¿Qué es una aplicación web?

Definición de **aplicación**: software que permite realizar al usuario una tarea específica o un conjunto de tareas en un dispositivo. Las aplicaciones permiten a los usuarios interactuar con el hardware y el sistema operativo para ejecutar funciones concretas, como escribir documentos, navegar por la web, reproducir música, entre otras actividades. Ejemplos:

- Aplicaciones de escritorio: Son programas que se instalan y se ejecutan en dispositivos como computadoras.

  Ejemplos: ¿?

- Aplicaciones móviles: Son programas diseñados para dispositivos móviles como teléfonos inteligentes o tabletas.

  Ejemplos: ¿?

- Aplicaciones web: es una aplicación que se ejecutan en un navegador y no requieren instalación local. (Google Chrome, Mozilla Firefox, Safari,..). Se accede a ellas a través de Internet.

  Ejemplos: ¿?

- Aplicaciones híbridas: combinan características de aplicaciones web y móviles. Se diseñan con tecnologías web, pero pueden instalarse como apps móviles.

Definición de **navegador**: software que permite el acceso a internet, interpretando la información de distintos tipos de archivos y sitios web para que éstos puedan ser visualizados. Los navegadores están presentes incluso en dispositivos más pequeños como tabletas o smartphones.

Funcionalidad: visualizar documentos, visitar páginas web, enlazar un sitio con otro, imprimir, enviar, recibir correos, etc...

Otros conceptos:

Hipervínculos: es un elemento de un página web que conecta con otro documento o página web dentro de la misma página. Al hacer clic en un hipervínculo, el navegador te lleva directamente a la ubicación vinculada. Generalmente, los hipervínculos se presentan como texto subrayado o botones, pero también pueden estar asociados a imágenes

Navegación: seguimiento de enlaces de una página a otra a través de un navegar. Al navegar, los usuarios pueden acceder a información, realizar búsquedas, interactuar con aplicaciones web y explorar enlaces dentro de diferentes sitios web.

URL (Uniform Resource Locator): dirección específica que se utiliza para acceder a un recurso de la web, como una página, una imagen, o un archivo. La URL indica la ubicación exacta de un recurso en Internet y normalmente comienza con "http://" o "https://", seguido del nombre del dominio y, opcionalmente, la ruta hacia un recurso específico.

Formato: protocolo://máquina/directorio/archivo

Ejemplo: https://www.wikipedia.org

La comunicación entre el servidor web y el navegador se realiza mediante el protocolo HTTP, aunque la mayoría de los navegadores soportan otros protocolos como FTP y HTTPS (una versión cifrada de HTTP basada en Secure Socket Layer o Capa de Conexión Segura (SSL)

Proceso de desarrollo de aplicaciones web

Cuando creamos aplicaciones web, se pueden utilizar diversos lenguajes y tecnologías. Principalmente hay dos tipos de tecnologías a tener en cuenta:

Tecnologías en el lado del cliente (Frontend)

Es la parte que tú como usuario, ves y experimentas directamente.

Ejemplo: Si accedes a la página web de un banco, puedes ver el contenido general y noticias. Todas las personas que acceden a dicha página ve lo mismo.

Tecnologías más comunes en el frontend:

- **HTML (HyperText Markup Language):** Es el lenguaje estándar para crear la estructura de las páginas web. Define los elementos como encabezados, párrafos, imágenes, enlaces, etc.

- **CSS (Cascading Style Sheets):** Es el lenguaje que se utiliza para darle estilo a las páginas web. Controla el diseño, los colores, los fondos, las fuentes y la disposición de los elementos.

- **JavaScript:** Es un lenguaje de programación que permite agregar interactividad y dinamismo a las páginas web. Con JavaScript, puedes manejar eventos, validar formularios, realizar animaciones y mucho más.

- **Frameworks y Bibliotecas de JavaScript:**

    - **React.js:** Una biblioteca para construir interfaces de usuario desarrollada por Facebook, basada en componentes.
    - **Angular:** Un framework completo mantenido por Google que ofrece una solución integral para desarrollar aplicaciones web dinámicas.
    - **Vue.js:** Un framework progresivo y ligero para construir interfaces de usuario que se ha vuelto muy popular por su simplicidad y flexibilidad.

