:** el sucesor del ya retirado Internet Explorer, basado en Chromium/Blink, integrado con Windows y Microsoft 365, muy buen crecimiento en entornos corporativos.
- **Mozilla Firefox:** con motor Gecko, enfoque fuerte de privacidad y código abierto, menor couta que Chrome pero muy usado en desarrollo y usuarios cuya mayor preocupación es la privacidad de sus búsquedas.
- **Opera:** incluye funciones de VPN, gestión de procesador y RAM, y gestión de gasto de ancho de banda pero no ha podido superar a los gigantes quedándose en un producto de nicho.
- **Safari:** con motor WebKit, es el navegador de los dispositivos Apple (iOS y macOS), buen rendimiento energético en portátiles y móviles.



Bibliografía:

- Los navegadores webs más usados en 2025: ¿está el tuyo en la lista?

Ejercicio 17: documentación

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

Los generadores de documentación HTML en PHP son herramientas que permiten crear documentación técnica de forma automática a partir del código fuente. Estas herramientas leen los comentarios especiales que el programador incluye en el código siguiendo un formato estándar llamado PHPDoc. Con esta información generan páginas HTML que describen las clases, funciones, métodos, variables y otros elementos del programa, facilitando la comprensión y el mantenimiento.

Ejemplos de generadores son PHPDocumentor que, mediante el reconocimiento de etiquetas PHPDoc realiza un HTML con toda la documentación necesaria. Otra alternativa menos usada pero aun así famosa es ApiGen que añade una interfaz moderna y fácil de usar para los principiantes en PHP.

Bibliografía:

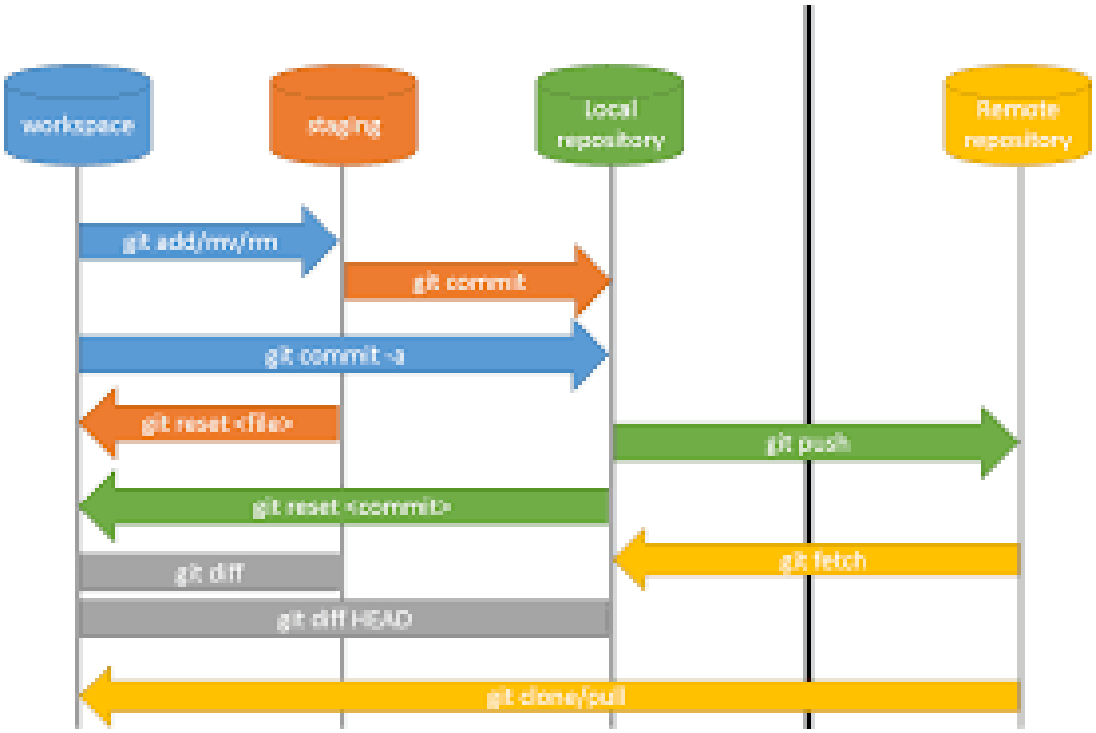
- PHP: Cómo se generan las documentaciones - Manual

