



ESTUDIO 1º UNIDAD

ÁLVARO GARCÍA GONZÁLEZ

1. Protocolos de comunicaciones: IP, TCP, HTTP, HTTPS. _____	4
2. Modelo de comunicaciones cliente – servidor y su relación con las aplicaciones web. _____	5
3. Estudio sobre los métodos de petición HTTP /HTTPS más utilizados. _____	6
4. Estudio sobre el concepto de URI (Identificador de Recursos Uniforme) /URL/URN, estructura, utilidad y relación con el protocolo HTTP/HTTPS. _____	7
5. Modelo de desarrollo de aplicaciones multicapa – comunicación entre capas – componentes – funcionalidad de cada capa. _____	8
6. Modelo de división funcional front-end / back-end para aplicaciones web. _____	9
7. Página web estática – página web dinámica – aplicación web – mashup . _____	10
8. Componentes de una aplicación web. _____	11
9. Programas ejecutados en el lado del cliente y programas ejecutados en el lado del servidor - lenguajes de programación utilizados en cada caso. _____	12
10. Lenguajes de programación utilizados en el lado servidor de una aplicación web (características y grado de implantación actual). _____	13
11. Características y posibilidades de desarrollo de una plataforma XAMPP. _____	14
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. _____	15
13. IDE más utilizados (características y grado de implantación actual). _____	15
14. Servidores HTTP /HTTPS más utilizados (características y grado de implantación actual). _____	16
15. Apache HTTP vs Apache Tomcat _____	16
16. Navegadores HTTP /HTTPS más utilizados (características y grado de implantación actual). _____	17
17. Generadores de documentación HTML (PHPDoc): PHPDocumentor, ApiGen, ... _____	17
18. Repositorios de software – sistemas de control de versiones: GIT, CVS, Subversion, ... _____	17
Estudios propuestos para el módulo Desarrollo Web en Entorno Servidor _____	18

Realiza un estudio sobre los siguientes conceptos:

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

El https es el **protocolo básico** de comunicación. El **puerto** por defecto es el 443.

Para que funcione necesita las siguientes partes que forman parte del protocolo TCP/IP:

HTTP: es el protocolo que realiza la petición de datos y recursos. La petición de datos es iniciada por el cliente. La **diferencia** con https es que este último añade una **capa de seguridad** a través del cifrado. Puerto por defecto:80

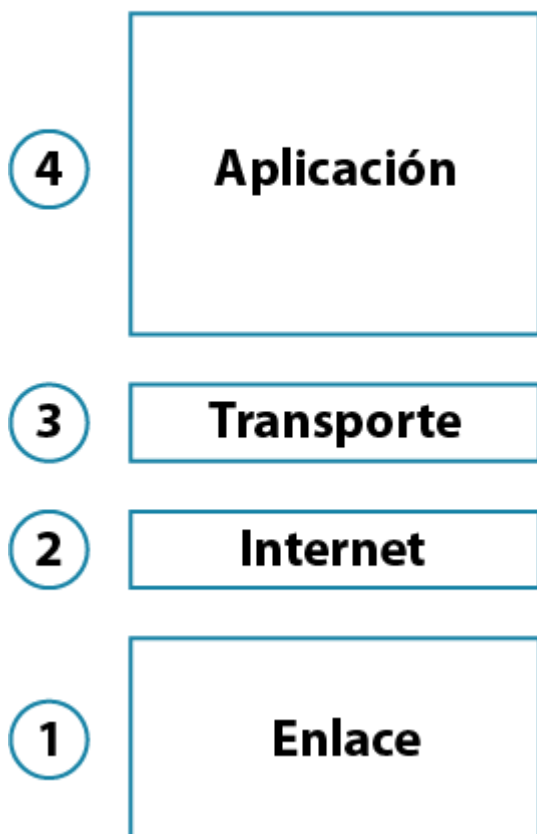
TCP: se encarga de **establecer** y **mantener** la conexión entre el emisor y receptor. TCP se considera más fiable en el apartado de entregas correctas comparado con el resto de protocolos.

