# ¿Qué es una aplicación Web?

Se define como aplicación Web aquellas herramientas donde los usuarios pueden acceder a un servidor web, ya sea mediante intranet o internet mediante un navegador.

# Arquitectura de una aplicación Web.

1. El cliente realiza una petición de un recurso
    -Introduce una dirección en el navegador.
    -A través de un servicio de DNS traductor los nombres de dominio a direcciones IP.
    -La dirección Ip le permite contactar con el servidor y enviarle la petición HTTP tipo get.
2. Se establece una conexión TCP.
3. El servidor proporciona el servicio solicitado.
4 Se cierra la conexión.

<u>Funcionamiento de protocolo HTTP.</u>

-El proceso se repite en cada acceso al servidor HTTP.
    Si se solicita un doc. HTML con 4 imágenes el proceso se repite 5 veces, 1 por imagen + el doc HTML.

-Tipo de mensajes que utilizan HTTP.
    -Get: recoge cualquier información del servidor, se usa al pulsar sobre un enlace o al teclear una URL.
    -Post: envía datos al servidor.
    -Head:solicita información sobre un fichero de tipo tamaño, fecha… Usado por gestores de cachés o servidores proxy para saber cuándo actualizar la copia de un fichero.

# Front-end VS Back-end

El front-end es la parte visual de una página, el diseño, contenidos, permite al usuario navegar por la página en cambio, el back-end se encarga de la gestión de datos de un servicio, de la conexión con las bases de datos, gestión de usuarios, distribución de información, permisos…

# ¿Qué es un cliente Web?

Un navegador web con el que interactúa el usuario.

Navegadores más usados:Chrome(64%), Safari(19%), Edge(3%), Opera(2%).

# Evolución y características de los navegadores.

World wide web(www).

-Conjunto de recursos interconectados que conforman el conocimiento humano.

. Hubs, repetidores, puentes…
. Protocolos de comunicación.
. Sistemas de nombres de dominio.

Configuración arquitectónica más habitual:
Cliente/servidor

-Cliente es un componente consumidor de servicios.
-Servidor es un proceso proveedor de servicios.

Navegador web:

-**Software** que utiliza el cliente para acceder al contenido ofrecido por los servidores de internet sin necesidad de instalar un programa.
-**Aplicación** (libre) que permite al usuario acceder a un recurso publicado por un servidor Web a través de Internet y descrito mediante una dirección URL.

Ejemplos de navegador Web:

+ **Mosaic**. Uno de los primeros navegadores Web y el primero con capacidades gráficas.
+ **Netscape Navigator** (después Communicator). Fue el primer navegador en incluir un módulo para la ejecución de código script (JavaScript). +
+ **Edge**. Es el navegador de Microsoft. +
Mozilla Firefox. Se trata de un navegador de código abierto multiplataforma de gran aceptación.
+ **Google Chrome**. Es el navegador de Google compilado a partir de componentes de código abierto.
+ **Safari**. Es el navegador por defecto de los sistemas de Apple. +
+ **Dolphin Browser**. Específico para el sistema operativo Android, fue uno de los primeros en incluir soporte para navegación multitáctil.

Criterios de clasificación:

- **Plataforma de ejecución**. Sistema operativo.

- **Características del navegador**. Funcionalidades adicionales.
- **Personalización de la interfaz**. Funciones de accesibilidad.
- **Soporte de tecnologías Web**. Grado de soporte de los estándares de la Web.
- **Licencia de software**. Código libre y navegadores propietarios.

# Arquitectura y ejecución.

Funcionamiento del Navegador Web:

1- Solicitar al servidor los **recursos** web que decida el usuario y mostrarlos en una ventana.
2- El **recurso** suele ser un doc. codificado en **HTML** aunque también pueden ser archivos (pdf, Word, audio, imagen…).
3- El usuario especifica la ubicación del recurso mediante el uso de una dirección **URI** o Identificador.

Estructura de una URI

Una Identificación Uniforme de Recursos (URI) es una cadena de caracteres que se utiliza para identificar de manera única un recurso en la web. Una URI consta de las siguientes partes:

**Esquema**: Indica el protocolo utilizado a la hora de solicitar el recurso, como "http://" , "https:// o ftp, file…".

**Parte jerárquica**: Puede incluir el nombre de dominio o la dirección IP del servidor y la **ruta**, la cual especifica la ubicación del recurso en el servidor.
Las dos barras inclinadas al principio “// ” indica que la dirección debe ser pasada al recurso para ser interpretada.
El servidor puede ser una dirección IP o un nombre de dominio y puede llevar parámetros como el puerto e información de control de acceso

**Solicitud**: Variables que se pasan al recurso. Se separan entre “?” y “#”,

**Fragmento**: Identifica una parte específica del recurso (subdirección)al que apunta la dirección.
El fragmento empieza despues del simbolo “#” y se extiende hasta donde se termina la URI.
La parte del fragmento en la que hace que sea una **URI** y una una **URL** ya que:

- Las URL no identifican fragmentos.
- se utiliza URI cuando se habla de direcciones completas.

