DESARROLLO WEB ENTORNO SERVIDOR

ALEJANDRO DE LA HUERGA FERNÁNDEZ

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 mas 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.

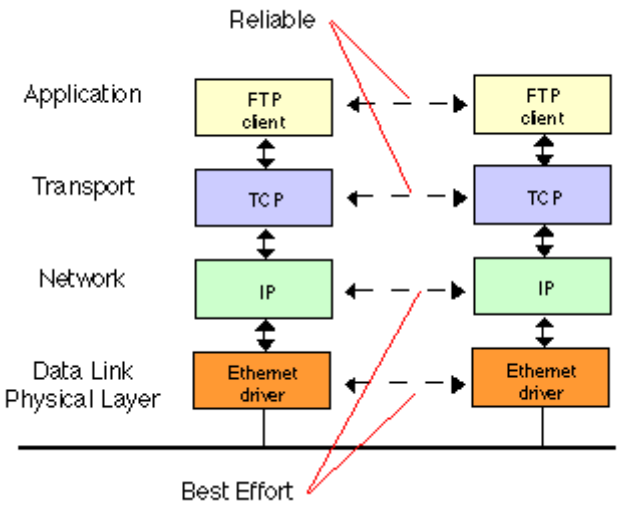
EJERCICIOS:

Realiza un estudio sobre los siguientes conceptos:

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

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

Dentro de dichos protocolos vamos a desglosarlos para ser mas específicos y explicar las características de cada uno:



- **Protocolo IP:**

  Es un protocolo, o conjunto de reglas, para enrutar y direccionar paquetes de datos para que puedan viajar a través de las redes y llegar al destino correcto. Los datos que atraviesan Internet se dividen en trozos más pequeños, llamados **paquetes**. La información IP se adjunta a cada paquete y esta información ayuda a los **enrutadores** a enviar los paquetes al lugar correcto.

  A cada dispositivo o dominio se le asigna una dirección ip.

  Una vez que los paquetes llegan a su destino se procesan de manera diferente dependiendo del protocolo de transporte que se utilice junto con el protocolo IP (TCP y UDP son los mas comunes).

- **Protocolo TCP:**

  El protocolo TCP es un protocolo de transporte , lo cual quiere decir que dicta la manera en que los datos viajan.

  Antes de enviar los datos TCP abre una conexión con el destinatario.

  TCP garantiza que todos los paquetes lleguen en orden una vez iniciada la transmisión.

  TCP e IP fueron diseñados para ser utilizados juntos, TCP/IP fue diseñado para **ser muy fiable**, **no para que fuera rápido**, ya que al tener que asegurar la llegada de todos los paquetes en orden hace que la velocidad disminuya.

- **Protocolo HTTP:**

  El protocolo de comunicación HTTP **(Protocolo de transferencia de hipertexto)**, constituye la base de toda la red es un protocolo el cual nos permite cargar sitios web mediante enlaces de hipertexto.

El flujo común de un protocolo HTTP implica una maquina cliente que envía una **solicitud** a un servidor , que inmediatamente manda una **respuesta**.

El puerto por defecto para el protocolo HTTP es el puerto **80.**

- **Protocolo HTTPS:**

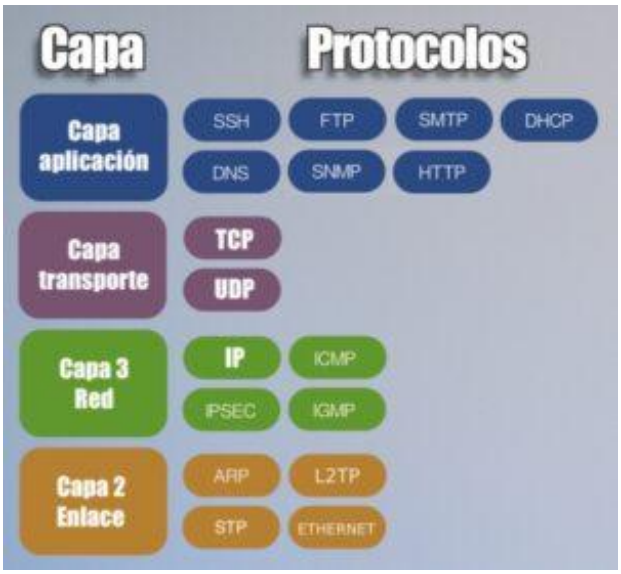
  El protocolo de comunicación HTTPS **(Protocolo de transferencia de hipertexto seguro),** actúa de la misma manera que el protocolo HTTP .

