DESARROLLO WEB
ENTORNO CLIENTE

2º DAW
CFGM

1. INTRODUCCIÓN AL DESARROLLO WEB

Natalia Escrivá Núñez

IES Serra Perenxisa

n.escrivanunez@edu.gva.es

Curso 2025-2026

CONTENIDO

1. INTRODUCCIÓN

Encontramos desarrollo web por todos los sitios: comercio electrónico, diseño web, aplicaciones móviles e Internet.

El desarrollo web ha evolucionado de manera muy significativa y ya no se limita únicamente a crear un sitio web atractivo. Hoy en día las aplicaciones web tienen funcionalidades que hace años eran inimaginables:

- Membresías
- Venta en línea
- Ocio
- Realización de cursos o impartición de clases regladas a distancia o en línea.

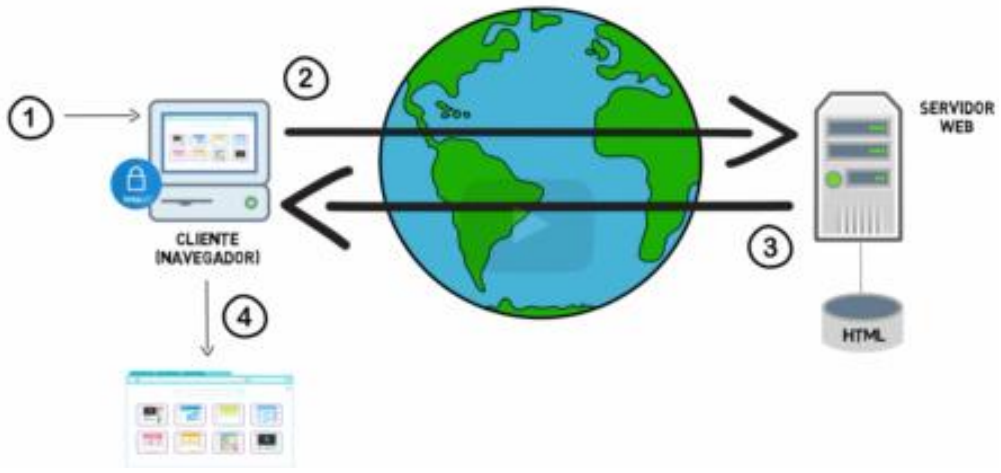
El desarrollo web está estrechamente relacionado con las **aplicaciones móviles** y con el **Internet de las cosas**:

Cuando accedemos a contenido de Internet a través de nuestra aplicación móvil o cuando nuestro reloj o nevera se está conectando es porque detrás hay servicios y aplicaciones web.

2. CÓMO FUNCIONA UNA PÁGINA WEB

Para crear una página web necesitamos escribir código en distintos lenguajes o utilizamos distintas tecnologías y bases de datos. Todos los **scripts** que hacemos los tendremos que almacenar en carpetas en un **servidor**.

Este servidor necesita estar **conectado a Internet**.



1. El usuario, a través de su cliente Web (navegador), escribe un **nombre en la barra de búsqueda** (url) y podrá ver el contenido de la página web. Esto es gracias al protocolo **DNS** (Domain Name System), ya que es el servicio que "traduce" un nombre de página web a su IP correspondiente.
2. En ese momento se genera una petición HTTP desde el lado cliente al servidor.
3. El servidor le contesta con un documento web.
4. En ese momento, ese documento web será visualizado por el cliente, es decir, por el navegador web.

Cualquier interacción provoca el envío de datos y la ejecución de código por parte del servidor generando a su vez nuevos documentos que podrán servir para seguir interactuando o, simplemente otra petición HTTP que nos devolverá otro documento web.

2.1. Lógica de programación de una página web

Si queremos crear una aplicación web, deberemos tener en cuenta que vamos a tener **lógica de programación tanto en el lado del cliente como en el lado servidor**.

Las aplicaciones web se basan en el patrón de **arquitectura de software MVC** (Modelo-Vista-Controlador).