IP: es el **conjunto de reglas** que define como se comunican los dispositivos. Más información

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

En la siguiente foto, la capa de aplicación correspondería con el HTTPS/HTTP, la capa de transporte con el TCP y la de internet con la IP

Modelo TCP/IP



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

El **modelo cliente-servidor** es una **arquitectura de comunicación** en la que dos entidades —el cliente y el servidor— intercambian información a través de una red, normalmente Internet.

- **Cliente:**
Siempre inicia la comunicación. Suele ser un navegador web, una aplicación móvil o de escritorio .
Ejemplo: Google Chrome solicitando una página web.
- **Servidor:**
Es el sistema que **espera y recibe las peticiones** de los clientes y les **responde** con los recursos solicitados (páginas HTML, imágenes, datos, etc.). Puede ejecutar código del lado del servidor (por ejemplo, en PHP, Python o Java) y acceder a bases de datos para generar respuestas dinámicas.

El funcionamiento suele ser el siguiente:

1. El cliente envía una **petición (request)** al servidor usando un protocolo, normalmente **HTTP o HTTPS**.
2. El servidor recibe la petición, la procesa (por ejemplo, consultando una base de datos o ejecutando un script).
3. El servidor envía una **respuesta (response)** al cliente, que puede ser una página HTML, un archivo JSON, una imagen o cualquier otro recurso.
4. El cliente interpreta la respuesta y la muestra al usuario.



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

Estos métodos son las peticiones que hacen los clientes para que el servidor entienda las solicitudes.

Algunos de los métodos más utilizados son “get”, “head”, “post”, “put” y “delete”

Método HTTP	Significado en Restful Web Services
GET	Se utiliza para operaciones de sólo lectura. No generan ningún cambio en el servidor.
DELETE	Elimina un recurso en específico. Ejecutar esta operación múltiples veces no tiene ningún efecto.
POST	Cambia la información de un recurso en el servidor. Puede o no regresar información.
PUT	Almacena información de un recurso en particular. Ejecutar esta operación múltiples veces no tiene efecto, ya que se está almacenando la misma información sobre el recurso.
HEAD	Regresa solo el código de respuesta y cualquier encabezado HTTP asociado con la respuesta.

Para información más detallada sobre los métodos: https://estilow3b.com/metodos-http-post-get-put-delete/#Metodo_HTTP_DELETE

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

Para que el servidor entienda las peticiones del cliente, tiene que realizar una petición formal que deben seguir un formato concreto.

URL: Las direcciones suelen ser mediante URL que contiene una **dirección**, la **referencia al protocolo utilizado** y la **ruta al objeto** que queremos

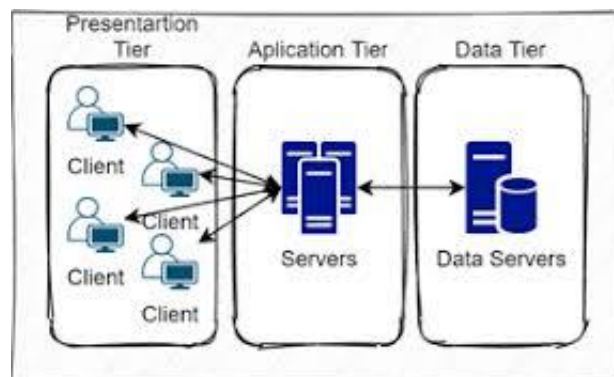
URN: Asigna un código de identificación a los objetos y este no varía, aunque el recurso se traslada. Las partes son: 1. información sobre el esquema URN 2. El NID (identificador de espacio de nombres) 3. El NSS, que es el identificador como tal.

URI: Es el identificador uniforme de recursos y sirve para acceder a todos los tipos de recursos. Abajo hay una foto que explica la estructura.