Tecnologías en el lado del servidor (Backend)

Es la parte que el usuario no puede ver pero que es imprescindible, ya que lleva a cabo las funcionalidades y servicios de la web o app. Gestiona la lógica del servidor, la base de datos y la autenticación. Se encarga de procesar las peticiones del usuario, manejar los datos y devolver la respuesta al frontend.

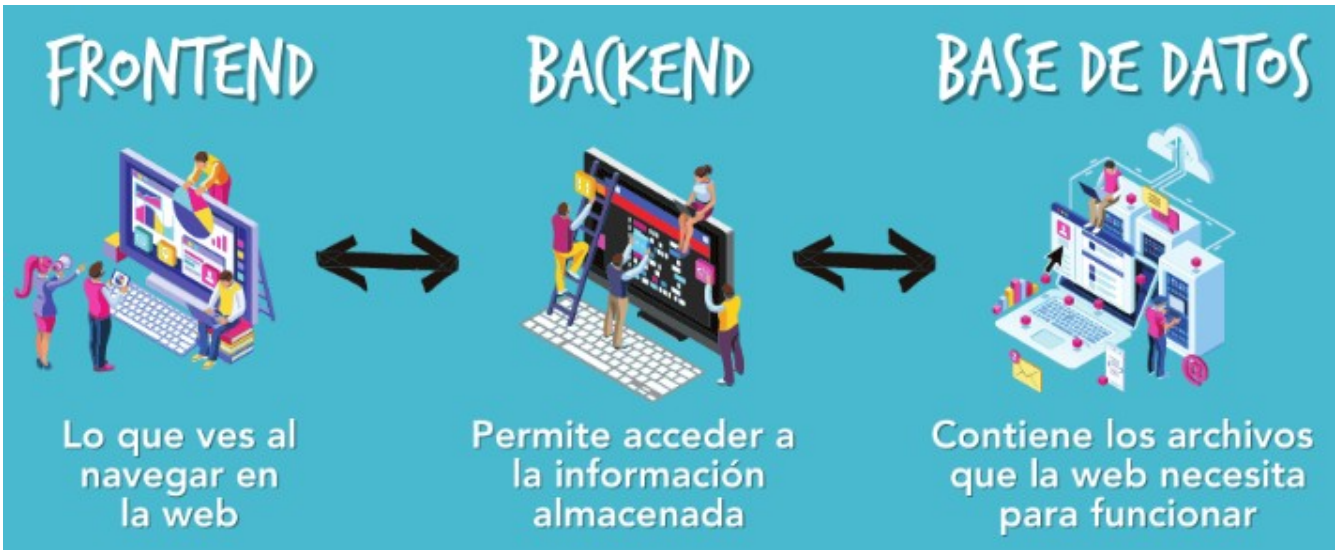
Ejemplo: si accedemos a la página del banco para poder ver nuestro saldo, tendremos que introducir claves. Aquí comienza a trabajar el backend.

Tecnologías y herramientas comunes en el backend:

- **Lenguajes de Programación Backend:**

    - **Node.js (JavaScript):** Permite utilizar JavaScript para el desarrollo del backend. Es muy popular debido a su capacidad para manejar aplicaciones en tiempo real y su

