TEMA 1 – DESARROLLO WEB EN ENTORNO SERVIDOR

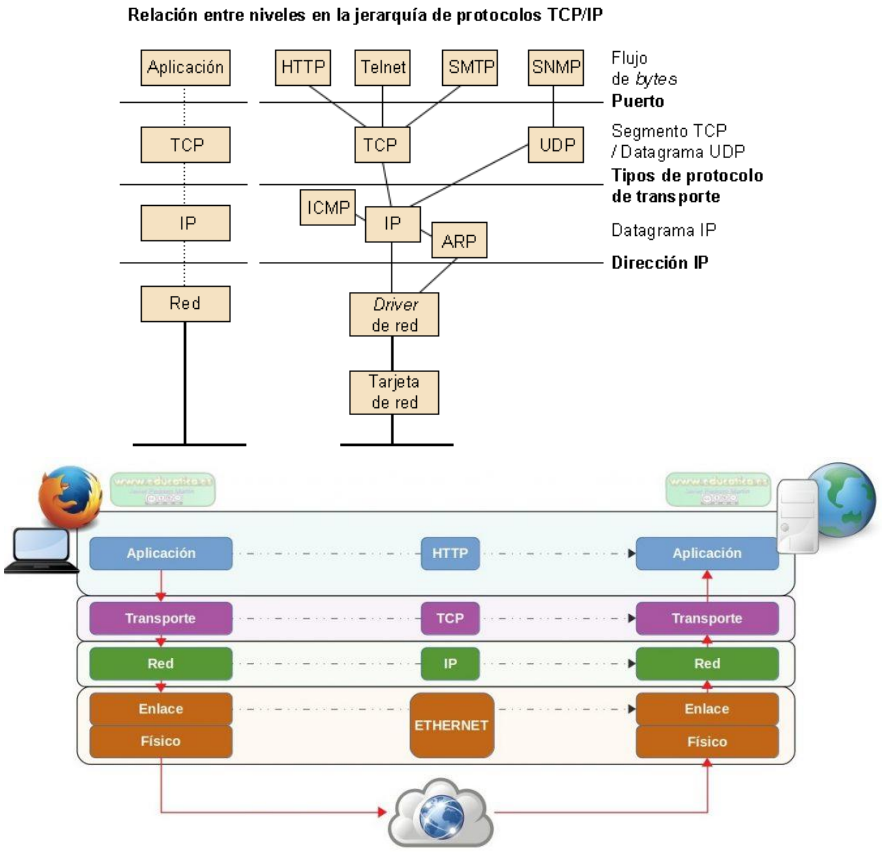
DAW2 - DWES

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 internet)
- **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)



[Explicación Detallada]

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

Cliente: Es el dispositivo o programa que solicita un servicio o información.

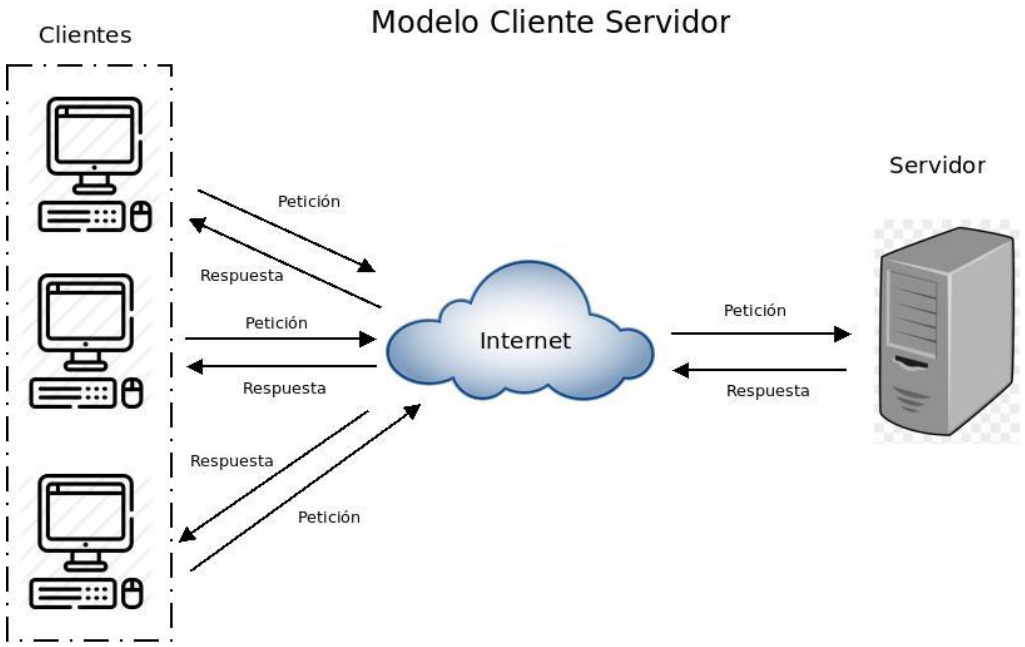
Ejemplo: tu navegador web (Chrome, Firefox, etc.).

Servidor: Es el sistema o programa que recibe las solicitudes, las procesa y envía una respuesta.

Ejemplo: el servidor donde está guardada una página web.

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.



Explicación detallada

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.