Ejercicio 18: repositorios

Repositorios de software – sistemas de control de versiones: GIT, CVS, Subversiones…

Los repositorios de software y los sistemas de control de versiones son herramientas fundamentales en el desarrollo de aplicaciones, ya que permiten gestionar los cambios realizados en el código fuente a lo largo del tiempo. Su función principal es almacenar diferentes versiones del proyecto, registrar quien realizó dichos cambios y facilitar el trabajo colaborativo. Gracias a este sistema de guardado se puede recuperar versiones anteriores, compararlas y evitar la perdida de la información.

El sistema de repositorios más famoso es GIT, creado por Linus Torvalds y se caracteriza por ser un sistema distribuido. Esto de que sea distribuido quiere decir que cada desarrollador tiene una copia completa del repositorio en local, es decir, no es necesario tener conexión a Internet. GIT es la base de GitHub, servicio online en el cual puedes guardar repositorios en una cuenta y compartir código con el resto de usuarios.



Bibliografía:

- ¿Qué es el control de versiones? - Azure DevOps | Microsoft Learn

Ejercicio 19: propuesta de configuración de entorno de desarrollo

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.

Ejercicio 20: propuesta de configuración de entorno de explotación

Propuesta de configuración del entorno de explotación para la asignatura de Desarrollo web del lado servidor en este curso (incluyendo las versiones): xxx-USEE.

Ejercicio 21: estudio sobre CMS y ERP

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

- **CMS – Sistema de gestión de contenidos.**
- **ERP – Sistema de planificación de los recursos empresariales.**

Ejercicio 22: estudio de arquitectura web

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

Glosario

Protocolos TCP/IP. Socket

TCP/IP es una familia de protocolos de comunicación utilizados para conectar sistemas en una red. También se le puede denomina modelo TCP/IP.

Estos protocolos están muy relacionados con el modelo OSI (Open Systems Interconnection). Desglosándolos tenemos el protocolo IP que está presente en la capa de red y proporciona un servicio de transmisión de datos sin conexión. Después tenemos el protocolo TCP situado en la capa de transporte y transmite datos fiables, dúplex y orientados a conexiones.

Bibliografía:

- Protocolos TCP/IP - Documentación de IBM

Protocolo HTTP/HTTPS

El protocolo de transferencia de hipertexto (HTTP) es un conjunto de reglas de comunicación entre cliente y servidor.

Este protocolo está integrado en la capa de aplicación en el modelo de comunicación TCP/IP. Este define el tipo de solicitud y de respuesta. HTTP tiene varios métodos de peticiones, cada uno para un fin en concreto:

- Get y Post: las peticiones más importantes de HTTP. La primera pide una representación del recurso especificado (HTML) y la segunda somete los datos a que sean procesados para el recurso identificado.
- Head, Put, Delete, Connect, etc…: son diferentes peticiones cada uno con una aplicación concreta.

HTTPS añade la seguridad que no tiene HTTP. La comunicación en ningún momento esta cifrada y, por tanto, sería tarea fácil ver la información que se está transmitiendo. Es por eso que HTTP junto con las tecnologías SSL y TLS cifran la información. La información se descifra obteniendo la clave de sesión una vez introducida la URL del servidor web que ahora pasa a ser https://...

Bibliografía

- HTTP y HTTPS: diferencia entre los protocolos de transferencia. AWS

HTML

Es un lenguaje de marcado de hipertexto o, en el argot común, un lenguaje de marcas y su codificación está basado en etiquetas como estas “<body></body>”. Es de tipo interpretado y, su interprete es un navegador.

Es el lenguaje web más importante componiendo todas las páginas web de Internet y, por eso, todos los navegadores están adaptados a él.

Bibliografía

- HTML - Concepto, historia, cómo funciona y etiquetas

XML

Es un lenguaje de marcas extensible que 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.

Este lenguaje no puede realizar operaciones de computación por sí mismo pero, por otra parte, se puede implementar cualquier software o lenguaje de programación para la administración estructurada de datos. Al igual que HTML, este lenguaje funciona con etiquetas XML que se utilizan para definir datos.

