DESARROLLO WEB ENTORNO SERVIDOR

Estudio teórico sobre el desarrollo de aplicaciones web del lado servidor

5 DE NOVIEMBRE DE 2025

ALEJANDRO DE LA HUERGA FERNÁNDEZ
Alejandro.huefer@educa.jcyl.es

Contenido

INTRODUCCIÓN

En el siguiente documento se va a llevar a cabo un estudio de las preguntas propuestas por el profesor en la asignatura de Desarrollo de aplicaciones Web Entorno Servidor.

Las preguntas serán respondidas con la siguiente estructura para una mejor expresión de la respuesta, lo cual las hará más sencillas a la hora de leer y de entender por parte del lector.

Estructura de las respuestas:

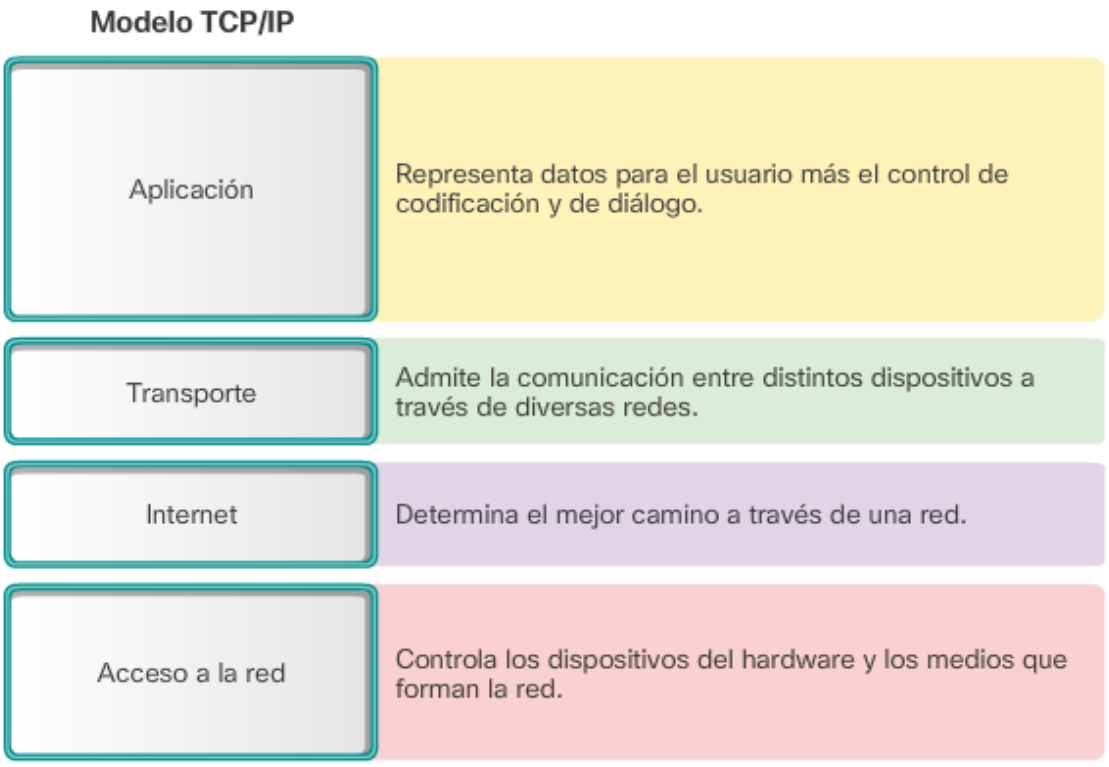
- Respuesta corta – resumen – titular.
- Respuesta texto largo.
- Imagen - modelo – mapa conceptual como respuesta.
- Enlaces externos (URL, video, tutorial, curso...).
- Referencia a las fuentes utilizadas.

Al final de cada pregunta se aportaran los recursos necesarios para simplificar el entendimiento de la pregunta, así como todos los recursos utilizados para formular la respuesta de esta.

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

Conjunto de reglas y estandarizaciones que establecen como los dispositivos y las redes se comunican entre sí y son capaces de intercambiar datos entre sí.

Para entender mejor los protocolos TCP/IP vamos a hacerlo mediante la siguiente imagen:



Esta imagen muestra las diferentes capas del protocolo TCP/IP.

El protocolo TCP/IP define cuidadosamente como se mueve la información desde el remitente hasta el destinatario, estos protocolos reciben los datos de la aplicación, los dividen en partes más pequeñas llamadas *paquetes*, añaden una dirección de destino y, a continuación, pasan los paquetes a la siguiente capa de protocolo, la capa de red de Internet.

Protocolo IP (Internet Protocol): Es un protocolo de comunicación el cual se encuentra dentro de la capa de Red.

Protocolo TCP (Transmission Control Protocol): Este protocolo garantiza que los datos lleguen a su destino.