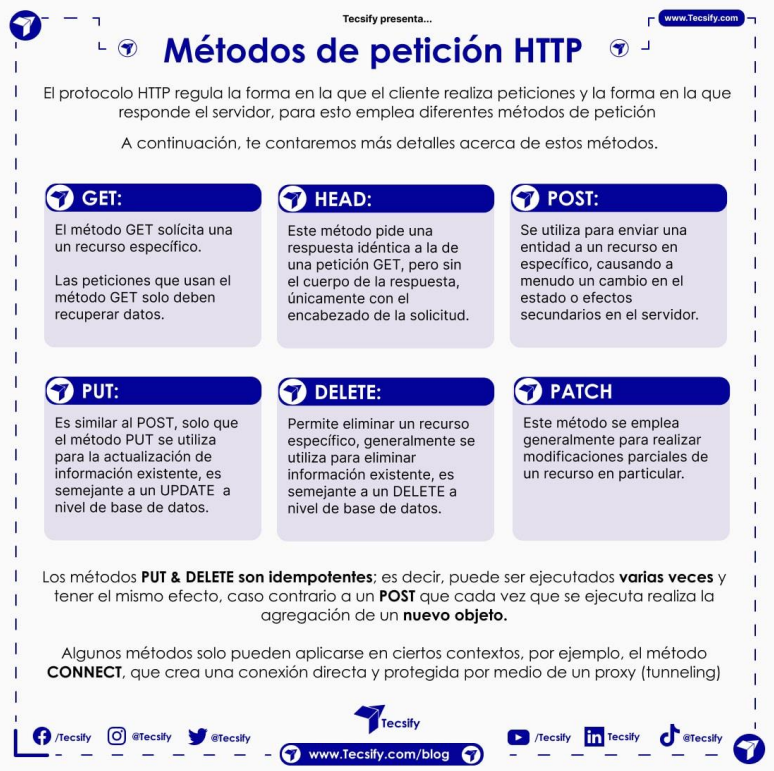
  Características:

  - 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

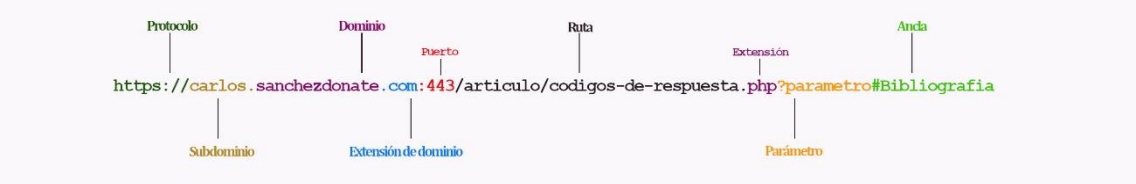


Explicación detallada

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

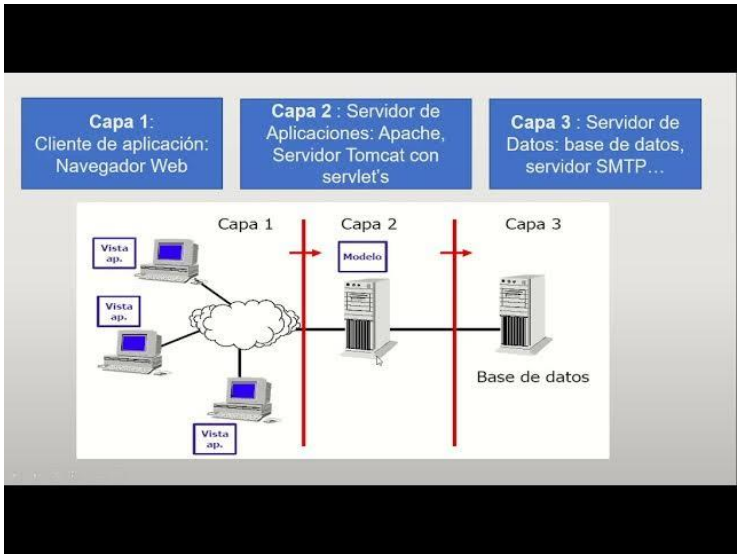


- **URN (Nombre de Recursos Uniforme):** Es otro tipo especifico de URI que identifica un recurso de manera única sin referirse a su ubicación, es decir, es un nombre permanente para el recurso. Estos tienen unos estándares de identificación específicos, como por ejemplo ISBN para los libros
  Ejemplo: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



Explicación Detallada

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

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

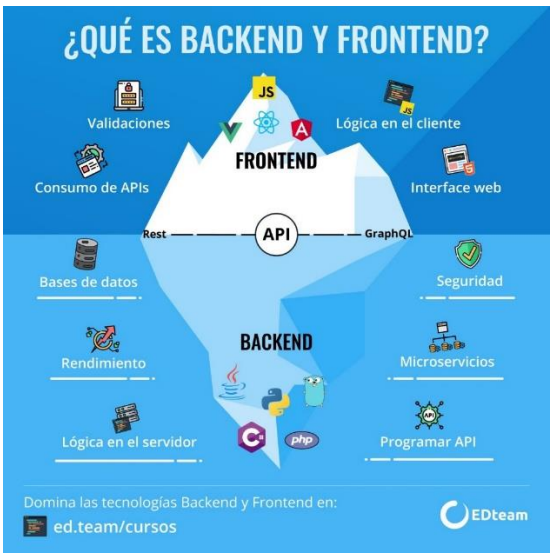
El front-end se encarga de mostrar la información al usuario y recoger sus interacciones**.** Es la parte visible del sistema.