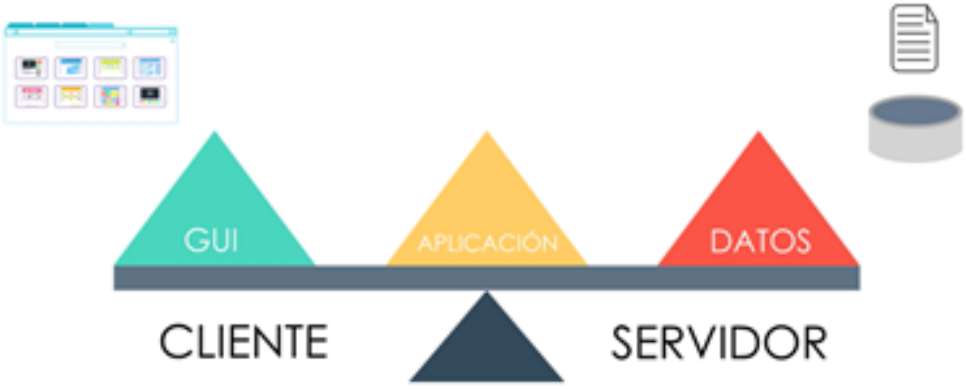
Esto implica que debemos separar el código atendiendo a sus responsabilidades.

→ Modelo: También conocida como capa de datos. Aquí hablamos de la información de la aplicación. Esta capa, que será dotada de una seguridad especial, se encuentra en un sistema gestor de base de datos (SGBD). Será este **software el que administre la información almacenada en la aplicación web o en la empresa**.

→ Vista: También conocida como capa de presentación, interfaz de usuario o Graphic User Interface. Contiene el **código de la aplicación que va a producir la visualización del usuario**. En la vista trabajamos con datos, pero no se realiza un acceso directo a éstos. Ese código lo envía el servidor web que lo entrega al navegador, a través del protocolo http.

→ Controlador: También llamada capa del servidor y capa de aplicación. Contiene el código necesario para responder a las acciones que se solicitan en la aplicación, como visualizar un elemento, realizar una compra, búsqueda de información… Esta capa hace de **enlace entre las vistas y los modelos**. Se programa en lenguajes como PHP, ASP, .NET, C#, Python, Java, JavaScript y otros. El código de esta capa se traduce por un servidor de aplicaciones que, suele ser un servidor web con módulos de traducción de esos lenguajes. Por ejemplo, el código PHP utilizado será traducido a HTML, CSS y JavaScript por el servidor de aplicaciones. Ese "paquete" será el que se entregue al navegador.

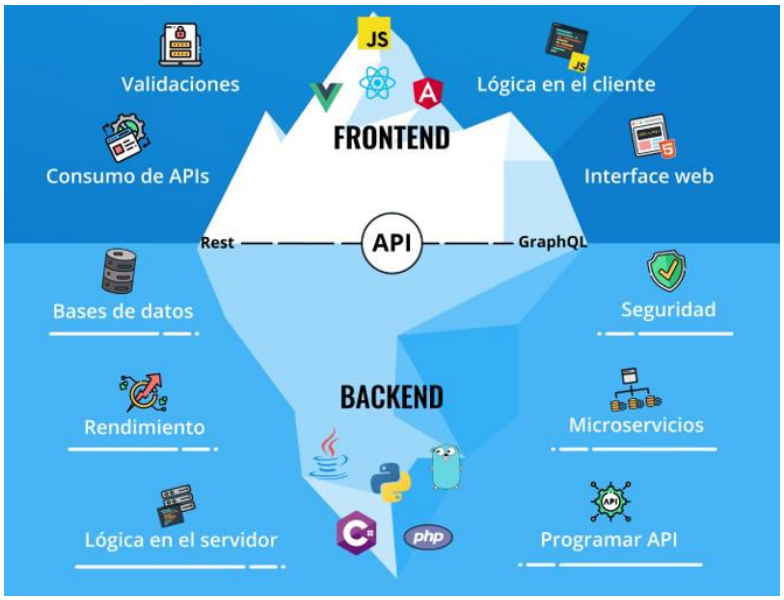
El controlador se comunica con el modelo para obtener datos que procesará y entregará en un formato interpretable, al navegador.



La división en capas del funcionamiento de las aplicaciones ha propiciado dos desarrollos diferentes: el **desarrollo FrontEnd** o del lado del cliente y el **desarrollo BackEnd** o del lado del servidor (de aplicaciones y base de datos).

El **FrontEnd** se ocupará de la capa de presentación, es decir, de la **experiencia del usuario y del funcionamiento final de la aplicación**. El objetivo es optimizar la presentación y conseguir una interfaz final agradable, funcional y eficiente.

El **BackEnd** se encarga de la **parte del desarrollo que queda oculto al usuario**. Los objetivos son conseguir un acceso rápido a los datos, una comunicación eficiente con la base de datos y el navegador, control de la seguridad, entre otros.



