Introducción a la Programación Web

Historia de la web

Inicios de la web (1989-1995)

- 1989: Tim Berners-Lee crea el concepto de la World Wide Web.
- 1990: Primer navegador web: WorldWideWeb.
- 1991: Publicación del primer sitio web.
- 1995: Nacimiento de JavaScript y CSS.

Era del diseño y dinamismo (1996-2004)

- 1996: Lanzamiento de Flash para animaciones.
- 1998: Creación de Google.
- 2000: Introducción de AJAX para interactividad.
- 2004: Fundación de Mozilla Firefox.

La web moderna (2005-presente)

- 2005: Creación de YouTube y auge de redes sociales.
- 2008: Nace Google Chrome.
- 2010: HTML5 y CSS3 revolucionan el diseño web.
- 2020: Aplicaciones web progresivas (PWA) y más.

¿Que es el desarrollo web ?

¿Qué es el desarrollo web?

- Es el proceso de crear, diseñar y mantener sitios web o aplicaciones web.
- Combina lenguajes de programación, diseño gráfico y herramientas para que los usuarios puedan interactuar con contenido en línea.
- Incluye aspectos como la funcionalidad, apariencia y experiencia del usuario.

Áreas principales del desarrollo web

- Frontend (desarrollo del cliente):
  - Se enfoca en la parte visible e interactiva del sitio web.
  - Incluye lenguajes como HTML, CSS y JavaScript.
  - Ejemplo: La disposición de elementos, colores y botones.
- Backend (desarrollo del servidor):
  - Maneja la lógica del negocio, bases de datos y servidores.
  - Usa lenguajes como PHP, Python, Java o Node.js.
  - Ejemplo: Procesar datos de un formulario o autenticar usuarios.

Áreas principales del desarrollo web

- Full-stack:
  - Combina frontend y backend.
  - Los desarrolladores full-stack pueden manejar todas las partes del desarrollo.

Componentes básicos de un sitio web

Componentes básicos de un sitio web

- Cliente:
  - El navegador web que usa el usuario para acceder al sitio.
- Servidor:
  - El sistema que procesa solicitudes y envía datos al cliente.
- Base de datos:
  - Almacena y administra la información que usa el sitio.
- Red:
  - Conecta al cliente con el servidor mediante protocolos como HTTP/HTTPS.

Etapas del desarrollo web

Etapas del desarrollo web

- Planificación
- Diseño
- Desarrollo
- Pruebas
- Implementación
- Mantenimiento

Tecnologías clave en el desarrollo web

Frontend

- HTML: Estructura del contenido.
- CSS: Apariencia y diseño.
- JavaScript: Interactividad.
- Frameworks: React, Angular, Vue.js.

Backend

- Lenguajes: PHP, Python, Java, Node.js.
- Frameworks: Django, Laravel, Spring.
- Bases de datos: MySQL, PostgreSQL, MongoDB.

Herramientas adicionales

- Control de versiones: Git.
- Entornos de desarrollo: Visual Studio Code, Sublime Text.
- Gestión de proyectos: Jira, Trello.

Conceptos importantes

Conceptos importantes

- Responsive Design:Asegura que los sitios web se vean bien en cualquier dispositivo (computadoras, tablets, teléfonos).
- Accesibilidad web:Hacer sitios usables para personas con discapacidades.
- Seguridad:Proteger los datos de los usuarios mediante cifrado (HTTPS), autenticación segura y validación de datos.

Navegadores

Navegadores

- Chrome: Popular por su velocidad y herramientas de desarrollo.
- Firefox: Famoso por su enfoque en la privacidad.
- Safari: Navegador nativo de Apple.
- Edge: Integrado en Windows.
- Opera: Con funciones únicas como VPN integrado.

Arq. Cliente y Servidor

Cliente

- Puede ser una aplicación, navegador web, app móvil, etc.
- Interactúa con el usuario final.
- Ejemplo: Navegador web (Google Chrome) solicitando una página web.

