ESTUDIO 1º UNIDAD

ÁLVARO GARCÍA GONZÁLEZ

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.

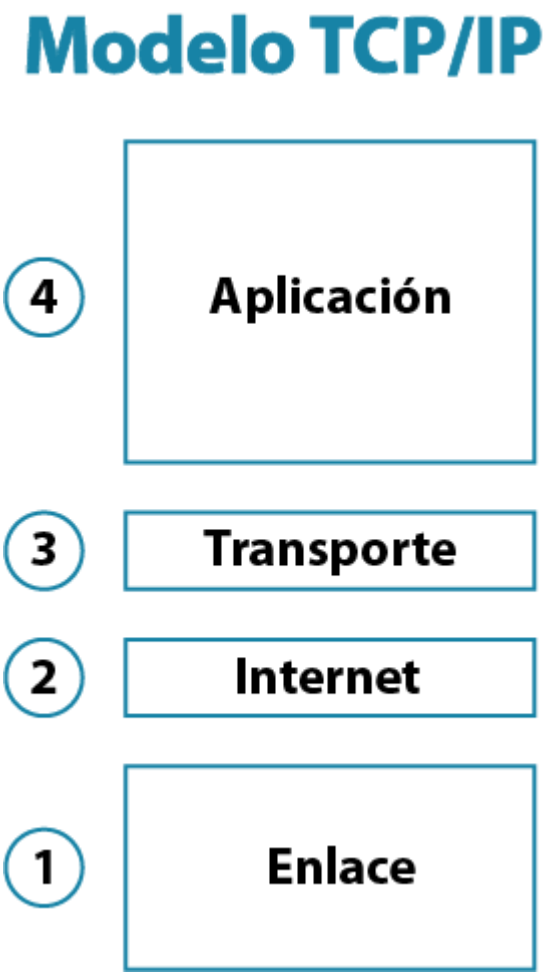
Para que funcione necesita las siguientes partes que forman parte del protocolo TCP/IP:

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



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 cabecero 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.

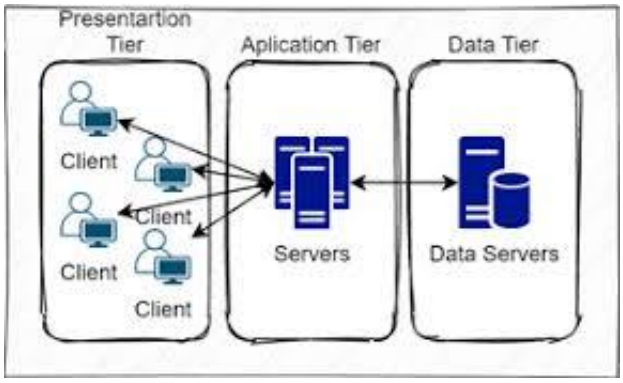
URL: Las direcciones suelen ser mediante URL que contiene una **dirección**, la **referencia al protocolo utilizado** y la **ruta al objeto** que queremos

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:



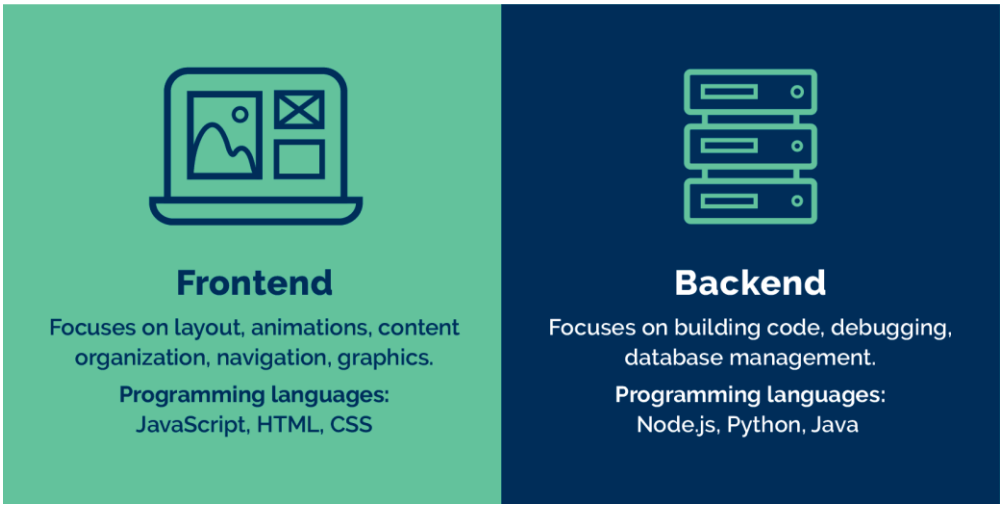
Capa de presentación: es lo que ve el usuario y recoge la interacción de este.