2.2. Lenguajes del lado del cliente (FrontEnd)

Son aquellos que **trabajan bajo un navegador web**. Serán los encargados de gestionar cómo se ve y cómo se interactúa con el usuario: botones, enlaces, mostrar información, formularios, imágenes… Se encarga de que el sitio web sea **responsive**, es decir, la vista se adapte a todo tipo de dispositivos.

Se hará cargo del área UI Design, gestionando los colores, tipografía…, es decir, los **diseños**.

Se encargará de validar **formularios**, enviar peticiones estructuradas hacia las **APIs**.

Encontramos:

HTML *(HyperText Markup Language)*: No es considerado como lenguaje de programación, sino un lenguaje de marcado de hipertexto para dar significado al contenido. **Se encarga de la parte semántica de la aplicación, dando estructura conceptual al contenido de la página web (texto, imágenes, vídeos…).**

CSS *(Cascading Style Sheets).* Es un **lenguaje de estilos**. Proporciona reglas que determinan el formato y maquetación de los elementos del HTML. **Se encarga de la apariencia y presentación visual (colores, tamaños, espacios, animaciones…). Aquí entrarían los frameworks de CSS como Bootstrap, Tailwind…**

JavaScript: **Lenguaje de programación que permite dinamizar la página web**. Permite la interacción con el usuario, como validar **formularios**, añadir **animaciones**, reaccionar a través de distintos **eventos**, **preprocesando datos** antes de enviar al servidor, modificando estilos y contenidos de forma dinámica, solicitando y enviando ficheros al servidor… Aquí entrarían sus **frameworks**, **bibliotecas**, todo un ecosistema (React, Vue, Angular…).

2.3. Lenguajes del lado del servidor (BackEnd)

Estos lenguajes corren en la parte del servidor.

Son los que se encargan de gestionar la **base de datos** (como se van a relacionar las diferentes tablas o entidades de información para evitar la redundancia de datos…).

Buscará también la forma de ganar **eficiencia** en el código, optimizarlo.

También será el encargado de revisar cuestiones de **seguridad**, evitar ataques e inyección SQL.

Por **ejemplo**, si tenemos una tienda virtual, en el proceso del pago, se encargará de que este proceso se haga de forma segura. Gestionará de conectarse con el método de pago, validarlo y almacenar la información.

Además, creará una **API** que el FrontEnd consumirá y mostrará.

Por ejemplo, Java, Python, Perl, .Net o C# pueden ser utilizados para desarrollar web.

Todos estos lenguajes funcionan acceden a una base de datos. Por ejemplo, MySQL funciona con todos los comentados, pero hay algunas bases de datos que solo funcionan con lenguajes específicos.

3. JAVASCRIPT

JavaScript se encuentra en la capa de Vista, junto con **HTML** y **CSS**. Se encarga de dar funcionalidad a los dos anteriores.

Sus principales **CARACTERÍSTICAS** son:

→ Es un lenguaje **del lado del cliente**. Esto quiere decir que se ejecuta en el **navegador** de la propia computadora del usuario.

JavaScript también se puede utilizar como lenguaje **del lado de servidor**, gracias al entorno **Node.js.**

→ Es un **lenguaje de programación interpretado**, por lo que no será necesario compilar los programas para ejecutarlos. En otras palabaras, los programas escritos con JavaScript se pueden **probar directamente en cualquier navegador** sin necesidad de procesos intermedios.

El código JavaScript se ejecuta de arriba abajo y el resultado de ejecutar el código se devuelve inmediatamente. Teóricamente no tendremos que transformar el código a código máquina para que sea ejecutado.

Debemos decir que esto **NO es completamente así**. Por las altas capacidades que ha ido adquiriendo este lenguaje, para mejorar su rendimiento, JavaScript utiliza un **compilador JIT (Just In Time)**, gracias a una máquina virtual (JVM) que proporcionan los navegadores.

Dicho esto, NO podemos olvidar que tiene mucho de interpretado por lo que **muchos de los errores NO se marcarán al comenzar a ejecutar el código**.

→ Es un lenguaje **orientado a objetos**. Hasta la gran evolución con ES6 de JavaScript, este no era un lenguaje orientado a objetos de la forma que estamos acostumbrados a Java o C++. No estaba basado en clases, sino en prototipos. Desde ES6 ya se incorporan aspectos similares a las clases.

→ Es un lenguaje **imperativo** con toques **declarativos** . **Imperativo** como la mayoría de los lenguajes de programación de alto nivel: le estamos **diciendo de forma explícita la instrucción que queremos que se ejecute**. Como ejemplo, imprimir por pantalla. A partir del estándar ES6, se introducen funcionalidades **declarativas**, es decir, expresa la lógica sin detallar el flujo de control (la sintaxis no es tan clara).

