

# TEMA 65. ARQUITECTURA DE DESARROLLO EN LA WEB. DESARROLLO WEB FRONT-END. SCRIPTS DE CLIENTE. FRAMEWORKS. UX. DESARROLLO WEB EN SERVIDOR, CONEXIÓN A BASES DE DATOS E INTERCONEXIÓN CON SISTEMAS Y SERVICIOS.

Actualizado a 13/04/2023

## 1. INTRODUCCIÓN

Las aplicaciones web son aquellas que permiten a los usuarios acceder a un servicio web alojado en un servidor a través de una red (Internet o intranet) mediante el uso de un navegador (por ejemplo, tu acceso al banco online o las redes sociales).

Utilizan estándares para la interfaz (HTML, CSS, JavaScript) y las comunicaciones (TCP/IP, DNS, HTTP).

## 2. ARQUITECTURA DE DESARROLLO WEB

Desde los primeros días de la World Wide Web en la década de 1990, la arquitectura de desarrollo web ha evolucionado enormemente para adaptarse a las necesidades de los usuarios y la tecnología.

En un principio, los sitios web eran simples páginas HTML con enlaces a otras páginas, pero a medida que la demanda de una experiencia más interactiva y personalizada creció, se crearon lenguajes de programación como JavaScript para crear aplicaciones web más dinámicas.

La arquitectura de desarrollo web tradicional es de tipo cliente-servidor y se estructura en N capas (lo más habitual son 2 o 3). En los últimos años se han creado nuevas arquitecturas y modelos de desarrollo que se resumen a continuación.

### 2.1 ARQUITECTURA CLIENTE-SERVIDOR

La arquitectura **cliente-servidor** es un modelo de diseño de software que se utiliza para dividir las tareas de una aplicación en dos partes principales: el cliente y el servidor.

**El cliente es la interfaz de usuario que se ejecuta en el dispositivo del usuario**, como una aplicación de escritorio o un navegador web, y se encarga de solicitar datos o servicios al servidor.

**El servidor es el componente que gestiona la lógica de la aplicación y los datos**, y responde a las solicitudes del cliente proporcionando los datos o servicios solicitados.

Esta arquitectura se utiliza ampliamente en el desarrollo de aplicaciones web, ya que permite una mayor escalabilidad y permite que los recursos del servidor sean compartidos entre múltiples clientes. Además, permite la separación de preocupaciones entre la interfaz de usuario y la lógica de negocio, lo que hace que la aplicación sea más fácil de mantener y mejorar. También es la base de otros servicios como el correo electrónico o redes de impresoras.

### 2.2 ARQUITECTURA EN CAPAS

En la **Arquitectura de Tres Niveles** aparecen tres funciones bien diferenciadas: el almacenamiento y gestión de los datos, el proceso de la lógica de la aplicación y la presentación de los datos e interacción con el usuario. Estas capas forman una única aplicación y se necesitan todas para poder desplegar una aplicación funcional. Este estilo de desarrollo recibe el nombre de **arquitectura monolítica**.

1. **Nivel de presentación:** Tiene como función realizar la presentación de la información extraída de la base de datos, interactuar con el usuario e interactuar con los objetos de aplicación. Los componentes visuales de la aplicación son Páginas HTML. (Cliente).

2. **Nivel de Aplicación o Lógica de negocio:** En este nivel intervienen los Objetos de Negocio. Dichos objetos están ubicados en el Servidor de Aplicaciones y son los encargados de procesar la lógica de la aplicación, así como de establecer el mecanismo de comunicación entre la información almacenada en la base de datos y el usuario. De esta manera, los componentes de las páginas HTML solicitarán cierta información a estos objetos de negocio, que a su vez solicitarán la información a la base de datos. (Servidor de Aplicaciones).
3. **Nivel de Datos:** Es el encargado del almacenamiento de los datos, y sus componentes son los Componentes de una Base de Datos Relacional: tablas, vistas, triggers, procedimientos almacenados, etc. (Servidor de Datos).

En el diseño en capas se recomienda utilizar patrones de diseño.

*Ampliar información en el T26 del temario de la Xunta en la carpeta de “Doc adicional”.*

## 2.3 MICROSERVICIOS

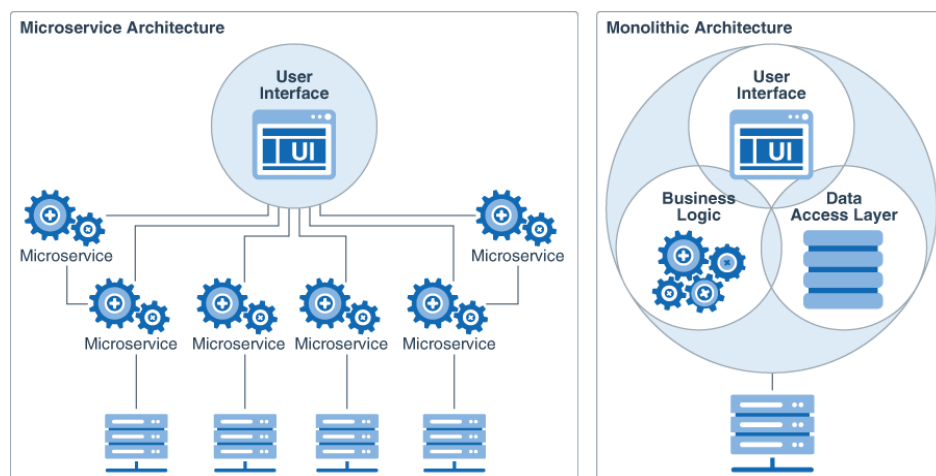
Es una nueva arquitectura que implica separar claramente la presentación (frontend) que se unifica de los distintos servicios a los que se llama de manera independiente (backend).

**Cada servicio se desarrollará con una aplicación diferente**, pueden hacerlo equipos diferentes y disponer de una tecnología y base de datos diferente. El objetivo es facilitar la escalabilidad, el rendimiento y el despliegue continuo de aplicaciones. Entre el frontend y el backend se suelen comunicar mediante una API REST mediante JSON, si bien puede haber múltiples alternativas.

**La arquitectura de microservicios es diferente de SOA, ya que cada servicio se implementa y opera de forma independiente, lo que permite la implementación frecuente de nuevas versiones y la escalabilidad autónoma de los microservicios.** Además, si un microservicio presenta un problema, sólo éste se verá afectado, mientras que los demás continuarán funcionando normalmente.

Aunque se pueden desarrollar con cualquier tecnología, existen varios frameworks especializados en el desarrollo de Microservicios:

- Java: Spring Boot, .NET: ASP.NET Core, PHP: Laravel



<https://docs.oracle.com/en/solutions learn-architect-microservice/>

Ante el incremento de implantación de microservicios y APIs, ha surgido un concepto reciente denominado **API GATEWAY**: es como un proxy inverso por delante de los microservicios, pero que permite añadir fácilmente ciertas funcionalidades a los microservicios. Estas serían algunas de las funcionalidades:

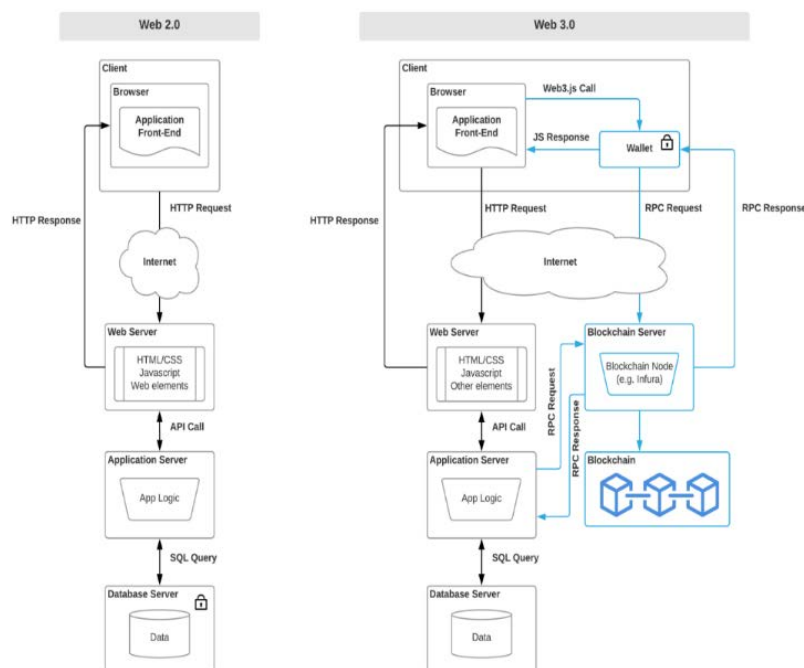
- Autenticación y Autorización (OAuth2, OpenID, SAML2...)
- Protección contra amenazas (inyección, límites de uso, monitorización...)
- Transformación de datos y *Content Negotiation*
- Traducción de protocolos (SOAP, REST...)
- Algunos ejemplos de Api Gateway son: KONG, Apigee (Google) o Amazon API Gateway.