- **Interfaz de usuario (UI)**
  - Construcción de páginas, botones, formularios, menús, etc.
  - Uso de tecnologías como HTML, CSS, y JavaScript (o frameworks como React, Angular, Vue).
- **Lógica de presentación (View Logic)**
  - Manejo de eventos del usuario (clics, desplazamientos, entradas).
  - Validaciones básicas (por ejemplo, que un campo de email tenga "@").
  - Renderización dinámica de datos (por ejemplo, mostrar una lista de productos traídos desde el back-end).
- **Comunicación con el back-end**
  - Envío y recepción de datos mediante API REST, GraphQL, o WebSockets.
  - Normalmente se hacen peticiones HTTP (GET, POST, PUT, DELETE).
- **Gestión de estado local y de sesión**
  - Almacena temporalmente datos del usuario o de la app (por ejemplo, usando localStorage, Redux, o Vuex).

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

El back-end se encarga de procesar la lógica de negocio**,** acceder a bases de datos, y proveer información al front-end**.**

- **Lógica de negocio (Business Logic)**
  - Implementa las reglas del sistema (por ejemplo, “solo los usuarios con rol admin pueden borrar registros”).
  - Procesa los datos recibidos antes de guardarlos o enviarlos.
- **Gestión de datos y persistencia**
  - Comunicación con bases de datos (SQL, NoSQL, etc.).
  - Modelado de entidades y relaciones (por ejemplo, “usuarios”, “productos”, “pedidos”).
- **Gestión de usuarios y seguridad**
  - Autenticación (login, tokens, OAuth).
  - Autorización (roles, permisos).
  - Encriptación de contraseñas, protección contra inyecciones SQL, etc.
- **Comunicación con el front-end**
  - Exposición de endpoints o APIs que devuelven datos en formato JSON o XML.
  - Procesamiento de solicitudes del cliente y envío de respuestas adecuadas (códigos HTTP, mensajes de error, etc.).
- **Integraciones externas**
  - Conexión con servicios de terceros (pasarelas de pago, APIs externas, etc.).



Explicación Detallada

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

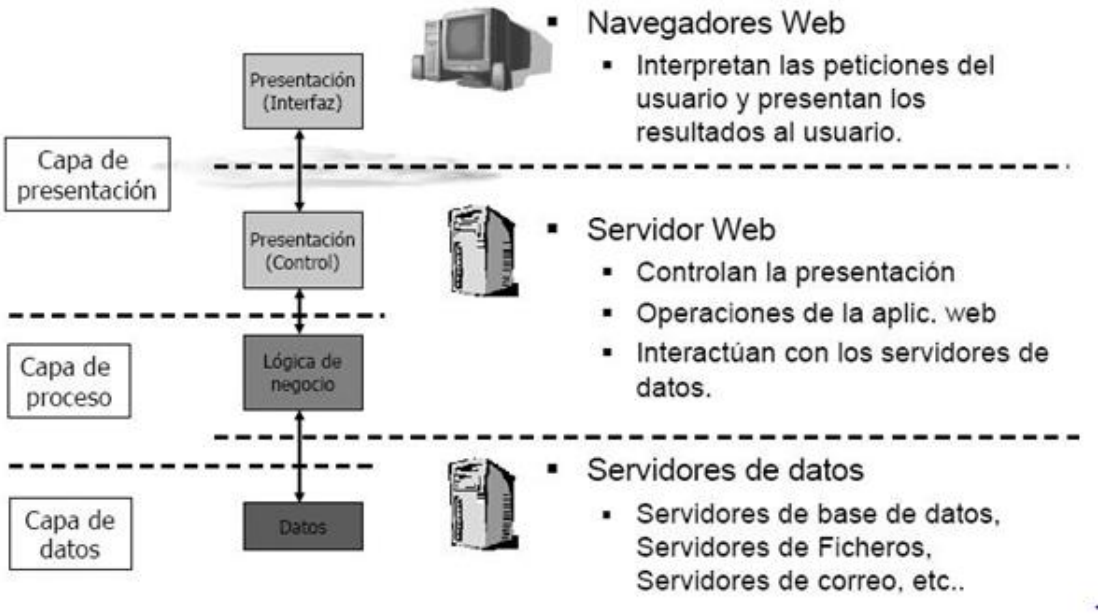
- **Página web estática**: Una página web estática es un conjunto de archivos (HTML, CSS, imágenes, etc.) que se muestran tal cual fueron creados. El contenido no cambia a menos que el desarrollador modifique el código manualmente.
    - El contenido es fijo (no cambia según el usuario ni el tiempo).
    - Se sirve directamente desde un servidor web, sin necesidad de bases de datos.
    - Se crea con HTML, CSS y, opcionalmente, JavaScript básico.
    - Es rápida, simple y barata de alojar.
- **Página web dinámica**: Una página web dinámica genera su contenido de forma automática o personalizada, normalmente con ayuda de un servidor y una base de datos.
    - El contenido se actualiza o cambia automáticamente (por usuario, fecha, búsquedas, etc.).
    - Usa lenguajes del lado del servidor como PHP, Python, Node.js, Java, etc.
    - Interactúa con bases de datos (MySQL, PostgreSQL, MongoDB, etc.).
    - Requiere más recursos y mantenimiento.
- **Aplicación Web**: Una aplicación web es un programa interactivo que se ejecuta en un navegador, pero funciona como una aplicación tradicional.
    - Ofrece funcionalidad compleja (formularios, autenticación, cálculos, gestión de datos).
    - El usuario interactúa activamente (crear cuentas, subir archivos, editar, etc.).
    - Puede tener front-end (interfaz) y back-end (lógica y base de datos).
    - Usa tecnologías como HTML, CSS, JavaScript, frameworks (React, Angular, Vue), y APIs.
- **Mashup**: Un mashup es una aplicación web o servicio que combina información o funcionalidades de varias fuentes distintas (generalmente mediante APIs públicas) para crear un nuevo servicio.
    - Integra datos o servicios de diferentes sitios web.
    - Suele usar APIs REST, JSON, XML, etc.
    - Puede ser una mezcla de mapas, datos sociales, noticias, clima, etc.