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

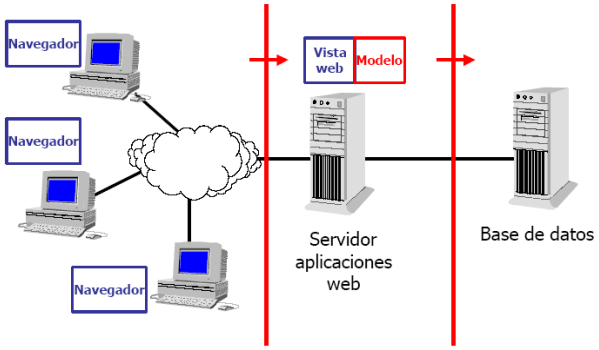
8. Componentes de una aplicación web.

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.
- Cloudfare 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 sus 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.

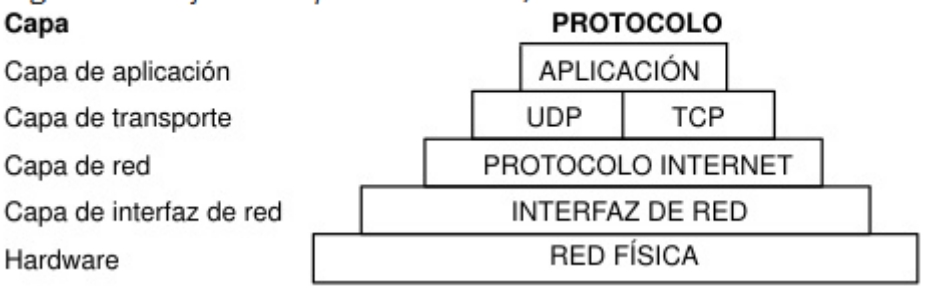
En DWEC estudiamos todo lo que tiene que ver en el desarrollo por parte del cliente, es decir, todo aquello que pueden ver los usuarios comunes. En DWES también desarrollamos, pero en la parte del servidor, es decir, la parte que realiza los cálculos y la que más cerca está de la información (aparte 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

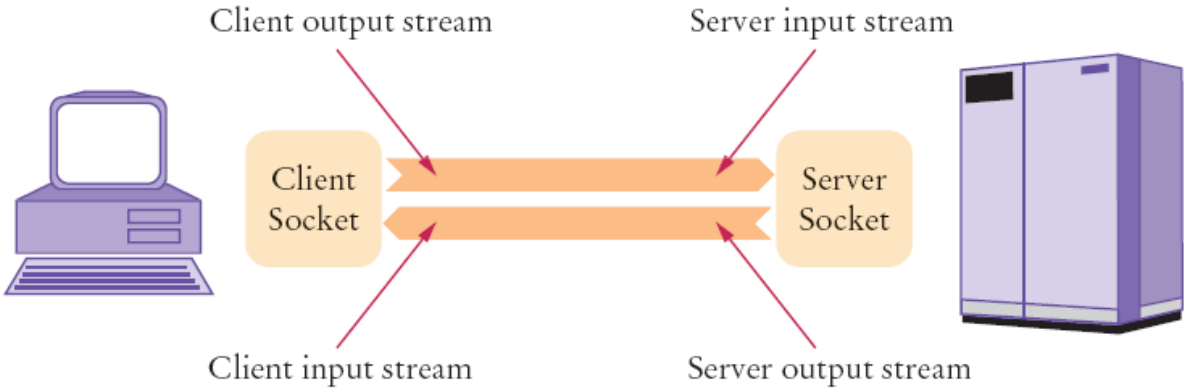


Figure 5 Client and Server Sockets

➢ 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 leguaje de marcas donde debes definir tus propias etiquetas y que sirve principalmente para compartir datos.