Bibliografía:

- ¿Qué es XML? - Explicación del lenguaje de marcado extensible (XML) - AWS

JSON

Es un formato de texto estándar para almacenar y transferir datos estructurados. Se basa en la sintaxis de objetos en JavaScript, pero no está ligado a él. Es soportado en muchos lenguajes como Python, Java y otros.

Su importancia se debe al aumento de velocidad de trabajo que proporcionó en las páginas web ya que las solicitudes AJAX (que funcionan con formato JSON), se ejecutan en segundo plano.

Bibliografía:

- JSON: ¿Qué es este Formato y Cómo Trabajar con Él? » CodigoNautas

Lenguajes de programación embebidos en HTML

En el ámbito de programación y desarrollo de aplicaciones web, embeber es el proceso de insertar código de un lenguaje dentro de otro lenguaje.

Los lenguajes embebidos en el desarrollo web hacen que las páginas sean dinámicas e interactivas. Ejemplos de lenguajes usados:

- **JavaScript:** utilizado para mejorar la interactividad y la validación.
- **PHP:** usado para generar contenido dinámico y procesar formularios en aplicaciones web.
- **Java:** proporciona una amplia gama de funciones y se utiliza para tareas más complejas en sistemas embebidos.
- **AJAX:** una técnica que utiliza JavaScript y XML para intercambiar datos con el servidor sin recargar la página.

ERP

(Enterprise Resource Planning) Es una herramienta indispensable para el éxito de una empresa. Permite integrar en un solo entorno digital los procesos fundamentales de una organización. Áreas como finanzas, gestión de personal, logística o compras pueden operar de forma coordinada. En resumen, facilita la optimización del trabajo aumentando así la actividad económica.

Bibliografía:

- ¿Qué es un ERP? Características, Funciones y Beneficios

CMS

(Content Management System) Es una aplicación de software que se ejecuta en un navegador. Proporciona a los usuarios una interfaz gráfica de usuario que les permite crear y administrar un sitio web sin necesidad de codificarlo desde cero.

Bibliografía:

- ¿Qué es un CMS? Definición, funcionamiento y características

PHP

(Hypertext PreProcesor) Es un lenguaje de scripts generalista y Open Source, especialmente concebido para el desarrollo de aplicaciones web. Se puede integrar fácilmente al HTML mediante la etiqueta reservada <? "código PHP" ?>

Este lenguaje está diseñado para servir del lado del servidor, por lo que se puede hacer todo lo que cualquier programa CGI puede hacer, como recolectar datos de formularios, generar contenido dinámico, o gestionar cookies.

Bibliografía:

- PHP: Introducción - Manual

IDE

(Integrated Development Enviroment) Es un sistema de software para el diseño de aplicaciones que combina herramientas de desarrollador comunes en una sola interfaz gráfica de usuario (GUI). Por lo general, contienen:

- **Editor de código fuente.**
- **Automatización de las compilaciones locales.**
- **Depurador**

Los IDEs permiten que los desarrolladores comienzan a programas aplicaciones nuevas con rapidez, ya que no necesitan establecer ni integrar manualmente varias herramientas como parte del proceso de configuración.

Bibliografía:

- ¿Qué es y para qué sirve un IDE?

Navegador

Es un software de aplicación que permite al usuario ingresar a diferentes páginas web en Internet a través de URLs. Algunos navegadores vienen preinstalados en los sistemas operativos como Edge o Chrome.

Bibliografía:

- Navegador web - Concepto, usos, cómo funciona y ejemplos

Repositorio

Es una ubicación centralizada donde se almacenan y gestionan archivos digitales, como código fuente y documentos. Su función principal que permite a los desarrolladores almacenar múltiples versiones de un proyecto y rastrear cambios a lo largo del tiempo. Esto facilita la colaboración y el trabajo en equipo, ya que los desarrolladores pueden trabajar en diferentes ramas del código y revertir cambios.

Bibliografía:

- ¿Qué son los repositorios? - Explicación sobre los repositorios - AWS

Entorno de desarrollo

Es un espacio de trabajo que permite a los desarrolladores crear una aplicación o realizar cambios en ella sin afectar a la versión real del producto de software. Estos pueden incluir el mantenimiento, la depuración y la aplicación de parches.