## 2.4 P2P

Para construir aplicaciones web también puede utilizarse una arquitectura P2P (peer-to-peer), **compuesta por una red de ordenadores que pueden operar tanto como clientes como servidores** (por ejemplo, el famoso Emule), siendo por tanto una arquitectura descentralizada. Esta solución está indicada para servicios que requieran muchos recursos, distribuye los costes y es altamente escalable. Además, aprovechan, administran y optimizan el uso del ancho de banda. Es necesario evitar los posibles nodos maliciosos, que pudieran infectar la red P2P y como la mayor parte de los nodos de internet no tienen una IP fija, surge el problema de cómo localizar un nodo dentro de la red o cómo conectar nodos sin IP pública. Esta tecnología es muy utilizada a nivel de protocolos de acceso anónimos como TOR o I2P.

## 2.5 WEB3

WEB3 es una arquitectura emergente que se centra en la creación de aplicaciones descentralizadas (dApps) y en la integración de tecnologías Blockchain en el desarrollo web.



<https://www.coinbase.com/blog/understanding-web-3-a-user-controlled-internet>

**Las aplicaciones descentralizadas son aplicaciones web que utilizan tecnologías Blockchain para descentralizar los procesos de transacción y almacenamiento de datos, lo que les permite ser más seguras y resistentes a la censura y al control centralizado.** Las dApps se construyen utilizando una combinación de tecnologías web tradicionales, como HTML, CSS y JavaScript, junto con frameworks y bibliotecas específicas de Blockchain, como Ethereum, Polkadot, Cardano, entre otros.

WEB3 también implica la creación de una red de nodos descentralizados que ejecutan las aplicaciones y verifican las transacciones, lo que permite la creación de sistemas confiables y transparentes sin necesidad de una autoridad central.

### 3. DESPLIEGUE DE APLICACIONES

Desplegar una aplicación web significa ponerla en funcionamiento y hacerla accesible en un servidor web para que los usuarios puedan acceder a ella a través de Internet. Esto implica la configuración y preparación del entorno de ejecución en el servidor y otros servicios. A continuación, se muestran las formas más comunes de despliegue de aplicaciones.

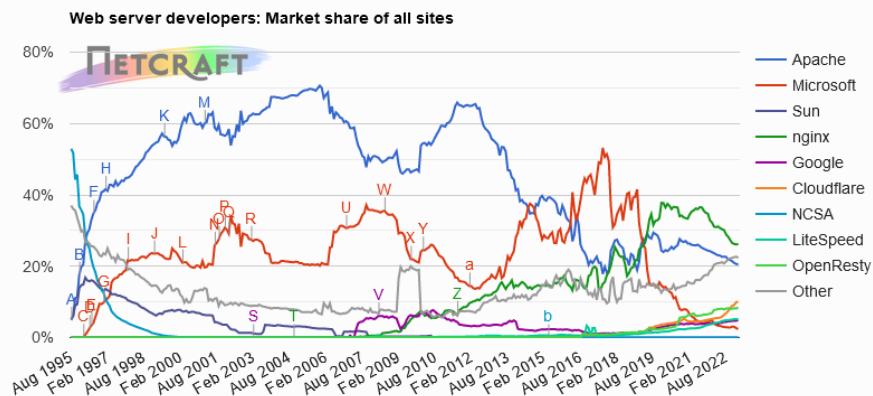
#### 3.1 SERVIDORES WEB

Un servidor web o servidor HTTP es un programa informático que procesa una aplicación del lado del servidor, realizando conexiones bidireccionales o unidireccionales y síncronas o asíncronas con el cliente y generando o cediendo una respuesta en cualquier lenguaje o Aplicación del lado del cliente.

Los servidores web más importantes para el desarrollo de aplicaciones web son los siguientes:

- ✓ **Apache** (líder del mercado en base instalada): servidor web de software abierto desarrollado por la Apache Software Foundation (ASF):
  - Disponible para GNU/Linux, Windows, MacOS.
  - Dispone de una gran comunidad (lo que agiliza el desarrollo).
  - Sus características son altamente configurables y expandibles vía plugins
  - Gran número de características de seguridad y buena compatibilidad
- ✓ **Apache Tomcat**: Contenedor de Servlets, que implementa las especificaciones de estos y de las JSP-Java Server Pages:
  - Software abierto, soportado en GNU/Linux, Windows, MacOS.
  - Puede usarse en combinación con Apache.
  - Principalmente usado para desplegar aplicaciones Java.
  - Existen otras alternativas comerciales para desplegar aplicaciones en Java:
    - JBoss (aunque tiene versión Open Source)
    - Weblogic
    - Glassfish
    - Websphere
- ✓ **Internet Information Services (IIS)**: Servidor web del entorno Windows:
  - Permite publicar páginas web de forma local (intranet) como remota (Internet).
  - Se basa en varios módulos que le dan capacidad para procesar distintos tipos de páginas (por ejemplo, Microsoft incluye los de Active Server Pages (ASP) y ASP.NET. También pueden ser incluidos los de otros fabricantes, como PHP3o Perl.4).

- ✓ **NGINX:** de software libre y código abierto, bajo licencia BSD simplificada:
  - Puede ofrecer funciones de proxy inverso ligero de alto rendimiento y de proxy para protocolos de correo electrónico (IMAP/POP3).
  - A diferencia de Apache (basado en procesos) se trata de un software asíncrono, lo que le permite mayor escalabilidad y rendimiento.
  - Por tanto, usado por aplicaciones muy conocidas como Facebook, Netflix o GitHub.
  - Disponible para GNU/Linux, BSD, Solaris, Mac OS X y Windows.
- ✓ **Node.JS:** Software libre, permite la ejecución de Javascript para el backend
- ✓ **Servidores embebidos como Spring Boot (java) o Kestrel (ASP.NET Core) para Microservicios**
  - Se arranca un servidor web al arrancar la aplicación.
  - Encajan con el desarrollo y despliegue de aplicaciones mediante contenedores.



Developer	January 2023	Percent	February 2023	Percent	Change
nginx	295,678,304	26.11%	295,723,793	26.23%	0.11
Apache	233,636,177	20.63%	231,042,423	20.49%	-0.15
Cloudflare	112,159,331	9.91%	113,829,198	10.09%	0.19
OpenResty	92,291,824	8.15%	95,176,082	8.44%	0.29

<https://news.netcraft.com/archives/2023/02/28/february-2023-web-server-survey.html>

### 3.2 CONTENEDORES

**Los contenedores de aplicaciones son entornos de ejecución portables y aislados que encapsulan una aplicación y sus dependencias, permitiendo su ejecución en cualquier sistema operativo y servidor compatible con contenedores.** Cada contenedor es una instancia aislada y portátil que incluye todo lo necesario para que la aplicación se ejecute, lo que facilita su despliegue y mantenimiento. Existen distintas plataformas de contenedores como Docker o LXC (Linux Containers) así como sistemas de gestión de clústeres como Kubernetes, Rancher o Amazon ECS.

*Ampliar información en el Tema 131 de PreparaTIC - Virtualización de sistemas y de centros de datos. Virtualización de puestos de trabajo. Maquetas de terminales Windows y de servidores Linux.*

Desplegar una aplicación web en contenedores implica encapsular la aplicación y todas sus dependencias en un entorno aislado y portable, que puede ser ejecutado en cualquier servidor compatible con contenedores. Esta técnica de despliegue es diferente de los servidores web tradicionales, ya que los contenedores proporcionan un entorno más flexible, escalable y portátil para el despliegue.

**En el contexto de microservicios, desplegar aplicaciones en contenedores permite implementar y escalar cada servicio de forma independiente, lo que mejora la flexibilidad y la disponibilidad de la aplicación.** En este enfoque, cada servicio se ejecuta en su propio contenedor, lo que permite una mayor granularidad y facilidad de mantenimiento.

Por otro lado, **en aplicaciones monolíticas, desplegar la aplicación en contenedores también puede proporcionar beneficios, como una mayor portabilidad y consistencia en el entorno de desarrollo y producción.** Además, la utilización de contenedores puede permitir la gestión más eficiente de los recursos del servidor y una mayor flexibilidad para el escalado y la gestión de la aplicación.

### 3.3 Cloud

Desplegar una aplicación web en entornos cloud significa alojarla en infraestructuras de computación en la nube, lo que proporciona una mayor flexibilidad, escalabilidad y disponibilidad en comparación con los servidores tradicionales.

Los sistemas **PaaS (Platform-as-a-Service) e IaaS (Infrastructure-as-a-Service)** son dos tipos de servicios en la nube que se utilizan para desplegar aplicaciones. PaaS proporciona una plataforma completa de desarrollo y despliegue de aplicaciones, mientras que IaaS proporciona acceso a servidores, almacenamiento y redes virtuales para desplegar aplicaciones personalizadas. Los sistemas PaaS e IaaS proporcionan diferentes niveles de control y flexibilidad en la gestión de la infraestructura respecto a un CPD tradicional.

Por otro lado, **los sistemas serverless o FaaS (Function-as-a-Service) son un tipo de servicio en la nube que permite a los desarrolladores desplegar funciones individuales de código en lugar de aplicaciones completas.** En este enfoque, el proveedor de la nube se encarga de la infraestructura, y los desarrolladores solo pagan por la ejecución de las funciones.

*Ampliar información en el Tema 55 de PreparaTIC - Cloud Computing. IaaS, PaaS, SaaS. Nubes privadas, pública e híbridas*

## 4. DESARROLLO WEB FRONTEND

**El desarrollo web frontend se refiere a la creación de la parte visible y funcional de un sitio web,** con la que interactúa el usuario final. Esto incluye la implementación de elementos visuales como la disposición de los elementos, el diseño gráfico, la navegación y la experiencia del usuario.

