ESTUDIO TEMA 1

CONTENIDO

REALIZA UN ESTUDIO SOBRE LOS SIGUIENTES CONCEPTOS.

1. PROTOCOLOS DE COMUNICACIONES

Añadir el concepto de socket

Los **protocolos de comunicaciones** son los encargados de determinar cómo deben circular los mensajes dentro de una red. Son conjuntos de normas que permiten la comunicación entre ordenadores, estableciendo la forma de identificación de estos en la red, la forma de transmisión de los datos y la forma en que la información debe procesarse.

IP, *Internet Protocol* o Protocolo de Internet

Este es un protocolo de la capa de red, cuya función es el **envío de paquetes de datos**, llamados datagramas, tanto a nivel local como a través de las redes.

Es un **protocolo no orientado a conexión**, lo que significa que el envío de los datos tratará de realizarse del mejor modo posible, pero sin garantías de que vaya a alcanzarse el destino final, siendo un protocolo de comunicación no fiable.

Para realizar el envío de paquetes de datos, este protocolo asigna una **dirección IP** a las máquinas de origen y destino. Esta dirección IP, será el identificador del dispositivo en el proceso de comunicación.

TCP, *Transmission Control Protocol* o Protocolo de Control de Transmisión

Este es un **protocolo orientado a conexión**, que permite un transporte fiable y bidireccional de los datos. En la pila de protocolos TCP/IP, este actúa como capa de transporte de datos entre el protocolo de red y la aplicación.

Mediante su uso, las aplicaciones pueden comunicarse de forma segura, puesto que el protocolo TCP se encarga de que los datos que emite el cliente sean recibidos por el servidor sin errores y en el mismo orden en que fueron emitidos, a pesar de trabajar con los servicios de la capa IP, la cual no es confiable.

HTTP, *Hypertext Transfer Protocol* o Protocolo de Transferencia de Hipertexto

Este es un protocolo de la capa de aplicación, que permite las **transferencias de información en la web**, definiendo la sintaxis y la semántica que utilizan los elementos de software de la arquitectura web para comunicarse.

Opera siguiendo el esquema petición-respuesta entre un cliente y un servidor, y es un **protocolo sin estado**, es decir, no guarda ninguna información sobre conexiones anteriores.

HTTPS, *Hypertext Transfer Protocol Secure* o Protocolo de Transferencia de Hipertexto Seguro

Este es un protocolo de aplicación **basado en el protocolo HTTP**, destinado a la transferencia segura de datos de hipertexto, es decir, es la versión segura de HTTP.

El sistema HTTPS utiliza un cifrado basado en la seguridad de textos SSL/TLS para crear un canal cifrado más apropiado para el tráfico de información sensible que el protocolo HTTP.

2. MODELO DE COMUNICACIONES CLIENTE – SERVIDOR

La **arquitectura cliente-servidor** es un modelo de diseño de software con dos partes claramente diferenciadas, el cliente y el servidor, que se reparten las tareas. En este modelo de comunicaciones, un cliente realiza una petición al servidor, que la procesa y envía una respuesta.

Este tipo de arquitectura nos permite conectar a varios clientes a los servicios que provee un servidor, lo cual es imprescindible para la mayoría de aplicaciones y servicios web hoy en día.

¿Cuál es su relación con las aplicaciones web?

Cualquier aplicación web que queramos desarrollar va a necesitar utilizar un servidor que haga uso del protocolo http, es decir, todas las aplicaciones web que hagamos, utilizarán una arquitectura cliente-servidor.

3. MÉTODOS DE PETICIÓN HTTP/HTTPS

Los métodos de petición o *HTTP verbs*, nos permiten **indicar la acción que se desea realizar sobre un recurso** determinado. HTTP define una serie predefinida de métodos que pueden utilizarse, los más usados son los siguientes.

GET

El método GET solicita la representación de un recurso específico. Las peticiones que usan el método GET sólo deben recuperar datos.

HEAD

El método HEAD pide una respuesta idéntica a la de una petición GET, pero sin el cuerpo de la respuesta. Es útil para recuperar los metadatos de los encabezados de respuesta, sin tener que transportar todo el contenido.

POST

El método POST se encarga de enviar datos para que sean procesados por un recurso en específico. Semánticamente, está orientado a la creación de nuevos recursos.

PUT

El modo PUT también envía datos al servidor, pero en lugar de especificar el recurso que los procesará, se encarga de identificar los propios datos. Semánticamente, está más orientado a la actualización de los contenidos.

DELETE

El método DELETE borra un recurso en específico.

CONNECT

El método CONNECT se utiliza para saber si se tiene acceso al servidor especificado.

OPTIONS

El método OPTIONS devuelve los métodos HTTP que soporta un servidor específico.