No hay que confundirlo con un IDE que es una herramienta del entorno de desarrollo. Tampoco hay que confundirlo con el entorno de explotación que es donde ejecutamos la aplicación una vez terminada y testeada.

Bibliografía:

- Qué es un entorno de desarrollo: definición, usos y tipos

Entorno de explotación o producción

Es un espacio donde alojamos la aplicación una vez realizada y ya está lista para ser usada por los usuarios. Una vez subida la aplicación deberemos de usar. Debe tener las máximas medidas de seguridad y de rendimiento dado que es el producto final y los clientes van a introducir en él datos sensibles.

Bibliografía:

- Entornos de desarrollo, pruebas y explotación

Gestión de la configuración. Control de cambios. Mantenimientos de la aplicación

Es un proceso de ingeniería de sistemas que sirve para establecer la coherencia de los atributos de un producto a lo largo de su vida. En informática la gestión de la configuración de software es un proceso de ingeniería de sistemas que supervisa y controla los cambios en los metadatos de configuración de los sistemas. En el desarrollo la gestión se utiliza habitualmente junto con el control de versiones y la infraestructura de CI y CD.

Su importancia reside en la solidez y estabilidad que dan a los sistemas gestionando y supervisando automáticamente las actualizaciones de los datos de configuración.

Bibliografía:

- ¿Qué es la gestión de la configuración? | Atlassian

Web Services

Es un sistema de software designado para soportar la interacción interoperativa de máquina a máquina a través de una red. Éste realiza una tarea específica o un conjunto de tareas, y se describe mediante una descripción de servicio en una notación XML estándar denominada Web Services Description Language. La descripción de servicio proporciona todos los detalles necesarios para interactuar con el servicio, incluidos los formatos de mensaje, los protocolos de transporte y la ubicación.

Bibliografía:

- ¿Qué es un servicio web? - Documentación de IBM

AJAX

(Asyncronous JavaScript and XML) Es una combinación de tecnologías de desarrollo de aplicaciones web que hacen que las aplicaciones respondan mejor a la interacción del usuario. Por ejemplo, cada vez que un usuario hace clic en botones o en casillas de verificación, el navegador intercambia datos con el servidor remoto. El intercambio de datos puede provocar que las páginas se vuelvan a cargar. AJAX realiza este intercambio en segundo plano provocando que solo se carguen las partes necesarias de la página.

Los ejemplos más claros de casos prácticos son, por ejemplo, el autocompletar en la barra de búsqueda de los navegadores o la verificación de formularios cuando los envían los usuarios.

Bibliografía:

- ¿Qué es AJAX? - Explicación de JavaScript asíncrono y XML - AWS

Desarrollo de aplicaciones multicapa. Estrategias de diseño de aplicaciones web.

La arquitectura de software multicapa es un enfoque estructurado que divide una aplicación en capas independientes, como presentación, lógica de negocio, acceso a datos y almacenamiento. Esto facilita el mantenimiento, la escalabilidad y la reutilización del código, al tiempo que promueve una organización clara del sistema.

En el diseño de aplicaciones web rentables, es fundamental seguir ciertas estrategias que pueden ayudar:

- **Investigación de usuarios:** comprender quiénes son los usuarios potenciales y sus necesidades y expectativas.
- **Definición del objetivo:** establecer objetivos específicos para guiar el diseño de la aplicación.
- **Wireframes y prototipos:** crear esquemas visuales y prototipos interactivos para visualizar la estructura y el flujo de la página.
- **Diseño visual:** utilizar una paleta de colores adecuada, tipografías legibles y elementos gráficos coherentes.
- **Navegación intuitiva:** diseñar una navegación clara y accesible para facilitar la búsqueda de información por parte del usuario.
- **Optimización para dispositivos móviles:** asegurar que la aplicación sea responsiva y funcioné de información por parte del usuario.

Bibliografía:

- Codideep
- Aplicaciones Web - 10 estrategias de diseño y desarrollo

Aplicaciones basadas en microservicios

Los microservicios son una forma de arquitectura de software donde una aplicación se divide en un conjunto de servicios pequeños y autónomos, es decir: cada microservicio se puede implementar y ejecutar de forma independiente. Este tipo de arquitectura ganó popularidad frente a la arquitectura monolítica para el diseño de aplicaciones, pues en el enfoque tradicional todos los elementos estaban contenidos en una sola aplicación. Estos microservicios ofrecen un mayor nivel de flexibilidad y escalabilidad, además de tener mayores niveles de automatización de los centros de datos.