➢ JSON

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

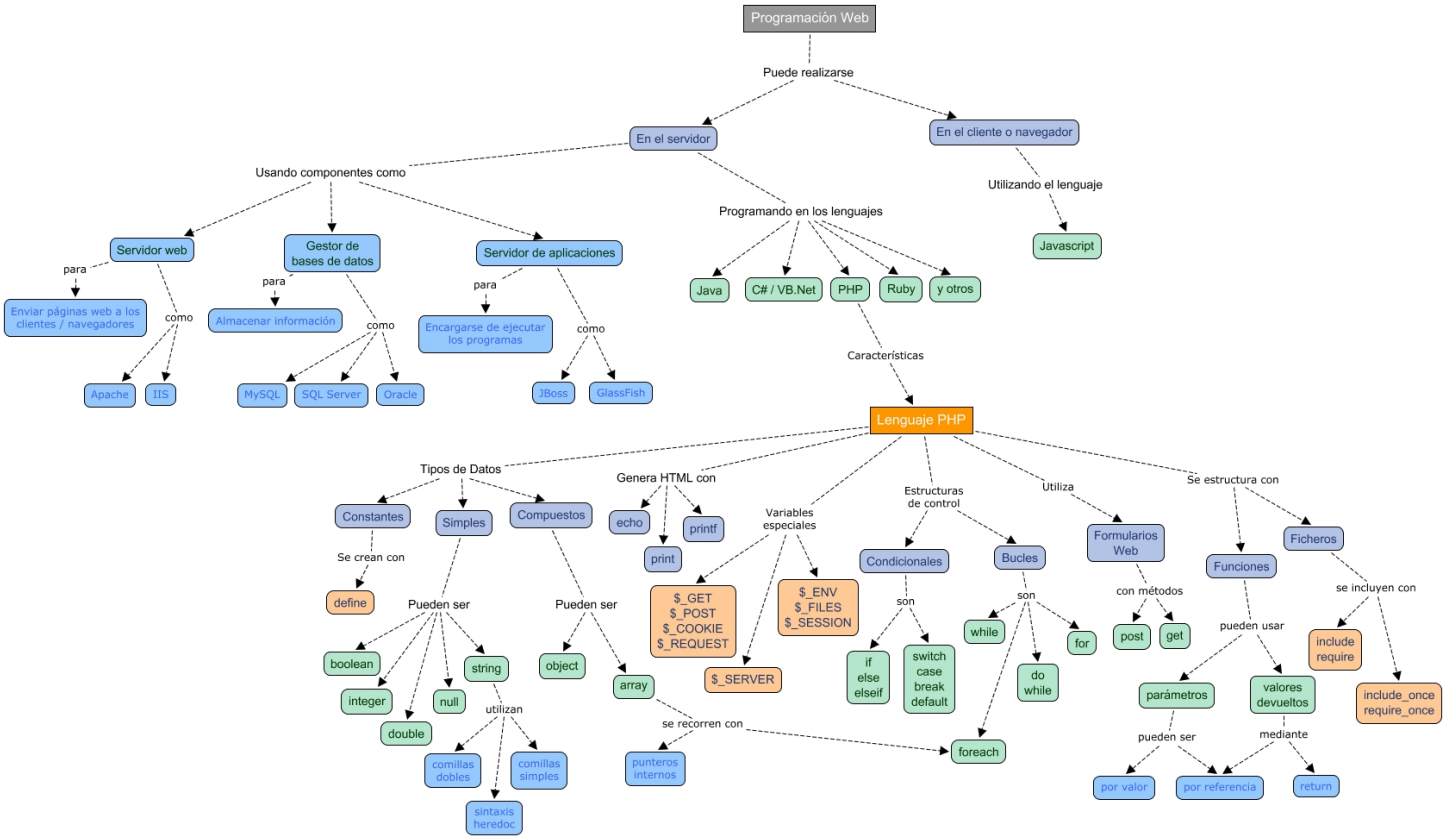
➢ Lenguajes de programación embebidos en HTML

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

Las arquitecturas web definen la forma en las que las páginas de un sitio web están estructuradas y enlazadas.