Estáticas VS Dinámicas detallado

Que es Mashup, ventajas y ejemplos

8. Componentes de una aplicación web.

- **Navegador:** Programa que permite a los usuarios acceder y visualizar páginas web (ej. Chrome, Firefox).
- **Servidor Web**: Sistema 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…

Explicación detallada 1

Explicación detallada 2

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

1. Node.js (JavaScript)

- Características: Basado en JavaScript, permite usar el mismo lenguaje en frontend y backend.
- Ventajas: Gran rendimiento con operaciones de entrada/salida, enorme ecosistema de paquetes.
- Ideal para: Aplicaciones en tiempo real como chats, etc.

2. Python

- Características: Sintaxis clara y legible, gran comunidad.
- Ventajas: Amplia gama de frameworks (Django, Flask), excelente para desarrollo rápido.
- Ideal para: Aplicaciones web, ciencia de datos, automatización.

3. PHP

- Características: Diseñado específicamente para la web.
- Ventajas: Fácil de aprender, ampliamente soportado por servidores, integración sencilla con bases de datos.
- Ideal para: Sitios web dinámicos, CMS como WordPress.

4. Java

- Características: Lenguaje orientado a objetos, robusto y seguro.
- Ventajas: Gran rendimiento, escalabilidad, frameworks como Spring.
- Ideal para: Aplicaciones empresariales, sistemas bancarios, comercio electrónico.

5. Ruby

- Características: Enfocado en la simplicidad y productividad.
- Ventajas: Convención sobre configuración, desarrollo rápido.
- Ideal para: Startups, prototipos rápidos, aplicaciones CRUD

Grado de implantación de lenguajes del lado del servidor (y otros datos)

Explicación detallada de estos y otros lenguajes

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

Guía sobre XAMPP: Características, beneficios e instalación paso a paso

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.

Entorno de Desarrollo

En este entorno se crean, prueban y depuran aplicaciones Java. Aquí sí es necesario instalar tanto el JDK como la JVM.

Se necesita el JDK porque:

- Incluye el compilador javac para convertir código fuente .java en bytecode .class.
- Proporciona herramientas como javadoc, jdb (depurador), y otras utilidades para desarrollo.
- Contiene bibliotecas de desarrollo esenciales (API estándar de Java).

Se necesita la JVM porque:

- Permite ejecutar el bytecode generado para probar la aplicación.
- Es parte del JDK, pero también puede instalarse por separado si se desea.

Entorno de Explotación

Este entorno ejecuta aplicaciones ya compiladas. Aquí solo es necesario instalar la JVM, no el JDK.

Se necesita la JVM porque:

- Ejecuta el bytecode .class o .jar generado en el entorno de desarrollo.
- Es suficiente para correr aplicaciones Java sin necesidad de compilar código.

No se necesita el JDK porque:

- No se compila ni modifica el código fuente en producción.
- Instalar el JDK sería redundante y podría representar un riesgo de seguridad si se usa incorrectamente.

Más información sobre JRE, JVM y JDK

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

1. **Visual Studio Code (70%)**

   El IDE de Visual Studio es una interfaz de desarrollo integrada impulsada por Microsoft desarrollada para ayudar a los desarrolladores de software con los desarrollos web. El IDE utiliza funciones de inteligencia artificial para aprender de la edición del programador en sus códigos, lo que facilita completar líneas de código automáticamente.

   El IDE también permite a los usuarios compartir servidores, comentarios y terminales.

   Además, Visual Studio tiene la capacidad de admitir el desarrollo de aplicaciones móviles, web y juegos. También es compatible con el lenguaje Python, Node.js, ASP.NET y Azure.