TRACE

El método TRACE solicita al servidor que introduzca en la respuesta todos los datos que reciba en el mensaje de petición. Se utiliza con fines de depuración y diagnóstico ya que el cliente puede ver lo que llega al servidor y de esta forma ver todo lo que añaden al mensaje los servidores intermedios.

PATCH

El método PATCH se utiliza para actualizar de manera parcial uno o varios recursos, sobrescribiéndolos.

Ejemplos: https://yosoy.dev/peticiones-http-get-post-put-delete-etc/

URI, URN Y URL: DEFINICIÓN, SINTAXIS Y EJEMPLOS

URI, UNIFORM RESOURCE IDENTIFIER O IDENTIFICADOR UNIFORME DE RECURSOS

Un URI es una cadena de caracteres que nos permite identificar recursos en una red. Gracias a esta sintaxis, el sistema **identifica a qué información debe acceder, así como dónde y cómo**.

Las URI constan de cinco partes:

- El **esquema** o *scheme*, que proporciona información sobre el protocolo utilizado.
- La **autoridad** o *authority*, que identifica el dominio.
- La **ruta** o *path*, que señala la dirección exacta en la que se encuentra el recurso.
- La **consulta** o *query*, que representa la acción de la consulta.
- El **fragmento** o *fragment*, que designa una parte del recurso principal.

Estructura: scheme :// authority path ? query # fragment

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

URN, Uniform Resource Name o Nombre de Recurso Uniforme

Tanto las URNs como las URLs son un tipo de URIs. En el caso de las URN, estas funcionan asignando un **código de identificación exclusivo a cada recurso** (algo así como el DNI de una persona), de forma fiable y permanente.

Las URN utilizan la sintaxis de URI, aunque con sus propias particularidades:

- La información sobre el **esquema URN**.
- El NID o **identificador de espacio de nombre**s.
- El NSS o **cadena específica del espacio de nombres**, que identifica el objeto en cuestión.

Estructura: urn:<nid>:<nss></nss></nid>

Ejemplo: urn:nbn:de:101:1-201907280240175770291 3

URL, Uniform Resource Locator o Localizador de Recursos Uniforme

Una URL es un tipo de URI que utilizamos para el **direccionamiento de páginas web**. Los recursos referidos por las URL pueden cambiar, es decir, los contenidos que se encuentran en la dirección a la que esta apunta pueden cambiar.

El esquema de URL se basa en la sintaxis de URI, aunque dividen su autoridad en segmentos más específicos:

- **User y password**, que contienen el nombre de usuario y la contraseña de una persona autorizada para acceder al recurso y solo son necesarios si el recurso exige una autenticación.
- El **host**, que indica el dominio en el que se encuentra el recurso.

- El **puerto**, que debe especificarse cuando no se haya definido un puerto general o no se deba utilizar un puerto estándar para las transmisiones.

Estructura: Schema:[//[user[:password]@]host[:port]][/path][?query][#fragment]

Ejemplo: http://www.example.org/index/pagina1

4. MODELO DE DESARROLLO DE APLICACIONES MULTICAPA

La **arquitectura multicapa** es un modelo de desarrollo de software que estructura el código de las aplicaciones en distintas capas o niveles.

El objetivo de dividir en capas el diseño de una aplicación es puedan separarse las diferentes funciones lógicas de la misma, de forma que podamos distribuir las tareas de creación de la aplicación por niveles, formando grupos de trabajo independientes.

En una aplicación que utiliza este tipo de arquitectura, podremos distinguir, de forma general, tres capas o niveles:

La **capa de presentación**, que se encarga de dar formato a los datos para presentárselos al usuario final. Es la interfaz gráfica de la aplicación. Esta capa se comunica únicamente con la capa de negocio.

La **capa de negocio**, que es donde se reciben las peticiones del usuario, se ejecutan los programas y se envían las respuestas tras el proceso. Aquí es donde se establecen todas las reglas que deben cumplirse. Esta capa, se comunica con la capa de presentación para recibir las peticiones, y con la capa de acceso a datos para solicitar al gestor de la base de datos que almacene o recupere información. **"Es donde se gestiona la lógica de la aplicación"**

La **capa de acceso a datos**, que es la encargada de almacenar la información de la aplicación y recuperarla cuando sea necesario. Esta capa está formada por uno o más gestores de bases de datos que reciben peticiones desde la capa de negocio.

5. MODELO DE DIVISIÓN FUNCIONAL FRONT-END Y BACK-END

A la hora de desarrollar una aplicación web, necesitaremos diferenciar entre el front-end y el back-end.