Imagen de referencia.



➢ Framework de desarrollo Web

Los frameworks se consideran como plantillas para que cada vez que empieces un proyecto, no tengas que empezar de cero. No es lo mismo que una librería, ya que tu desarrollas dentro de la estructura del framework, siguiendo sus reglas.

➢ ERP

Es un sistema de software que se usa en empresas para la gestión de las finanzas, recursos humanos, cadena de suministros, clientes, etc. Es decir, es un sistema especializado que permite la unificación y organización **de todas las áreas, que permite la trazabilidad de todos los procesos.**

➢ CMS

CMS (Sistema de Gestión de Contenidos) es un software que permite la creación y administración de páginas web de forma simple. Con el puedes controlar bases de datos y manejar el diseño y contenido.

➢ PHP

Lenguaje de programación centrado en el lado del servidor más usado del mundo (fecha 05/11/2025).

Es un lenguaje interpretado y de tipado dinámico. Además, es fácil integrarlo en código HTML.

Introducción de su web oficial: https://www.php.net/manual/es/introduction.php

➢ IDE

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

- **Editor de código fuente**: editor de texto que ayuda a escribir el código de software con funciones como el resaltado de la sintaxis con indicaciones visuales, el relleno automático específico para el lenguaje y la comprobación de errores a medida que se escribe el código.
- **Automatización de las compilaciones locales**: herramientas que automatizan las tareas sencillas y repetitivas como parte de la creación de una compilación local del software para que use el desarrollador, como la compilación del código fuente de la computadora en código binario, el empaquetado de ese código y la ejecución de pruebas automatizadas.
- **Depurador**: programa que sirve para probar otros programas y mostrar la ubicación de un error en el código original de forma gráfica.

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

➢ Navegador

Un navegador web es un programa que permite el acceso a la web. Permite la visualización de documentos de texto, visitar páginas webs y hacer actividades en ella. La mayor parte de la interfaz del navegador es usado para mostrar el contenido. Se puede visitar direcciones web mediante la barra de direcciones con la que el navegador cuenta.

➢ Repositorio

Un repositorio es un "almacén virtual" donde guardar archivos informáticos, en nuestro caso generalmente código. Es usado también para organizar dichos archivos. El repositorio puede ser tanto en local (git) como vía red (github). En estos, últimos los repositorios pueden ser públicos, con la posibilidad de que otros pueden acceder, o privado para solo poder verlo tú mismo

➢ Entorno de Desarrollo

El entorno de desarrollo es el entorno donde trabajamos para desarrollar nuestras aplicaciones y debe reunir todas las características necesarias para poder desarrollar dichas aplicaciones. Es decir, incluye el IDE, los servidores (o máquinas virtuales).

➢ Entorno de Explotación o Producción

Es el entorno donde se publica nuestro contenido para que pueda ser visto por los usuarios en internet. Es el lugar en el cuál tenemos nuestro dominio web

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

Gestión de la configuración:

La gestión de la configuración es un proceso para mantener el estado deseado de los sistemas y los elementos de la TI. Garantiza el funcionamiento uniforme durante todo el ciclo de vida.

Los administradores pueden utilizar herramientas de gestión de la configuración para preparar un sistema de TI, como un servidor o una estación de trabajo, y diseñar y mantener otros con los mismos parámetros. También pueden usar las evaluaciones de la configuración y los análisis de los desajustes para identificar constantemente los sistemas que no se encuentran en el estado deseado, por lo que es necesario actualizarlos, volver a configurarlos o aplicarles parches.