  Es el protocolo por excelencia para enviar entre de un navegador web a un sitio web.

  A diferencia del protocolo HTTP , el protocolo HTTPS esta encriptado para aumentar la seguridad al enviar y recibir datos.

  Normalmente los sitios web con inicio de sesión utilizan protocolo HTTPS.

Los navegadores nos indicaran si la web que queremos visitar utiliza protocolo HTTPS.



Esta imagen es un esquema de los diferentes protocolos que existen y la capa en la que trabajan.

ENLACES DE INTERES:

https://www.cloudflare.com/es-es/learning/network-layer/internet-protocol/

La página web de la empresa **"Cloudfare"** tiene mucha información de gran valor a la hora de hablar de protocolos con diferentes artículos para muchos de ellos.

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

El modelo cliente – servidor es uno de los conceptos de arquitectura mas comunes en la tecnología de redes. Regula la interacción entre el cliente y el servidor.

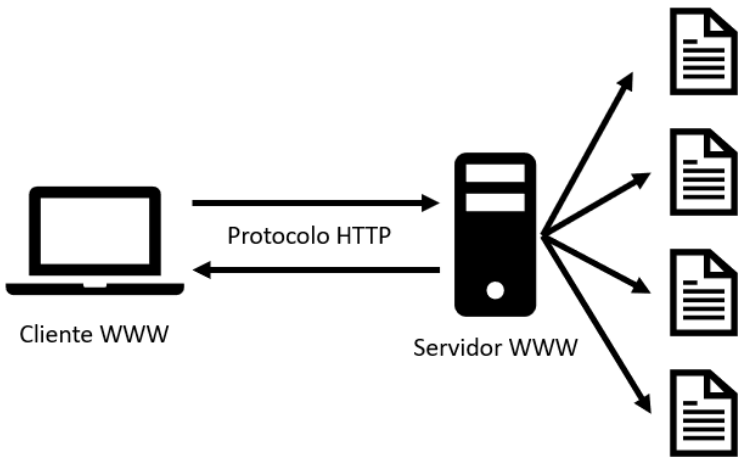
Es un modelo el cual permite la distribución de tareas dentro de una red de ordenadores.

Hoy en día la mayoría de paginas web utilizan el modelo cliente servidor , el cual podemos dividir en varias partes:

1. **Cliente**: Es un proceso de software ,si hablamos de desarrollo web este lugar lo ocupa el navegador el cual inicia la conversación mandando una petición al servidor usando la red como mecanismo de comunicación.

2. **Servidor:** El servidor es el componente que proporciona los servicios a los clientes , el servidor siempre será un sistema que espera peticiones de los clientes.

3. **Protocolos:** Para la comunicación entre el cliente y el servidor se usan los protocolos , el mas utilizado hoy en día es el protocolo HTTPS ya que es el protocolo que sustenta la red .



En la siguiente imagen vemos el esquema de la comunicación cliente servidor.

ENLACES DE INTERES:

En el siguiente articulo de la web "Arsys", nos explica de manera detallada la comunicación entre cliente y servidor:

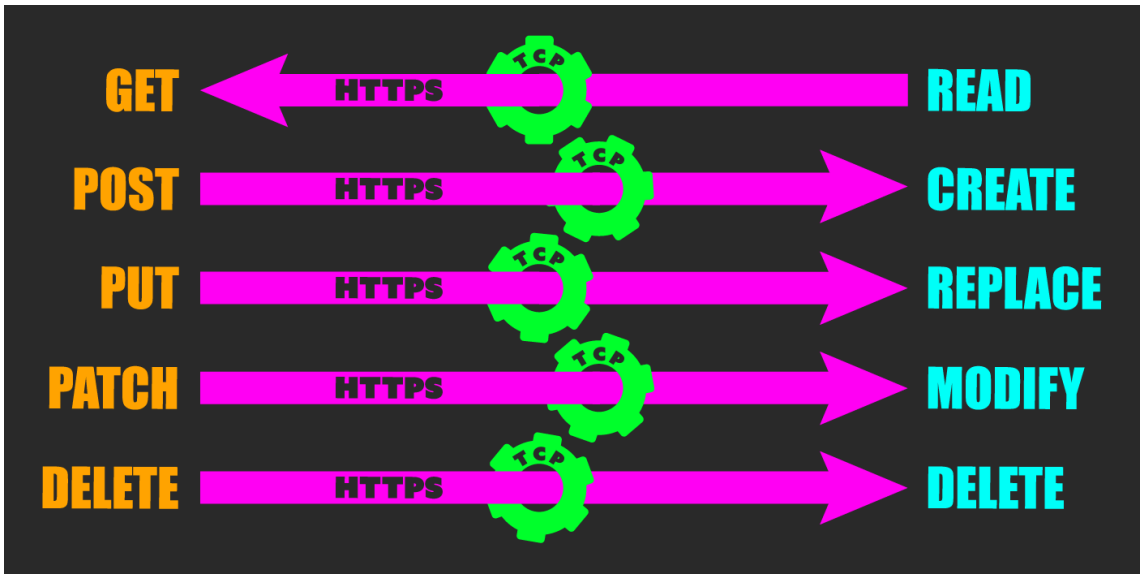
https://www.arsys.es/blog/todo-sobre-la-arquitectura-cliente-servidor

La información de esta pregunta ha sido obtenida de la web de "Arsys" que se pone anteriormente , así como las imágenes obtenidas de la búsqueda en Google.

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.



Dentro de estas peticiones tenemos distintos tipos que vamos a explicar a continuación:

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 sinllegar a modificarlos, por ejemplo visitas una pagina web de un periódico tu navegador usa “GET” para traer los textos , imágenes y videos que componen la pagina.

POST: Somete los datos para que sean procesados por el recurso identificado, por ejemplo al regístrate en una pagina 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 casa de que todavía no exista , este se suele usar para actualizar datos de una base de datos.

DELETE: Como su nombre lo indica el “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.

TRACE: Este método solicita al servidor que envíe de vuelta en un mensaje , lleva un seguimiento de la ruta que toma una solicitud a través de proxies, después devuelve el mensaje al servidor destino (muy útil para posibles problemas de red).

OPTIONS: Devuelve los métodos HTTP que el servidor soporta para un URL específico, es decir se utiliza para saber las opciones de comunicación que tiene un recurso específico.

CONNECT: Se utiliza para saber si se tiene acceso a un host o no, establece un túnel hacia el servidor identificado por el recurso.

A continuación tenemos una tabla con los principales métodos de petición y su compatibilidad con los navegadores:

	Chrome	Edge	Firefox	Opera	Safari	Chrome Android	Firefox for Android	Opera Android	Safari on iOS	Samsung Internet	WebView Android	WebView on iOS
CONNECT	✓ 1	✓ 12	✓ 1	✓ 15	✓ 1	✓ 18	✓ 4	✓ 14	✓ 1	✓ 1	✓ 4.4	✓ 1
DELETE	✓ 1	✓ 12	✓ 1	✓ 15	✓ 1	✓ 18	✓ 4	✓ 14	✓ 1	✓ 1	✓ 4.4	✓ 1
GET	✓ 1	✓ 12	✓ 1	✓ 2	✓ 1	✓ 18	✓ 4	✓ 10.1	✓ 1	✓ 1	✓ 1	✓ 1
HEAD	✓ 1	✓ 12	✓ 1	✓ 15	✓ 1	✓ 18	✓ 4	✓ 14	✓ 1	✓ 1	✓ 4.4	✓ 1
OPTIONS	✓ 1	✓ 12	✓ 1	✓ 15	✓ 1	✓ 18	✓ 4	✓ 14	✓ 1	✓ 1	✓ 4.4	✓ 1
POST	✓ 1	✓ 12	✓ 1	✓ 15	✓ 1	✓ 18	✓ 4	✓ 14	✓ 1	✓ 1	✓ 4.4	✓ 1
PUT	✓ 1	✓ 12	✓ 1	✓ 15	✓ 1	✓ 18	✓ 4	✓ 14	✓ 1	✓ 1	✓ 4.4	✓ 1