**El desarrollo web backend se refiere a la creación de la lógica del servidor que se ejecuta detrás de la interfaz de usuario.** Esto incluye la gestión de la base de datos, la implementación de la lógica de negocio y la programación de los algoritmos necesarios para que la aplicación web funcione correctamente.

La principal diferencia entre el desarrollo frontend y backend es que el frontend se enfoca en la experiencia del usuario, mientras que el backend se enfoca en la gestión de la lógica del servidor y la manipulación de datos. Ambos aspectos son cruciales para el éxito de una aplicación web, y los desarrolladores frontend y backend trabajan juntos para crear un sistema funcional.

#### 4.1 ELEMENTOS COMUNES

**HTML (Hypertext Markup Language)** es un lenguaje de marcado utilizado para crear la estructura y el contenido de una página web. **CSS (Cascading Style Sheets)** es un lenguaje de estilo que se utiliza para definir la presentación y el aspecto visual de una página web. **JavaScript** es un lenguaje de programación que se utiliza para agregar interactividad y dinamismo a una página web, permitiendo la creación de efectos visuales, animaciones y la manipulación de contenido. En conjunto, estos tres elementos son la base para la creación de páginas web modernas y atractivas.

**XML (Extensible Markup Language) y JSON (JavaScript Object Notation)** son formatos de intercambio de datos utilizados para transmitir información entre servidores y clientes. XML es un lenguaje de marcado extensible que permite definir etiquetas personalizadas para representar datos y estructuras complejas, mientras que JSON es un formato más ligero y fácil de leer que se basa en objetos y arrays. Es posible además validar ambos utilizando esquemas y reglas personalizadas.

*Ampliar información en el Tema 77 de PreparaTIC - Lenguajes y herramientas para la utilización de redes globales. HTML, CSS y XML. Navegadores web y compatibilidad con estándares*

#### 4.2 JAVASCRIPT Y ECMASCRIPT

JavaScript es un lenguaje de programación de alto nivel que se utiliza principalmente para crear aplicaciones web interactivas. **ECMAScript es un estándar de scripting para la creación de lenguajes de programación basados en JavaScript. Por lo tanto, JavaScript es uno de los lenguajes de programación que se basa en el estándar ECMAScript.**

JavaScript fue creado por Brendan Eich mientras trabajaba en Netscape Communications Corporation en 1995. La primera versión de JavaScript se llamaba Mocha, que luego se convirtió en LiveScript y finalmente se renombró como JavaScript. A medida que el lenguaje se hizo más popular, se convirtió en un estándar de facto y se implementó en diferentes navegadores web.

*Ampliar información con el fichero "XVII\_JavaScript" de la carpeta de "Doc adicional".*

En 1997, **Ecma International (European Computer Manufacturers Association) comenzó a trabajar en la estandarización de JavaScript y creó el estándar ECMAScript.** La primera versión de ECMAScript, llamada ECMAScript 1, se basó en la versión original de JavaScript de Netscape. Desde entonces, se han lanzado varias versiones de ECMAScript, cada una con nuevas características y mejoras.

Cada vez que se lanza una nueva versión de ECMAScript, los navegadores web y otros entornos de JavaScript, como Node.js, implementan estas nuevas características en sus motores de JavaScript. Los desarrolladores pueden utilizar estas características para escribir código más moderno y eficiente.

**TC39 es el Comité Técnico 39 de Ecma International, el organismo que desarrolla y mantiene el estándar ECMAScript. El trabajo de TC39 se centra en la evolución del lenguaje,** que implica el diseño y la implementación de nuevas características y mejoras en ECMAScript. Las propuestas para nuevas características son presentadas por miembros de TC39 y otros miembros de la comunidad de desarrolladores de JavaScript. Estas propuestas son evaluadas por el comité y, si son aceptadas, se incorporan al estándar ECMAScript. TC39 tiene un proceso formal para la aprobación de nuevas características de ECMAScript, que se divide en varias etapas. Cada etapa representa un nivel de madurez de la propuesta, desde la etapa de propuesta inicial hasta la etapa de aprobación final y la inclusión en la especificación ECMAScript.



La relación entre TC39 y ECMAScript es estrecha, ya que TC39 es responsable de desarrollar y mantener el estándar ECMAScript. Las nuevas características y mejoras de ECMAScript son diseñadas y desarrolladas por TC39 y luego se incorporan al estándar ECMAScript. La evolución continua de ECMAScript a través de TC39 asegura que JavaScript siga siendo un lenguaje de programación moderno y relevante para las necesidades de los desarrolladores y la industria en general.



<https://www.freecodecamp.org/news/tc39-and-its-contributions-to-ecmascript-c178b77f32e1/>

#### 4.2.1 ESTÁNDARES ECMA

Ecma Internacional y posteriormente el TC39 han publicado varias revisiones de la especificación de ECMAScript desde su creación en 1997. El desarrollo original de ECMAScript es el siguiente:

- ✓ **ECMAScript 1 – ECMA-262 (ISO/IEC 16262) - 1997**
  - JavaScript Standard
- ✓ ECMAScript 2 – ECMA-262 (ISO/IEC 16262) - 1998
- ✓ ECMAScript 3 – ECMA-262 (ISO/IEC 16262) - 1999
- ✓ **ECMAScript 4 – Abandonado – 2003 (last draft)**

A comienzos de los años 2000, el ecosistema web se encontraba muy fragmentado. Por un lado, Microsoft conseguía el monopolio de los navegadores web con Internet Explorer destronando a Netscape Navigator (IE 3 utilizaba JScript, una variante de JavaScript diseñada mediante ingeniería inversa a partir de la implementación JavaScript de Netscape) y tecnologías como Macromedia Flash (posteriormente Adobe Flash) y su lenguaje ActionScript, así como las Applets de Java, muy populares en entornos empresariales.

En 2003 Netscape propone la cuarta revisión de ECMAScript que es descartada por considerarse muy extensa, compleja y romper los navegadores y sitios web de la época. Netscape pierde casi toda su cuota de mercado mientras que .NET, Java Applets y sobre todo Flash ganan popularidad. En un ecosistema web cada vez más complejo y frágil, se forman dos grupos de trabajo dentro de TC39: por una parte, una reorganización de ECMAScript 3 (ECMAScript 3.1) y un rediseño completo de ECMAScript llamado ECMAScript 4. Finalmente, a finales 2009, con la llegada de los smartphones, nuevos navegadores como Chrome o Firefox y descontento general de los desarrolladores por la fragmentación, la seguridad y la usabilidad de la web, se publica ECMAScript 5, que es un rediseño de ECMAScript 3 y se descarta finalmente ECMAScript 4.

- ✓ **ECMAScript 5 – ECMA-262 (ISO/IEC 16262) - 2009**
  - Gran reorganización del entorno JS, soporte JSON y modo estricto
- ✓ ECMAScript 5.1 – ECMA-262 (ISO/IEC 16262:2011) - 2011
  - Reajuste para adaptarse a la nueva ISO

**ECMAScript 5 fue una actualización importante de la especificación ECMAScript que incluyó nuevas características como métodos de Array y Object, la función `JSON.parse()` y `JSON.stringify()`, y**

una variedad de mejoras de rendimiento. A principios de 2010, Node.js empieza a ganar popularidad a la vez que Flash y otras tecnologías empiezan a perder tracción mientras que JavaScript se empieza a consolidar como el nuevo estándar de la web. A mediados de 2012 los grandes navegadores son 100% compatibles con ECMAScript 5. Con este cambio de paradigma, en 2015 se publica ECMAScript 6, que se considera el inicio del JavaScript contemporáneo

#### ✓ **ECMAScript 6 – ECMA-262 (ISO/IEC 16262:2011) - 2015**

- Cambio sustancial del entorno JS/HTML5. Inicio del JS contemporáneo
- No soportado al completo en IE11 y entornos móviles antiguos
- Uso de transpiladores y código asíncrono

ECMAScript 6 o ECMAScript 2015: lanzado en 2015, ES6 fue una actualización significativa de ECMAScript que introdujo nuevas características como el operador spread, las funciones de flecha, el uso de `let` y `const` para declarar variables, los módulos `import/export`, funciones asíncronas, nuevas colecciones de datos y una mejora considerable de la eficiencia y seguridad del código.