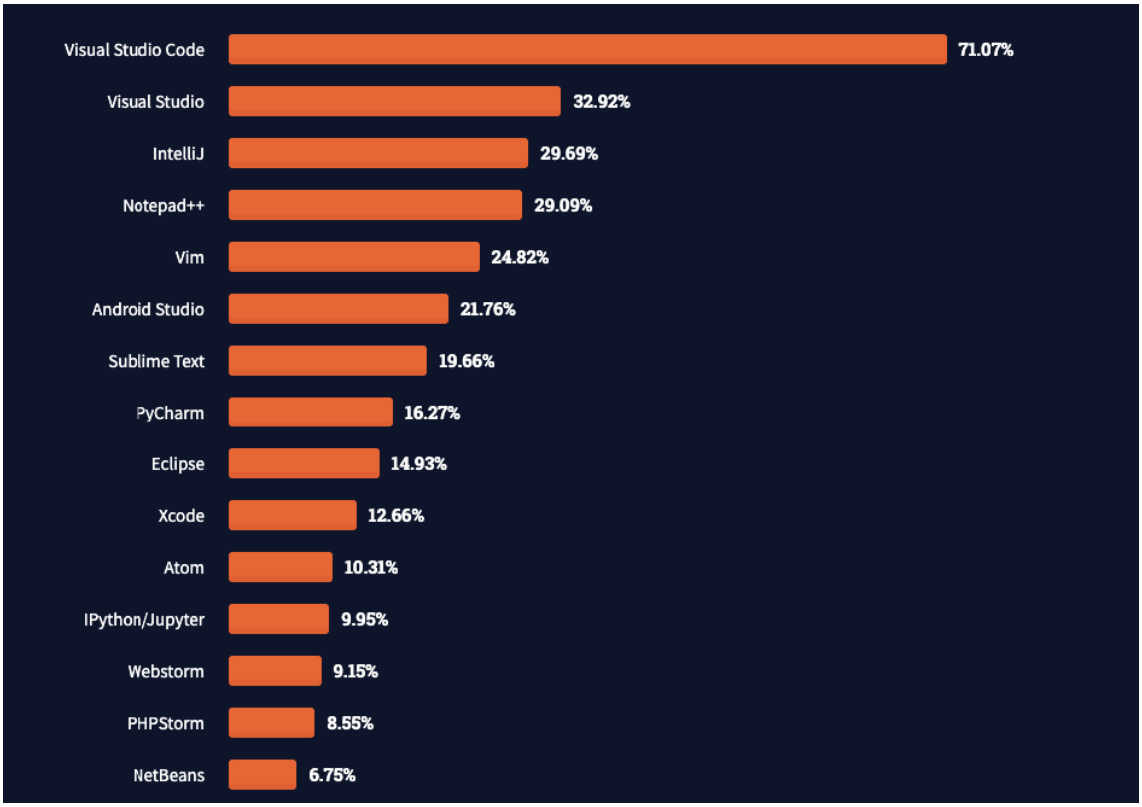
2. **IntelliJ IDEA (25%)**
   Ha existido durante años y ha servido como uno de los mejores IDE para la programación Java. La interfaz de usuario de IntelliJ Idea está diseñada de una manera elegante que hace que la codificación sea atractiva para muchos desarrolladores de Java.
   Con este IDE, el código se puede indexar, proporcionando sugerencias relevantes para ayudar a completar las líneas de código. También lleva esta codificación sugerente más allá al automatizar varias tareas que pueden ser repetitivas.

   Además de ser compatible con la programación Java web, empresarial y móvil, también es una buena opción para la programación JavaScript, SQL y JPQL.

3. **Eclipse (10-15%)**
   Eclipse es uno de los IDE más populares. Es una herramienta multiplataforma con una potente interfaz de usuario que admite arrastrar y soltar. El IDE también incluye algunas características importantes, como herramientas de análisis estático, capacidades de depuración y creación de perfiles. Eclipse es compatible con el desarrollo empresarial y permite a los desarrolladores trabajar fácilmente en el desarrollo de software escalable y de código abierto.

   Aunque Eclipse se asocia mejor con Java, también admite varios lenguajes de programación. Además, los usuarios pueden agregar sus complementos preferidos al IDE para respaldar proyectos de desarrollo de software.

4. **Android Studio (20%)**
   Android Studio es uno de los mejores IDE para el desarrollo de aplicaciones de Android. Este IDE es compatible con los lenguajes de programación Kotlin y Java. Algunas funciones importantes que los usuarios pueden obtener de Android Studio son alertas automáticas, integraciones de cámara y otras funciones de tecnología móvil.
   Los desarrolladores también pueden crear variantes y diferentes APK con la ayuda de este IDE flexible, que también ofrece compatibilidad con plantillas ampliadas para los servicios de Google.



Estos y más IDES

Más información

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%)

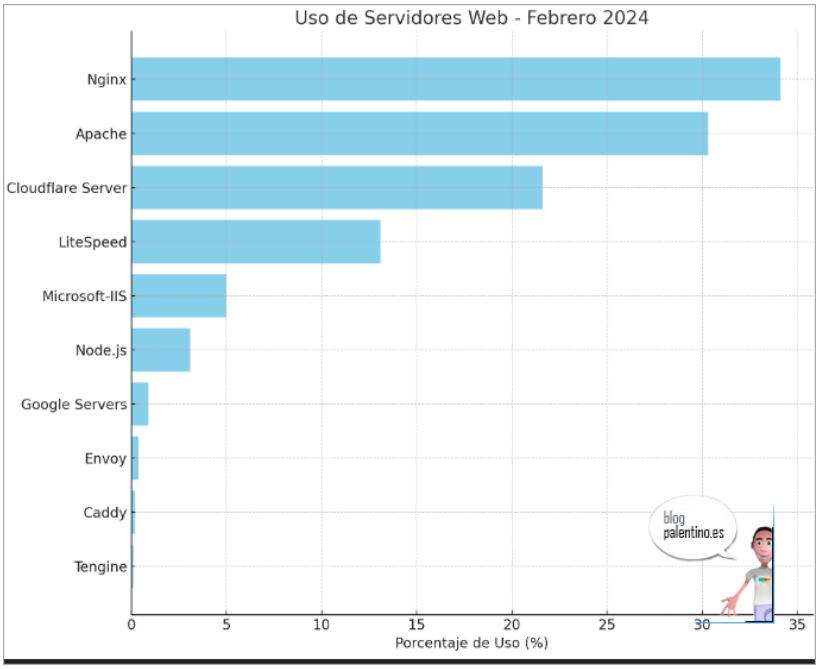
- Mejora la seguridad, acelera la carga de las páginas y además protege de ataques