ENLACES DE INTERES:

Toda la información sobre los métodos de petición ha sido tomada de las dos siguientes paginas webs :

Mozilla :

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

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

Keepcoding :

La página Keepcoding nos presenta un análisis mas exhaustivo de los principales métodos de petición que Mozilla pero menos amplio en cuanto a número.

https://keepcoding.io/blog/metodos-de-peticion-http/

Toda la información de esta pregunta ha sido obtenida de estas dos páginas webs así como de los apuntes dados en clase sobre los métodos de petición HTTP y HTTPS.

La tabla presentada en cuanto a las peticiones soportadas por los navegadores ha sido obtenida de la pagina web presentada de Mozilla.

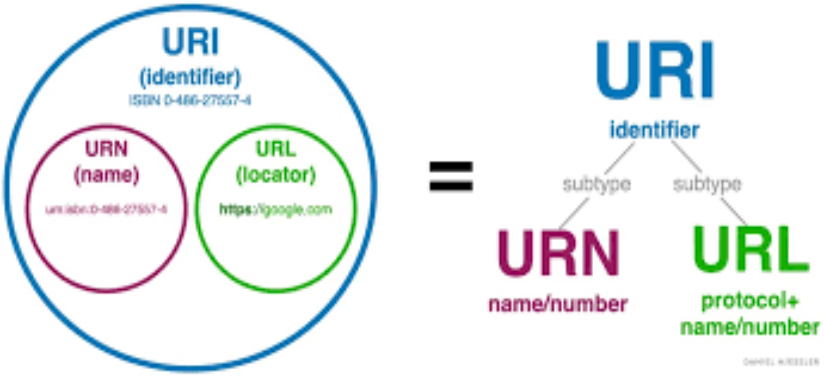
4. **Estudio sobre el concepto de URI (Identificador de Recursos Uniforme) /URL/URN, estructura, utilidad y relación con el protocolo HTTP/HTTPS.**

URI:

Un URI (Identificador Uniforme de Recursos) es una cadena de caracteres que se refiere a un recurso.

Se utiliza para acceder a un recurso físico o abstracto en internet.

Protocolos como el HTTP funcionan gracias al URI ya que la forma de direccionamiento se establece en base al URI.



El URI consta de un máximo de 5 partes:

1. **Scheme** *(esquema):* información sobre el protocolo utilizado.
2. **Authority** *(autoridad):* identifica el dominio.
3. **Path** *(ruta):* Muestra la ruta exacta al dominio.
4. **Query** *(consulta):* Representa la acción de consulta.
5. **Fragment** *(fragmento):* Designa una parte del recurso principal.

Los dos elementos imprescindibles que deben contener todos los identificadores son *scheme* y *path*.

scheme :// authority path ? query # fragment

EJEMPLO:

https://example.org/test/test1?search=test-question#part2

Scheme: https.

Authority: example.org

Path: test/test1

Query: search=test-question

Fragment: part2

URL Y URN:

Normalmente las siglas URI , URL y URN suelen confundirse por que aparte de sonar muy parecidas , se refieren a tres conceptos muy similares .

El URL **(Uniform resource locator)** se utiliza para indicar donde se encuentra un recurso, lo cual hace que también se utilice para buscar páginas web en internet.

El URL funciona completamente independiente de la ubicación y designa el recurso de forma permanente.

Tanto el URL como el URN presenta la misma estructura que el URI ya que estos dos son subgéneros del mismo.