Protocolo HTTP y HTTPS:

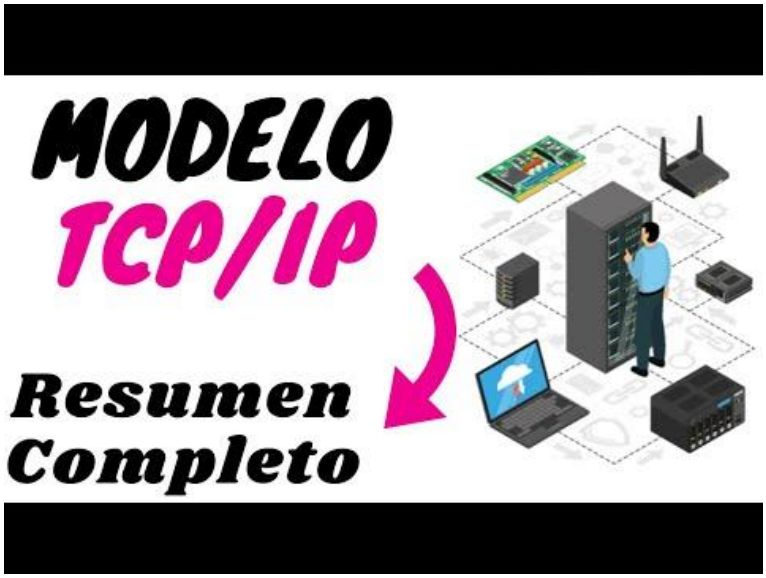
Ambos protocolos pertenecen a la capa de aplicación y son los encargados de enviar datos entre un navegador web y un sitio web.

El protocolo HTTPS es la versión segura del protocolo HTTP ya que este está encriptado para así poder proteger la información.

HTTPS utiliza el protocolo TSL para encriptar la información.

Recursos y enlaces de interés:

Video explicativo del protocolo TCP/IP:



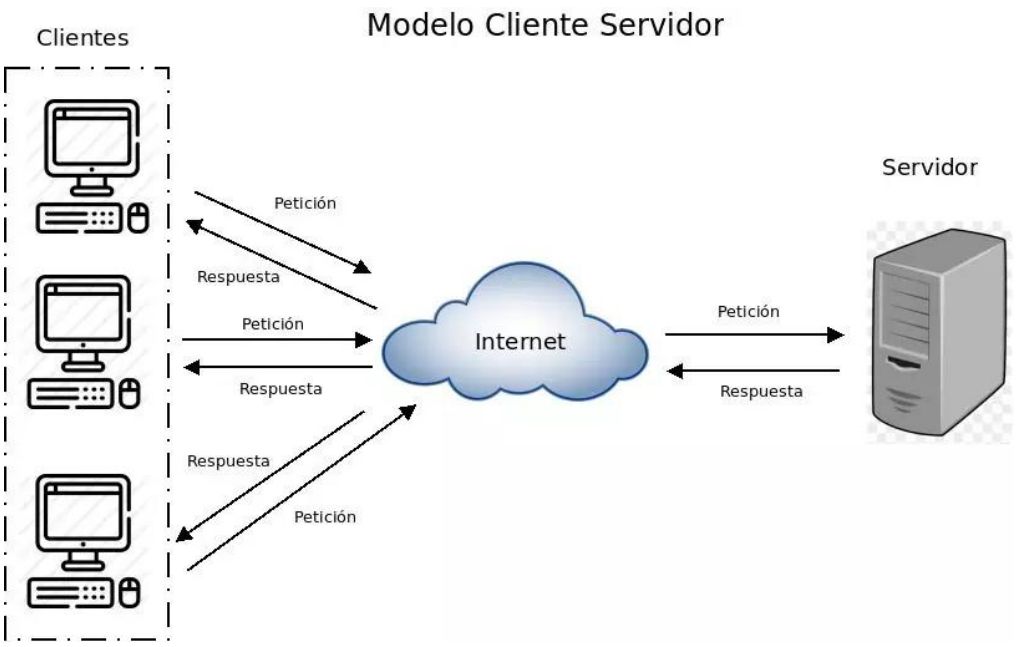
Página web de IBM (Explicación concisa con gráficos):

[Página Web IBM protocolos]

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

El modelo de comunicaciones cliente – servidor se basa en el lanzamiento de una petición por parte del cliente hacia el servidor el cual esta escuchando por un determinado puerto, el servidor al recibir la petición manda una respuesta con el recurso que el cliente ha pedido.

Esta petición del cliente al servidor se hace mediante una serie de protocolos como HTTP o FTP (Dependiendo del tipo de servicio).



En esta imagen podemos ver un diagrama claro de la comunicación entre clientes y servidor.

Dentro de este tipo de modelo podemos distinguir varias partes:

1. **Cliente:** Navegador web que hace una petición.
2. **Servidor:** Servidor web el cual procesa la petición y manda el recurso pedido.
3. **Protocolos:** Lenguaje mediante el cual se comunican ambas partes.
4. **Red:** Conexión entre varios ordenadores.