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

El modelo multicapa que se usa para poder **separar un proyecto en varias partes**. Es útil para poder **desarrollar una capa sin afectar a las otras**. Por ejemplo, cuando actualizamos la capa de negocio, nos aseguramos de no afectar a los datos que se encuentran en la capa de persistencia. El modelo funciona de esta manera:



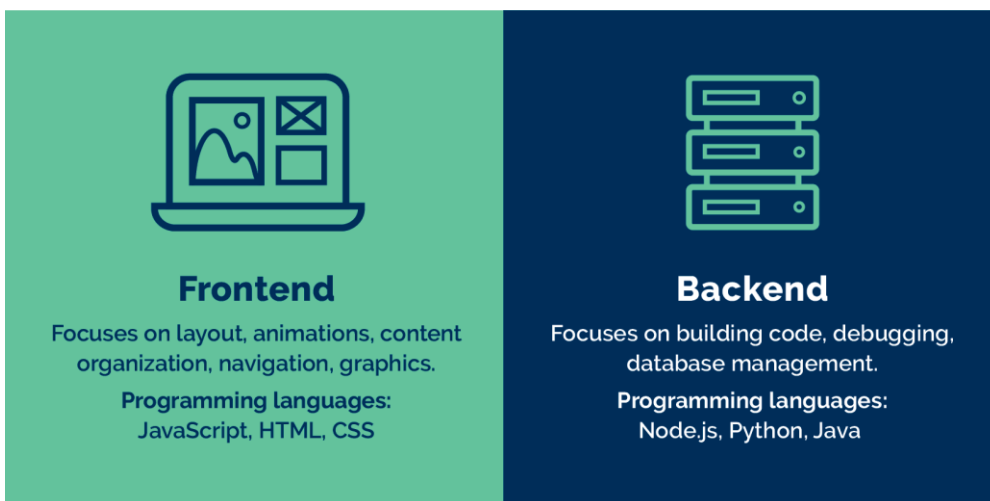
Capa de presentación: es lo que ve el usuario y recoge la interacción de este.

Capa de negocio: gestiona las funcionalidades de la aplicación web. Es donde, normalmente, se reciben las peticiones de los usuarios y se devuelven las respuestas.

Capa de persistencia: es donde se almacenan los datos y desde donde se accede a ellos. Reciben solicitudes de almacenamiento o de recuperación de información.

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

El **front-end** trata todo aquello que los **usuarios normales pueden ver e interactuar**, mientras que el **back-end** es lo que hace que la **aplicación funcione y la parte de** la aplicación dedicada a **los usuarios especiales** (administrador, publicador, etc....). Ambas partes se deben **desarrollar de forma consciente la una de la otra**, ya que ciertos aspectos del front-end pueden mejorar la velocidad y el rendimiento de la aplicación.



Para indagar más en las diferencias: <https://aws.amazon.com/es/compare/the-difference-between-frontend-and-backend/>

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

Página web estática: Su contenido **nunca varía** (salvo que el programador lo haga, claro). Y solo necesitan un servidor comunicándose con la web para que funcionen.

Página web dinámica: su contenido **cambia dependiendo de ciertas variables** (por ejemplo, las acciones que has realizado anteriormente en esa web). También puede ser que la página sea **generada completamente** de forma dinámica (Por ejemplo, tu correo, cada uno tiene sus mensajes).

Aplicación web: es la evolución de una página web dinámica, ya que usa estas últimas para **crear aplicaciones** (sin necesidad de instalación) que se **ejecutan en un servidor y se muestran en la web**. Se empieza a **diferenciar entre aplicación web y página dinámica** cuando tiene **control de acceso**.