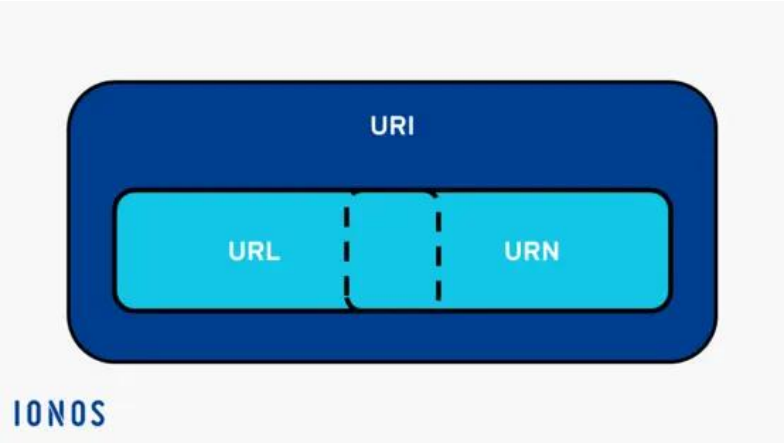


Imagen que representa la relación de URL y URN con URI.

Relación con el protocolo HTTP/ HTTPS:

La URL especifica la ubicación de un recurso , incluyendo el protocolo como HTTP para poder acceder a él.

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.

La URN a diferencia de la URL no utiliza un método HTTP directamente por que no tenemos la dirección.

ENLACES DE INTERÉS:

IONOS:

Esta página explica a la perfección lo que es una URI , su estructura y ambos subgéneros.

https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/uri-identificador-de-recursos-uniformes/

La imagen esquemática también ha sido obtenida de esta página web.

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

El desarrollo de una aplicación muticapa , es una arquitectura cliente ‘ servidor cuyo objetivo primordial es la separación de la lógica de negocios de la de diseño , es decir separar la capa de datos de la capa de presentación del usuario.

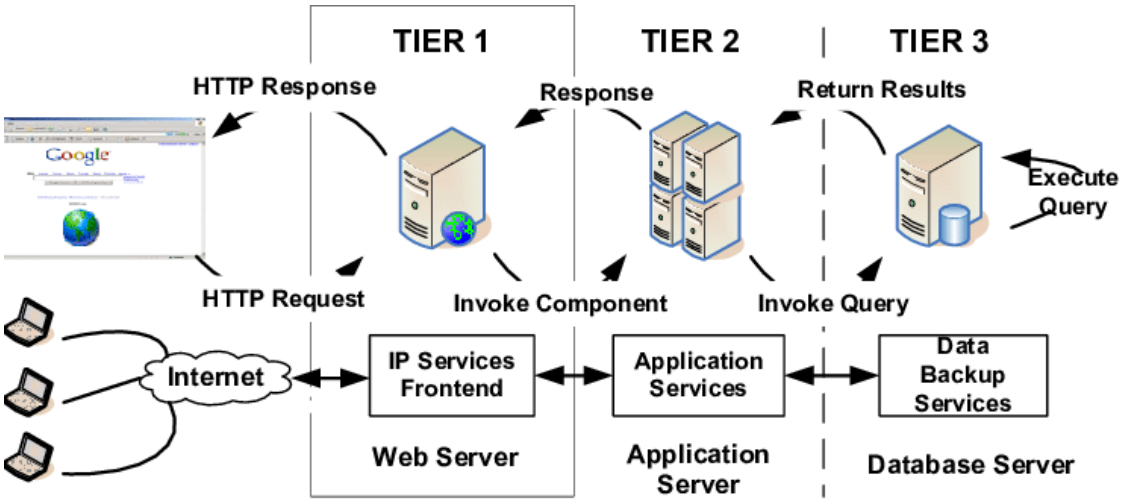
ARQUITECTURA DE 3 CAPAS

Tenemos que distinguir entre capas físicas y capas lógicas.

- Capa de Presentación.
- Capa de negocio.
- Capa de acceso a datos.