Recursos y enlaces de interés:

Video explicativo del modelo cliente- servidor (Fazt):

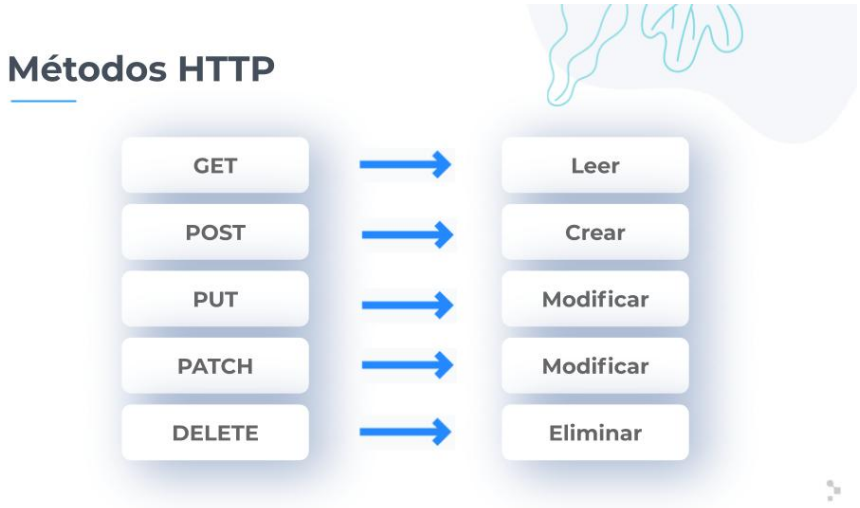
Articulo de arsys.es sobre el modelo cliente – servidor:

Articulo arquitectura cliente - servidor arsys

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

Los métodos de petición HTTP y HTTPS son las instrucciones que el cliente le manda al servidor indicándole la acción que debe realizar con un recurso determinado.

En resumen, estas peticiones le dicen al servidor si debe recibir, enviar, actualizar o borrar algo.



En esta imagen podemos ver los diferentes tipos de métodos y la instrucción que mandan al servidor cada una de ellas.

Definición de métodos:

HEAD: El método “Head” pide una respuesta al servidor, solamente recupera los encabezados para tener que evitar descargar el recurso completo.

GET: Método que se utiliza para pedir recursos al servidor sin llegar a modificarlos, por ejemplo, visitas una página web de un periódico tu navegador usa “GET” para traer los textos, imágenes y videos que componen la página.

POST: Somete los datos para que sean procesados por el recurso identificado, por ejemplo, al regístrate en una página web, los datos personales que ingresas, se envían al servidor con este método para que los procese.

PUT: Se utiliza para actualizar un recurso ya existente o para crearlo en caso de que todavía no exista, este se suele usar para actualizar datos de una base de datos.

DELETE: Se utiliza para borrar un recurso del servidor, por ejemplo, si decides eliminar un artículo de tu página web este recurso lo eliminara de forma permanente.

CONNECT: Inicia la comunicación entre cliente y el servidor , también se utiliza para crear una comunicación de túnel.

Recursos y enlaces de interés:

Video explicativo del modelo cliente- servidor (TodoCode):

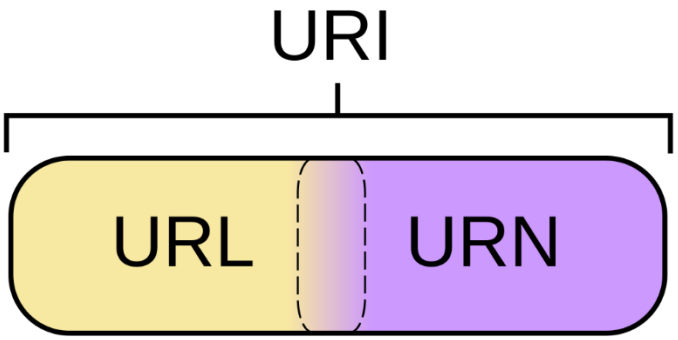
Video breve del canal TodoCode nos explica con detalle los diferentes métodos de manera muy visual y utilizando tablas para su explicación.

Articulo de Mozilla:

Articulo de mozilla sobre metodos de petición

Mozilla nos presenta un análisis exhaustivo de todos los métodos de petición saliendo de los mas comunes y cubriendo todos los métodos existentes.

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

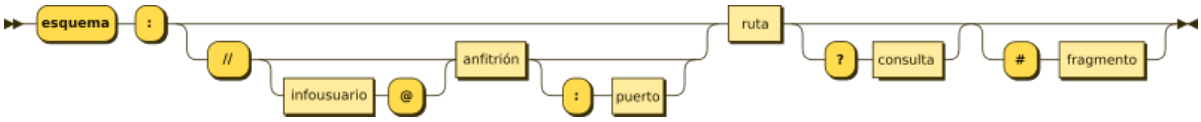


URI (Identificador de Recursos Uniforme): Cadena de caracteres que se refiere a un recurso, se utiliza para acceder a un recurso físico o abstracto en internet. Está compuesto por URL y URN.

URL: Es el identificador del servidor, indica donde se encuentra un recurso.

URN: Identifica el recurso que le vamos a pedir al servidor.

Estructura del URI:



- **Scheme:** protocolo (HTTP, HTTPS, FTP,...) que se va a usar en la petición.
- **Userinfo:** Información del usuario.
- **Host:** IP o dominio del servidor.
- **Port:** Puerto de la conexión.
- **Path:** Dirección del recurso dentro del servidor.
- **Query**
- **Fragment**

Relación con los protocolos HTTP y HTTPS:

La relación del URI con ambos protocolos se debe a que ambos se utilizan para obtener recursos de un servidor web.

La URL especifica la ubicación del recurso , incluyendo el protocolo HTTP o HTTPS para poder acceder a el.