Mashup: Es una forma de integración en la que una aplicación web es usada por otra para utilizar su funcionalidad, normalmente mediante APIs públicas

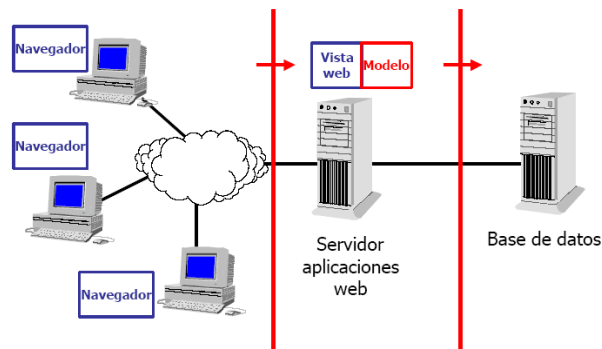
8. Componentes de una aplicación web.

Por parte del cliente los componentes son:

1. Interfaz de usuario (UI)
 - Es todo lo que el usuario ve e interactúa: botones, formularios, menús, gráficos, etc.
 - Su objetivo es ofrecer una experiencia clara y amigable.
 - Lenguajes: HTML (estructura), CSS (estilo), JavaScript (interactividad).
 - Ejemplo: el diseño de un formulario de login o el panel de control de una web.
2. Código ejecutado en el navegador
 - Son scripts que se ejecutan en el cliente para mejorar la experiencia sin depender del servidor.
 - Incluye JavaScript y librerías/frameworks como jQuery, React o Vue.js.
 - Funciones: validación de formularios, animaciones, actualización de contenido dinámico sin recargar la página (AJAX).
 - Ejemplo: mostrar automáticamente nuevos mensajes en una app de chat sin recargar la web.

Por parte del servidor son los siguientes:

1. Servidor web.
 - Programa que **recibe las solicitudes HTTP/HTTPS** del cliente y envía las respuestas correspondientes.
2. Modulo encargado de ejecutar el código.
 - Función: aplicar la lógica de negocio, procesar formularios, validar datos, generar contenido dinámico.
3. Sistema gestor de base de datos.
 - Se encarga de **almacenar, recuperar y administrar información** de forma eficiente.
4. Ficheros escritos en lenguaje de programación.
 - Define cómo se comporta la aplicación y cómo se procesan los datos.



9. 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 del **cliente**: HTML, CSS, Lenguajes para interactuar con el navegador y librerías.

Lenguaje más usado es: **JavaScript** con un **98,9%**

Librerías de JavaScript más usadas:

1. JQuery: 72,5%
2. Bootstrap: 17,2 %
3. Underscore: 8,1 %

En el lado del **servidor**: Servidor web, gestor de base de datos y lenguajes de programación.

















Lenguajes de programación más usados:

1. PHP: 73,3%
2. Ruby: 6,4%
3. Java: 5,4%

Servidores web más usados:

1. NGinx: 33,4%
2. Apache: 25,4%
3. Cloudfare: 24,5%

Sistemas gestores de bases de datos más usadas

Rank			DBMS	Database Model	Score		
Oct 2025	Sep 2025	Oct 2024			Oct 2025	Sep 2025	Oct 2024
1.	1.	1.	Oracle	Relational, Multi-model 	1212.77	+42.15	-96.67
2.	2.	2.	MySQL	Relational, Multi-model 	879.66	-12.11	-143.09
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model 	715.05	-2.27	-87.04
4.	4.	4.	PostgreSQL	Relational, Multi-model 	643.20	-13.97	-8.96
5.	5.	5.	MongoDB 	Document, Multi-model 	368.01	-12.49	-37.20
6.	6.	 7.	Snowflake	Relational	198.65	+8.46	+58.05
7.	7.	 6.	Redis	Key-value, Multi-model 	142.33	-2.84	-7.30
8.	 9.	 14.	Databricks	Multi-model 	128.80	+4.74	+43.21
9.	 8.	9.	IBM Db2	Relational, Multi-model 	122.37	-1.81	-0.40
10.	10.	 8.	Elasticsearch	Multi-model 	116.67	-1.59	-15.18