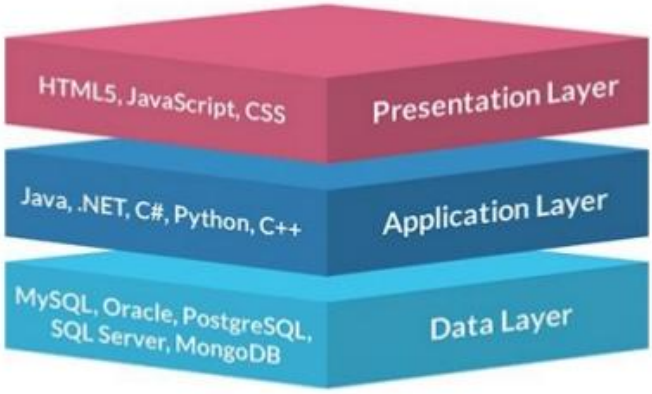
Ejemplo de arquitectura en tres capas:

- *Servidor Web*
- *Servidor de Aplicaciones*
- *Servidor Base de Datos*



Arquitectura de 3 capas físicas

En cuanto a las capas lógicas organizan su código respecto a su funcionabilidad:



En cada capa se puede implementar diferentes lenguajes de programación y herramientas.

ENLACES DE INTERÉS:

La información puesta en esta pregunta ha sido obtenida de la siguiente página web , la cual incluye información bastante valiosa y un pdf bastante extenso sobre las aplicaciones multicapa.

https://es.slideshare.net/slideshow/arquitectura-multicapa/12516686

Las imágenes ilustrativas y esquemáticas proporcionadas han sido obtenidas de la siguiente página web :

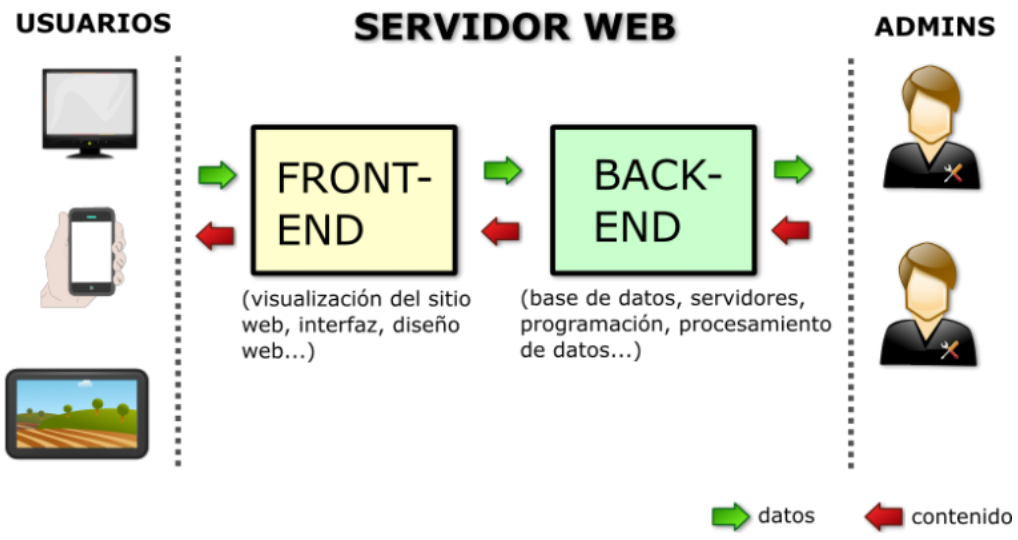
https://aitor-medrano.github.io/dwes2122/01arquitecturas.html#arquitectura-de-3-capas

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:** visualización del sitio web , interfaz, diseño web , esta parte esta diseñada para el usuario normal (autenticado o no) .

Back – **end :** bases de datos , servidores, programación , procesamiento… Esta parte de la aplicación esta diseñada para un usuario administrador , publicador , censor , regulador…



La siguiente imagen muestra el flujo de trabajo con el Front'end y el Back-end

ENLACES DE INTERÉS:

TeoCompDa:

https://perez15damian.wordpress.com/2017/10/16/que-es-frontend-y-backend/

Las imágenes han sido obtenidas de esta pagina web.

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:

Es aquella página web la cual esta compuesta por archivos html individuales , lo que hace que la única manera de que el cliente interactúa con la página sea mediante los elementos html , como enlaces formularios ...

Siempre veras la misma información cada vez que la visites a no ser que alguien cambie información modificando el código fuente.