Cuando utilizas una URL en una aplicación web, el navegador o la aplicación pueden usar métodos HTTP (como GET para obtener datos, POST para enviar datos, etc.) para interactuar con el servidor en esa URL y manipular el recurso.

Recursos y enlaces de interés:

Articulo de la página web de IONOS:

Artículo de IONOS sobre el URI

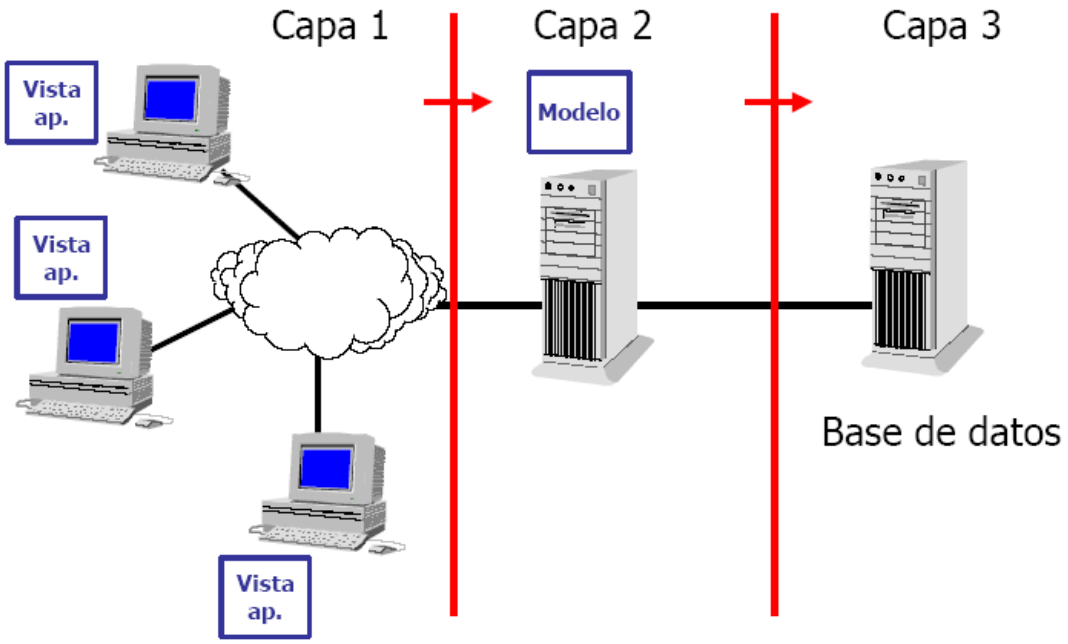
En este articulo del blog de IONOS podemos ver una explicación detallada sobre el concepto de URI.

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

La programación por capas es un modelo de desarrollo de software en el que el objetivo es la separación (desacoplamiento) de las partes que componen un sistema software o también una arquitectura cliente – servidor.

El modelo multicapas se divide en 3 capas diferentes:

- **Capa de presentación:** Es la capa encargada de interactuar con el usuario (autenticado o no) mediante una interfaz de usuario. Esta capa se comunica con la capa de negocio solamente.
- **Capa de negocio:** En esta capa se encuentran los programas que son ejecutados, se reciben las peticiones que lanza el usuario y se envían las respuestas tras procesar la petición. Esta capa se comunica con la capa de presentación para recibir las peticiones y presentar las respuestas y con la capa de datos para solicitar al gestor de base de datos los datos que el almacena.
- **Capa de datos:** Esta capa es la encargada de gestionar el almacenamiento de los datos, generalmente estamos hablando en esta capa de un sistema gestor de bases de datos relacionales.



En esta imagen podemos ver un esquema de la estructura y la comunicación entre diferentes capas de una aplicación multicapa.

Recursos y enlaces de interés:

Video corto explicativo del modelo multi-capas (TechTalks):

Video corto que nos da una visión general del modelo mutli-capas.

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

El modelo de división funcional en aplicaciones web, divide en dos partes las aplicaciones web, Front-end y Back-end.

Front-end: Es el conjunto de páginas que ven la gran mayoría de los usuarios (autenticado o no), en esta parte se incluyen partes como la visualización del sitio web, la interfaz de usuario o el diseño web.

Back-end: Es un conjunto de páginas dinámicas que utilizan las personas que producen el contenido y las administradoras de la página web.

Es donde se encuentra la lógica del servidor, se combina la información encontrada y se transforma.