FRONT END

El **front-end** es la parte de la aplicación que pensamos y programamos para la interacción con los usuarios normales. Es la interfaz de nuestra aplicación, que contendrá toda la

información gráfica: estilos, colores, fondos, tamaños, animaciones... y deberá ser lo más atractiva posible, puesto que está orientada a la interacción con los usuarios.

Además, será la parte responsable de recolectar los datos de entrada del usuario y transformarlos ajustándolos a las especificaciones que demanda el back-end para poder procesarlos.

BACK END

El **back-end** es la parte de la aplicación que programamos para que sea utilizada por los usuarios especiales, es decir, aquellos que tendrán permisos para administrar la web, o añadir contenido. Es lo que conocemos como el lado del servidor.

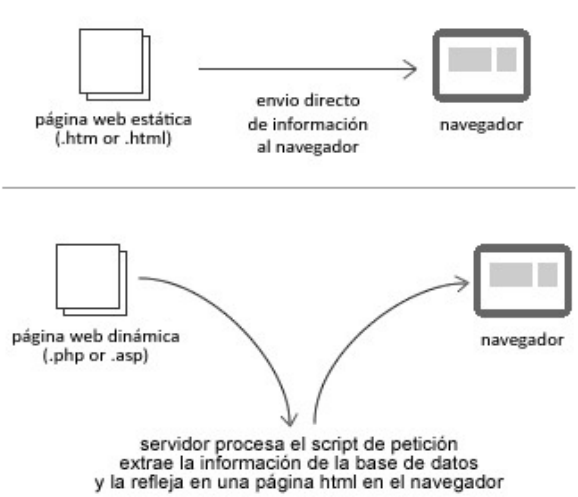
Cuando hablamos de back-end, estamos hablando de la parte del desarrollo de nuestra web que se encargará de que todo funcione, es decir, será el encargado de llevar a cabo todas las funciones que simplifiquen el proceso de desarrollo, de las conexiones con las bases de datos, de la gestión de las librerías, de la optimización de los recursos y de la seguridad de nuestra aplicación.

Tanto el front-end como el back-end **son esenciales para la construcción de una aplicación web**. Abarcan distintos aspectos del desarrollo de la misma, pero deben trabajar juntos, de forma coordinada, para que esta se entienda y funcione correctamente.

6. PÁGINAS WEB ESTÁTICAS Y DINÁMICAS, APLICACIONES WEB Y MASHUP

PÁGINA WEB

Una **página web** es un documento electrónico, capaz de contener texto, imágenes, sonido, vídeo, enlaces, programas.... al que se puede acceder a través de un navegador. Generalmente, se encuentran en formato HTML o XHTML y pueden estar almacenadas en un ordenador, o en un servidor web. Además, cuando hablamos de páginas web, podemos diferenciar entre páginas web estáticas y páginas web dinámicas.



Una **página web estática** es aquella enfocada principalmente a mostrar una información permanente. Son funcionalmente básicas, con poca programación, principalmente en lenguajes HTML, CSS o JavaScript.

Una **página web dinámica** ofrece una mayor interactividad con los usuarios que la visitan y permite la creación de aplicaciones dentro de la página web. Su creación es más compleja ya que utiliza lenguajes de programación y programas de gestión de bases de datos.

Dentro de las páginas web dinámicas, podemos distinguir **dos tipos**: aquellas que están escritas en HTML, pero contienen código que debe ser ejecutado por el navegador (normalmente en JavaScript); y otras que, en lugar de almacenar en el servidor las páginas que se van a mostrar en el navegador, son generadas por la ejecución de un programa.

Características: página web estática vs página web dinámica

Realizadas en HTML o XHTML	Utilizan varias técnicas de programación
Proceso de actualización lento y manual	Proceso de actualización sencillo
Ausencia de movimiento y funcionalidades	Múltiples funcionalidades: bases de datos, foros…
Es necesario acceder al servidor para realizar cambios	Pueden concedérsele derechos de administrador a los usuarios, para que puedan alterar el diseño o los contenidos

APLICACIÓN WEB

Una **aplicación web** es una página web dinámica que ejecutamos en un servidor web y mostramos en un navegador. Las aplicaciones web contienen elementos que permiten que el usuario acceda a los datos de modo interactivo, gracias a que la página responderá a cada una de sus acciones.

Las aplicaciones web tienen grandes ventajas, como por ejemplo que no necesitan ningún tipo de instalación, que son multiplataforma y multidispositivo, que no necesitamos tener un ordenador muy potente para utilizarlas, puesto que la mayor parte del peso de estas lo soporta el servidor en el que están alojadas, y que son adaptables, intuitivas y muy fáciles de actualizar.

MASHUP

En desarrollo web, una **mashup** es una forma de integración y reutilización. Los mashup son aplicaciones web que utilizan los servicios que ofrecen otras aplicaciones web, con el fin de reutilizar su contenido o funcionalidad.