Ejemplo de una web estática:

https://million-devs.netlify.com

Página web dinámica:

Una página web dinámica es aquella que sus elementos cambian continuamente y permiten al usuario interactuar con los diferentes elementos.

Ha diferencia de las estáticas para elaborar una página web dinámica se necesita algo mas que simplemente html y css, además de permitir que la página web tenga diversas funcionalidades.

Aplicación web:

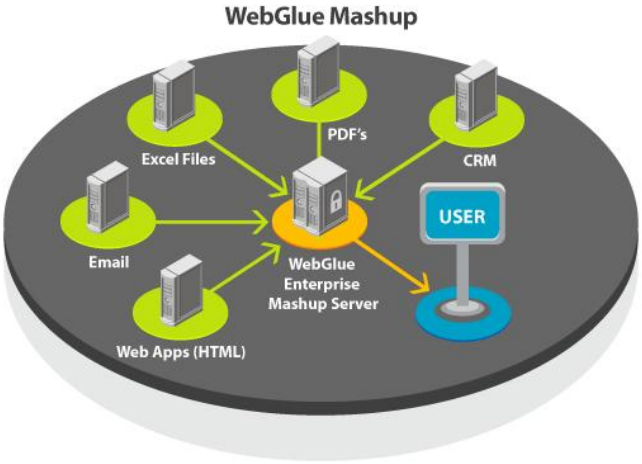
A diferencia de una pagina web dinámica la aplicación web agrega el control de acceso , pudiendo así tener un área de acceso a administradores o a diferentes funcionalidades como un panel de control.

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

Mashup:

Un mashup o aplicación web hibrida es aquella aplicación la cual integra los servicios de otras páginas web en una sola para ofrecer otro servicio.

Un ejemplo claro de mashup son las páginas comparadoras de precios , las cuales integran las demás páginas webs donde se encuentran los productos con su precio e información.



ENLACES DE INTERES:

OpenWebinars:

Toda la información de web estática , web dinámica y aplicación web esta obtenida de un articulo publicado en dicha web.

https://openwebinars.net/blog/paginas-web-estaticas-vs-paginas-web-dinamicas/

Arimetrics:

Toda la información utilizada en esta pregunta sobre lo que es un mashup ha sido obtenida de esta página web.

8. Componentes de una aplicación web.

Los componentes de una página web son aquellos elementos que **forman una aplicación web** , podemos dividirlos **en dos grupos** diferentes:

- **Componentes Front-end**:

  Los componentes para el lado del cliente son los siguientes: HTML (Para la estructura), CSS (Para el estilo) y JavaScript (Para la interactividad y el dinamismo).

  También hay frameworks de javascript dedicados al Front- end como React , Angular o Vue.

- **Componentes Back-end:**

  Dentro de los componentes de una aplicación web del lado del servidor podemos distinguirlos en varios sub grupos como son el servidor web , el módulo encargado de ejecutar el código , el sistema gestor de base de datos o los ficheros escritos en lenguajes de programación.

- **Servidor Web:** Apache, Nginx...
- **Sistema gestor de bases de datos:** MySQL o MongoDB.
- **Lenguajes de programación:** PHP , Java, JavaScript...

ENLACES DE INTERES:

ApiumHub:

La siguiente página web presenta un árticulo y bastante interesante sobre los componentes de una página web.

https://apiumhub.com/es/tech-blog-barcelona/componentes-web/

Amazon web services:

La siguiente página web muestra un articulo entre la diferencia de Front-end y Back-end , indicando también los componentes utilizados sobre todo en el lado del cliente.

https://aws.amazon.com/es/compare/the-difference-between-frontend-and-backend/

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

Dentro del desarrollo de una aplicación web se ejecutan distintos programas tanto del lado del cliente como del lado del servidor , también para ambos casos el lenguaje de programación cambia.

Lenguajes de programación utilizados del lado del cliente:

El que predomina con una claridad absoluta en el lado del cliente es sin duda alguna JavaScript .

