TEMA 1 – DESARROLLO WEB EN ENTORNO SERVIDOR

DAW2 - DWES

CRISTIAN MATEOS VEGA

Contenido

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

- **IP**: Se encarga de la dirección y envío de datos entra dispositivos. Usa direcciones IP para identificar origen y destino y no garantiza la entrega. (Capa de red)
- **TCP**: Establece la conexión fiable entre dos dispositivos. Controla orden, integridad y transmisión de los datos y los divide en segmentos asegurando que lleguen completos y en orden. (Capa de transporte)
- **HTTP**: Es el protocolo que usan los navegadores y servidores web para pedir y enviar páginas web en texto plano. (Capa de aplicación y usa el puerto 80)
- **HTTPS**: Es lo mismo que http, pero con seguridad, cifrando la información pedida y enviada. (Capa de aplicación y usa el puerto 443)

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

Cliente: Es un proceso sencillo que solicita recursos o servicios

Servidor: Es un proceso que ofrece esos recursos o servicios que se han pedido

Funcionamiento:

1. El cliente (navegador) envía una solicitud al servidor (por ejemplo, pedir una página web) mediante HTTP/HTTPS.
2. El servidor procesa esa solicitud, accede a la información necesaria y envía la respuesta (la página web) al cliente.
3. El cliente recibe y muestra esa información para que el usuario pueda interactuar con ella.

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

- **GET**: Solicita la representación de un recurso especifico. Es el método más usado para obtener datos

  Características:

  - No tiene cuerpo en la petición
  - Si se hace la misma petición varias veces devuelve lo mismo
  - Los parámetros suelen enviarse en la URL
  - Se usa para obtener páginas web con recursos estáticos (imágenes, CSS…)

- **POST**: Envía datos al servidor para crear o modificar recursos.

  Características:

  - Puede contener un cuerpo con datos (JSON, formularios, archivos…)
  - Si se hace la misma petición varias veces puede devolver cosas diferentes
  - Se usa para enviar formularios, subir archivos, crear registros en una base de datos…

- **PUT**: Reemplaza un recurso existente o crea uno nuevo si no existe.

  Características:

  - El cuerpo contiene el recurso completo
  - Actualiza completamente un recurso

- **DELETE**: Elimina un recurso específico.

  Características:

  - Se usa para eliminar recursos como usuarios o archivos

- **HEAD**: Obtiene los encabezados de una respuesta, sin el cuerpo del recurso.

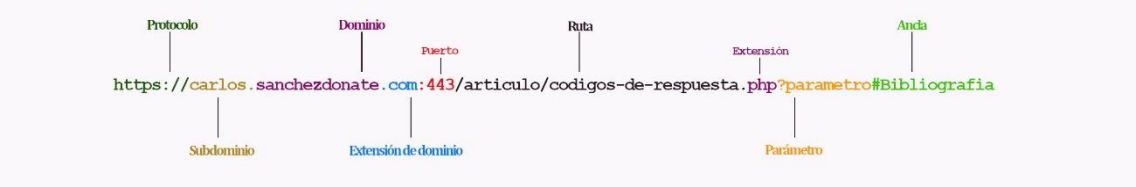
  Características:

  - Sin cuerpo
  - Se trata igual que el GET, pero omite el cuerpo
  - Es muy eficiente para comprobar el estado de un recurso sin descargarlo completamente

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

- **URI (Identificador de Recursos Uniforme):** Es una cadena de caracteres que identifica de manera única un recurso de internet o en cualquier red
  Ejemplo:
  https://www.ejemplo.com
  Explicación detallada de URI

- **URL (Localizador de Recursos Uniforme):** Es un tipo especifico de URI que no solo identifica un recurso, sino que también proporciona los medios para localizarlo, es decir, la dirección o ruta para acceder al recurso)
  Ejemplo:
  https://www.ejemplo.com/index.html
  Explicación detallada de URL