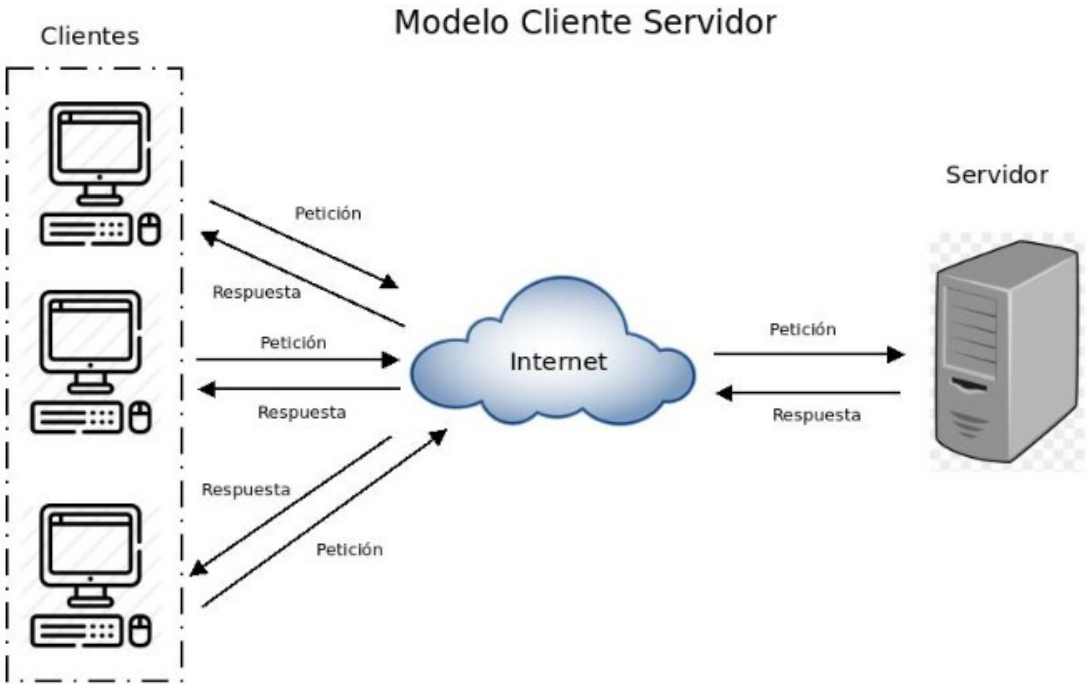
ecosistema de paquetes (npm).
    - **Python (Django, Flask):** Django es un framework robusto y completo, mientras que Flask es más ligero y flexible, ideal para proyectos pequeños o microservicios.
    - **Ruby (Ruby on Rails):** Un framework basado en Ruby que sigue el enfoque "convención sobre configuración", facilitando el desarrollo rápido de aplicaciones.
    - **PHP (Laravel):** PHP es uno de los lenguajes más tradicionales para desarrollo web, y Laravel es un framework que ha modernizado su uso.
    - **Java (Spring Boot):** Java es muy utilizado en aplicaciones empresariales, y Spring Boot es un framework potente para crear microservicios y aplicaciones robustas.
- **Bases de Datos:**
    - **SQL (Relacionales):** MySQL, PostgreSQL, SQLite. Se utilizan para manejar bases de datos estructuradas con tablas y relaciones.
    - **NoSQL:** MongoDB, Firebase, Cassandra. Adecuadas para manejar grandes volúmenes de datos no estructurados, como documentos o datos en tiempo real.
- **APIs y Comunicación:**
    - **REST:** Estándar para construir APIs web que permiten la comunicación entre el frontend y el backend.
    - **GraphQL:** Una alternativa a REST, que permite al cliente especificar exactamente qué datos necesita, optimizando las peticiones.
    - **WebSockets:** Tecnología para manejar comunicación en tiempo real entre el cliente y el servidor (chat en vivo, notificaciones en tiempo real).



Desarrollo web no es lo mismo que diseño web. El diseño es la parte de la estética de la web, deciden dónde se sitúan todos los elementos. El desarrollador a partir del diseño, construye la web. Se encarga de toda la parte funcional.

Arquitecturas web

Esquema básico de una arquitectura web cliente-servidor



1.- El usuario o cliente especifica en el navegador web la dirección de la página que desea consultar. (Escribe en el navegador la dirección (URL) de la página a visitar). Ejemplo: **https://www.ceuandalucia.es**

2.- El cliente establece una conexión con el servidor web.

3.- El servidor recibe la solicitud y procesa la entrada de datos y devuelve una repuesta.

