UD1. Arquitecturas y tecnologías web.

Contenido

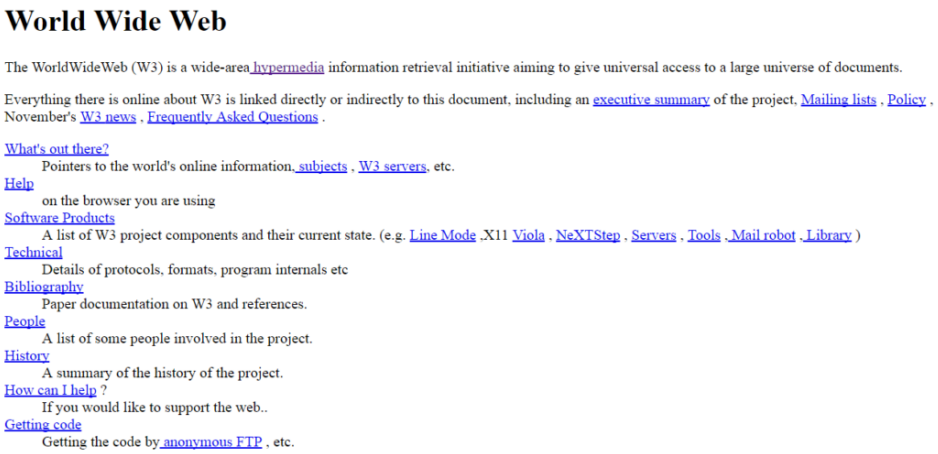
Introducción

La Web es un conjunto de protocolos inventado por Tim Berners-Lee en el CERN con la finalidad de compartir información. La idea clave es el "hipertexto" el cual permite enlazar otros documentos generando un modelo estructural de información en vez de un secuencial.

El World Wide Web Consortium (W3C) es la organización encargada de estandarizar la web. Algunos de sus estándares definen los lenguajes HTML, CSS y JavaScript.

Enlace a la primera web (te recomiendo que le dediques un minuto al código fuente).

https://info.cern.ch/hypertext/WWW/TheProject.html



World Wide Web

The WorldWideWeb (W3) is a wide-area hypermedia information retrieval initiative aiming to give universal access to a large universe of documents.

Everything there is online about W3 is linked directly or indirectly to this document, including an executive summary of the project, Mailing lists , Policy , November's W3 news , Frequently Asked Questions .

What's out there?
Pointers to the world's online information, subjects , W3 servers, etc.
Help
on the browser you are using
Software Products
A list of W3 project components and their current state. (e.g. Line Mode ,X11 Viola , NeXTStep , Servers , Tools , Mail robot , Library)
Technical
Details of protocols, formats, program internals etc
Bibliography
Paper documentation on W3 and references.
People
A list of some people involved in the project.
History
A summary of the history of the project.
How can I help ?
If you would like to support the web..
Getting code
Getting the code by anonymous FTP , etc.

Componentes de la Web

De manera muy básica los principales componentes web son.

- **Hipertexto e hipervínculos.** Documentos más enlaces.
- **Protocolo HTTP, actualmente HTTPS.** Protocolo de transferencia de documentos.
- **Servidores y Clientes.** Los servidores alojan o generan los documentos, los clientes (navegadores) los descargan y visualizan.
- **DNS.** Sistema de nombres de dominio. Traduce las direcciones IP de los servidores por nombres legibles, por ejemplo, https://example.com/

Perfiles en el desarrollo Web

En el desarrollo de un proyecto web podemos encontrarnos con diversos perfiles profesionales. El número de perfiles y de tipos variara en función del tamaño y alcance del proyecto.

- **Desarrolladores de Frontend.** Se encargan de la interfaz de usuario y la experiencia visual.
- **Desarrolladores de Backend**. Gestionan la lógica del servidor y las bases de datos.
- **Diseñadores.** Crean la estética y la usabilidad de la plataforma.
- **Gestores de despliegue.** Aseguran la correcta implementación y actualización de la aplicación.