→ Es un **lenguaje estructurado**. **Estructurado** implica que **está basado en tres pilares o estructuras básicas**: la **secuencia** (una instrucción va detrás de otra), la **selección** (estructuras de control condicionales) y la **iteración** (estructuras de control repetitivas).

→ Es un **lenguaje objetual**, con **tipificación débil y dinámica**. Un lenguaje **objetual** significa que **los objetos con los que trabaja tendrá sus propios atributos y métodos**. Con **tipificación débil** se refiere que **no se tiene la obligación de definir de qué tipo van a ser las variables que utilizamos**. **Tipificación dinámica**

implica que **se puede modificar el tipo de variable durante la ejecución del programa.**

Podemos encontrar distintos tipos de aplicaciones web o, mejor dicho, distintas herramientas para su desarrollo:

- Con JavaScript "Vanilla" o puro.
- Con bibliotecas (aportan funcionalidades al lenguaje): JQuery, Prototype, D3.js, React.
- Con frameworks (modifican totalmente la forma de programar porque se encargan de todo: funcionalidades y patrones de diseño): Angular, Vue.js, Phaser.
- Con frameworks full Stack: Odoo, Blitz, Next.js, SvelteKit

3.1. Breve historia

El desarrollo de aplicaciones web ha contado con diversas tecnologías FrontEnd a lo largo de la historia. Podemos citar:

→ XML: Lenguaje de marcado que se sigue teniendo en cuenta ya que permite definir formatos documentales a voluntad. En la actualidad se sigue utilizando principalmente como formato de intercambio de datos entre aplicaciones y servicios.

→ JSON: Formato documental procedente del lenguaje JavaScript y que ha reemplazado a XML como formato de intercambio de datos.

→ SVG: Lenguaje basado en XML que permite mostrar imágenes vectoriales en los navegadores. Actualmente sigue implantado y reconocido en los navegadores actuales.

→ Flash: Tecnología creada para crear, de forma rápida y eficiente, aplicaciones web enriquecidas. Tuvo mucho éxito a finales de los 90s, pero ya no se utiliza, los navegadores ya no tienen soporte (plugin incorporado) para visualizar Flash.

Vemos que algunas se mantienen y otras no. Ha sido JavaScript el único lenguaje de programación que los navegadores son capaces de traducir sin necesidad de añadir plugins o extensiones.

Inicialmente **JavaScript** se ideó para dotar de **dinamismo a las páginas web**.

Las **primeras páginas web se hacían en HTM**L y, **con las primeras versiones de CSS,** el cambio que ofrecían las páginas web era muy ligero: eran **páginas estáticas con opciones de formato** como colores, tamaños de letra, tablas, etc.

A principios de los 90s, los usuarios se conectaban a Internet a una velocidad media de 28,8kbps.

Se iniciaba el desarrollo de las aplicaciones web y muy velozmente, estas aplicaciones se fueron haciendo más complejas por lo que surgió la **necesidad de un lenguaje que se ejecutara en el navegador del usuario**. De esa forma, si el usuario no rellenaba correctamente un formulario, no se le hacía esperar mucho tiempo hasta que el servidor volviera a mostrar el formulario indicando los errores existentes.

Brendan Eich, un programador que trabajaba para **Netscape**, pensó que podría solucionar esto adaptando otras tecnologías ya existentes (como **ScriptEase**) al navegador Netscape Navigator 2.0, que se iba a lanzar en 1995.

Inicialmente, **Eich** denominó a su lenguaje **LiveScript**.

Pasado un tiempo, Netscape firmó con **Sun Microsystems** un contrato para el desarrollo del nuevo lenguaje de programación. Justo antes del lanzamiento Netscape cambió el nombre por el de **JavaScript**.

La primera versión de JavaScript fue un éxito y Netscape Navigator 3.0 ya incorporaba la versión 1.1 del lenguaje.

Microsoft lanzó **JScript** con su navegador Internet Explorer 3. Era prácticamente una copia de JavaScript al que rebautizaron para evitar problemas legales.

El hecho de que los navegadores **Microsoft Internet Explorer y Netscape Navigator coparan el mercado** hizo que ambas versiones, con sus diferencias, compitieran entre sí generando un problema a los desarrolladores de aplicaciones web.

3.2. Normalización de JS

Durante mucho tiempo la tecnología de creación de páginas web se llamó **DHTML** (*Dynamic HTML*) porque con JavaScript las páginas ya eran dinámicas y empezaban a reaccionar a las acciones del usuario.