- Las URL son identificadores que permiten acceder a recursos Web (páginas), mientras que las URI identifican.

Los dos elementos imprescindibles que deben contener todos los identificadores son esquema y ruta.

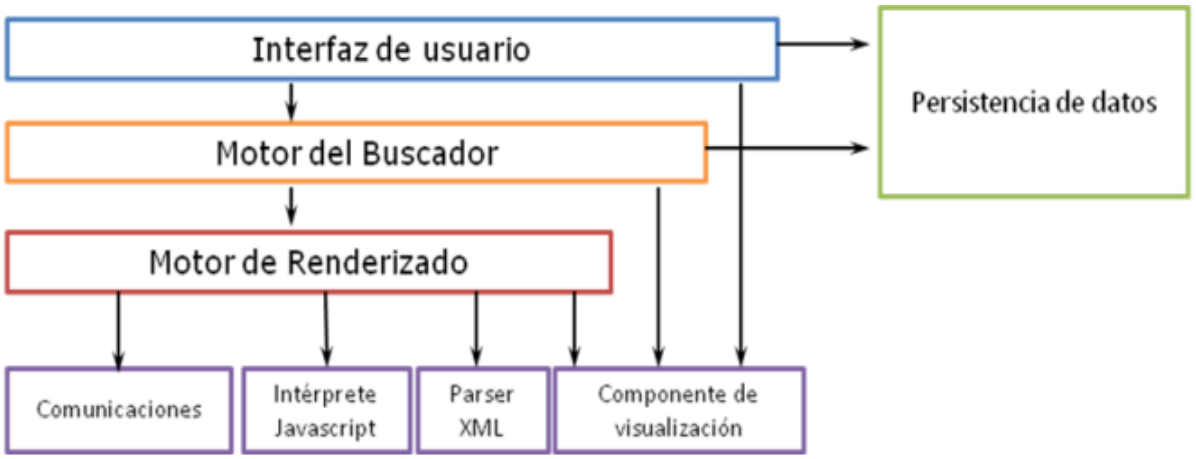
En la estructura del URI, los componentes se enumeran uno tras otro por este orden y están separados por caracteres estándar.

Esquema : Parte jerárquica path ? Solicitud # Fragmento

Proceso de ejecución:

El proceso se inicia con el usuario indicando la dirección del recurso al que quiere acceder y termina visualizando el recurso por pantalla.

Arquitectura de referencia de un navegador Web:




La "Arquitectura de referencia de un navegador web" es un término que se utiliza para describir la estructura general de un navegador web y cómo se dividen sus componentes principales. Aquí explico cada uno de los conceptos en el contexto de esta arquitectura:

1. **Interfaz de usuario**: Esta es la parte del navegador que **interactúa directamente con el usuario**. Incluye elementos como la barra de direcciones, botones de navegación (anterior, siguiente, recargar), pestañas, barras de herramientas y otros controles que permiten a los usuarios interactuar con el navegador.

2. **Motor de búsqueda**: Es el componente que **facilita la búsqueda en la web**. Los navegadores a menudo incluyen un cuadro de búsqueda en la interfaz de usuario que utiliza un motor de búsqueda para encontrar sitios web o información en línea.

3. **Motor de renderizado**:Este motor se encarga de **procesar y mostrar el contenido de las páginas web**. **Convierte el código HTML, CSS** y otros recursos en una representación visual que los usuarios pueden ver en la pantalla.

4. **Comunicaciones**: Este componente gestiona la **comunicación entre el navegador y los servidores** web. Incluye protocolos como **HTTP y HTTPS**, que permiten la descarga de páginas web y otros recursos desde servidores remotos.

5. **Intérprete JavaScript**: Los navegadores modernos incluyen un **motor de JavaScript que interpreta y ejecuta el código JavaScript** en las páginas web. Esto permite que las páginas web sean **interactivas** y **dinámicas**.

6. **Parser XML**:El parser XML en el navegador se encarga de **analizar documentos XML para** que puedan ser **procesados** y **presentados** de manera adecuada en la interfaz de usuario.

7. **Componente de visualización**: Este componente se encarga de mostrar el contenido de las páginas web en la ventana del navegador. Esto incluye la renderización de texto, imágenes, videos y otros elementos multimedia.