- **Creadores de contenido.** Generan el material que nutre y da valor a la plataforma.
- **Desarrolladores Fullstack.** Combina las habilidades de frontend y backend, gestionando tanto la interfaz como la lógica del servidor.
- **QA o Tester.** Verifican la calidad del software mediante pruebas, asegurando su correcto funcionamiento antes del despliegue.

La amplitud de posibilidades que ofrece el desarrollo web se resume parcialmente en "el camino del frontend" que puedes consultar en https://roadmap.sh/frontend

Arquitectura Cliente – Servidor

La arquitectura cliente-servidor es el modelo fundamental en el que se basa la Web. Se divide en dos partes principales:

- **Server-side** (Lado del servidor)
  - Hardware: Incluye servidores y elementos de red, máquinas virtuales y contenedores.
  - Software: Involucra servidores web (como Apache, IIS, NGINX) y lenguajes CGI (como Perl, PHP, C). También incluye lenguajes y frameworks con servidores web integrados, como Python, Java, Node.js y C++.
- **Client-side** (Lado del cliente)
  - Clientes web: Los navegadores web como Firefox, Chrome, Safari, Vivaldi, Opera, Edge e Internet Explorer.
  - Lenguajes de marcas: HTML, XHTML, HTML5 y CSS.
  - Lenguajes de programación del entorno Cliente: Principalmente JavaScript. Otros lenguajes como TypeScript son transpilados, es decir, traduce el código fuente a JS.
  - Frameworks cliente: React, Vue.js, Angular.
  - Bibliotecas: JQuery, D3.js, Bootstrap, Pico.css.

Arquitectura de tres capas

La arquitectura de tres capas es un enfoque de desarrollo que divide una aplicación en tres niveles claramente diferenciados: presentación, aplicación (lógica de negocio) y datos. Esta separación permite que cada nivel tenga responsabilidades claras, facilitando la escalabilidad, el mantenimiento y la evolución de la aplicación.

- **Nivel de presentación.** Se encarga de la interacción con el usuario. Este nivel maneja cómo se muestran los datos y recibe las entradas del usuario. La capa de presentación suele ejecutarse en el cliente (el navegador) y puede hacer uso de un **View Model** para optimizar y estructurar los datos de una manera que sea más sencilla de mostrar en la interfaz. Esto mejora la separación entre la lógica de la interfaz y la lógica de negocio.
- **Nivel de aplicación o de lógica de negocio.** Se encarga de procesar las solicitudes de la capa de presentación y ejecutar las reglas de negocio que rigen el funcionamiento de la aplicación. Aquí se gestiona la lógica central, como cálculos, validaciones y flujos de negocio. Este nivel se ejecuta típicamente en el servidor, procesando los datos de acuerdo con las reglas de negocio antes de enviarlos de vuelta a la capa de presentación o comunicarse con la capa de datos.

- **Nivel de datos.** Se encarga de la persistencia de la información. Este nivel se ocupa de acceder y gestionar las bases de datos, asegurando que la aplicación pueda almacenar y recuperar información de forma eficiente. Al igual que la lógica de negocio, esta capa se ejecuta en el servidor, ya que es donde residen las bases de datos, que pueden ser relacionales o no relacionales.

Este modelo permite que cada capa sea desarrollada y mantenida de manera independiente, asegurando una separación clara de responsabilidades y facilitando la modularidad en el desarrollo de software.

Tipos de páginas Web/Aplicaciones…

El desarrollo web ha ido ganando importancia adaptándose para poder competir en otro ámbito de aplicaciones. Ten en mente que la mayoría de aplicaciones son aplicaciones de tipo gestión.