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



Datos sobre los servidores más utilizados

Explicación Detallada

15. Apache HTTP vs Apache Tomcat

HTTP Apache y Apache Tomcat, ambas mantenidas y desarrolladas por la ASF (Apache Software Fundation), son software que se ejecuta en un servidor con la finalidad de ayudar a desplegar (deploy) un proyecto web. La diferencia radica en el tipo de proyecto que se planea implementar.

HTTP Apache es usado para almacenar páginas web estáticas y dinámicas, usando lenguajes como PHP, Perl, Python y Ruby.

Apache Tomcat también soporta páginas web estáticas (aunque su rendimiento es menor), pero su especialidad son las páginas web dinámicas desarrolladas con la tecnología Java, como Java Servlet, JavaServer Page (JSP).

Más información general

Apache HTTP Server: Qué es, cómo funciona e instalación

Introducción a Apache Tomcat

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

1. **Chrome (69,23%)**

   Google Chrome es el navegador más utilizado en el mundo, desarrollado por Google y basado en el motor Blink y el motor de JavaScript V8. Se destaca por su velocidad, compatibilidad con tecnologías web modernas y una integración profunda con el ecosistema de Google.

2. **Safari (14,98%)**

   Safari es el navegador web desarrollado por Apple, diseñado para integrarse de forma óptima con el ecosistema de macOS, iOS y iPadOS. Utiliza el motor de renderizado WebKit, que destaca por su eficiencia energética y rendimiento optimizado en dispositivos de Apple.

3. **Edge (5,03%)**

   Microsoft Edge es el navegador desarrollado por Microsoft, basado en el motor Chromium desde su renovación en 2020. Esta actualización le permitió mejorar significativamente su rendimiento, compatibilidad y eficiencia en comparación con su versión anterior basada en EdgeHTML. Actualmente, es el navegador predeterminado en Windows y también está disponible para macOS, Linux, Android y iOS.

4. **Firefox (2,26%)**

   Mozilla Firefox, desarrollado por la Mozilla Foundation, es uno de los navegadores más populares y reconocidos por su enfoque en la privacidad, la personalización y el código abierto. Utiliza el motor Gecko y se distingue de los navegadores basados en Chromium, ofreciendo una alternativa sólida y con características propias.

5. **Opera (1,85%)**

   Opera es un navegador web con una larga trayectoria, conocido por su enfoque innovador y la incorporación de características únicas que mejoran la experiencia de navegación. Utiliza el motor Blink, al igual que Google Chrome y Microsoft Edge, pero se diferencia por su integración de herramientas adicionales que no se encuentran en otros navegadores.

Más información

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

PHPDocumentor

- Descripción: Es el generador de documentación más popular y veterano para PHP, similar a Javadoc pero adaptado al ecosistema PHP.
- Características clave:
  - Analiza el código fuente y los comentarios DocBlock para generar documentación API completa.
  - Soporta múltiples formatos: HTML, PDF, CHM, XML DocBook.
  - Permite diagramas UML, listas de tareas (@todo), ejemplos, y enlaces entre clases y métodos.
  - Se puede ejecutar desde la línea de comandos o mediante interfaz web.
  - Compatible con sintaxis moderna de PHP y altamente personalizable.

Sitio oficial: phpDocumentor

ApiGen

- Descripción: Herramienta ligera y rápida para generar documentación API desde comentarios PHPDoc.
- Características clave:
  - Genera documentación HTML con navegación clara y estructura jerárquica.
  - Soporta namespaces, clases, interfaces, traits y funciones.
  - Menos complejo que PHPDocumentor, ideal para proyectos pequeños o medianos.
  - Compatible con PHP 5.3+ y versiones posteriores.

Sitio oficial: ApiGen

phpDox

- Descripción: Alternativa moderna centrada en documentación extensible y enriquecida.
- Características clave:
  - Utiliza PHP-Parser para analizar el código y generar archivos XML.
  - Permite enriquecer la documentación con datos de herramientas como PHP_CodeSniffer, PHPMD o PHPUnit.
  - Usa transformaciones XSL para generar HTML personalizado.
  - Soporta plugins para ampliar funcionalidades y anotaciones DocBlock.

Sitio oficial: phpDox

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

¿Qué es un Sistema de Control de Versiones (VCS)?

Un Sistema de Control de Versiones es una herramienta que permite:

- Registrar los cambios realizados en archivos (especialmente código fuente).
- Recuperar versiones anteriores.
- Facilitar el trabajo colaborativo entre desarrolladores.
- Evitar conflictos y pérdidas de información.

1. Git

Ventajas:

- Trabajo sin conexión.
- Ramas ligeras y eficientes.
- Alta velocidad y rendimiento.
- Amplio soporte en plataformas como GitHub, GitLab, Bitbucket.