Enlace a las estadísticas de las bases de datos => <https://db-engines.com/en/ranking>

Enlace al resto de estadísticas => <https://w3techs.com/>

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

Lenguajes de programación más usados:

1. PHP: 73,3%. Algunas características de PHP es que es de **código abierto y gratuito, orientado a objetos y gran implementación con HTML**
2. Ruby: 6,4%: Ruby también está orientado a objetos, pero con un **tipado dinámico**, es decir, es un lenguaje de **programación interpretado**. Los errores se producen cuando se ejecuta el programa, ya que aunque haya errores, te permite ejecutar igualmente.
3. Java: 5,4%: Una de sus características es su **gran robustez y su recolección de basura, además de también ser orientado a objetos**.

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

Características: Es un paquete que incluye **Apache** como **servidor web**, **MariaDB/MySQL** como **base de datos** y **PHP/Perl** como **lenguajes de programación**.

Con la plataforma XAMPP puedes hacer todo el desarrollo de una aplicación web desde tu dispositivo, ya que te permite tener el servidor, la base de datos, el entorno de desarrollo y el de explotación **todo en el mismo lugar**. Útil para proyectos en solitario y de desarrollo relativamente rápido.



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.

La máquina virtual de Java es necesaria si pretendemos usar cualquier aplicación desarrollada en Java, sino no podremos ejecutarla. Sin embargo, el JDK solo será necesario si queremos desarrollar en Java.

Si eres un **desarrollador** y quieres crear una aplicación Java:

- Necesitas **JDK** para escribir y compilar el código.
- La **JVM** se ejecuta automáticamente para probar la aplicación.

Si solo quieres **ejecutar una aplicación Java** en un servidor (sin modificarla):

- Solo necesitas instalar la **JVM**, no el JDK.

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

Los IDEs más usados a fecha de creación de este documento son los siguientes:

- **Visual Studio**: 28.1%. Compatible con cualquier entorno .NET.
- **Visual Studio Code**: 15.32%. Alta capacidad de personalización.
- **Eclipse**: 12.65 %. Amplio repertorio de plugins

Caso del IDE NeatBeans.

- **NeatBeans**: 3.63%. Especializado en Java

La información ha sido recopilada de: <https://pypl.github.io/IDE.html>

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

A la fecha de creación de este documento, los servidores más usados son:

- Nginx: Con un 33.4% de uso actual, soporta lenguajes como PHP, Python y Java (a través de Node.js). También puede soportar otros lenguajes gracias a ciertos módulos.
- Apache HTTP: Con 25.4%, puede soportar Perl o Python, pero por lo general, se usará para soportar PHP.
- Cloudflare Server: Tiene una implementación actual del 24.5%. Soporta JavaScript, Rust y PHP. Destaca especialmente en su seguridad (se considera como uno de los mejores servicios contra ataques DDoS).
- Apache Tomcat: Menos de 0.1% de implementación. Es un contenedor de servlets (<https://www.tokioschool.com/noticias/que-es-servlet/>) específico para Java. Aunque puede funcionar como servidor web por el mismo, lo normal es que se combine con otros servicios como Apache HTTP.

Link de referencia -> https://w3techs.com/technologies/overview/web_server

15. Apache HTTP vs Apache Tomcat

Ambos servicios de Apache tienen en común que son servidores web (aunque no es el fuerte de Tomcat). Lo que los diferencia son el tipo de proyecto a implementar.

Apache HTTP se usa en páginas webs estáticas y dinámicas implementando lenguajes como PHP o Python, mientras que **Tomcat** se especializa en páginas web dinámicas desarrolladas con Java, aunque también se puede usar para “hostear” webs estáticas, no es su fuerte.