SaaS: Software as a Service

Es una aplicación de correo electrónico basada en la web, como Gmail o Outlook. Estas plataformas permiten a los usuarios enviar y recibir correos electrónicos sin necesidad de instalar software localmente ni gestionar servidores o sistemas operativos subyacentes.

En el modelo SaaS, el proveedor aloja la aplicación y los datos en sus propios servidores o en la nube, ofreciendo acceso a los usuarios a través de un navegador web. Esto elimina la necesidad de preocuparse por el mantenimiento, actualizaciones o infraestructura, ya que todo es gestionado por el proveedor.

Bibliografía:

- ¿Qué es el SaaS? - Explicación del software como servicio - AWS

Control de acceso a la aplicación web o los Web Services

Es un mecanismo popular para exigir la autorización en las aplicaciones. Cuando una organización usa RBAC, el desarrollador de una aplicación define los roles para la aplicación. Después, el administrador puede asignar roles a usuarios y grupos diferentes para controlar quien tiene acceso al contenido y la funcionalidad de la aplicación.

Bibliografía:

- Uso del control de acceso basado en roles para las aplicaciones - Microsoft Entra External ID | Microsoft Learn

Validación de entrada de datos a una aplicación Web

La validación de datos es un proceso que asegura la entrega de datos limpios y claros a los programas, aplicaciones y servicios que lo utilizan. Comprueba la integridad y validez de los datos que se están introduciendo en diferentes softwares y sus componentes. La validación de los datos garantiza que los datos cumplen con los requisitos y los parámetros de calidad.

También es conocida como la validación de entrada y es parte esencial del procesamiento de datos. No hay que confundirla con la verificación de datos.

Bibliografía:

- Validación de Datos: Definición, elementos y métodos

Posicionamiento de una aplicación Web

Es el conjunto de acciones que se realizan para situar páginas web en la parte de arriba de la lista de resultados de los buscadores.

Tipos de posicionamiento web:

- **Posicionamiento orgánico:** también llamado posicionamiento natural o SEO es una serie de estrategias, técnicas y tácticas utilizadas para aumentar la cantidad de visitantes a un sitio web y que busca posicionar en lo más alto de la página de resultados de los buscadores.
- **Posicionamiento PPC:** consiste en realizar una estrategia de campañas de anuncios pagados en Googles Ads o cualquier otra plataforma de marketing como Bing Ands, Facebook Ads, etc…

Bibliografía:

- Qué es Posicionamiento web - Definición, significado y para qué sirve

Historia, situación actual y evolución del diseño de aplicaciones Web

El primer servidor web fue puesto en línea en el mes de agosto de 1991 y se ejecutaba con el navegador de la época Mosaic el cual era gratuito. Estas primeras páginas eran muy básicas con poca decoración, mucho texto y pocas imágenes debido a la sobrecarga que suponían en aquella época. De aquellas el leguaje HTML estaba dedicado más para mostrar información que para entretener.

Por aquellos años el desarrollo y la evolución de la programación web era lenta y caótica. Es por ello que se creó W3C para poder controlar la codificación. Esta comunidad es responsable de organizar las normas, directrices y codificación para hacer una web de máxima calidad.

A partir de 1992 tirando a 1994 apareció el HTML 2 el cual era más rápido y unos diseñadores web con más opciones de HTML y más capacidades. El código cuenta con una mayor de gráficos y se gana en complejidad. Ya empiezan a aparecer iconos para sustituir ciertas partes del texto, botones, listados y menús.

Llega 1995 y con él HTML 3. Ahora se tienen más posibilidades a la hora de diseñar una web, como el uso de tablas y hojas con estilos CSS. Justo este año es en el que nace Internet Explorer, navegador creado por Microsoft que iba a marcar una época de hegemonía absoluta hasta su desaparición. Aparece el GIF como formato de imagen animada. Como última aparición está JavaScript el cual venía a resolver ciertas limitaciones del lenguaje HTML. Entro en escena el dinamismo actual de las páginas web.

Bibliografía:

- Historia del Diseño Web. La historia del diseño web a lo largo de los años

Filosofías de desarrollo del software