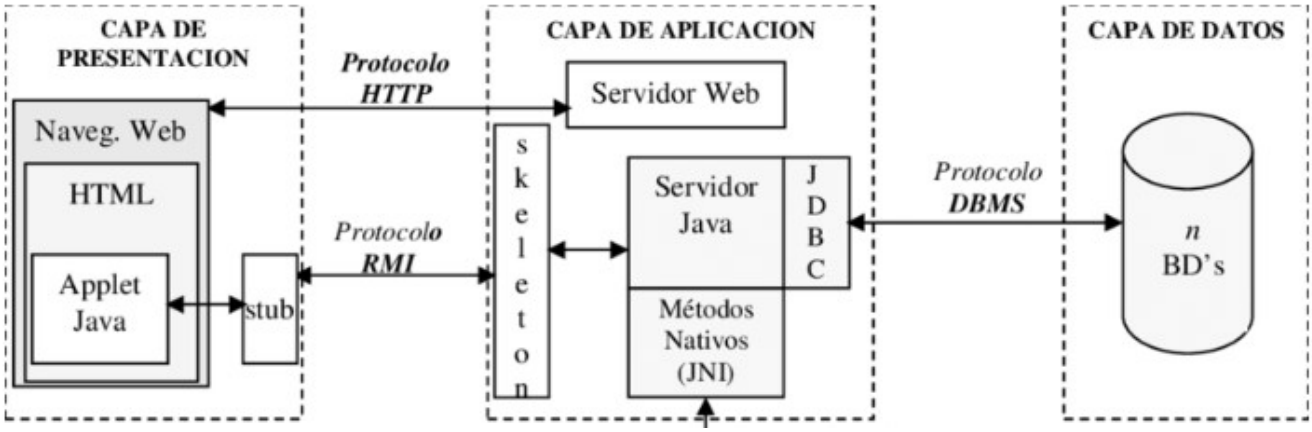
4.- Si se trata de una página HTML, el cliente inicia sus labores de interpretación de los códigos HTML. Si el cliente web encuentra instrucciones que hacen referencia a otros objetos que se tienen que mostrar con la página (imágenes, sonidos, animaciones multimedia, etc.), establece automáticamente comunicación con el servidor web para solicitar dichos objetos.

Arquitectura en tres capas: Separa la aplicación en tres capas independientes, cada una con una responsabilidad distinta.

- **Capa de presentación. Front-end**
  Conocida como capa del cliente o el navegador.
  Controla la parte de la aplicación web que llega al navegador.
  Ejemplos: HTML, CSS, Javascript

- **Capa lógica o proceso. Back-end**
  Conocida como capa del servidor o de aplicación
  Gestiona el funcionamiento interno de la aplicación.
  Ejemplos: PHP, ASP, Java, Javascript (lado servidor)

- **Capa de negocio o datos. Back-end**
  Conocida como capa de datos. Se encarga del acceso a base de datos y gestión de datos, recursos correo, audio, certificados..



¿Cuáles de estas tecnologías son del lado del cliente y cuáles de servidor?

HTML, CSS, JavaScript, XML, JSON, SVG, Flash, Applets

JavaScript (JS). Un poco de historia

JavaScript es un lenguaje de programación creado por Brendan Eich en 1995 con el objetivo de hacer la navegación web más dinámica e interactiva. Se creó para el navegador Netscape Navigator. Se convertió en standard en el año 1997.

Los programas escritos en JS, se ejecutan por el navegador web y no en el servidor donde se encuentra alojado el sitio.

Por otro lado, Microsoft creó en el años 1996 un lenguaje para Internet Explorer 3, jScript. Con características específicas y comportamientos distintos a los de JavaScript. Por lo tanto los sitios desarrollados con una de las dos versiones del lenguaje no eran compatibles con todos los navegadores.

Esta competitividad y guerra de tecnologías provocó la estandarización del lenguaje: ECMA (European Computer Manufacturers Association).