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

- Google: Es el más usado por mucha diferencia. Algunas de sus **ventajas** son su amplia gama de extensiones, su integración con otros servicios de Google, como Gmail o Drive, o su simpleza. Pero Google presenta unas **desventajas** considerables como su alto uso de recursos o su falta de privacidad para los usuarios.
- Mozilla Firefox: Una de sus **ventajas** es la privacidad, de la que carece Google, además de ser un software de código abierto, que contribuye a su constante mejora. Mas no carece de **desventajas** tales como su velocidad, más lenta que la de la competencia, o falta de compatibilidad con muchas extensiones.
- Microsoft Edge: Ya es un navegador cuyo uso es bastante bajo, aunque tenga **ventajas** tales como una velocidad comparable a la de Google y su eficiencia de recursos. Tiene gran integración con el sistema operativo Windows aunque eso afecta a la privacidad. Pero desventajas claras que tiene es su escasísima personalización y su limitada biblioteca de extensiones.
- Safari: Es veloz, sencillo y te da la posibilidad de sincronizarte con el resto de dispositivos de Apple, y justo esto también es una desventaja ya que solo es compatible con este tipo de dispositivos y su biblioteca de extensiones es ínfima.

Estadísticas -> <https://gs.statcounter.com/browser-market-share#monthly-202409-202509-bar>

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

https://manual.phpdoc.org/HTMLSmartyConverter/HandS/phpDocumentor/tutorial_tags.pkg.html

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

Un repositorio es un almacén en el cual se puede guardar código y recuperarlo. Un sistema de control de versiones se encarga de gestionar los cambios que realizamos a nuestro código.

El software que nosotros usaremos es GIT y su versión en web GITHUB. Nos sirve tanto como de repositorio como de control de versiones gracias a su sistema de commits.

CVS: es un sistema de control de versiones muy popular entre desarrolladores de software libre.

19. Propuesta de configuración del entorno de desarrollo para la asignatura de Desarrollo web del lado servidor en este curso (incluyendo las versiones): xxx-USED y xxx-WXED. (Explicado en tema2)
20. Propuesta de configuración del entorno de explotación para la asignatura de Desarrollo web del lado servidor en este curso (incluyendo las versiones): xxx-USEE.(Explicado en tema 2)
21. 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
22. Elegir y realizar un estudio y una presentación para la exposición del trabajo sobre una de las siguientes arquitecturas de desarrollo de Aplicaciones Web:
- MEAN (con MongoDB y con MySQL)
 - Java EE vs Spring
 - Microsoft .NET
 - Angular 7
 - Symfony
 - Laravel
 - CakePHP
 - CodeIgniter

Estudios propuestos para el módulo Desarrollo Web en Entorno Servidor

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

DWES, DWEC, DIW, DAW.

En DWEC estudiamos todo lo que tiene que ver en el desarrollo por parte del cliente, es decir, todo aquello que pueden ver los usuarios comunes. En DWES también desarrollamos, pero en la parte del servidor, es decir, la parte que realiza los cálculos y la que más cerca está de la información (aparte de tener que crear otra web para los usuarios especiales que administran la aplicación). DIW trata sobre el diseño de nuestra aplicación, que la hará bonita y apetecible para que nos quedemos en ella. Por último esta DAW, que trata sobre el despliegue de la aplicación.

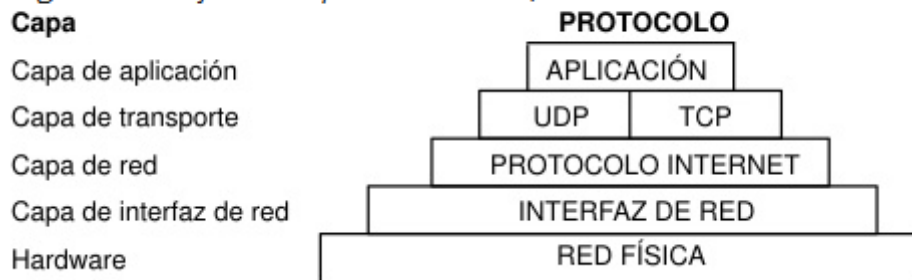
➤ Protocolos TCP/IP. Socket.