Un ejemplo de mashup sería, por ejemplo, la página web de un hotel, que utilice el servicio de Google Maps para integrar un mapa con la ubicación del mismo.

7. COMPONENTES DE UNA APLICACIÓN WEB.

Una aplicación web que reside en uno o varios servidores, normalmente está estructurada en varias capas o componentes: **servidor web**, **módulo encargado de ejecutar el código**, un **sistema gestor de bases de datos** y el **lenguaje de programación**. El primero de ellos mantiene la comunicación con el navegador del usuario que es quién proporciona la interfaz de acceso al servicio web. El módulo que ejecuta el código está formado por los scripts o programas que se ejecutan en el servidor para atender las solicitudes del usuario, el gestor de base de datos es el repositorio en el que se guardan los datos con los que trabaja el servicio y el lenguaje de programación que se va a utilizar para desarrollar la aplicación.

Las capas pueden estar montadas sobre una o varias máquinas dependiendo del volumen de carga que tenga que soportar el servicio, y todo corriendo sobre algún sistema operativo.

SERVIDOR WEB

El servidor web se encarga de la comunicación a través de la red con el navegador del usuario.

Debe conocer el procedimiento a seguir para generar la página web: qué módulo se encargará de la ejecución del código y cómo se debe comunicar con él.

Normalmente escucha en el puerto TCP 80 cuando se trata de una conexión HTTP, y en el TCP 443 cuando se trata de una HTTPS, aunque este parámetro es configurable.

Cuando recibe una petición del usuario puede atenderla:

- de manera estática cuando contesta con un fichero que está en el sistema de archivos (la descarga de una imagen, un fichero css o html)
- o de manera dinámica cuando se ejecuta algún programa y se envía al cliente el resultado de dicha ejecución.

El servidor web evidentemente es capaz de atender muchas peticiones de forma concurrente. Pudiendo realizar diferentes funciones habituales entre las que se encuentran:

- Registro de actividad y errores
- Control de acceso basado en la dirección del cliente, contenido o usuario/contraseña
- Virtual Hosts, para mantener diferentes webs (por ejemplo http://elpuig.xeill.net y http://blog.elpuig.xeill.net)
- Proxy, para reenviar las peticiones a otro servidor
- Reescritura de URLs

Aunque existen diferentes alternativas **Apache** es el servidor web más utilizado en Internet. Se trata de una aplicación libre, robusta y modular con una impresionante colección de funciones, pudiendo servir tanto para contenido estático como para prácticamente cualquier contenido dinámico.

Otras alternativas pueden ser:

Cherokee: Un servidor web más moderno que Apache con la pretensión de ser eficiente y fácil de configurar (tiene una interfaz web de administración). Está bajo la GPL.

NGINX: Orientado a sitios con una gran carga de trabajo en los que es necesario un gran rendimiento.

Tomcat: Contenedor de servlets para aplicaciones web escritas en Java.

MÓDULO ENCARGADO DE EJECUTAR EL CÓDIGO

Este módulo o programa es el encargado de generar la página web resultante, debe integrarse con el servidor web y dependerá del lenguaje y tecnología que utilicemos para programar la aplicación web.

Antes los servidores web solo permitían visualizar información estática, con la llegada de la publicidad y la necesidad de crear aplicaciones web dinámicas se buscó desarrollar una tecnología que permitiera ejecutar, en un único proceso del servidor, todos los pedidos de ejecución de código sin importar la cantidad de clientes que se conectaban concurrentemente. Así surgieron los denominados servlets, basados en la tecnología Java de Sun Microsystems, y los filtros ISAPI de Microsoft. Estos permitían ejecutar código en un único proceso externo que gestionaba todas las integraciones realizadas por el servidor web, impidiendo al mismo tiempo que el servidor web pueda ejecutar programas del sistema operativo.

SISTEMA GESTOR DE BASES DE DATOS

La aplicación de bases de datos normalmente también será un servidor.

Una web sencilla puede trabajar únicamente con ficheros, pero en cuanto el servicio deja de ser trivial, que las aplicaciones empiecen a utilizar grandes cantidades de datos, aparece la necesidad de utilizar un sistema gestor de bases de datos que organice la información con la que se trabaja.

Entre los modelos de Bases de datos podemos diferenciar:

SGBD relacional: La más extendida y con una larga vida por delante. Guardan la información en registros de tablas y mediante SQL se realizan consultas u operaciones para manipular los datos. Dependiendo de la herramienta utilizada hay grandes diferencias en cuanto a sus capacidades y administración, pero en general, para las aplicaciones web prima la velocidad sobre las características.