2. CVS (Concurrent Versions System)

Ventajas:

- Simplicidad.
- Adecuado para equipos pequeños.

Desventajas:

- Manejo limitado de ramas.
- Riesgo de pérdida si el servidor falla.

3. Subversion (SVN)

Ventajas:

- Mejor manejo de ramas que CVS.
- Control de acceso granular.

Desventajas:

- Menor flexibilidad que Git.
- Dependencia del servidor central.

Explicación Detallada

Subversion explicación detallada VS Git

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.
Configuración del servidor

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…

Explicación Detallada

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…

Explicación Detallada

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**

MÁS INFORMACION COMPLETA GENERAL

Wordpress (próximamente)

GLOSARIO DE TÉRMINOS RELACIONADOS CON DWES

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

➢ **Protocolos TCP/IP. Socket.**

El conjunto de protocolos TCP/IP (Transmission Control Protocol / Internet Protocol) es la base de la comunicación en Internet. IP se encarga de dirigir los paquetes de datos desde un origen hasta un destino a través de direcciones IP, mientras que TCP garantiza que esos datos lleguen completos y en el orden correcto mediante una conexión fiable.

Un socket es un punto de conexión entre dos programas que se comunican por la red, representando un canal bidireccional para enviar y recibir información. Por ejemplo, un servidor web abre un socket en un puerto (como el 80 o 443) para atender las peticiones HTTP de los navegadores.

Más información

➢ **Protocolo HTTP / HTTPS**

HTTP (HyperText Transfer Protocol) es el protocolo que regula la transferencia de información entre clientes (navegadores) y servidores web. Define cómo deben formularse las peticiones y respuestas (métodos como GET, POST, PUT, DELETE).

HTTPS es la versión segura de HTTP, que añade una capa de cifrado mediante SSL/TLS para proteger los datos transmitidos (como contraseñas o información personal).

Más información

➢ **HTML**

HTML (HyperText Markup Language) es el lenguaje estándar para estructurar el contenido de una página web. Utiliza etiquetas (tags) que indican el tipo de información: títulos (<h1>), párrafos (<p>), enlaces (<a>), imágenes (<img>), entre otros.

Aunque HTML no define el diseño visual (eso se logra con CSS), es fundamental para la accesibilidad y el SEO.

Más información

➢ **XML**

XML (eXtensible Markup Language) es un lenguaje diseñado para almacenar y transportar datos de forma estructurada y legible tanto por humanos como por máquinas.

A diferencia de HTML, XML no define cómo se muestra la información, sino qué significa. Se usa en configuraciones, servicios web o intercambio de datos entre aplicaciones.

Más información

➢ **JSON**

JSON (JavaScript Object Notation) es un formato ligero y fácil de leer para el intercambio de datos, especialmente en aplicaciones web. Es actualmente el formato más usado en APIs REST por su simplicidad.

Es independiente del lenguaje y se basa en pares "clave: valor".

Más información

➢ **Lenguajes de programación embebidos en HTML**

Son lenguajes que se insertan dentro del código HTML para dotar a la página de funcionalidad o dinamismo.

JavaScript se usa para manipular el contenido del lado del cliente, mientras que PHP o ASP.NET permiten generar contenido dinámico desde el servidor.

➢ **Arquitecturas de desarrollo web**

La arquitectura web define cómo se organizan y comunican las distintas partes de una aplicación. Las más comunes son:

- **Cliente-servidor**: el cliente solicita y el servidor responde.
- **MVC (Modelo-Vista-Controlador)**: separa la lógica, los datos y la presentación.
- **Microservicios**: divide la aplicación en pequeños servicios independientes.

Elegir una arquitectura adecuada mejora la escalabilidad y el mantenimiento del sistema.

Más información

➢ **Framework de desarrollo Web**

Un framework es un conjunto estructurado de herramientas, librerías y convenciones que facilitan el desarrollo de aplicaciones. Permite reutilizar código y aplicar buenas prácticas.
Ejemplos:

- **Laravel** (PHP)
- **Django** (Python)
- **Angular / React** (JavaScript)

Estos frameworks agilizan tareas comunes como autenticación, enrutamiento o conexión a bases de datos.

Más información

➢ **ERP**

Un ERP (Enterprise Resource Planning) es un sistema que integra todos los procesos y recursos de una empresa (producción, inventario, ventas, finanzas, recursos humanos) en una única plataforma.

Mejora la eficiencia y la toma de decisiones al centralizar los datos.

Ejemplos: SAP, Odoo, Microsoft Dynamics.

Más información

➢ **CMS**

Un CMS (Content Management System) permite crear, editar y gestionar contenido web sin necesidad de conocimientos técnicos avanzados.

Facilita la publicación de artículos, imágenes, menús o páginas con una interfaz visual.

Ejemplo: WordPress, Drupal, Joomla.

Más información

➢ **PHP**

PHP es un lenguaje de programación de propósito general, enfocado al desarrollo web del lado del servidor. Se ejecuta en el servidor y puede generar código HTML dinámico que se envía al navegador.

Es ampliamente usado junto con bases de datos MySQL.

Más información

➢ **IDE**

Un IDE (Integrated Development Environment) es un entorno de software que agrupa herramientas esenciales para programar: editor de texto, compilador, depurador y control de versiones.