Las bases de datos de la gestión de la configuración (CMDB) son parte del proceso de gestión de servicios de TI (ITSM) y llevan a cabo un seguimiento de las opciones de configuración (CI) individuales, es decir, cualquier recurso o elemento que se incluya en la distribución de los servicios de TI. Las CMDB almacenan información sobre los atributos y las dependencias de las CI, además de

los cambios en su configuración que se implementan con el tiempo, lo cual permite que los equipos de TI diseñen y mantengan las relaciones que unen a las CI.

https://www.redhat.com/es/topics/automation/what-is-configuration-management

Control de cambios:

El control de cambios es un proceso que se usa para gestionar las solicitudes de cambio para proyectos y otras iniciativas importantes. Forma parte de un plan de gestión de cambios que define los roles para gestionar el cambio dentro de un equipo o empresa. Si bien un proceso de cambios tiene muchas partes, la forma más sencilla de visualizarlo es mediante la creación de un registro de cambios para dar seguimiento a las solicitudes de cambios del proyecto.

En la mayoría de los casos, cualquier involucrado podrá solicitar cambios. Una solicitud puede ser tan pequeña como una modificación en el programa del proyecto o tan grande como un nuevo entregable. Sin embargo, es importante tener en cuenta que no todas las solicitudes serán aprobadas, ya que depende de los participantes clave aprobar o rechazar las solicitudes de cambio.

Dado que un proceso de control de cambios incluye muchas piezas en movimiento y este difiere de una empresa a otra, es conveniente incorporar herramientas que ayuden a que los ciclos del proceso fluyan sin problemas. Herramientas como un software de gestión de flujos de trabajo pueden ayudarte a gestionar el trabajo y las comunicaciones en un solo lugar.

https://asana.com/es/resources/change-control-process

Mantenimiento de la aplicación:

Dentro de la ingeniería de software, el mantenimiento de aplicaciones tiene como objetivo modificar y actualizar la aplicación después de su entrega para corregir fallos, mejorar el rendimiento, mejorar el diseño, crear interfaz con otros sistemas, implementar mejoras o desarrollar nuevas funcionalidades. Todo este proceso suele estar gestionado en base al control de calidad, la visibilidad del cliente y ciclos ágiles de desarrollo.

Las actividades de mantenimiento de aplicaciones se clasifican en cuatro tipos:

Correctivo: orientado a diagnosticar y corregir errores comúnmente detectados por los usuarios, subsanando cualquier mal funcionamiento de la aplicación.

Preventivo: su objetivo es mejorar el rendimiento y otros factores no funcionales de las aplicaciones para evitar incidencias futuras.

Perfectivo o Evolutivo: orientado a mejorar la usabilidad y abarcar nuevos requisitos, evolucionando las aplicaciones en función de las nuevas necesidades del negocio y las novedades tecnológicas.

Adaptable: consiste en realizar modificaciones al software para conseguir que se adapte a los cambios del entorno (sistemas operativos, aplicaciones interconectadas, etc.).

https://talenticbpo.com/mantenimiento-de-aplicaciones/

➢ Web Services

El web service es una vía de intercomunicación entre maquinas conectadas en red.

La interacción se basa en el envío de solicitudes y respuestas entre un cliente y un servidor, que incluyen datos.

➢ AJAX

Es una técnica de desarrollo web para crear aplicaciones web asíncronas. Estas aplicaciones se ejecutan en el cliente mientras se mantiene la comunicación asíncrona con el servidor en segundo plano.

Tecnologías incluidas en Ajax:

- XHTML y CSS:https: //es.wikipedia.org/wiki/XHTML
- DOM: https://es.wikipedia.org/wiki/Document_Object_Model
- XMLHttpRequest:https: //es.wikipedia.org/wiki/XMLHttpRequest
- XML: https://es.wikipedia.org/wiki/Extensible_Markup_Language

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

Aplicaciones multicapa:

Es una arquitectura cliente-servidor donde las funciones de presentación, lógica de negocio y gestión de datos están separadas.

De esta forma, es sencillo hacer cambios en una parte del código sin afectar a las demás.