Ejemplos de uso: MySQL y PostgreSQL

SGBD orientado a objetos: En lugar de almacenar registros, directamente guardan y recuperan objetos. De manera que cuando se utiliza un lenguaje de programación orientado a objetos nos ahorramos el trabajo de adaptar las entidades con las que trabajamos al modelo relacional.

Ejemplos de uso: DB4O y Zope Object Database

SGBD NoSQL: Aquí se agrupan un conjunto de técnicas diferentes que tienen en común que no cumplen con los requisitos ACID (ACID es un grupo de 4 propiedades que garantizan que las transacciones en las bases de datos se realicen de forma confiable: atomicidad, consistencia, aislamiento y durabilidad) y/o no estructuran la información en tablas como las bases de datos relacionales. La ventaja es que consiguen romper algunas de las limitaciones de los sistemas relacionales en cuanto a escalabilidad y rendimiento permitiendo crear ingentes bases de datos distribuidas.

Ejemplos de uso: Apache Cassandra y Apache CouchDB

LENGUAJES DE PROGRAMACIÓN

Los programas que determinan qué se debe responder a las solicitudes del usuario, pueden estar escritos en una gran variedad de lenguajes de programación.

Una posible clasificación es: lenguajes compilados y lenguajes interpretados o scripts.

Los **lenguajes compilados** obligan a compilar el código fuente antes de obtener un ejecutable, en líneas generales priman el rendimiento en tiempo de ejecución y cuando

una aplicación se debe utilizar en diferentes plataformas debe ser compilada para cada una de ellas. Ejemplos típicos son los lenguajes C, C++ y Java.

Los **lenguajes interpretados**, o scripts, no necesitan ser compilados pues un programa llamado intérprete realiza las operaciones indicadas por el código fuente. En líneas generales permiten un desarrollo más rápido, aunque no alcanzan en velocidad de ejecución a los lenguajes compilados. Ejemplos típicos son Bash, Perl, PHP, Python y Ruby.

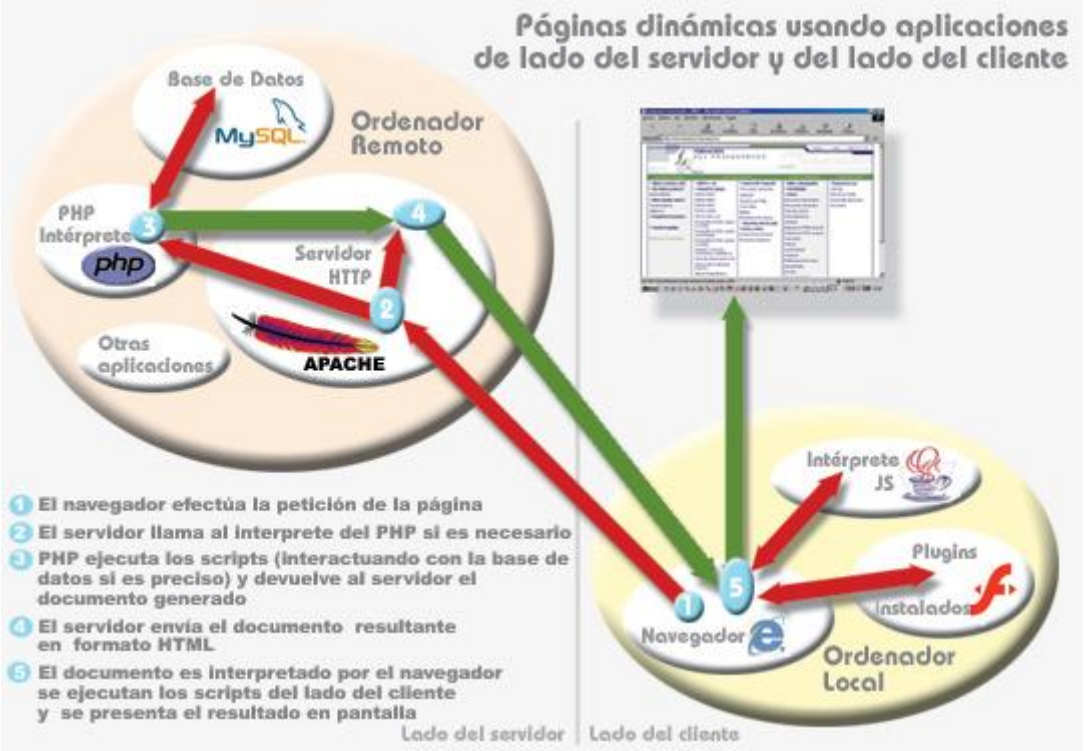
Falta revisión

8. PROGRAMAS EJECUTADOS EN EL LADO DEL CLIENTE Y PROGRAMAS EJECUTADOS EN EL LADO DEL SERVIDOR – LENGUAJES DE PROGRAMACIÓN EN CADA CASO.