Facilita el desarrollo mediante resaltado de sintaxis, autocompletado y ejecución de código.

Ejemplo: Visual Studio Code, IntelliJ IDEA, Eclipse.

Más información

➢ **Navegador**

El navegador web es el programa que interpreta y muestra las páginas web. Procesa el código HTML, CSS y JavaScript para renderizar la interfaz.

Además, gestiona cookies, historial, seguridad y extensiones.

Ejemplo: Google Chrome, Mozilla Firefox, Safari.

Más información

➢ **Repositorio**

Un repositorio es un espacio donde se guarda y gestiona el código fuente de un proyecto, generalmente usando sistemas de control de versiones como Git.

Permite registrar los cambios, colaborar entre programadores y mantener un historial completo.

Ejemplo: GitHub, GitLab, Bitbucket.

Más información

➢ **Entorno de Desarrollo**

Es el ambiente donde los programadores construyen y prueban la aplicación antes de publicarla.

Incluye el servidor local, las bases de datos de prueba y herramientas de depuración. Permite experimentar sin afectar al entorno real.

Más información

➢ **Entorno de Explotación o Producción**

Es el entorno donde la aplicación se ejecuta de forma estable y accesible a los usuarios finales.

Debe garantizar disponibilidad, seguridad y rendimiento, ya que cualquier error puede afectar a los clientes o usuarios reales.

Más información

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

Consiste en administrar las versiones del software, controlar los cambios realizados y garantizar que las modificaciones sean seguras y documentadas.

El control de versiones (Git, SVN) permite trabajar en equipo sin sobrescribir código. El mantenimiento incluye corregir errores, mejorar rendimiento y actualizar dependencias.

Más información

➢ **Web Services**

Los servicios web permiten la comunicación entre aplicaciones a través de la red, usando estándares como SOAP o REST.

Son la base de muchas integraciones modernas (por ejemplo, conectar una app con PayPal o Google Maps).

Ejemplo: una API REST que devuelve datos JSON sobre el clima.

Más información

➢ **AJAX**

AJAX (Asynchronous JavaScript And XML) permite que las páginas web soliciten datos al servidor sin tener que recargarse completamente.

Mejora la experiencia de usuario al hacer las aplicaciones más rápidas e interactivas.

Ejemplo: un formulario que valida el correo en tiempo real sin refrescar la página.

Más información

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

Esta arquitectura separa la aplicación en capas lógicas independientes:

- **Capa de presentación** (interfaz web).
- **Capa de negocio** (procesos y lógica).
- **Capa de datos** (base de datos).
  Facilita la escalabilidad y el mantenimiento. Un ejemplo típico es una app con React (frontend), Node.js (backend) y MySQL (datos).

Más información

➢ **Aplicaciones basadas en microservicios**

Los microservicios dividen una aplicación en pequeños servicios independientes que se comunican por APIs.

Cada servicio gestiona una función específica (usuarios, pagos, productos) y puede actualizarse sin afectar al resto.

Ejemplo: Netflix o Amazon, donde cada parte del sistema opera de forma autónoma.

Más información

➢ **SaaS: Sofware as a Service**

El modelo SaaS ofrece aplicaciones como servicios accesibles desde Internet, sin necesidad de instalación local.

El proveedor se encarga del mantenimiento, actualizaciones y seguridad.

Ejemplo: Google Workspace, Salesforce, Canva, Dropbox.

Más información

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

Son los mecanismos que regulan quién puede acceder a qué recursos dentro de una aplicación.

Incluyen autenticación (verificar identidad) y autorización (asignar permisos).

Ejemplo: un usuario “admin” puede modificar datos, mientras que uno “invitado” solo los consulta.

Más información

➢ **Validación de entrada de datos a una aplicación Web**

Es el proceso de comprobar que los datos que introduce el usuario son correctos, coherentes y seguros antes de procesarlos.

Se realiza tanto en el cliente (JavaScript) como en el servidor (PHP, Python) para prevenir errores o ataques como la inyección SQL.

Ejemplo: comprobar que el email tenga formato válido o que un número esté dentro de un rango permitido.

Más información

➢ **Posicionamiento de una aplicación Web**

También conocido como SEO (Search Engine Optimization), consiste en optimizar el contenido, la estructura y el rendimiento de un sitio web para aparecer en los primeros resultados de los motores de búsqueda.

Incluye el uso correcto de etiquetas HTML, palabras clave, enlaces internos y velocidad de carga.

Más información

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

En los años 90, las webs eran estáticas (solo HTML). En los 2000 surgieron los sitios dinámicos con PHP, bases de datos y JavaScript.

La década de 2010 trajo el auge de frameworks front-end (React, Angular, Vue) y la nube.

Hoy las aplicaciones son interactivas, responsivas y distribuidas, con un fuerte enfoque en la experiencia del usuario y la seguridad.

Más información

➢ **Filosofías de desarrollo del software**

Son enfoques o metodologías que guían el proceso de creación del software.
Las más conocidas son:

- **Cascada**: desarrollo lineal, con fases fijas.
- **Ágil (Scrum, Kanban)**: trabajo iterativo, entregas frecuentes y comunicación constante.
- **DevOps**: colaboración entre desarrollo y operaciones para automatizar despliegues.
  Cada filosofía busca mejorar la calidad, productividad y adaptabilidad del proyecto.

Más información