8. **Persistencia de datos**: La persistencia de datos se refiere a cómo se **almacenan** y **gestionan** los datos locales en el navegador, como **cookies**, **caché** y datos de formularios. Este componente gestiona la forma en que se almacenan y recuperan estos datos.

Proceso de carga en un navegador Web:

- A medida que el servidor va recibiendo el código el navegador lo va mostrando en el área destinada a ello.
- Se interpreta la estructura del documento y la búsqueda de recursos externos, scripts para su descarga.
- Imágenes, scripts y más archivos se guardan en carpetas temporales.
- El antivirus analiza estos archivos.
- La velocidad de carga es mayor si se repiten las peticiones, osea si se piden recursos al servidor los cuales todavía están almacenados en las carpetas temporales y en navegador verifica que esos recursos ya están descargados.

# Tipos de aplicaciones Web

No hay un número determinado de tipos de páginas web pero las vamos a clasificar por estáticas y dinámicas, múltiple page o single page y SPA vs MPA.

## Estáticas VS Dinámicas

**Estáticas:**

- Contenido fijo
- Nula interactividad
- Simples (doc HTML y doc CSS), cargan rápido.
- Actualización más tediosa.
- Ejemplo: portfolios, páginas de presentación, cv digitales…

**Dinámicas:**

- Generan datos en tiempo real en función de las peticiones del usuario.
- Utiliza bases de datos.
- Más complejas -> mayor tiempo de carga.
- Actualización más sencilla gracias a herramientas de gestión de contenido (CMS) como WordPress.
- -Ejemplos: Webs comerciales.

## Multi VS Single

**Multiple Page Applications (MPA):**

- Enfoque clásico de programación.
- La web está compuesta por varias páginas, que se cargan según el usuario va navegando.
- La carga inicial es muy rápida pero la experiencia puede ser peor a la hora de navegar entre páginas ya que el tiempo de carga es mayor.
- Protegida frente a vulnerabilidades.

**Single Page Applications (SAP):**

- Enfoque más actual de programación.
- Una única página web.
- El navegador solo carga las secciones necesarias para las peticiones del usuario.
- Carga inicial más lenta.
- La página se actualiza localmente en el navegador del cliente, en lugar de hacer consultas al servidor.
- Navegación más fluida -> mejor experiencia de usuario.
- -Más fácil de sufrir ciberataques.

# Capas de una aplicación Web

- Capa de **estructura** de la Web: Indica que elementos tiene y cómo se relacionan (cabecera, footer…) se escribe en HTML o derivados.
- Capa de **presentación** de los elementos Web: Indica cómo se deben mostrar los elementos (color, tamaño…). Se escribe en CSS.
- Capa de **comportamiento** de la Web:Gestiona los cambios producidos por interacciones del usuario con el navegador o porque llegan nuevas secciones desde el servidor (como nuevas noticias).

Los lenguajes de programación de entorno cliente son aquellos que se ejecutan en el navegador.

- Lenguajes principales:
  - Html, Dhtml, Xml, Xhtml.
- Lenguajes de scripting:
  - JavaScript, VBScript.
- Otros:
  - ActionScript, AJAX.

**HTML:**

- Lenguaje más utilizado en la World Wide Web.
- Sistema de etiquetas.
- Lenguaje interpretado.
- Uso de hipervínculos.

**XML:**

- Lenguaje parecido con etiquetas extensibles, se usa para organizar y transferir datos más que mostrarlos.

**DHTML:**

- Integración de HTML con lenguajes de scripting (JavaScript), hojas de estilo personalizadas (CSS) y la identificación de los contenidos de una página Web en formato de árbol (DOM).

**CSS:**

- Es el estilo que va a recibir la página.

**JAVASCRIPT:**

- Lenguaje interpretado normalmente insertado en el doc HTML.

**AJAX:**

- Acrónimo de JavaScript y XML.
- Conjunto de técnicas y métodos de desarrollo Web para la creación de aplicaciones Web interactivas y asíncronas.

**FrameWorks:** Permite la programación front-end de una forma más sencilla y estructurada. Ej: Angular, Vue,js, React…

# Herramientas de Edición

Es esencial utilizar un editor de texto para la programación web, se podría programar HTML, CSS y JS en el bloc de notas pero existen herramientas que facilitan la programación web formateando el código, autocompletando texto, detectando errores…

Notepad++, VisualStudioCode, SublimeText…

Herramientas **Online:** W3School, Fiddle, CodePen…

# Herramientas de depuración en navegadores.

Es posible ver el código de la página web a través del navegador pero gran parte está “enmascarado”. Los navegadores disponen de “herramientas para desarrolladores” que ayudan al desarrollo front-end. Las funciones que ofrecen los distintos navegadores son muy parecidas.

Podemos ver errores y mensajes en la parte de la “consola”.