Para realizar una aplicación web podemos utilizar lenguajes del lado cliente o lado servidor.

Los lenguajes de programación de lado cliente (entre los que se encuentra HTML, JavaScript…) son aquellos que pueden ser directamente procesados por el navegador y no necesitan de un pretratamiento.

Los lenguajes de lado servidor son aquellos que son reconocidos, ejecutados e interpretados por el propio servidor y que se envían al cliente en un formato comprensible



para él.

El código que se ejecuta en el lado cliente tiene como principal interés la mejora de la apariencia y el comportamiento de la página web, es decir, la selección de estilos, componentes UI, creación de layouts, navegación.....

Este código del lado cliente está escrito usando **HTML, CSS y JavaScript**. Para ello necesitaremos un IDE (NetBeans, Eclipse...)

La programación de lado servidor en su mayor parte implica la elección de qué contenido se ha de devolver al navegador como respuesta a sus peticiones. Este código gestiona también bases de datos para almacenar.

Este código del lado servidor está escrito en **PHP, Perl, ASP**, y la gestión de datos en **MySQL, MariaDB**. Para realizar este código necesitamos tener Apache, PHP, MySQL o MariaDB, un Git para los respositorios.

Falta revisión

9. LENGUAJES DE PROGRAMACIÓN UTILIZADOS EN EL LADO SERVIDOR DE UNA APLICACIÓN WEB (CARACTERÍSTICAS Y GRADO DE IMPLANTACIÓN ACTUAL).

Profundizamos en las tecnologías que se utilizan del lado del servidor: Spring, XAMPP, .NET, MEAN.

10. CARACTERÍSTICAS Y POSIBILIDADES DE DESARROLLO DE UNA PLATAFORMA XAMPP.

Es la forma que tengo yo de instalar rápidamente un servidor web en mi ordenador, con todos sus componentes.

11. 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.

12. REALIZA UN ESTUDIO SOBRE LOS IDE MÁS UTILIZADOS

Los **entornos de desarrollo** o IDEs son aplicaciones informáticas que facilitan el desarrollo de software. Están diseñados para maximizar la productividad del programador, ofreciendo, en un mismo programa, herramientas para la creación, modificación, compilación, implementación y depuración de sus aplicaciones.

Existen muchos entornos de desarrollo distintos, cada uno de ellos con sus particularidades, pero todos ellos tienen ciertas características en común. Normalmente, los IDE ponen a nuestra disposición, como mínimo:

- Herramientas de **resaltado de texto**, que se adaptan al lenguaje que estamos utilizando.
- **Completado automático**.
- Herramientas para facilitar la **navegación en el texto**.
- **Comprobación de errores**.
- **Generación automática de código**.
- **Ejecución y depuración** de programas.
- **Gestión de versiones**.

Además, es importante que sean **multiplataforma**, que tengan una **interfaz práctica** para trabajar y que cuenten con un **asistente de ayuda** y con **foros** donde los usuarios puedan plasmar sus dudas.

NetBeans

NetBeans es un entorno de desarrollo integrado libre, orientado principalmente para el lenguaje de programación Java (aunque ofrece la posibilidad de instalar módulos que nos permiten trabajar con otros lenguajes).

Esta plataforma nos permite crear nuestro software mediante módulos que se desarrollan independientemente, lo que facilita la ampliación de las aplicaciones.

Otras de sus características son:

- Sistema de proyectos basado en Ant.
- Control de versiones.
- Sistema de refactorización.

Eclipse

Eclipse es un entorno de desarrollo integrado de código abierto multiplataforma. Emplea un sistema de módulos que nos permite incluir únicamente las funcionalidades que necesitemos.

El proyecto Eclipse se define a sí mismo como: *"Una especie de herramienta universal – Un IDE abierto y extensible para todo y nada en particular"*.

Entre sus características, están las siguientes:

- Un editor de texto con analizador sintáctico.
- Compilación en tiempo real.

- Pruebas unitarias con Junit.
- Control de versiones con CVS.
- Integración con Ant.
- Asistentes para la creación de proyectos.
- Sistema de refactorización.

Además, nos permite añadir mediante plugins otros lenguajes de programación, otros sistemas de control de versiones u otros sistemas de integración.

VISUAL STUDIO

Visual Studio es

Características

Ofrece opciones para depurar el código, generar perfiles y emitir diagnósticos de forma fácil.

Permite integración de pruebas.

Permite acceder a un marketplace para instalar extensiones.

Permite utilizar el control de versiones de Git para rastrear y guardar los cambios en los ficheros.

Permite desarrollar e implementar bases de datos de SQL Server y Azure SQL fácilmente.

Permite desarrollar extensiones propias.