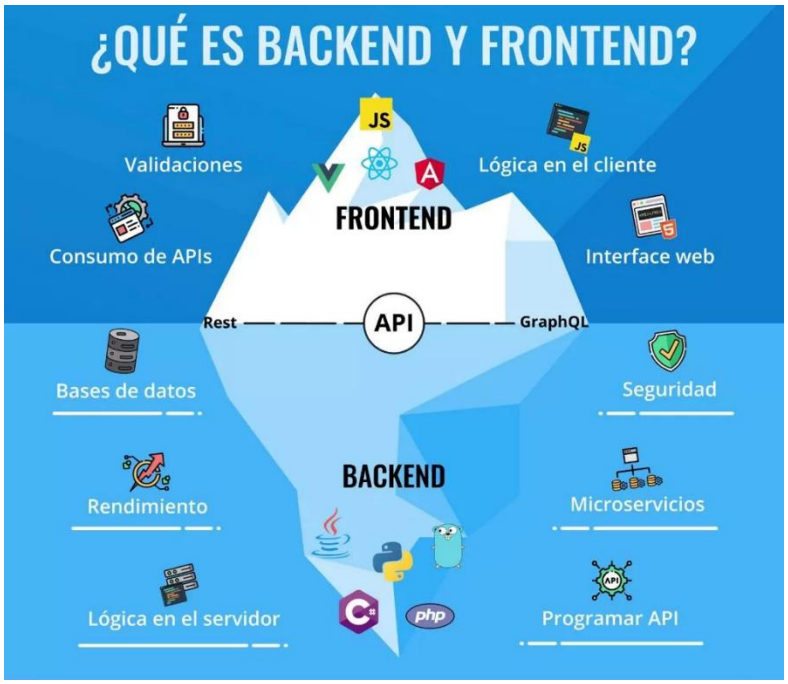


Imagen que nos muestra las diferentes tecnologías, lenguajes y frameworks que se utilizan tanto en el front-end como en el back-end.

Recursos y enlaces de interés:

Articulo sobre el Front-end y el Back-end de Pingback:

Articulo Front-end y Back-end

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

Dentro del mundo del desarrollo web podemos diferenciar varios tipos de páginas web o aplicaciones web, organizando cada una de ellas debido a sus características.

Página web estática:

Se conoce como página web estática aquella que es meramente informativa , la cual esta compuesta únicamente con archivos .html individuales, este tipo de página web muestran la misma información de manera permanente sin cambiarla con el paso del tiempo.

Si quisiéramos modificar la información mostrada tendríamos que modificar el código fuente del archivo .html

Se puede dar el caso de formularios de contacto o suscripciones en este tipo de páginas web.

Página web dinámica:

Una página web dinámica es una página web generada bajo demanda, es decir según lo que el usuario necesite en cada momento mostrara una información u otra.

Las páginas web dinámicas pueden gestionar la información de una base de datos.

La información o contenido que muestran puede ser gestionado a través de un CMS.

Aplicación web:

Una aplicación web, es una aplicación web dinámica, pero en este caso tiene la implementación del control de acceso, lo cual nos permite tener un área de acceso a administradores a un panel de control.

Las aplicaciones web normalmente están diseñadas para realizar algunas funcionalidades más complejas, mientras que la página web dinámica simplemente muestra información que se actualiza automáticamente.

Mashup o aplicación web híbrida:

Es una aplicación la cual combina diferentes tipos de tecnologías o servicios de otras páginas web, normalmente la integración de datos se hace mediante diferentes API' s abiertas.

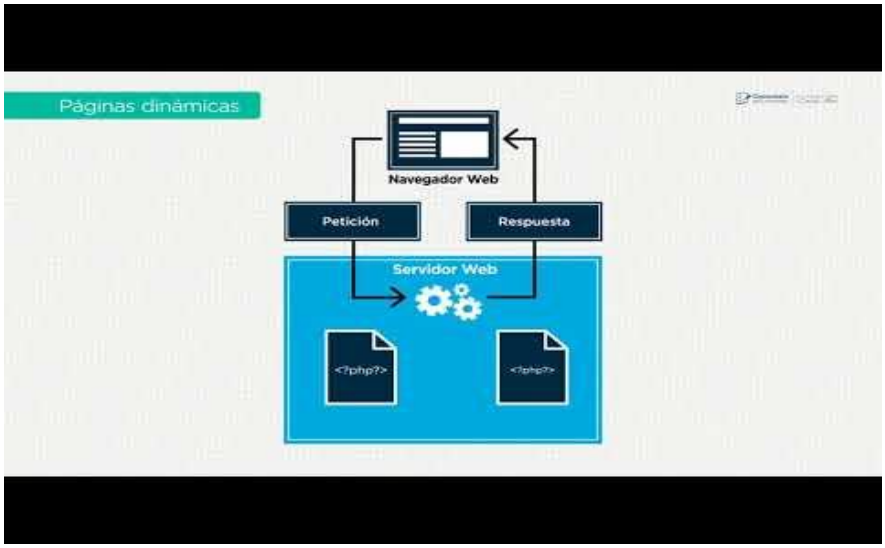
Un ejemplo claro de Mashup puede ser **Google Maps**.

Recursos y enlaces de interés:

Artículo sobre las diferencias entre las diferentes páginas web de OpenWebinars:

Árticulo de OpenWebinar sobre las diferencias entre páginas web.

Video breve sobre las diferencias entre página web dinámica y estática (Network Business School):



8. Componentes de una aplicación web.

Una aplicación web esta compuesta por diferentes componentes que hacen posible su funcionamiento de manera correcta y eficiente.

Componentes:

- **Servidor web:** Su función es recibir las peticiones que le manda el cliente (navegador) y enviarles el recurso o página que solicitan , el servidor web conoce que procedimiento seguir para mostrar la página web o que módulo es el encargado de ejecutar el código.

- **Módulo encargado de ejecutar el código:** Es el encargado de interpretar el código escrito en cualquier lenguaje de programación y generar la página web, el módulo esta integrado con el servidor web y dependerá del lenguaje de programación que utilicemos.