**Esta nueva versión es un cambio significativo y no es compatible con navegadores antiguos (como Internet Explorer 11). Para ello se diseñaron “transpiladores” que convierten código escrito en ECMAScript 6 y lo convierten a ECMAScript 5.** Algunos de los transpiladores más conocidos son Babel o TypeScript. ECMAScript 6 viene acompañado del nuevo paradigma de desarrollo de aplicaciones web que gana popularidad: Single Page Applications (SPA). A partir de entonces, cada año se publica una revisión menor de ECMAScript:

#### ✓ **ECMAScript 2016 – ECMA-262 (ISO/IEC 16262:2011) – 2016**

- Primera versión post-ES6

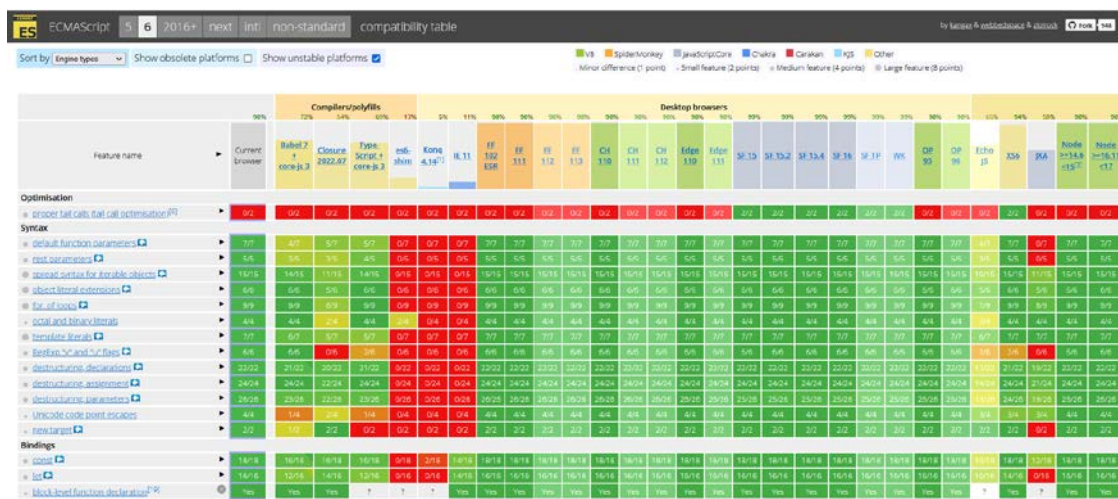
#### ✓ **ECMAScript 2018 – ECMA-262 (ISO/IEC 22275:2018) – 2018**

- Nueva ISO **ISO/IEC 22275:2018**

#### ✓ **ECMAScript 2022 – ECMA-262 (ISO/IEC 22275:2018) – 2022**

- Última versión publicada

Además se crea el concepto de **ES.Next** para referirse a las nuevas características experimentales o de futuras revisiones que los navegadores añaden progresivamente



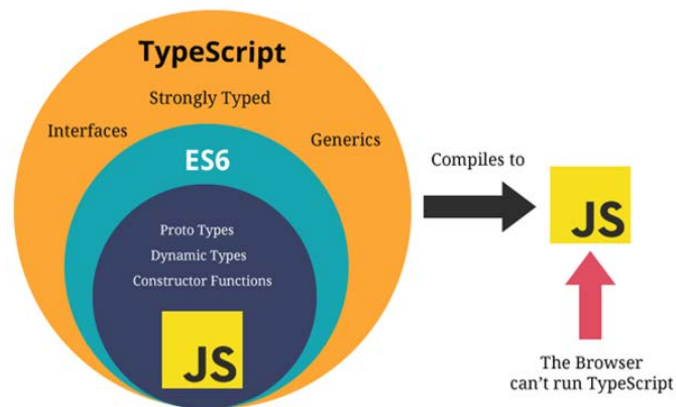
The screenshot shows the ECMAScript Compatibility Table for ES6 and ES7 features. The table lists features like 'proper tail calls', 'default function parameters', 'rest parameters', 'spread syntax', 'for...of loops', 'async functions', 'destructuring', 'Unicode code point escapes', and 'new targets'. Each feature is evaluated across various browsers and engines, with support levels indicated by colored cells (green for full support, yellow for partial, red for not supported).

<https://kanqax.github.io/compat-table/es6/>

Ecma Internacional ha desarrollado además otros estándares de tecnologías muy relevantes en el desarrollo web como:

- ✓ **ECMA-334 – C# programming language (ISO/IEC 23270)**
- ✓ ECMA-376 – Office Open XML
- ✓ **ECMA-402 – ECMAScript Internationalization API Specification**
- ✓ **ECMA-404 – The JSON Data Interchange Format**
- ✓ ECMA-408 – Dart Programming Language Specification

#### 4.2.2 TYPESCRIPT



<https://openwebinars.net/blog/que-es-typescript/>

JavaScript se ha convertido en el nuevo estándar de la web con amplio soporte de herramientas y librerías. Sin embargo, el lenguaje creado en 10 días sigue teniendo aspectos complejos y que limitan el desarrollo. Con el cambio a ECMAScript 6, los transpiladores como Babel han ganado popularidad para soportar navegadores antiguos como Internet Explorer 11. Este mismo concepto de transpilación o compilar un lenguaje de alto nivel a otro de alto nivel, se utiliza también para utilizar nuevos lenguajes de programación.

**TypeScript** llega de la mano de Microsoft en 2012 y se posiciona como una alternativa sólida a JavaScript, con soporte completo en frameworks como Angular o React. TypeScript trata de solucionar uno de los problemas de JavaScript: tipado estático. **Es una extensión de JavaScript que añade nuevas funcionalidades relacionadas con la definición de tipos de objetos y clases**, uso de interfaces y genéricos, así como una mejor gestión de grandes proyectos.

**El lenguaje TypeScript (y similares como Dart) no son compatibles directamente con la mayoría de los navegadores y necesita ser transpilado a JavaScript para ser utilizado.**

Además de TypeScript, existen otros lenguajes similares como Dart (creado por Google) o Flow (creado por Facebook). Además de lenguajes de uso general, varios frameworks han desarrollado sintaxis especiales para desarrollar webs de forma más eficiente que son 100% transpilables a JavaScript como es el caso de los ficheros JSX de componentes de React que permiten definir código HTML dentro de funciones JavaScript o los ficheros VUE del framework Vue.js que combinan HTML, JavaScript y CSS en el mismo fichero.

### 4.3 APIs DE JAVASCRIPT Y HTML5

Las APIs de JavaScript son conjuntos de funciones y métodos que permiten a los desarrolladores acceder a funcionalidades específicas del navegador y del sistema operativo. Estas APIs son proporcionadas por los navegadores web modernos y son accesibles a través de clases estáticas u objetos globales en JavaScript, como `window` o `navigator`. HTML5 también incluye una serie de APIs que se pueden acceder desde JavaScript y que añaden funcionalidad a la estructura estática de HTML4.

A continuación, se muestra una lista de las APIs más utilizadas:

- ✓ **Ejecución Asíncrona y obtención de datos: AJAX, Promises y API Fetch**
  - **AJAX (Asynchronous JavaScript and XML) es una técnica de programación que permite actualizar partes de una página web sin tener que recargar la página completa.** AJAX utiliza JavaScript para enviar y recibir datos en segundo plano, lo que permite a las aplicaciones web ofrecer una experiencia de usuario más rápida y dinámica.
  - Una **Promise (en español Promesa) es un objeto que representa el resultado de una operación asíncrona,** que puede ser una respuesta de una solicitud AJAX, un resultado de una operación de base de datos o cualquier otro tipo de operación que pueda tardar un tiempo en completarse. Las *Promises* utilizando los métodos `then()` y `catch()`, lo que hace que el código sea más legible y fácil de seguir.
  - La API *Fetch* es una API nativa de JavaScript que se utiliza para realizar solicitudes HTTP (por ejemplo, GET, POST, PUT, DELETE) a un servidor web. Fetch utiliza *Promises* para manejar las respuestas de la solicitud, lo que permite a los desarrolladores escribir código asíncrono más limpio y fácil de leer. Fetch es la opción recomendada en aplicaciones modernas, pero está soportada en navegadores antiguos como IE11. Son la base de librerías como Axios
- ✓ **Modificación de DOM**
  - **El DOM (Document Object Model) es una interfaz de programación que representa los elementos de una página web como objetos,** permitiendo a los desarrolladores manipular y actualizar el contenido de la página mediante JavaScript.
  - APIs de búsqueda de DOM:
    - `document.getElementById("mi-elemento");`
    - `document.getElementsByClassName("mi-clase");`
    - `document.querySelector("#mi-elemento .mi-clase");`
    - `document.querySelectorAll(".mi-clase");`
  - APIs de modificado de DOM:
    - `document.createElement("div");`
    - `parent.appendChild(elem);`
    - `parent.removeChild(child);`
  - Esta es la forma nativa de acceder a los elementos, sin embargo, es bastante limitada y existen diversas librerías como jQuery que facilitan esta tarea.
- ✓ **Almacenamiento**
  - **`window.localStorage`**
    - Esta API permite almacenar datos en el navegador de forma persistente, incluso después de cerrar el navegador o reiniciar el dispositivo. Los datos se almacenan como pares clave-valor.