El protocolo TCP/IP es un conjunto de reglas que permiten la comunicación y la transferencia de datos entre dispositivos.

Las diferentes capas son: Aplicación, transporte, Internet, Acceso a red. La capa de acceso a red a veces se divide en dos: Interfaz de red y red física.

<https://www.ibm.com/docs/es/aix/7.1.0?topic=protocol-tcpip-protocols>

Figura 1. Conjunto de protocolos TCP/IP



Socket

<https://psp.codeandcoke.com/apuntes:sockets>

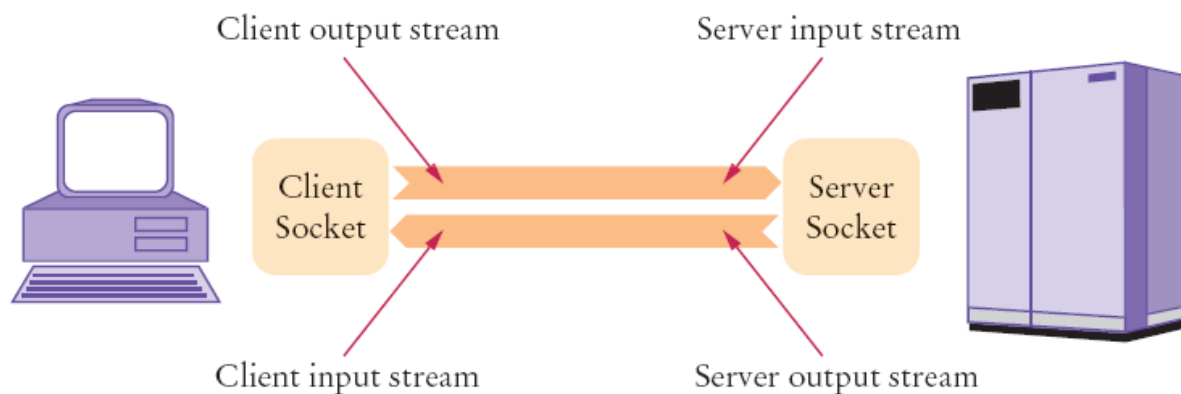


Figure 5 Client and Server Sockets

➤ Protocolo HTTP / HTTPS

Siglas de **protocolo de transferencia de hipertexto**. Forma parte de la capa de aplicación del protocolo TCP/IP y se utiliza para cargar páginas web mediante los enlaces de hipertexto. La diferencia con el protocolo HTTPS es que añade una capa de cifrado, el resto es igual.

➤ HTML

Es un lenguaje de marcas que sirve para darle la estructura a una página web.

➤ XML

Es un lenguaje de marcas donde debes definir tus propias etiquetas y que sirve principalmente para compartir datos.

➤ JSON

<https://www.json.org/json-es.html>

➤ Lenguajes de programación embebidos en HTML

Instrucciones que se encuentran dentro del código de HTML y enmarcados en etiquetas específicas. Así puedes ejecutar instrucciones para crear contenido interactivo o generar HTML. Ejemplos: PHP, JS, JSP

➤ Arquitecturas de desarrollo web

➤ Framework de desarrollo Web

➤ ERP

➤ CMS

➤ PHP

➤ IDE

➤ Navegador

➤ Repositorio

➤ Entorno de Desarrollo

➤ Entorno de Explotación o Producción

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

➤ Web Services

➤ AJAX

➤ Desarrollo de aplicaciones multicapa. Estrategias de diseño de aplicaciones Web. ➤ Aplicaciones basadas en microservicios

➤ SaaS: Software as a Service

➤ Control de acceso a la aplicación web o los Web Services

➤ Validación de entrada de datos a una aplicación Web

- Posicionamiento de una aplicación Web
- Historia, situación actual y evolución del diseño de aplicaciones Web
- Filosofías de desarrollo del software