ECMA: https://www.w3schools.com/js/js_versions.asp

ECMA es una asociación fundada en Ginebra en 1961 para crear estándares del sector informático y de las telecomunicaciones. El primer estándar de JS que creó se denominó ECMA-262, en 1997, en el que se definió por primera vez el lenguaje ECMAScript. (dicta la base del lenguaje ECMAScript a través de su sintaxis, tipos, sentencias, palabras clave y reservadas, operadores y objetos)

Por este motivo, algunos programadores prefieren la denominación ECMAScript para referirse al lenguaje JavaScript. De hecho, JavaScript no es más que la implementación que realizó la empresa Netscape del estándar ECMAScript.

Desde entonces se han ido sucediendo distintas versiones de ECMAScript de forma anual:

EN 1998 se realizaron pequeñas modificaciones para adaptarlo al estándar ISO/IEC-16262 y se creó la segunda edición. *ECMAScript 2*

La tercera edición del estándar ECMA-262 (publicada en Diciembre de 1999) *ECMAScript 3* con soporte de expresiones regulares, nuevas sentencias de control, manejo de excepciones (bloque try-catch), definición de errores más precisa, formateo de salidas numéricas de datos, manejo de strings más avanzado...

ECMAScript 4, que aparece en 2004. (Nunca se lanzó, se canceló). Fue una versión muy ambiciosa que buscaba incorporar muchas características avanzadas, pero debido a diferencias entre las partes interesadas, nunca se lanzó oficialmente.

Las versiones de referencia son la 5 y la 6.

Las características que se incluyen en ECMAScript5 o ES5 (2009) (Versión actual de la mayoría de los navegadores con la aceptación de HTML 5 como estándar) son las siguientes: Getters y setters, Array extras y reductions, Rediseño de los atributos internos de las propiedades, Introducción de métodos estáticos de Object, cambios en el objeto Date, soporte nativo de JSON...

ECMAScript 6 o ES6 (2015): ES2015: Módulos, Ámbito a nivel de bloque (sentencia let),Iterators, String templates, hash tables..

Así sucesivamente...

Diferentes versiones de JavaScript junto con su nombre oficial y el año de su lanzamiento:

Nombre	Versión	Fecha
ES14	ES2023	Junio del 2023
ES13	ES2022	Junio del 2022
ES12	ES2021	Junio del 2021
ES11	ES2020	Junio del 2020
ES10	ES2019	Junio del 2019
ES9	ES2018	Junio del 2018
ES8	ES2017	Junio del 2017
ES7	ES2016	Junio del 2016
ES6	**ES2015**	**Junio del 2015**
ES5.1	ES5.1	Junio del 2011
ES5	ES5	Diciembre del 2009
ES4	ES4	Descartado
ES3	ES3	Diciembre de 1999
ES2	ES2	Junio de 1998
ES1	ES1	Junio de 1997

¿Cómo integramos código JavaScript en HTML?

Hay tres maneras de poder integrar código JS en HTML

1.- Embebido dentro de la etiqueta <script>

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
   <title>Ejemplo de código JavaScript en el propio documento</title>
  <script type="text/javascript">
    console.log("Un mensaje de prueba");
  </script>
</head>
<body>
<p>Un párrafo de texto.</p>
</body>
</html>
```

2.- Instrucciones dentro del código

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
   <title>Ejemplo de código JavaScript en el propio documento</title>
</head>
<body>
   <p onclick="console.log('Un mensaje de prueba')">Un párrafo de texto.</p>
</body>
</html>
```

3.- Archivo externo

```
<head>
   <script type="text/javascript" src="/js/codigo.js"></script>
</head>
```

Tendremos un fichero: codigo.js por ejemplo con el código: console.log("Un mensaje de prueba");

¿Cual pensáis que es la mejor de las tres?

Ventajas de utilizar fichero externo:

- Carga más rápida de páginas. Los archivos JavaScript externos pueden ser almacenados en caché por los navegadores, lo que reduce el tiempo de carga en visitas repetidas.
- Se separa el comportamiento de la estructura
- Se comparte código entre páginas. Reutilización
- Es más sencillo depurar errores
- Modularidad

Para finalizar...

1. ¿Sabríais explicar con vuestras palabras del funcionamiento de las aplicaciones web?
2. ¿Sabríais encontrar qué versiones de los navegadores Chrome, Firefox, Safari, Edge empezaron a soportar el ES2015?