- **window.sessionStorage**
  - Esta API permite almacenar datos en el navegador de forma temporal, solo durante la sesión actual. Los datos se almacenan como pares clave-valor
- **window.indexedDB**
  - Esta API permite almacenar grandes cantidades de datos estructurados en el navegador. Los datos se almacenan en una base de datos local y se pueden acceder utilizando transacciones.
- Cookies
- FileAPI

✓ **WebSockets**

- Los **WebSockets** son una API de JavaScript que proporciona **una forma bidireccional de comunicación en tiempo real entre el navegador del usuario y un servidor web**. Los WebSockets permiten una comunicación más eficiente y escalable que otras tecnologías de comunicación como AJAX, porque establecen una conexión persistente entre el navegador y el servidor, lo que permite una comunicación bidireccional en tiempo real.
- Para utilizar WebSockets en JavaScript, se debe crear un objeto WebSocket, que se conecta al servidor utilizando una URL de WebSocket. Una vez establecida la conexión, el objeto WebSocket puede enviar y recibir mensajes a través de la conexión. El protocolo de WebSockets es generalmente `wss://mi-servidor.com/mi-ruta`
- Existen otras variantes a esta tecnología para comunicación en tiempo real
  - Polling
  - Publish/Subscribe
  - Server-sent Events (SSE)
- Servidores de WebSockets
  - Socket.IO (Node.js)
  - ws (Node.js)
  - javax.websockets (Java)
  - SignalR (.NET)

✓ **WebAssembly - WASM**

- **WebAssembly es un lenguaje de bajo nivel diseñado para mejorar el rendimiento de las aplicaciones web**. Es un formato binario portátil que se ejecuta dentro del navegador web y se puede utilizar para ejecutar código escrito en una variedad de lenguajes, incluyendo C, C++, Rust y otros.
- WebAssembly se creó para mejorar el rendimiento de las aplicaciones web, especialmente para aquellas que requieren una gran cantidad de cálculos y procesamiento de datos. A diferencia de JavaScript, que se ejecuta en un motor de scripting dentro del navegador, el código de WebAssembly se compila en código de máquina nativo, lo que permite una ejecución más rápida y eficiente.
- Soportado por la mayoría de los navegadores modernos
- Frameworks de desarrollo web basados en WASM
  - Blazor (.NET)
  - Vugu (Go)
- Proyectos que usan WASM
  - Unity
  - AutoCAD
  - Google Earth

✓ **WebRTC**

- *Web real-time communications* o **comunicaciones web en tiempo real** es una tecnología que permite a aplicaciones y sitios web capturar y opcionalmente retransmitir audio/vídeo, así como intercambiar datos arbitrarios entre navegadores sin necesidad de un intermediario. El conjunto de estándares que comprende WebRTC hace posible compartir datos y realizar teleconferencias de *peer-to-peer*, sin requerir que el usuario instale complementos o cualquier otro software de terceros.
- Soporte por la mayoría de los navegadores modernos
- Para utilizar WebRTC en JavaScript, se debe crear una conexión `RTCPeerConnection` y establecer una conexión con otro navegador web. Una vez establecida la conexión, se pueden enviar y recibir datos de audio y video utilizando los objetos `MediaStream` y `RTCDataChannel`.
- Productos que usan WebRTC
  - Jitsi
  - Discord
  - Blackboard

✓ **WebWorkers**

- Los Web Workers hacen posible ejecutar la operación de un script en un hilo en segundo plano separado de la ejecución del hilo principal de la aplicación web. La ventaja de esto es que un proceso laborioso puede actuar en un hilo separado, permitiendo al hilo principal (normalmente la UI) ejecutarse sin ser bloqueado o ralentizado.

✓ **WebComponents**

- Es un estándar creado por la W3C, que permite la creación de componentes personalizados reutilizables (un ejemplo sería crear una nueva etiqueta HTML que se llame `<opositor>` y que el browser sea capaz de interpretarla y renderizarla).
- Actualmente todos los browsers modernos soportan este estándar de forma nativa.
- Uso de Shadow DOM que encapsula el DOM y el CSS de un componente

✓ **APIs Multimedia**

- AudioAPI
- VideoAPI
- WebGL
- CanvasAPI
- WebUSB
- Web MIDI

#### 4.4 TECNOLOGÍAS OBSOLETAS

La web ha sufrido una transformación radical desde sus inicios en los años 90, por ello algunas tecnologías con las que inicialmente se desarrollaban las webs dinámicas ya se han quedado obsoletas. Algunas como JavaScript han ido evolucionando con los años y otras por cuestiones de seguridad, complejidad o popularidad se quedan sin soporte progresivamente de los navegadores. A pesar de que estas tecnologías no tienen soporte (o limitado) y algunas de ellas se consideran inseguras, siguen teniendo un nicho de mercado en software legacy de empresas y sector público.

Algunas de las tecnologías obsoletas más conocidas son las siguientes:

- ✓ JScript
  - Implementación de Microsoft de ECMAScript y la versión más reciente es JScript .NET. Se basó en su versión 4, si bien ahora es incompatible tanto con ella como con JavaScript y solo sirve para IE.
- ✓ VBScript
  - Ampliar información con el archivo “XVII\_VBScript” de la carpeta “Doc adicional”
- ✓ **Java Applets**
  - Miniaplicación en Java que se descarga del servidor y se ejecuta en la máquina virtual java (JVM) del navegador
  - En la página web se incluye la llamada al applet(.class) y los parámetros que se le pasan
  - JDK8 es la última versión que soporta esta tecnología
- ✓ **Controles ActiveX**
  - Son pequeños programas informáticos que aumentan las posibilidades de los navegadores como, por ejemplo: escuchar audio, ver vídeo, visualizar ficheros PDF, etc.
  - Soporte en varios lenguajes, incluyendo Java, C++ y Microsoft Visual Basic
  - Al contrario que los Applets, ActiveX y Java refuerzan el control de la seguridad para los datos del ordenador del cliente haciendo necesario que los usuarios tengan que aceptar permitir el acceso para garantizar la seguridad y la privacidad de los datos
  - Soporte principal en navegadores Internet Explorer
- ✓ **Adobe Flash Player / Adobe Animate**
  - Permite crear animaciones audiovisuales con un alto grado de compresión y nitidez
  - Utiliza ActionScript como lenguaje de scripts
  - Cuando no existía HTML5, era la alternativa de crear interfaces de usuario con contenido dinámico e interactividad. Creación basada en componentes
  - Es independiente del navegador y el módulo de extensión (Flash Player) es universal

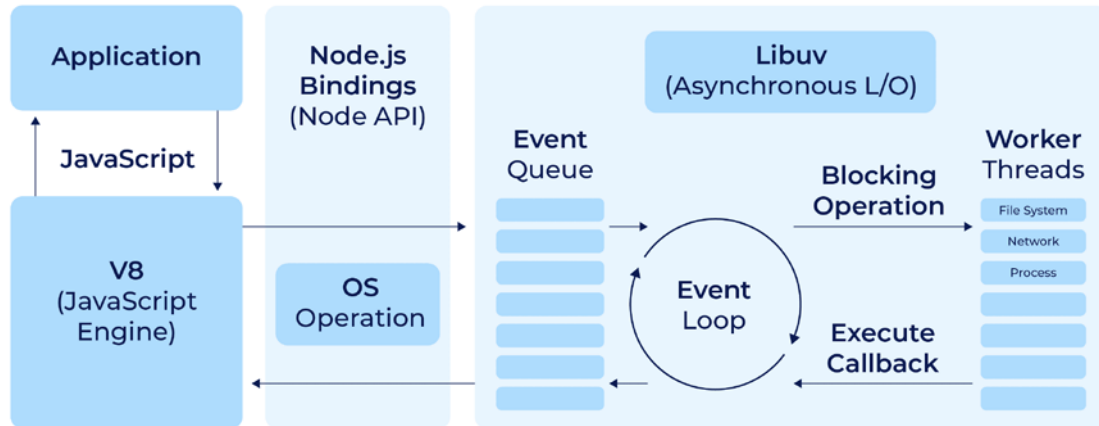
## 5. NODE.JS

**Node.js es un entorno de ejecución de JavaScript en el lado del servidor creado en 2009, que permite a los desarrolladores crear aplicaciones web escalables y de alta velocidad utilizando JavaScript tanto en el cliente como en el servidor.** También es posible desarrollar para Node.js utilizando otros lenguajes como TypeScript con una transpilación previa.

Node.js se basa en el motor JavaScript V8 de Google Chrome, que compila y ejecuta el código JavaScript de manera muy eficiente. Además, Node.js utiliza un modelo de E/S sin bloqueo y orientado a eventos, lo que significa que las operaciones de entrada y salida no bloquean el hilo de ejecución principal, permitiendo que Node.js procese múltiples solicitudes de manera eficiente.



## Node.js Architecture



<https://litslink.com/blog/node-js-architecture-from-a-to-z>

El ecosistema de Node.js tiene los siguientes elementos

### ✓ **Node Runtime**

- **El runtime de Node.js es un entorno de ejecución de JavaScript que permite a los desarrolladores crear aplicaciones de servidor en JavaScript utilizando el motor de JavaScript V8 de Google.** Node.js se basa en un modelo de operaciones no bloqueantes y utiliza un sistema de E/S asíncrono para admitir aplicaciones escalables y de alta concurrencia. Node.js también cuenta con un amplio ecosistema de módulos y paquetes de terceros que los desarrolladores pueden utilizar para crear aplicaciones web más rápidamente.
- **La última versión LTS de Node.js es la 20.x**
  - Soporte LTS en versiones pares desde la 16.x
- Recientemente se han desarrollado variantes de Node.js que buscan ser compatibles con parte del ecosistema actual mejorando algunos aspectos como la seguridad o el rendimiento
  - Deno
  - Bun

### ✓ **Sistema de eventos**

- El bucle de eventos es una estructura que permite que Node.js pueda manejar varias operaciones asíncronas al mismo tiempo
- Node.js registra todas las operaciones asíncronas que se deben realizar. En cada iteración del bucle de eventos, Node.js revisa la cola de eventos y toma la siguiente operación asíncrona en la cola. Node.js comienza a realizar la operación asíncrona, pero no espera a que se complete.
- En Node.js, el núcleo de este sistema de eventos es la clase `EventEmitter`, que se puede utilizar para definir y emitir eventos personalizados.