Permite crear, administrar e implementar aplicaciones de escala de nube en Azure.

Permite desarrollar tanto aplicaciones nativas como híbridas para Android, iOS y Windows.

Es multiplataforma.

Permite desarrollar con múltiples lenguajes de programación.

Utiliza la tecnología IntelliSense.

Utiliza gráficos de vanguardia.

PHPSTORM

Características

Php, html, javascript

Compatibilidad con diversas versiones de PHP

Función de autocompletado de código, clases, métodos, los nombres de las variables y las palabras clave de PHP, además de los nombres utilizados comúnmente para los campos y las variables según su tipo.

Soporte de estilo de codificación (PSR1/PSR2, Drupal, Symfony, Zend).

Soporte PHPDoc

Detector de código duplicado.

PHP Code Sniffer (phpcs)

Refactorizaciones (Cambiar nombre, Introducir variable, Introducir constante, Introducir campo, Variable en línea, Mover miembro estático, Extraer interfaz).

Edición de plantillas de Smarty y Twig (resaltado de errores de sintaxis, finalización de funciones y atributos de Smarty, inserción automática de llaves emparejadas, comillas y etiquetas de cierre y más).

Vista MVC para Symfony y frameworks Yii.

Soporte PHAR.

GRADO DE IMPLANTACIÓN ACTUAL

Worldwide, Oct 2020 compared to a year ago:

Rank	Change	IDE	Share	Trend
1	↑	Visual Studio	25.0 %	+3.4 %
2	↑	Eclipse	16.7 %	-1.0 %
3	↓↓	Android Studio	12.18 %	-11.6 %
4	↑	Visual Studio Code	8.49 %	+3.4 %
5	↑	pyCharm	7.58 %	+2.5 %
6	↑	IntelliJ	5.93 %	+1.2 %
7	↓↓↓	NetBeans	5.1 %	-0.3 %
8		Xcode	4.55 %	+0.6 %
9	↑	Atom	3.87 %	+0.8 %
10	↓	Sublime Text	3.75 %	+0.1 %
11		Code::Blocks	1.84 %	+0.4 %
12		Vim	0.96 %	+0.1 %
13	↑	PhpStorm	0.65 %	+0.0 %

HTTPS://PYPL.GITHUB.IO/IDE.HTML

Falta desarrollo

13. SERVIDORES HTTP/HTTPS MÁS UTILIZADOS (CARACTERÍSTICAS Y GRADO DE IMPLANTACIÓN ACTUAL).

Apache y Apache Tomcat. Tabla de servidores web y de quien es cada uno de ellos.
Mención a servidores web portables, para llevar el un pincho

14. APACHE HTTP VS APACHE TOMCAT

Apache orientado a aplicaciones web

Tomcat orientado a java

15. REALIZA UN ESTUDIO SOBRE NAVEGADORES HTTP/HTTPS MÁS UTILIZADOS

¿Qué es un navegador web HTTP/HTTPS?

Los **navegadores web** son un tipo de software que nos permite tener acceso a internet y que interpreta la información de los sitios web para que éstos puedan ser leídos. La función de un navegador es permitir al usuario la visualización de páginas web y la interacción con las mismas.

Google Chrome

Características

Grado de implantación actual

Safari

Características

Grado de implantación actual

Mozilla Firefox

Características

Grado de implantación actual

Microsoft Edge

Características

Grado de implantación actual

OPERA

Características

Grado de implantación actual

Hay que saber jugar con todo lo que nos ofrecen los distintos navegadores.

16. GENERADORES DE DOCUMENTACIÓN HTML (PHPDOC): PHPDOCUMENTOR, APIGEN, ...

17. REPOSITORIOS DE SOFTWARE – SISTEMAS DE CONTROL DE VERSIONES: GIT, CVS, SUBVERSION, ...

Git, GitHub, GitLab

18. 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-W10ED.

Sistema Operativo: Windows 10.

- Servidor administración remota: SSH.
- Servidor de transferencia de ficheros: SFTP (SSH) Puerto 22.
- Repositorio: GitHub.
- Servidor web: Apache 2.4 HTTP
- Gestor de Bases de Datos: MySQL 8.0.
- Navegador: Mozilla Firefox 81.0.
- IDE: NetBeans 12.0.
- Ofimática, multimedia, generador html: LibreOffice 6.4.
- Frameworks PHP: Pendiente de actualizar.
- Cliente SSH: Filezilla 3.50.0.

19. PROPUESTA DE CONFIGURACIÓN DEL ENTORNO DE EXPLOTACIÓN PARA LA ASIGNATURA DE DESARROLLO WEB DEL LADO SERVIDOR EN ESTE CURSO (INCLUYENDO LAS VERSIONES): XXX-USEE

Sistema Operativo: Ubuntu Server 20.04.