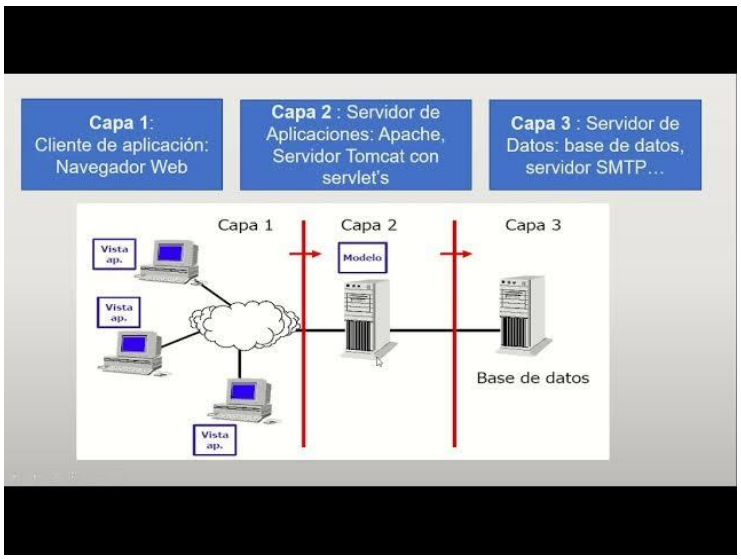


- **URN (Nombre de Recursos Uniforme):** Es otro tipo especifico de URI que identifica un recurso de manera única sin referirse a su ubicación, es decir, es un nombre permanente para el recurso. Estos tienen unos estándares de identificación específicos, como por ejemplo ISBN para los libros
  Ejemplo:
  urn:isbn:979-10-91414-08-1
  Explicación detallada de URN

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

El modelo de desarrollo de aplicaciones multicapa es una forma de organizar una aplicación dividiéndola en varias capas o niveles, donde cada capa tiene una función específica. Esto facilita el mantenimiento, la escalabilidad y la reutilización del código.

- **Capa de Presentación:** Se encarga de mostrar y captar la información
  - Interfaz con el usuario.
  - Muestra datos y recibe inputs.
- **Capa de Lógica de Negocio:** Toma decisiones y ejecuta reglas
  - Procesa la información.
  - Contiene las reglas y operaciones del negocio.
- **Capa de Datos:** Guarda y recupera la información
  - Gestiona el almacenamiento y recuperación de datos.
  - Se conecta con bases de datos o servicios externos.



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

1. **Front-end (lado cliente)**

- Es la parte que ve y usa el usuario directamente en su navegador.
- Incluye todo lo relacionado con la interfaz visual: diseño, botones, menús, animaciones, formularios, etc.
- Se desarrolla principalmente con HTML, CSS y JavaScript.
- Su función es presentar los datos y facilitar la interacción del usuario con la aplicación.
- Puede comunicarse con el back-end para pedir o enviar datos.

2. **Back-end (lado servidor)**

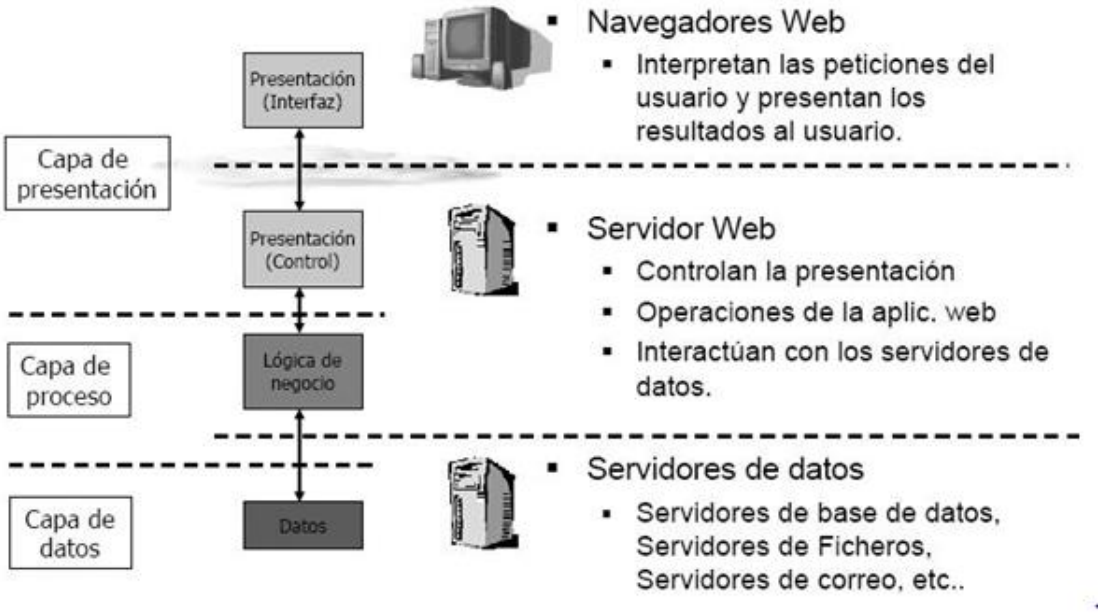
- Es la parte que corre en el servidor y se encarga de la lógica, gestión de datos y seguridad.
- Procesa las solicitudes que llegan desde el front-end, accede a bases de datos, realiza cálculos o validaciones, y prepara las respuestas.
- Se desarrolla con lenguajes como Python, Java, PHP, Node.js, Ruby...
- Proporciona servicios web que el front-end consume para mostrar información actualizada.

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

- **Página web estática**: página con contenido fijo que no cambia a menos que se modifique manualmente.
- **Página web dinámica**: página web cuyo contenido puede cambiar en función del usuario o de datos externos, generalmente mediante programación del lado del servidor.
- **Aplicación Web**: programa completo que se ejecuta con un navegador.
- **Mashup**: aplicación web que combina datos o funcionalidades de dos o más fuentes externas para crear un nuevo servicio.