- **Sistema gestor de base de datos:** Este componente o módulo no es estrictamente necesario, pero normalmente se utiliza en todas las aplicaciones web que necesitan grandes cantidades de datos.

- **Lenguajes de programación (Ficheros):** Los utilizaremos para desarrollar nuestras aplicaciones web.

- **Aplicación:** Herramienta la cual sirve para que los usuarios la utilicen para acceder al servidor web.

Recursos y enlaces de interés:

Artículo de el Puig sobre los componentes de una aplicación web:

Artículo sobre los componentes de una aplicación web

Este articulo explica de manera detallada los diferentes componentes de una aplicación web, ha servido como fuente de información para formular la res puesta a la pregunta.

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

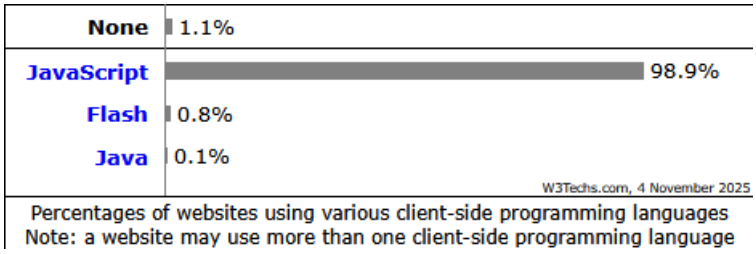
El programa que se ejecuta del lado del cliente es el navegador web, dependiendo del lenguaje utilizado del lado del cliente harán un uso mas exhaustivo o no del navegador.

Del lado del servidor podemos incluir sistemas gestores de bases de datos o interpretes del código escrito dependiendo del lenguaje de programación utilizado.

Hay que tener en cuenta tanto en el lado del cliente y del servidor el uso de frameworks o librerías los cuales nos permiten simplificar el desarrollo de nuestra aplicación.

Lenguajes del lado del cliente:

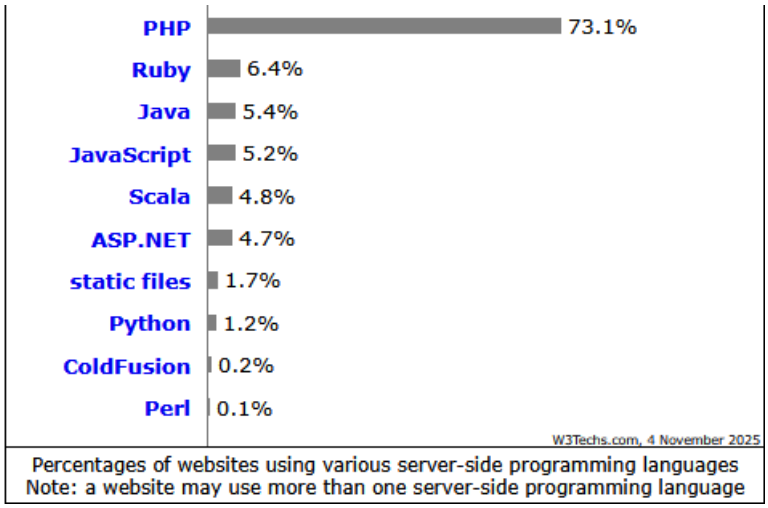
- JavaScript
- Java
- TypeScript
- Frameworks como React,Vue, Angular,...



En esta imagen se pueden ver como JavaScript domina el lado del cliente como lenguaje de programación.

Lenguajes del lado del servidor:

- JavaScript.
- PHP.
- Java.
- Ruby.
- Python.
- PL / SQL.
- Frameworks como Laravel (PHP) o Symfony (PHP), Spring (Java).



En esta imagen se puede ver como PHP domina como lenguaje de progrmación del lado del servidor.

Recursos y enlaces de interés:

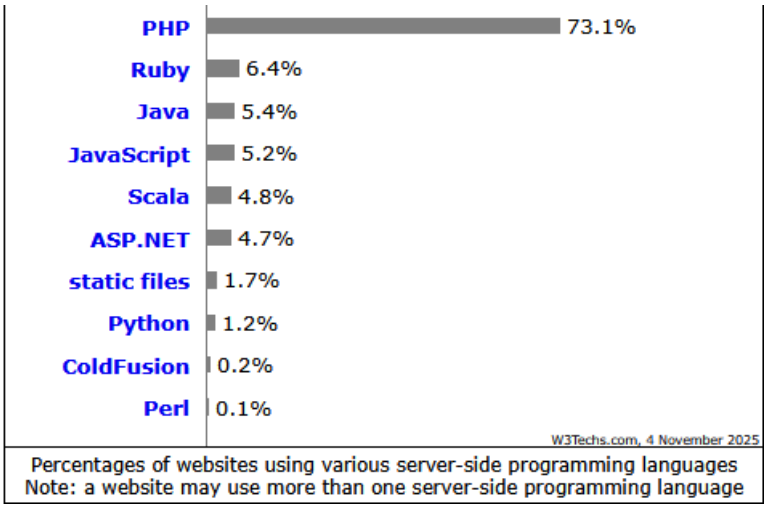
Toda la información y las tablas han sido obtenidas de la web W3Techs:

Página W3Techs con diferentes estadisticas de uso

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

Para el grado de implantación de los lenguajes de programación del lado del servidor voy a utilizar la información y estadísticas que proporciona la web de w3techs.com.