Además de este lenguaje de programación también se utilizan diversos frameworks basados en JavaScript como lo son React.js, Angular, Vue.js

Most popular client-side programming languages

© W3Techs.com	usage	change since 1 August 2025
1. JavaScript	98.9%	
2. Flash	0.8%	-0.1%
3. Java	0.1%	

percentages of sites

Tabla con los lenguajes mas utilizados del lado del cliente.

Lenguajes de programación utilizados del lado del servidor:

Al igual que del lado del cliente en el lado del servidor tenemos un claro ganador cuando hablamos de lenguaje de programación, en este caso estamos hablando de **PHP**.

A diferencia del lado del cliente en el lado del servidor las opciones son mas amplias con lenguajes como **Java** , mas utilizado para aplicaciones a medida o Ruby .

JavaScript a pesar de ser utilizado en su mayoría en el lado del cliente también se puede utilizar del lado del servidor.

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

Tabla con los lenguajes de programación mas usados del lado del servidor en el ultimo mes.

ENLACES DE INTERES:

W3techs:

La siguiente página web nos muestras un análisis mediante tablas y porcentajes de los diferentes lenguajes de programación , servidores , frameworks y muchas cosas , que mas se están utilizando en el ultimo mes.

https://w3techs.com/

Las imágenes proporcionadas de las tablas se han obtenido de esta página web.

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

En cuanto a los lenguajes de programación utilizados del lado del servidor tenemos un claro ganador en cuanto a su porcentaje de utilidad hoy en día:

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

Tabla que representa el porcentaje de utilidad de los lenguajes de programación del lado del servidor.

PHP :

El lenguaje de programación por excelencia del lado del servidor es PHP , debido a que es un lenguaje de programación embebido , orientado a objetos y de código abierto ha hecho que ocupe la primara posición con mucha ventaja del segundo.

RUBY :

En segundo lugar tenemos a Ruby , debido a su sintaxis legible y por estar inspirado en lenguajes como Python y Perl ha hecho que se coloque como el segundo lenguaje del lado del servidor preferido por los profesionales.

JAVASCRIPT :

Tenemos que tener en cuenta en este apartado a JavaScript , al ser un lenguaje dedicado al lado del cliente , llama la atención verlo en esta lista , pero JavaScript es capaz de ejecutar lógica de backend , interactuar con bases de datos o crear APIS , al poder programar con el mismo lenguaje en ambos lados hace que JavaScript sea la opción de muchos profesionales en los últimos años.

ENLACES DE INTERES:

PHP Web Oficial:

https://www.php.net/manual/es/introduction.php

RUBY Web Oficial:

https://www.ruby-lang.org/es/about/

W3SCHOOLS JAVASCRIPT:

https://www.w3schools.com/js/default.asp

W3techs:

La siguiente página web nos muestras un análisis mediante tablas y porcentajes de los diferentes lenguajes de programación del lado del servidor más utilizados.

https://w3techs.com/

Las imágenes proporcionadas de las tablas se han obtenido de esta página web.

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

XAMP es un paquete de software el cual ya incluye Apache, MariaDB, PHP y Perl m lo que permite a los usuarios todos los componentes necesarios y un entorno de servidor web local para el desarrollo y las pruebas de aplicaciones web.

CARACTERÍSTICAS:

- Es multiplataforma (Windows , Linux y macOS).
- Es compatible con muchas plataformas CMS como WordPress.
- Esta formado por estos componentes clave (PHP, MySQL, PHP, Perl).
- Fácil instalación y uso.
- Te permite instalar un servidor web en tu propia computadoras.
- Soporte para múltiples CMS.

POSIBILIDADES DE DESARROLLO CON XAMP:

- La ejecución de un entorno de servidor completo en el propio ordenador del usuario.
- Facilidad para configurar y gestionar diferentes versiones de los componentes.
- La posibilidad de trabajar sin conexión a internet.
- Su principal propósito es el desarrollo de manera local.

ENLACES DE INTERES:

GoDaddy:

La siguiente pagina web muestra un articulo muy interesante que cubre todo lo relacionado con XAMP desde la instalación hasta sus características, la información redactada en la respuesta a la pregunta ha sido obtenida de esta pagina web.

https://www.godaddy.com/resources/es/crearweb/xampp-que-es

Última actualización: 30/09/2025 17:39