- **Estáticas.** Contenido estático accesible mediante una URL.
- **Aplicaciones Web.** Generadas dinámicamente en el servidor o SPA (Single Page Application).
- **Aplicaciones Web Responsivas.** Diseñadas para adaptarse a distintos tipos de dispositivos, principalmente resoluciones y capacidades del hardware (no es lo mismo emplear un ratón que una pantalla táctil).
- **PWA (Progressive Web App).** Mezcla de aplicación web y móvil.
- **Aplicaciones híbridas.** Aplicaciones que pueden adaptarse a múltiples plataformas.

NOTA: ten en cuenta que JavaScript se ejecuta dentro de entornos seguros en el navegador, esto implica que no puedas acceder a elementos del sistema operativo. Algunos frameworks añaden funcionalidad adicional que se saltan las restricciones del navegador.

Aplicaciones de una sola página (SPA)

Una **Single Page Application (SPA)** es una aplicación web que a partir de una única página HTML va generando el contenido dinámicamente a través de JavaScript, normalmente al servidor únicamente se le solicita los datos en formato JSON o XML y el JavaScript se encarga de presentarlos.

Ventajas.

- Carga más rápida tras el inicio, ya sólo solicitamos datos.
- Mejora la experiencia del usuario, aspecto de aplicación nativa.
- Posibilidad de gestionar el enrutado en el frontend.
- Reduce la carga de trabajo y el envío de datos del servidor.

Desventajas

- Añade complejidad al desarrollo.
- Complica el SEO. Obliga a los buscadores a renderizar JavaScript.
- Mayor carga inicial.

Navegadores e Intérpretes de JavaScript

Cada navegador cuenta con su propio motor de JavaScript, que es el encargado de interpretar y ejecutar el código. Estos interpretes implementan las recomendaciones del W3C en mayor o menor medida, pudiéndose dar el caso de que una misma web según que navegador empleemos. Podemos verificar el nivel de soporte de cada navegador o de elementos HTML específicos en la web "**Can I Use It**" https://caniuse.com/

Principales Navegadores e Intérpretes de JavaScript:

- **Google Chrome:** Utiliza el motor **V8**, uno de los más rápidos y utilizados también fuera del navegador, como en Node.js.
- **Mozilla Firefox:** Usa el motor **SpiderMonkey**, que fue el primer intérprete de JavaScript y sigue evolucionando para optimizar el rendimiento.
- **Microsoft Edge:** Utiliza Chakra en versiones antiguas, pero las más recientes (**basadas en Chromium**) usan el motor V8, al igual que Chrome.
- **Safari:** Implementa **JavaScriptCore**, también conocido como Nitro, desarrollado por Apple.
- **Opera:** Al igual que Chrome, usa el motor V8 debido a su **base en Chromium**.

Herramientas de desarrollo

Para el desarrollo web disponemos de multitud de herramientas, desde el bloc de notas hasta IDEs avanzados. Durante el curso vamos a emplear las siguientes herramientas:

- Herramientas integradas en el navegador (**tecla F12**).
- Editor de texto avanzado **VS Code**. A destacar las extensiones:
    - EsLint. Linter para JavaScript que proporciona información y ayuda sensible a nuestro código.
    - Live Server. Refresca automáticamente el navegador cuando detecta un cambio en el código fuente.
- **Node.js.** Es un entorno en tiempo de ejecución de lado servidor que emplea JavaScript como lenguaje de programación. Adicionalmente nos permite probar código JavaScript sencillo. A destacar las extensiones:
    - **NPM**. Node Package Manager. Permite gestionar los paquetes JavaScript de nuestro proyecto. Ten en cuenta que cada versión de Node.js tiene su propia versión de NPM.
    - **NVM**. Node Version Manager. Permite gestionar la versión de Node.js de nuestro proyecto.

Bibliografía

- Desarrollo Web en Entorno Cliente. Lope González Vázquez. Editorial Paraninfo.
- Desarrollo Web en Entorno Cliente (DWEC). https://xxjcaxx.github.io/libro_dwec/intro.html