8. Componentes de una aplicación web.

- **Navegador:** Programa que permite a los usuarios acceder y visualizar páginas web (ej. Chrome, Firefox).
- **Servidor We**b: Computadora o software que almacena, procesa y entrega páginas web a los navegadores.
- **Módulo encargado de ejecutar el código:** Componente del servidor que interpreta y ejecuta código dinámico (ej. PHP, Node.js).
- **Base de datos:** Sistema que almacena y organiza información para ser consultada y manipulada por la web.
- **Control de acceso:** Mecanismo que gestiona permisos y autentica usuarios para restringir acceso a recursos.
- **Ficheros escritos en lenguajes de programación:** Archivos con código fuente que definen la funcionalidad y lógica de la aplicación web.



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

Programas del lado del cliente:

Estos programas se ejecutan directamente en el navegador o dispositivo del usuario. Su función principal es la interacción con el usuario y la manipulación de la interfaz.

Características:

- Se ejecutan en el navegador o dispositivo del usuario.
- Permiten interacción directa con la interfaz.
- No necesitan comunicación con el servidor para funcionar
- Seguridad limitada, ya que el código es visible y accesible para el usuario.

Lenguajes comunes: JavaScript, HTML y CSS, TypeScript…

Programas del lado del servidor:

Estos programas se ejecutan en el servidor, procesan datos, gestionan bases de datos, autenticación, lógica de negocio, etc.

Características:

- Ejecutados en un servidor remoto.
- Control total sobre la lógica, seguridad y acceso a datos.
- Pueden interactuar con bases de datos, servicios externos, etc.
- No son visibles directamente para el usuario final.

Lenguajes comunes: JavaScript (Node.js), Python, PHP, Ruby, Java…

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

1. PHP (73,4%)
2. Ruby (6,4%)
3. Java (5,3%)
4. JavaScript (5%)
5. ASP.NET (4,8%)

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

Características:

- Facilidad de uso y configuración
- Portabilidad y compatibilidad multiplataforma
- Soporte para múltiples lenguajes de programación y bases de datos

Posibilidades de Desarrollo:

- Desarrollo de sitios web dinámicos
- Pruebas locales de aplicaciones web
- Integración con CMS populares como WordPress
- Desarrollo de scripts automatizados
- Configuración personalizada
- Desarrollo colaborativo en local
- Migración y respaldo de bases de datos y proyectos entre entornos

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.

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

1. Visual Studio Code (70%)
   - Ligero, rápido y personalizable con muchas extensiones
2. IntelliJ IDEA (25%)
   - Muy potente para Java y otros lenguajes tiene autocompletado inteligente
3. Eclipse (10-15%)
   - IDE gratuito y abierto, principalmente para Java
4. PyCharm (15%)
   - IDE especializado en Python
5. Android Studio (20%)
   - IDE oficial para desarrollo Android

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

1. Nginx (33,3%)

- Alta eficiencia para manejar tráfico masivo
- Rendimiento ligero y asíncrono
- Funciona también como proxy

2. Apache (25,5%)

- Gran compatibilidad con varias tecnologías y lenguajes
- Facilita la gestión de contenidos dinámicos

3. Cloudflare Server (24,5%)

- Mejora la seguridad, acelera la carga de las páginas y además protege de ataques

4. LiteSpeed (14,8%)

- Muy rápido y eficiente para sitios con mucho tráfico
- Funciona muy bien sobre todo con PHP y WordPress

5. Node.js (5%)

- Usa JavaScript para crear aplicaciones web interactivas y en tiempo real
- Soporta muchas conexiones simultaneas

15. Apache HTTP vs Apache Tomcat

Apache HTTP: Es un servidor web estándar para páginas estáticas, usa php

Apache Tomcat: Es un servidor para páginas dinámicas basadas en servlets y JSP (JavaServer Pages).

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

1. Chrome (69,23%)
2. Safari (14,98%)
3. Edge (5,03%)
4. Firefox (2,26%)
5. Samsung Internet (1,97%)
6. Opera (1,85%)

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

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

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.

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.

21. Realizar un estudio sobre los siguientes conceptos y su relación con el desarrollo de aplicaciones web:

CMS – Sistema de gestión de contenidos:

Es un software que permite a los usuarios crear, gestionar, editar y publicar contenido en un sitio web sin necesidad de programar esa web.

Ejemplos: WordPress, Joomla, Shopify, Wix…

ERP – Sistema de planificación de los recursos empresariales

Es un software que unifica y automatiza todos los procesos de una empresa en una sola plataforma utilizando una base de datos centralizada.

Ejemplos: SAP, Microsoft Dynamics…

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**