Servidor

- Procesa las solicitudes del cliente y envía respuestas.
- Puede manejar datos, lógica de negocio o almacenamiento.
- Ejemplo: Un servidor web como Apache o Nginx.

Protocolo HTTP

¿Qué es HTTP?

- HTTP significa HyperText Transfer Protocol.
- Es el protocolo que permite la comunicación entre un cliente (navegador, app) y un servidor web.
- Funciona en la capa de aplicación del modelo TCP/IP.
- Es un protocolo sin estado, lo que significa que cada solicitud es independiente.

¿Cómo funciona HTTP?

- El cliente envía una solicitud (request) al servidor.
- El servidor procesa la solicitud y envía una respuesta (response).
- Ejemplo cotidiano: Un cliente pide un menú (solicitud), el mesero entrega el menú (respuesta).

HTML

Qué es HTML

- Es el lenguaje de marcado para estructurar el contenido de las páginas web.
- HTML significa HyperText Markup Language.
- Es la base de toda página web, utilizado para definir elementos como textos, imágenes, enlaces, tablas, formularios, etc.

Estructura básica de un documento HTML

```
<!DOCTYPE html>
<html>
  <head>
    <title>Título de la Página</title>
  </head>
  <body>
    <h1>Hola Mundo</h1>
  </body>
</html>
```

CSS

¿Qué es CSS?

- CSS significa Hojas de Estilo en Cascada.
- Es el lenguaje que controla la apariencia y diseño de las páginas web.
- Se utiliza junto con HTML para separar el contenido (HTML) de la presentación (CSS).

¿Por qué usar CSS?

- Mejora la apariencia visual de los sitios web.
- Facilita la reutilización de estilos en varias páginas.
- Separa el contenido del diseño, lo que hace que el código sea más limpio y mantenible.

JavaScript

¿Qué es JavaScript?

- JavaScript es un lenguaje de programación utilizado para agregar interactividad a las páginas web.
- Se ejecuta principalmente en el navegador, pero también puede ejecutarse en el servidor (por ejemplo, con Node.js).
- Es uno de los pilares del desarrollo web junto con HTML (estructura) y CSS (estilo).

¿Por qué usar JavaScript?

- Permite crear sitios web dinámicos e interactivos.
- Controla elementos HTML y responde a eventos como clics, movimientos del mouse, y entrada del teclado.
- Ejemplo: Mostrar alertas, validar formularios, cambiar el contenido de una página sin recargarla.

Editores de código

¿Qué es un editor de código?

- Es una herramienta diseñada para escribir, editar y organizar el código fuente de aplicaciones y sitios web.
- Ayuda a los desarrolladores a trabajar de manera eficiente mediante funcionalidades como resaltado de sintaxis, autocompletado y depuración.

Tipos de editores de código

Editores simples

- Ejemplo: Bloc de notas (Windows) o TextEdit (macOS).
- Ventaja: Son ligeros y no requieren instalación.
- Desventaja: No tienen funciones avanzadas como resaltado de sintaxis o autocompletado.

Editores avanzados

- Ejemplo: Visual Studio Code, Sublime Text, Atom.
- Incluyen herramientas que facilitan el desarrollo web.

Características clave de los editores avanzados

- Resaltado de sintaxis:
  - Colorea diferentes partes del código según el lenguaje.
  - Ayuda a identificar errores y mejorar la legibilidad.
- Autocompletado:
  - Sugiere automáticamente nombres de variables, funciones y etiquetas.
  - Ahorra tiempo y reduce errores.

Mas características

- Previsualización en tiempo real:
  - Algunos editores permiten ver los cambios en el navegador a medida que se escribe el código.
- Depuración integrada:
  - Herramientas para detectar y corregir errores en el código.
- Integración con Git:
  - Facilita el control de versiones directamente desde el editor.