✓ **Npm**

- **Npm es el acrónimo de Node Package Manager, es una herramienta de línea de comandos que se utiliza para instalar, actualizar y administrar paquetes de software en Node.js.**
- Los paquetes de npm contienen módulos de Node.js que pueden incluir desde pequeñas funciones hasta aplicaciones completas. Estos paquetes se pueden descargar de un repositorio central llamado el Registro de npm
- Se instala junto con Node.js por defecto, versión actual 9.x
- Comandos más utilizados de npm
  - `npm init`
  - `npm install`
  - `npm start`
  - `npm run <<script>>`
- Existen otros gestores de paquetes alternativos
  - Yarn
  - Pnpm

✓ **Librerías de Node.js**

- `fs`
- `http`
- `events`
- `process`

✓ **Módulos**

- los módulos son bloques de código reutilizable que se utilizan para separar y organizar el código de una aplicación en archivos separados. Los módulos permiten una estructura modular para aplicaciones y bibliotecas, lo que facilita la reutilización del código.
- En Node.js, hay dos tipos de sistemas de módulos: CommonJS (CJS) y ECMAScript Modules (ESM).
- CommonJS (CJS) es el sistema de módulos predeterminado en Node.js. En CJS, los módulos se cargan de forma sincrónica en el momento en que se solicitan. La sintaxis para importar módulos en CJS es el uso de `require()`. Estos módulos son de ámbito global y se cargan en tiempo de ejecución.
- ECMAScript Modules (ESM) es un sistema de módulos que se basa en la sintaxis de los módulos de JavaScript introducidos en ECMAScript 6 (ES6). Los módulos ESM se cargan de forma asíncrona y dinámica en el momento en que se necesitan. La sintaxis para importar módulos en ESM es el uso de `import`. Los módulos ESM tienen su propio ámbito y se cargan en tiempo de compilación.

El ecosistema de Node.js dispone de una gran cantidad de librerías y utilidades que se utilizan durante el ciclo de vida completo de las aplicaciones. Algunas de las herramientas y librerías más utilizadas en Node.js son las siguientes

✓ **Build Tools**

- Task Runners de JavaScript:
  - Gulp, Grunt, Broccoli, Uglify...
  - Ejecutan tareas para generar código front-end en HTML, CSS o JavaScript realizando procesos de minimizando, compresión, compilación, agrupación de ficheros...

- Bundlers
  - Webpack, parcel, pkg, rollup...
  - se utiliza para compilar y empaquetar el código fuente de una aplicación en una sola salida optimizada para producción.
- Transpiladores
  - Babel
- ✓ **Linters**
  - Una herramienta que se utiliza para analizar el código fuente de un programa y detectar errores, inconsistencias o prácticas no recomendadas en el código. Su objetivo principal es ayudar a los desarrolladores a mejorar la calidad y la consistencia del estilo del código y reducir los errores en el proceso de desarrollo.
  - En Node.js los más populares son
    - ESLint
    - Prettier
- ✓ **Testing**
  - Pruebas Unitarias
    - Mocha y Chai
    - Jest
  - Pruebas e2e
    - Nightwatch
    - Cypress
    - Playwright
- ✓ **IDE**
  - VSCode
  - VSCodeium (Software Abierto basado en VSCode)
  - JetBrains WebStorm
  - Eclipse Che

## 6. FRAMEWORKS FRONTEND

Un framework (marco de trabajo o estructura) es una herramienta de software que proporciona una estructura y un conjunto de herramientas para desarrollar aplicaciones de software de manera más rápida y fácil. En lugar de comenzar desde cero, los desarrolladores pueden utilizar el framework para aprovechar la funcionalidad ya implementada y concentrarse en las características específicas de su aplicación.

Un framework suele incluir una serie de bibliotecas, herramientas y directrices para desarrollar aplicaciones de manera consistente y eficiente. También puede proporcionar una arquitectura de aplicación predefinida, que puede ayudar a los desarrolladores a estructurar su código y a garantizar que cumpla con las mejores prácticas de desarrollo.

Un framework frontend se enfoca en la creación de interfaces de usuario interactivas y responsivas en el lado del cliente. Estos frameworks incluyen características como la gestión del estado, el enrutamiento, la manipulación del DOM y la integración con APIs y servicios.

En este ámbito los frameworks más populares son aquellos basados en JavaScript y Node.js. A principios del 2010 se empezaron a popularizar este tipo de frameworks con el auge de Node.js, algunos de los frameworks iniciales de Node.js eran:

✓ **AngularJS**

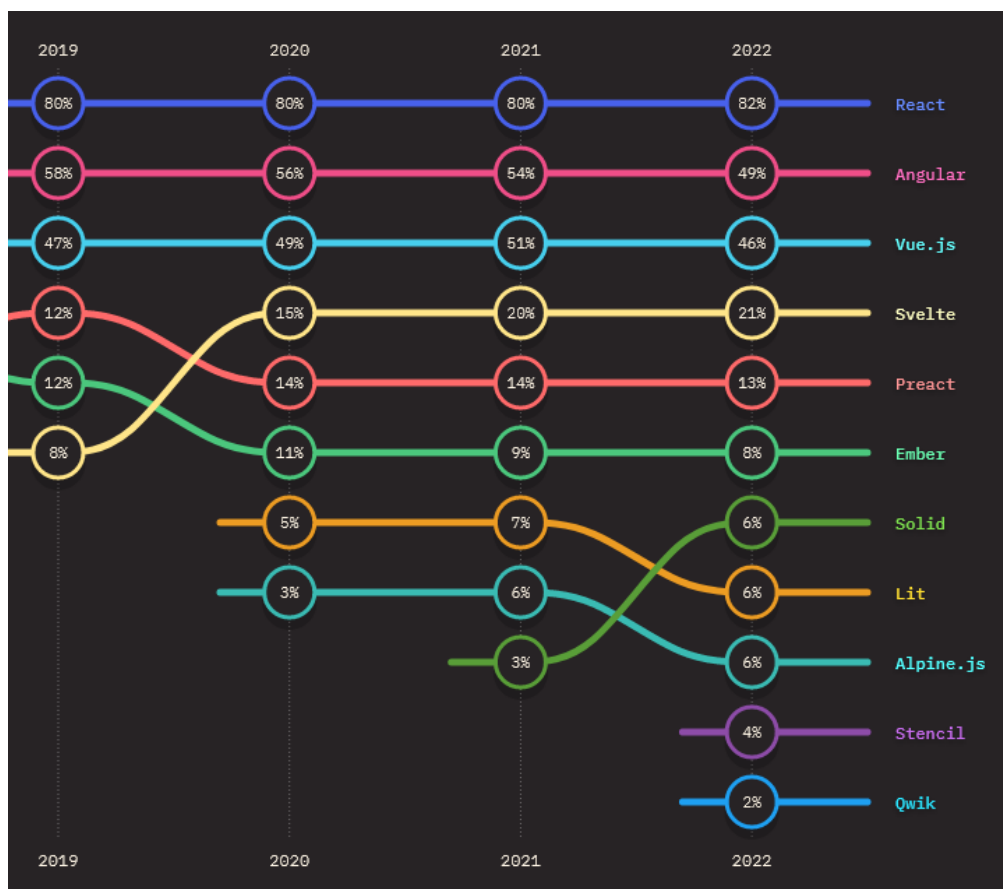
- También conocido como Angular 1
- Esta versión es muy distinta a la versión actual de Angular y no tiene soporte

✓ **Handlebars.js**

✓ **Ember.js**

✓ **Meteor.js**

**El entorno de JavaScript es muy volátil y los frameworks sufren una transformación muy rápida. Un framework que es popular actualmente puede dejar de serlo en unos años** como ha ocurrido con los frameworks anteriores que tienen ya muy poca cuota de mercado. A continuación, se muestra la popularidad de los frameworks frontend JS en el año 2022



(% de uso de frameworks frontend) <https://2022.stateofjs.com/en-US/libraries/front-end-frameworks/>

Actualmente los frameworks frontend JavaScript más populares son los siguientes:

✓ **React**

- Impulsado por Facebook/Meta y publicado en 2013
- Framework más popular actualmente con gran soporte de librerías y comunidad
- Define los componentes mediante ficheros JSX que combinan código JS y HTML
- Next.js permite realizar SSR
- React Native permite desarrollar aplicaciones híbridas usando React

✓ **Angular**

- También conocido como Angular 2 o Angular 2+ (Actualmente Angular 15)
- Evolución de AngularJS
- Impulsado por Google y publicado en 2016
- Framework más completo, complejo y maduro. Uso intensivo de TypeScript
- Angular Universal permite realizar SSR
- Ionic permite desarrollar aplicaciones híbridas usando Angular

✓ **Vue**

- Desarrollado por Evan You en 2014, proyecto de comunidad con financiación privada
- Ha sufrido un cambio completo de la versión 2 a la 3 que es la actual
- Menor curva de aprendizaje y sintaxis más sencilla
- Diseño basado en componentes VUE que combinan HTML, JS y CSS en un único fichero
- Nuxt.js permite realizar SSR (inspirado en Next.js y añadiendo más librerías)

✓ **Svelte**

✓ **Solid**

Además de los frameworks frontend JavaScript, existe una librería que ha sido crucial en el desarrollo de webs interactivas y que ha día de hoy se sigue utilizando a gran escala: **jQuery**.

jQuery es una biblioteca de JavaScript diseñada para simplificar la escritura de scripts de JavaScript y la manipulación de documentos HTML. Fue desarrollado por John Resig en 2006 y se convirtió rápidamente en una de las bibliotecas de JavaScript más populares y utilizadas en la web.