(Para entender cada capa, revisar el punto número 5)

Estrategias de diseño de aplicaciones web

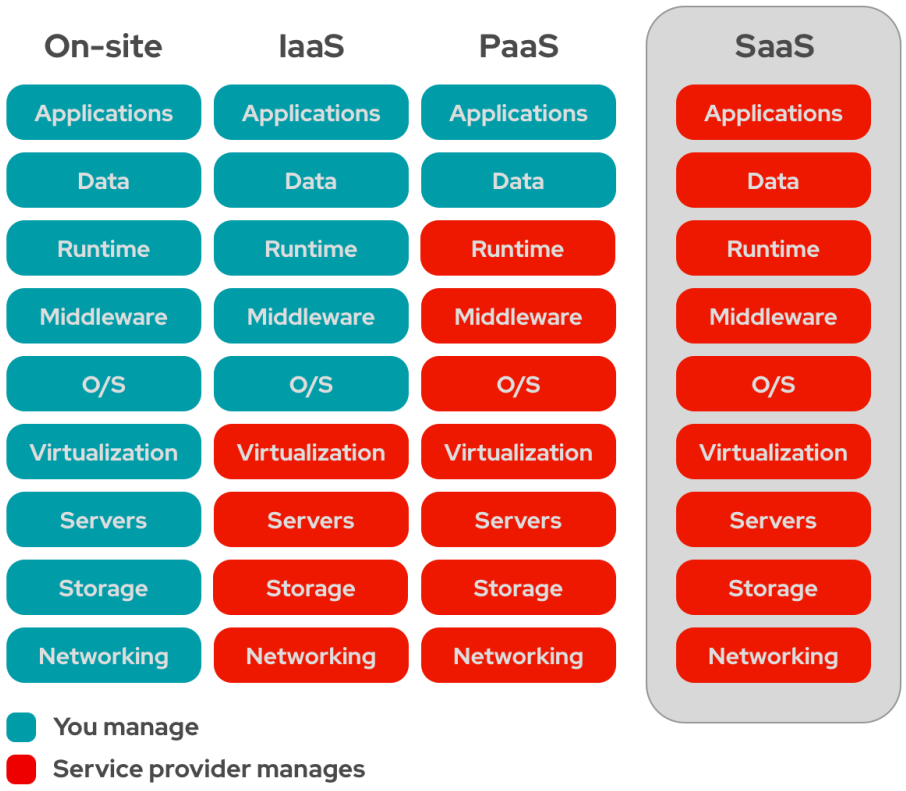
➢ Aplicaciones basadas en microservicios

Con los microservicios las aplicaciones se dividen en elementos más pequeños y elementos.

Es más fácil diseñar, probar, implementar y actualizar microservicios que aplicaciones monolíticas.

➢ SaaS: Sofware as a Service

El software como servicios es un modelo de distribución donde tu pagas por un software y después lo puedes usar SIN tener que instalar nada NI configurar nada. En la siguiente imagen hay una comparación con otros modelos con lo que se verá clara la diferencia.



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

El control de acceso un proceso de seguridad que verifica quien es el usuario que accede a la aplicación y que permisos tiene.

Se basa en:

Autenticación: Verificar que el usuario es quien dice ser

Autorización: Determinar que acciones puede realizar cada usuario

Registro y supervisión: Registra la actividad de los usuarios.

➢ Validación de entrada de datos a una aplicación Web

Es el proceso de comprobación de los datos introducidos por el usuario. Se debe hacer en el lado del servidor y también se puede hacer de forma secundaria y opcional en el cliente. La validación no permite que el usuario meta valores incorrectos que hagan fallar la aplicación.

➢ Posicionamiento de una aplicación Web

El posicionamiento web, también conocido como SEO se basa en la optimización en los motores de búsqueda para intentar que nuestro sitio web salga en los primeros puestos de los resultados de las búsquedas. Cosas que puedes cambiar para mejorar el SEO

- los títulos
- las descripciones
- el contenido
- las imágenes
- la forma en que se enlazan las páginas del sitio entre sí
- Etc.
- Etiquetas meta