Editores populares para desarrollo web

- Visual Studio Code
- Sublime Text
- Atom
- Brackets
- Notepad++

Editor	Resaltado de sintaxis	Autocompletado	Extensiones	Previsualización en tiempo real
Visual Studio Code	✅	✅	✅	✅
Sublime Text	✅	✅	✅	❌
Atom	✅	✅	✅	❌
Brackets	✅	✅	✅	✅
Notepad++	✅	❌	❌	❌

Control de versiones

¿Qué es el control de versiones?

- Es un sistema que permite rastrear los cambios realizados en archivos o proyectos a lo largo del tiempo.
- Ayuda a los desarrolladores a colaborar, revertir cambios y mantener un historial completo del proyecto.
- Se usa principalmente en el desarrollo de software, pero también puede aplicarse a documentos y otros archivos.

¿Por qué es importante?

- Rastreo de cambios: Puedes ver qué cambios se hicieron, quién los hizo y cuándo.
- Reversión: Si algo sale mal, puedes regresar a una versión anterior.
- Colaboración: Permite que varios desarrolladores trabajen en el mismo proyecto sin conflictos.
- Seguridad: Los datos no se pierden porque el sistema guarda copias del historial.

Herramientas para usar control de versiones

- Interfaz de línea de comandos (CLI):
  - Ejemplo: Git Bash, Terminal.
- Interfaz gráfica (GUI):
  - GitHub Desktop, Sourcetree.
- Plataformas remotas:
  - GitHub: Popular para proyectos públicos y privados.
  - GitLab: Similar a GitHub, con más opciones de autoalojamiento.
  - Bitbucket: Ideal para proyectos pequeños y equipos.

Servidores locales

¿Qué es un servidor local?

- Es un servidor que se ejecuta en tu propia computadora para probar y desarrollar aplicaciones web.
- Emula el comportamiento de un servidor real, pero sin necesidad de conexión a Internet.
- Permite trabajar de forma segura y eficiente antes de publicar un proyecto en un servidor remoto.

¿Por qué usar un servidor local?

- Pruebas rápidas: Puedes probar tu sitio web en tiempo real sin subirlo a Internet.
- Compatibilidad: Simula un entorno similar al de un servidor en producción.
- Privacidad: Tus archivos y datos permanecen en tu máquina mientras desarrollas.
- Ejecución de tecnologías del lado del servidor:
  - Lenguajes como PHP, Python o Node.js requieren un servidor para ejecutarse.
  - Bases de datos como MySQL también necesitan un entorno configurado.

Ejemplos de servidores locales

- XAMPP:
  - Incluye Apache, MySQL, PHP y Perl.
  - Ideal para proyectos en PHP y bases de datos.
- WAMP:
  - Similar a XAMPP, pero solo para Windows.
- MAMP:
  - Diseñado para macOS y Windows.
  - Incluye Apache, MySQL y PHP.

Ejemplos de servidores locales

- Node.js:
  - Sirve como base para aplicaciones JavaScript en el servidor.
  - Incluye un servidor ligero con herramientas como Express.js.

Componentes principales de un servidor local

- Servidor web:
  - Maneja solicitudes HTTP y entrega archivos al navegador.
  - Ejemplo: Apache, Nginx.
- Lenguaje del lado del servidor:
  - Procesa la lógica del backend.
  - Ejemplo: PHP, Python, Node.js.
- Base de datos:
  - Almacena y administra información dinámica.
  - Ejemplo: MySQL, PostgreSQL, MongoDB.

Tarea

Tarea: Estándares W3C

- ¿Que son los Estándares W3C?
- ¿Qué son los estándares del W3C?
- ¿Por qué son importantes los estándares del W3C?
- Principales estándares desarrollados por el W3C
- ¿Cómo trabaja el W3C?
- Impacto de no seguir los estándares
- Herramientas para verificar estándares
- Buenas prácticas para cumplir con los estándares

Conclusiones

¿Preguntas?

Gracias por su atención!