**La idea detrás de jQuery era simplificar la escritura de código JavaScript para la manipulación del DOM, lo que anteriormente requería mucho código.** En lugar de tener que escribir código JavaScript complejo para manipular el DOM, los desarrolladores podían simplemente utilizar las funciones y métodos proporcionados por jQuery para lograr el mismo resultado.

Con el tiempo, jQuery se expandió para incluir soporte para AJAX, animaciones, eventos, efectos visuales y mucho más. jQuery también se convirtió en una biblioteca de JavaScript multiplataforma, lo que significa que se puede utilizar en cualquier navegador web.

## 6.1 ARQUITECTURA DE FRAMEWORKS

La arquitectura de frameworks se refiere a la estructura, los patrones diseño y organización de un framework de software. Cada patrón de arquitectura de software tiene sus propias ventajas y desventajas, y elegir el patrón adecuado puede hacer una gran diferencia en la calidad y escalabilidad de la aplicación resultante. Los más comunes en aplicaciones web son:

- ✓ **MVC**: Model-View-Controller (MVC) es un patrón de arquitectura de software que separa una aplicación en tres componentes principales: el modelo, la vista y el controlador. El modelo representa los datos y la lógica de la aplicación, la vista es la interfaz de usuario y el controlador actúa como un intermediario entre el modelo y la vista
- ✓ **MVVM**: Model-View-ViewModel (MVVM) es un patrón de arquitectura de software que se utiliza para separar la lógica de la interfaz de usuario de la lógica de negocio. El modelo representa los datos y la lógica de la aplicación, la vista es la interfaz de usuario y el ViewModel actúa como un intermediario entre el modelo y la vista. Utilizado por frameworks como React o Vue

- ✓ **VIPER:** VIPER es un patrón de arquitectura de software utilizado para crear aplicaciones escalables y mantenibles. VIPER se divide en cinco componentes principales: vista, presentador, interactor, enrutador y entidad. Cada componente tiene un rol específico en la aplicación y se encarga de un conjunto específico de responsabilidades.
- ✓ **MVU:** Model-View-Update (MVU) es un patrón de arquitectura de software utilizado para construir aplicaciones de una sola página (SPA). El modelo representa los datos y la lógica de la aplicación, la vista es la interfaz de usuario y la actualización actúa como un intermediario entre el modelo y la vista.

## 6.2 CLASIFICACIÓN DE APLICACIONES Y RENDERIZADO

A la hora de clasificar las aplicaciones web existen 3 clasificaciones generales

- ✓ **Webs Estáticas:** No tienen interacción ni elementos dinámicos
- ✓ **MPA:** significa "Multi-Page Application" o Aplicación Multi-página. Es un tipo de aplicación web en la que cada página se carga por separado en el navegador del usuario, lo que resulta en una recarga completa de la página. MPA es una arquitectura tradicional de aplicaciones web y se utiliza en proyectos donde la navegación entre páginas es importante, como en sitios de noticias
- ✓ **SPA:** significa "Single-Page Application" o Aplicación de página única. Es un tipo de aplicación web que funciona en una única página y no requiere la recarga de la página completa para actualizar su contenido. SPA utiliza el poder del JavaScript para actualizar dinámicamente la página a medida que el usuario interactúa con ella.

El renderizado de la interfaz de usuario puede ser realizado en el servidor, en el cliente o en ambos. Muchos frameworks modernos ofrecen varios tipos de despliegue como opciones o utilizando librerías adicionales (como Next.js y Nuxt). Cada formato tiene ventajas y desventajas que deben tenerse en cuenta a la hora de publicar la aplicación.

- ✓ **SPA:** utiliza una sola página para renderizar todo el contenido, en lugar de tener múltiples páginas que se cargan por separado. Proporciona una experiencia más fluida y natural. Sin embargo, esto produce problemas con la primera carga y el SEO
- ✓ **SSR:** significa "Server-Side Rendering" o Renderizado en el servidor. Es un método en el que el servidor web genera el HTML de la aplicación y lo envía al navegador del usuario con cada petición. SSR se utiliza para mejorar la velocidad de carga de la página, mejorar la experiencia del usuario y mejorar la optimización de motores de búsqueda
- ✓ **SSG:** significa "Static Site Generation" o Generación de sitio estático. Es un método en el que el contenido de la aplicación se genera de antemano, antes de ser servido al usuario. SSG se utiliza para generar sitios web estáticos que son rápidos y fáciles de cargar. Este enfoque es muy útil para aplicaciones web que no cambian frecuentemente su contenido, como los blogs.

## 6.3 APLICACIONES HÍBRIDAS

Los frameworks frontend también se pueden utilizar para desarrollar aplicaciones multiplataforma a partir de una versión web como es el caso de **React Native** o **Flutter** para móviles o **Electron** para crear aplicaciones multiplataforma (como VSCode). Además, es posible instalar un servicio web como una aplicación utilizando las **PWA**, los WebWorkers y el **manifest.json**.

*Ampliar información en el Tema 123 de PreparaTIC - Aplicaciones móviles. Características, tecnologías, distribución y tendencias*

## 7. UX. EXPERIENCIA DE USUARIO

Para favorecer la usabilidad y la experiencia de usuario han aparecido frameworks para la generación de aplicaciones frontend con Responsive Web Design de tal manera que los controles y componentes HTML se visualicen, se redimensionen y se ubiquen de manera agradable a los usuarios. Si bien pueden ocupar un mayor espacio de visualización que otras soluciones.

- ✓ **Bootstrap 5.3. Es el líder del mercado**
- ✓ **TailwindCSS**
- ✓ Materialize - Material Design de Google

Otras tecnologías utilizadas para lograr un mejor diseño y que se adapte a distintos dispositivos son:

- ✓ **Precompiladores de CSS**
  - SASS, LESS, STYLUS
  - Los precompiladores permiten programar hojas de estilo CSS parametrizables con variables y con funciones que se compilan en la hoja de estilos definitiva
- ✓ **CSS3 y Media Queries**
  - Las media queries se utilizan para establecer condiciones que permiten a los estilos CSS ser aplicados solo si se cumplen ciertos criterios. Por ejemplo, se puede establecer una media query para cambiar el tamaño de fuente de un texto solo si la pantalla del dispositivo es más pequeña que una cierta resolución.

*Ampliar información en el Tema 44 de PreparaTIC - Accesibilidad y usabilidad. W3C. Diseño universal. Diseño web adaptativo y el Tema 93 de PreparaTIC - La elaboración de prototipos en el desarrollo de sistemas. Diseño de interfaces de aplicaciones*

## 8. FRAMEWORKS BACKEND

En general, los frameworks backend web ofrecen una serie de características comunes, como una capa de abstracción de base de datos para facilitar el acceso y la manipulación de datos, un sistema de enrutamiento para gestionar las solicitudes HTTP entrantes y una API de controladores para procesar las solicitudes y generar respuestas adecuadas. Además, suelen incluir herramientas de seguridad y autenticación para proteger la aplicación y sus datos de ataques externos.

La gran mayoría de lenguajes de programación de propósito general incluyen librerías o frameworks para desarrollar aplicaciones web o APIs. Estos frameworks pueden ser simples como Flask (Python) o Express (JavaScript) que incluyen una base simple y flexible o frameworks completos y complejos como ASP.NET Core (.NET) o Spring (Java).

<b><u>Frameworks de Java-JVM</u></b>	<b><u>Frameworks Ruby</u></b>	<b><u>Frameworks JavaScript-Node.js</u></b>
<ul style="list-style-type: none"> <li>● <b>Spring 6.0.2</b></li> <li>● Apache Struts 6.1.2</li> <li>● Play! 2.8.19</li> <li>● Javalin 5.4.2</li> </ul>	<ul style="list-style-type: none"> <li>● Ruby on Rails 7.0.4</li> <li>● Sinatra 3.0.5</li> </ul>	<ul style="list-style-type: none"> <li>● <b>Express 4.18.2</b></li> <li>● Meteor 2.12.0</li> <li>● Nest 9.14.0</li> <li>● Hapi 20.3.0</li> </ul>

Frameworks <u>PHP</u>	Frameworks <u>de Python</u>	Frameworks <u>de .NET</u>
<ul style="list-style-type: none"> <li>• <b>Symfony 5.4 (LTS), 6.2</b></li> <li>• <b>Laravel 10</b></li> <li>• Laminas Project 3.6.1 (antiguamente Zend)</li> <li>• CakePHP 4.4.11</li> </ul>	<ul style="list-style-type: none"> <li>• <b>Django 4.2</b></li> <li>• Flask 2.2.x</li> <li>• FastAPI 0.95.0</li> <li>• Web2py 2.22.5</li> </ul>	<ul style="list-style-type: none"> <li>• <b>ASP.NET MVC 5.2.9</b></li> <li>• <b>ASP.NET Core 6 (LTS), 7</b></li> <li>• Suave.IO (F#) 2.6.2</li> <li>• Giraffe (F#) 6.0.0</li> </ul>

## 8.1. MOTORES DE PLANTILLAS

Herramientas que sirven para generar código HTML de forma independiente a la lógica de la aplicación, permitiendo controlar la presentación sin tener que modificar a mano el código HTML). Muchos se basan en una versión modificada de HTML o un DSL propio que simplifique el desarrollo de páginas. En el caso de los framework de desarrollo frontend como React o Vue, utilizan un sistema de desarrollo combinado como es el caso de los ficheros JSX o VUE.