- Servidor administración remota: SSH.
- Servidor de transferencia de ficheros: SFTP (SSH) Puerto 22.
- Repositorio: GitHub.
- Servidor Web: Apache 2.4 HTTP
- Gestor de Bases de Datos: MySQL 8.0.
- Cliente SSH: NetBeans 12.0 / Filezilla client 3.50.0.
- Navegador: Mozilla Firefox 81.0.

21.

Año, versión, desarrolladores, características, componentes, funcionalidad

BIBLIOGRAFÍA

PROTOCOLOS DE COMUNICACIONES

https://es.wikipedia.org/wiki/Protocolo_de_comunicaciones

https://definicion.de/protocolo-de-comunicacion/

https://es.wikipedia.org/wiki/Protocolo_de_internet

https://es.wikipedia.org/wiki/Protocolo_de_control_de_transmisi%C3%B3n

https://es.wikipedia.org/wiki/Protocolo_de_transferencia_de_hipertexto

https://es.wikipedia.org/wiki/Protocolo_seguro_de_transferencia_de_hipertexto

MODELO DE COMUNICACIONES CLIENTE-SERVIDOR

https://es.wikipedia.org/wiki/Cliente-servidor

p.8-9 Teoría **Desarrollo Web Entorno Servidor**, por *José Luis Comesaña*.

MÉTODOS DE PETICIÓN

https://es.wikipedia.org/wiki/Protocolo_de_transferencia_de_hipertexto

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

https://diego.com.es/metodos-http

MODELO DE DESARROLLO DE APLICACIONES MULTICAPA

p.8-9 Teoría **Desarrollo Web Entorno Servidor**, por *José Luis Comesaña*.

https://es.wikipedia.org/wiki/Programaci%C3%B3n_por_capas

MODELO DE DIVISIÓN FUNCIONAL FRONT-END Y BACK-END

https://es.wikipedia.org/wiki/Front_end_y_back_end

https://nestrategia.com/desarrollo-web-back-end-front-end/

https://descubrecomunicacion.com/que-es-backend-y-frontend/

PÁGINAS WEB ESTÁTICAS Y DINÁMICAS, APLICACIONES WEB Y MASHUP

https://es.wikipedia.org/wiki/P%C3%A1gina_web

p.3 Teoría **Desarrollo Web Entorno Servidor**, por *José Luis Comesaña*.

https://brandmedia.es/diferencias-pagina-web-estatica-dinamica-mejor/

https://www.antevenio.com/blog/2017/11/web-estatica-vs-dinamica-que-es-mejor-para-seo/

p.5 Teoría **Desarrollo Web Entorno Servidor**, por *José Luis Comesaña*.

https://www.neosoft.es/blog/que-es-una-aplicacion-web/

https://es.wikipedia.org/wiki/Mashup_(aplicaci%C3%B3n_web_h%C3%ADbrida)

https://sg.com.mx/content/view/256

SERVIDORES WEB

p.8 Teoría **Desarrollo Web Entorno Servidor**, por *José Luis Comesaña*.

https://elpuig.xeill.net/Members/vcarceler/asix-m09/uf1/nf1/a1

https://dosideas.com/noticias/base-de-datos/973-acid-en-las-bases-de-datos

https://es.wikipedia.org/wiki/Script_del_lado_del_servidor

APLICACIONES WEB

p.8 Teoría **Desarrollo Web Entorno Servidor**, por *José Luis Comesaña*.

LADO DEL CLIENTE Y LADO DEL SERVIDOR

http://www.adelat.org/media/docum/nuke_publico/lenguajes_del_lado_servidor_o_cliente.html

https://developer.mozilla.org/es/docs/Learn/Server-side/Primeros_pasos/Introducci%C3%B3n

ENTORNOS DE DESARROLLO

p.15-16 Teoría **Desarrollo Web Entorno Servidor**, por *José Luis Comesaña*.

https://es.wikipedia.org/wiki/Entorno_de_desarrollo_integrado

https://www.arimetrics.com/glosario-digital/entorno-de-desarrollo

https://es.wikipedia.org/wiki/NetBeans

https://es.wikipedia.org/wiki/Eclipse_(software)

https://conectasoftware.com/apps/visual-studio/

https://www.linuxadictos.com/phpstorm-un-excelente-ide-para-php-multiplataforma.html

https://pypl.github.io/IDE.html

NAVEGADORES WEB

https://es.wikipedia.org/wiki/Navegador_web

REALIZADO EN COLABORACIÓN CON SONIA ANTÓN LLANES

Susana: 1, 2, 3, 4, 5, 12

Sonia: 3.1, 6, 7, 8

Todos los ejercicios realizados por Sonia son revisados y ampliados por mí.