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

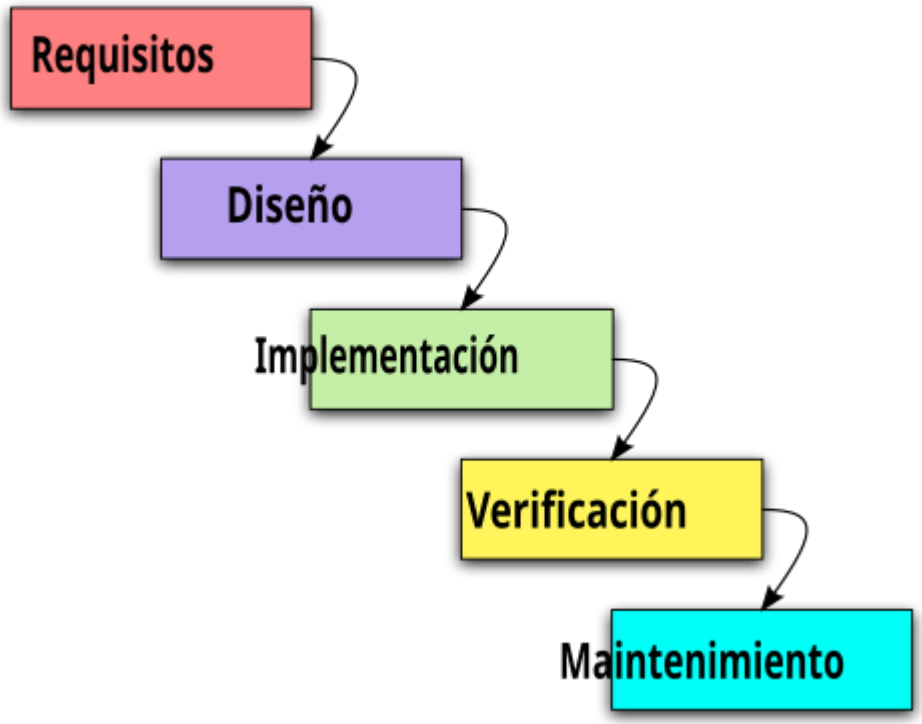
https://www.databot.es/blog/historia-diseno-web

➢ Filosofías de desarrollo del software

Hay muchas filosofías en el desarrollo de software, vamos a explicar algunas de las más relevantes

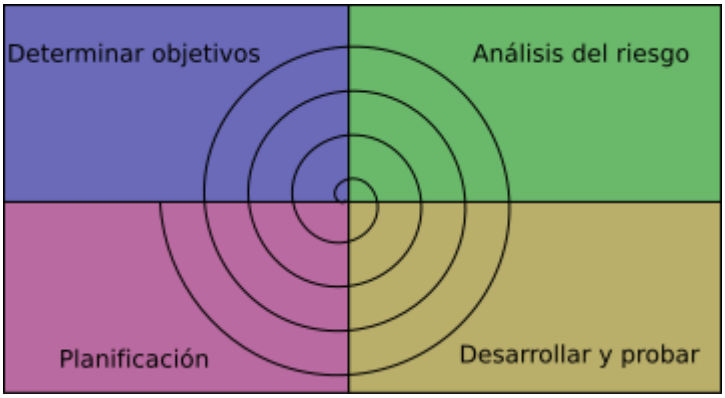
Desarrollo en cascada:

También llamado desarrollo secuencial, es un enfoque en el que se ordena rigurosamente las etapas del proceso de desarrollo, de manera que no se empieza una etapa hasta que no se finaliza la anterior, y nunca se vuelve atrás.



Desarrollo en espiral:

El desarrollo en espiral se basa en bucles sobre las fases, revisando los bucles anteriores y pudiendo volver a hacer modificaciones en esos bucles



Desarrollo ágil:

No hay planificación (o si la hay es muy corta) y trata de entregar el producto lo antes posible para satisfacer al cliente, aceptando cambios dando igual el punto del desarrollo en el que se encentra el programa. Estrecha colaboración entre los empresarios que quieren el software y sus desarrolladores

Para profundizar en alguna de las cuestiones o ver otras filosofías:

https://es.wikipedia.org/wiki/Anexo:Filosof%C3%ADas_del_desarrollo_de_software