Algunos de los motores de plantillas para frameworks backend más conocidos son:

<u>Templates de Java-JVM</u>	<u>Templates Ruby</u>	<u>Templates JavaScript-Node.js</u>
<ul style="list-style-type: none"> <li>• <b>JSP: Java/Jakarta Server Pages 3.1.1</b></li> <li>• <b>JSF: Java/Jakarta Server Faces 4.0</b></li> <li>• <b>Thymeleaf 3.1.1</b></li> </ul>	<ul style="list-style-type: none"> <li>• <b>Haml 6.1.1</b></li> <li>• <b>Slim 5.1.0</b></li> <li>• <b>Liquid 5.4.0</b></li> </ul>	<ul style="list-style-type: none"> <li>• Plantillas propias de frameworks frontend</li> <li>• <b>EJS: EmbeddedJS 3.1.9</b></li> <li>• <b>Pug (Jade) 3.0.2</b></li> <li>• <b>Marko 5.25.4</b></li> </ul>
<u>Templates PHP</u>	<u>Templates de Python</u>	<u>Templates de .NET</u>
<ul style="list-style-type: none"> <li>• <b>Smarty 4.3.0</b></li> <li>• <b>Twig 3.5.1</b></li> </ul>	<ul style="list-style-type: none"> <li>• <b>Django 4.2</b></li> <li>• <b>Jinja2 3.1.2</b></li> <li>• <b>Mako 1.2.4</b></li> </ul>	<ul style="list-style-type: none"> <li>• <b>Razor 3.2.9</b></li> <li>• <b>Razor Core 2.2.0</b></li> <li>• <b>Blazor 6 (LTS), 7</b></li> </ul>

## 8.2. JAMSTACK

JAMStack es una arquitectura web moderna que vuelve a los orígenes de la web estática. A diferencia de un servidor dinámico que genera cada petición, las páginas se compilan y se sirven de forma estática. Estas webs utilizan elementos modernos como HTML5 y JavaScript. **Las siglas JAM en JAMStack significan JavaScript, APIs y Markup. En lugar de usar servidores web tradicionales, JAMStack utiliza una combinación de precompilación y caching para generar contenido estático que se sirve a través de una CDN (Content Delivery Network) o un servidor web estático.**

Esto significa que los sitios web JAMStack pueden ser más rápidos y escalables que los sitios web tradicionales, ya que el contenido estático se puede servir desde múltiples servidores en todo el mundo.

Además, los sitios JAMStack son más seguros, ya que no hay lógica del lado del servidor que pueda ser explotada por atacantes.

JAMStack es una arquitectura muy popular en sitios de blogging y CMS (Sistemas de Gestión de Contenido). Algunas de las plataformas más populares basadas en JAMStack son:

- ✓ Gatsby
- ✓ Jekyll
- ✓ Vuepress
- ✓ Netlify

Estos sitios además tienen en común el lenguaje Markdown. Markdown es un lenguaje de marcado ligero que se utiliza para dar formato a documentos de texto en la web. Fue creado en 2004 por John Gruber como una alternativa más fácil de usar a HTML. Markdown utiliza ficheros de texto plano y una sintaxis simple y fácil de aprender para aplicar formato a texto, como encabezados, listas, enlaces e imágenes. Muchas plataformas de blogging y CMS (Sistemas de Gestión de Contenido) lo admiten de forma nativa o con la ayuda de complementos.

## 9. CONEXIÓN A BD

Una aplicación puede realizar una conexión a una BD (SQL o NoSQL) de forma directa, mediante drivers y librerías intermedias o ORM (mapeadores objeto-relacionales). Un ORM es una técnica de programación que permite trabajar con una base de datos utilizando objetos en lugar de consultas SQL.

Un ORM es una capa de abstracción que se sitúa entre la aplicación y la base de datos. En lugar de tener que escribir consultas SQL y manejar los resultados en formato de tablas, el ORM proporciona una interfaz orientada a objetos que permite trabajar con los datos como si fueran objetos.

*Ampliar información en el Tema 64 de PreparaTIC - El modelo relacional. El lenguaje SQL. Normas y estándares para la interoperabilidad entre gestores de bases de datos relacionales*

<u>ORM de Java-JVM</u>	<u>ORM de Ruby</u>	<u>ORM de JavaScript-Node.js</u>
<ul style="list-style-type: none"> <li>• Especificación Java Persistence API (JPA)</li> <li>• Hibernate 6.1.4</li> <li>• iBATIS/MyBatis 2.3.3/3.5.13</li> </ul>	<ul style="list-style-type: none"> <li>• ActiveRecord 4.2.6</li> <li>• Sequel 4.36.0</li> </ul>	<ul style="list-style-type: none"> <li>• Sequelize 6.30.0</li> <li>• Bookshelf.js 1.2.0</li> <li>• Waterline 0.15.2</li> </ul>
<u>ORM de PHP</u>	<u>ORM de Python</u>	<u>ORM de .NET</u>
<ul style="list-style-type: none"> <li>• Doctrine 2.14.1</li> <li>• Laravel Eloquent 10</li> </ul>	<ul style="list-style-type: none"> <li>• SQLAlchemy 2.0.9</li> <li>• Django ORM 4.2</li> <li>• Peewee ORM 3.16.0</li> </ul>	<ul style="list-style-type: none"> <li>• ADO.NET y LINQ</li> <li>• EntityFramework 6.4.4</li> <li>• EntityFramework Core 6 (LTS), 7</li> <li>• NHibernate 5.4.2</li> <li>• iBATIS/MyBatis 1.6.1/3.0.0</li> <li>• Dapper 2.0.x</li> </ul>



## 10. INTERCONEXIÓN CON SISTEMAS Y SERVICIOS

Las soluciones más habituales para interconectar servicios y microservicios son:

- ✓ **Servicios WEB**
  - Es habitual que la información se serialice con XML
  - El estándar **SOAP (Simple Object Access Protocol)** es el más utilizado
  - Obliga a generar un esquema WSDL, es menos flexible y eficiente que REST
- ✓ **RPC (Remote Call Procedure)**
  - Permite que una aplicación en una máquina pueda invocar una función o procedimiento en otra máquina de manera transparente como si fuera una llamada local.
  - Algunos protocolos basados en RPC son DCOM, CORBA o XML-RPC
- ✓ **API REST**
  - Es habitual que la información se serialice mediante JSON o Binary JSON
  - Se pueden usar herramientas como Swagger para generar la API y la documentación
  - Existe el standard OpenAPI (esquema) pero es menos formal que los WebServices. Disponible en formato YAML o JSON
- ✓ **API RESTful**
  - Las APIs RESTful utilizan los mismos elementos que las APIs REST y se basan también en HTTP. La diferencia radica a la hora de desarrollarlas y definir las rutas de acceso. Las APIs RESTful utilizan los verbos de HTTP (GET, POST, DELETE, PATCH, PUT) para definir la acción a realizar y el recurso afectado. De esta forma se consigue una API más uniforme y con convenciones universales de uso.
- ✓ **GRAPH-QL**
  - Es una evolución a las API RESTs. Menos conocido que REST, pero ampliamente implantado en grandes compañías como todo el API de Facebook. Se diferencia de REST, porque GraphQL sólo tiene un *endpoint* de entrada. Permitiendo en una sola *request* realizar múltiples peticiones funcionales simultáneamente, e incluso especificar qué campos se quieren recuperar. De modo que GraphQL permite que el rendimiento del cliente sea mejor que en REST (que tendría que hacer múltiples request). Por otro lado, REST permite mejor nivel de caching para mejorar escalabilidad.
- ✓ **gRPC**
  - gRPC es un sistema de comunicación entre procesos de código abierto desarrollado por Google inspirado en RPC.
  - gRPC utiliza el protocolo HTTP/2 y HTTP/3 para la transmisión de datos, lo que le permite tener un alto rendimiento y eficiencia en la comunicación. Además, admite la serialización de datos en varios formatos, como Protocol Buffers, JSON y XML
  - Soporte limitado (aunque en aumento) en servidores y frameworks
  - Actualmente utilizado en aplicaciones de altamente distribuidas y de gran rendimiento
- ✓ URL que devuelva texto o contenido binario
  - En algunos casos es más conveniente generar directamente el contenido a servir
- ✓ Comunicación de información binaria mediante protocolos como FTP, SMTP, TCP, HTTPS
- ✓ Uso de colas de mensajería: (RabbitMQ, JMI)

*Ampliar información en el Tema 58 de PreparaTIC - El procesamiento cooperativo y la arquitectura cliente-servidor. Arquitectura SOA.*