Las tablas que nos proporciona w3techs están ordenadas del mas usado al menos junto con los porcentajes de uso.



Percentages of websites using various server-side programming languages
Note: a website may use more than one server-side programming language

Como podemos ver en la tabla de uso de w3techs PHP domina como lenguaje de programación en un 73.1% de uso.

- **PHP:** es un lenguaje interpretado, libre y orientado a objetos. Diseñado sobre todo para hacer aplicaciones web y es especialmente útil en el acceso a datos.
- **Ruby:** Software libre, interpretado y orientado a objetos. Tercer puesto en el ranking de los 11 principales lenguajes de programación w3techs.
- **Java:** No es solo un lenguaje, es toda una plataforma de aplicaciones Informáticas, también es orientado a objetos. Es multiplataforma gracias a JMV (Java Virtual Machine).
- **JavaScript:** Es un lenguaje interpretado orientado a objetos, cuando se utiliza del lado del servidor se le suele llamar Server Side JavaScript, en los últimos años ha subido su porcentaje de uso debido a que se puede utilizar en ambos lados (cliente – servidor).
- **Scala:** Es un lenguaje de programación multi-paradigma, compilado e interpretado, diseñado para expresar patrones comunes de programación en forma concisa y con tipos seguros.
- **ASP.net:** Es un entorno para aplicaciones web desarrollado por Microsoft.

- **Python:** Es un lenguaje interpretado y multiparadigma, aunque su fuerte es la orientación a objetos. No fue diseñado con la idea de hacer aplicaciones Web, pero gracias al framework Django está ganando popularidad para la creación de páginas HTML.

Recursos y enlaces de interés:

Toda la información y las tablas han sido obtenidas de la web W3Techs:

Página W3Techs con diferentes estadisticas de uso

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

XAMP es un paquete de software el cual incluye Apache, MariaDB, PHP y Perl.

Nos permite instalar en nuestro ordenador Apache, sin importar el sistema operativo que utilicemos, también nos permite probar nuestras aplicaciones web en nuestro ordenador sin la necesidad de tener que acceder a internet.

Componentes de la plataforma XAMPP

- **Apache:** El servidor HTTP Apache es un servidor web HTTP de código abierto, es el servidor HTTP más utilizado.
  Fuente: Wikipedia.

- **MariaDB:** Es un sistema de gestión de bases de datos relacional desarrollado bajo licencia dual: Licencia pública general/Licencia comercial por Oracle.
  Fuente: Wikipedia.

- **PHP:** Es un lenguaje de programación de uso general que se adapta especialmente al desarrollo web. El código PHP suele ser procesado en un servidor web por un intérprete PHP implementado como un módulo.
  Fuente: Wikipedia.

- **Perl:** Lenguaje de programación diseñado por Larry Wall en 1987.
  Fuente: Wikipedia.

12. En qué casos es necesaria la instalación de la máquina virtual Java (JVM) y el software JDK en el entorno de desarrollo y en el entorno de explotación.

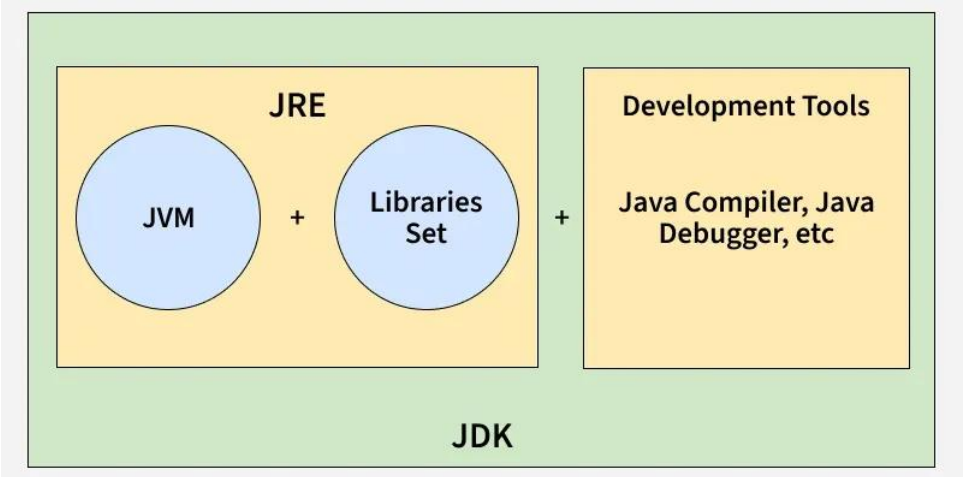
JVM (Java Virtual Machine) es un software que actúa como un intérprete y un entorno de ejecución para las aplicaciones Java.

Es necesaria tanto en el entorno de desarrollo como en el entorno de explotación.

- **Entorno de desarrollo:** Si, necesaria para ejecutar y probar aplicaciones.
- **Entorno de explotación/producción:** Sí, necesaria para ejecutar las aplicaciones Java.

JDK (Java Development Kit) es un paquete de software que incluye todas las herramientas necesarias para desarrollar aplicaciones Java.

JDK también incluye el JRE y por lo tanto el JVM, solo estará presente en **el entorno de desarrollo** al tratarse de la principal herramienta de escritura y prueba de aplicaciones Java.



Recursos y enlaces de interés:

Video de explicación de diferencias entre JDK ,JVM Y JRE (Programando en Java):

Video explicacion de diferencias Programando en Java

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

Definición: Un IDE es la herramienta de desarrollo de software la cual nos permite escribir, generar código, depurar y administrar la base de datos de un programa o aplicación web.

Los IDE' s documentados son **NetBeans, Eclipse, Visual Studio Code (Editor de texto), Intelli J.**

Eclipse:

Desarrollado por IBM, es un entorno de desarrollo de código abierto y multiplataforma.

Última versión: 25-09

Características:

- Editor de texto.
- Compilación en tiempo real.
- Pruebas unitarias con JUnit.
- Refactorización.
- Control de versiones.

NetBeans:

Entorno de desarrollo de código libre, gratuito y orientado principalmente a Java, fundado **por Sun Microsystems** en el año 2000.

Última versión: 27

Características:

- Gestión de la interfaz de usuario.
- Gestión de configuración de usuario.
- Herramientas de desarrollo integrado.
- Control de versiones Git.

Visual Studio Code:

Realmente es un editor de texto, pero mediante la instalación de extensiones lo podemos convertir en un entorno de desarrollo según nuestras necesidades, desarrollado por Microsoft es de código abierto y gratuito.

Última versión: 1.105.1

Características:

- Resaltado de la sintaxis.
- Autocompletado de código.
- Refactorización.
- Depuración.

IntelliJ IDEA:

Desarrollado por **JetBrains** con dos versiones, comercial y para la comunidad.

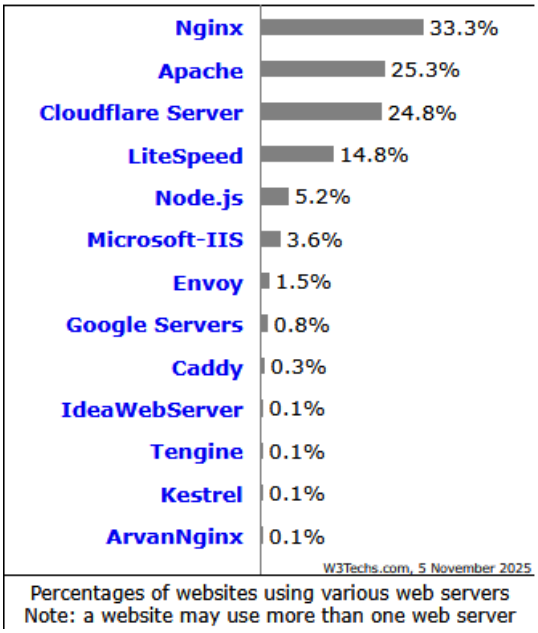
Última Versión: 2025.2.4

Características:

- Automatización de flujos de trabajo.
- Control de versiones.
- Pruebas unitarias.
- Descompilador integrado.

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

Para responder a la pregunta vamos a utilizar la información y las estadísticas proporcionadas por la web de W3Techs.com, la cual proporciona una tabla con los porcentajes y el orden de mas usados a menos de servidores HTTP Y HTTPS.



Nginx: El servidor web HTTP mas utilizado con un 33.3% de uso global, gratuito y programado en C es un servidor de archivos estáticos, permite el balanceo de cargas.

Apache HTTP: Servidor web HTTP de código abierto, desarrollado y mantenido por una comunidad de usuarios bastante activa en torno a Apache Software, soporte para lenguajes como Perl, PHP y Python , muy adaptable mediante módulos.

Apache Tomcat: Contenedor de aplicaciones contiene gratuito programado en Java. El servidor Apache tomcat se puede usar para manejar páginas estáticas y dinámicas.

Recursos y enlaces de interés:

Toda la información y las tablas han sido obtenidas de la web W3Techs:

Página W3Techs con diferentes estadísticas de uso

15. Apache HTTP vs Apache Tomcat

Dentro de los servidores Apache podemos diferenciar dos productos de Apache Software como lo son Apache HTTP y Apache Tomcat.

Apache Tomcat:

Apache Tomcat fue creado principalmente para el desarrollo y programación de aplicaciones Java.

Implementa especificaciones de Java Server Pages (JSP) y se enfoca en procesar código del lado del servidor escrito en java.

Su principal función es recibir peticiones HTTP y HTTPS y como respuesta dar contenido HTML basado en Servelets.

Apache HTTP:

Apache HTTP fue creado como servidor web de propósito general.

Su principal función es proporcionar contenido estático como HTML, imágenes, audio y texto programado en diferentes lenguajes de programación como PHP, Perl,...

Recursos y enlaces de interés:

En el siguiente árticulo de la página web ktsample explica a la perfección las diferencias entre Apache HTTP y Apache TomCat:

Enlace al artículo de ktsample

TRABAJO REALIZADO POR: Alejandro De la Huerga Fernández.

ULTIMA ACTUALIZACIÓN: 05/11/2025