El desarrollo de la llamada **Guerra de navegadores entre Microsoft y Netscape** agravó el problema de las versiones de JavaScript y se empezó a tratar la cuestión de estandarizar el lenguaje.

Netscape envió el borrador a **ECMA** (*European Computer Manufacturers Association)* para su normalización.

La **World Wide Consortium**, organismo de estándares para las tecnologías relacionadas con HTML y, sobre todo **ECMA**, organismo de estándares para la comunicación e información decidieron **estandarizar** el lenguaje.

ECMA normalizó el lenguaje por primera vez en 1997, llamándole **ECMAScript**.

Actualmente, el nombre de JavaScript, como marca registrada, es propiedad de **Oracle Corporation** al adquirir la empresa Sun Microsystems.

Netscape se disolvió a principios de siglo y dejó el proyecto como proyecto de código abierto impulsado por la **Fundación Mozilla**. Por ello, esta fundación obtuvo el derecho de usar también el nombre de **JavaScript**.

Google desarrolló el navegador **Chrome** y consiguió crear un **motor**, llamado **V8**, capaz de ejecutar JavaScript a una gran velocidad.

El crecimiento continuo de JavaScript hizo que aparecieran **aplicaciones web que ofrecían una experiencia al usuario semejante a la de las aplicaciones de escritorio**. Se las llamó **RIA** (*Rich Internet Applications*) y en su potencia fue fundamental el aumento de las capacidades y la velocidad del lenguaje JavaScript.

3.3. JS como lenguaje multipropósito

En el año 2009, Ryan Dahl desarrolló un software llamado **Node.js** o **Node**, siendo éste un **entorno de ejecución de JavaScript**.

Puesto que el motor V8 de Google que servía para traducir JavaScript en Chrome era un proyecto de código abierto, Dahl decidió utilizar el código para crear un entorno independiente del navegador, capaz de interpretar código JavaScript.

Esto ha permitido, entre otras posibilidades, crear aplicaciones JavaScript en el lado del servidor.

Apuntamos pues otro éxito que permite ver **JavaScript como un lenguaje completo capaz de crear aplicaciones de todo tipo.**

El **impulso de Node.js** con ayuda de múltiples bibliotecas y frameworks de las que se puede dotar, ha permitido a JavaScript ser un lenguaje capaz de:

- → Crear **aplicaciones de escritorio** con ayuda de **frameworks** como **Electron** o **NW.js**.
- → **Programar dispositivos hardware** con ayuda de **frameworks** como **Johnny-Five** (para programar placas Arduino), **Cylon.js** o **Node-Red**.
- → **Generar aplicaciones móviles nativas**. Por ejemplo, usaríamos Java para Android y Swift u Objective C para iOS. Ahora ya podemos hacer una aplicación HTML5 usando **frameworks** de JavaScript como son **PhoneGap/Cordova**, **Ionic**, **Flutter** o **React Native**.
- → Ser el **lenguaje de manipulación de Bases de Datos**, por ejemplo, con los sistemas **MongoDB.**

3.4. Lenguajes preprocesados

El auge y evolución de JavaScript ha provocado que empresas y otras entidades creen nuevos lenguajes para permitir la mejora del original.

El primero en conseguir popularidad fue **CoffeScript**: Permitió dotar a JavaScript de una sintaxis inspirada en Python para poder escribir JavaScript de forma rápida y eficaz.

Actualmente está en retroceso, pero durante muchos años fue utilizado por miles de programadores.

Este lenguaje NO respeta los estándares porque, de hecho, es un lenguaje nuevo. Eso significa que ningún navegador, de forma nativa, es capaz de interpretar código CoffeScript. Por ello **el código debe ser convertido a JavaScript**.

Lenguaje preprocesado → Se escribe lenguaje en un código y se convierte con ayuda de un software (llamado preprocesador) a JavaScript estándar.

Bajo esta idea aparece en 2012 **TypeScript**, lenguaje creado por **Microsoft** que amplía JavaScript para que reconozca, entre otras muchas cosas, tipos de datos avanzados y las ventajas de los lenguajes fuertemente tipados.

Ha tenido mucho éxito y es todavía muy utilizado, más aún al ser el lenguaje base del exitoso framework **Angular** de **Google**.

La misma idea está detrás de **Dart**, desarrollado por **Google** para modernizar JavaScript, **Elm** que permite programar en un lenguaje puramente funcional o los casos de **ClojureScript** o **Scala.js** para convertir código de esos lenguajes en código JavaScript.

4.4. Versiones de JS

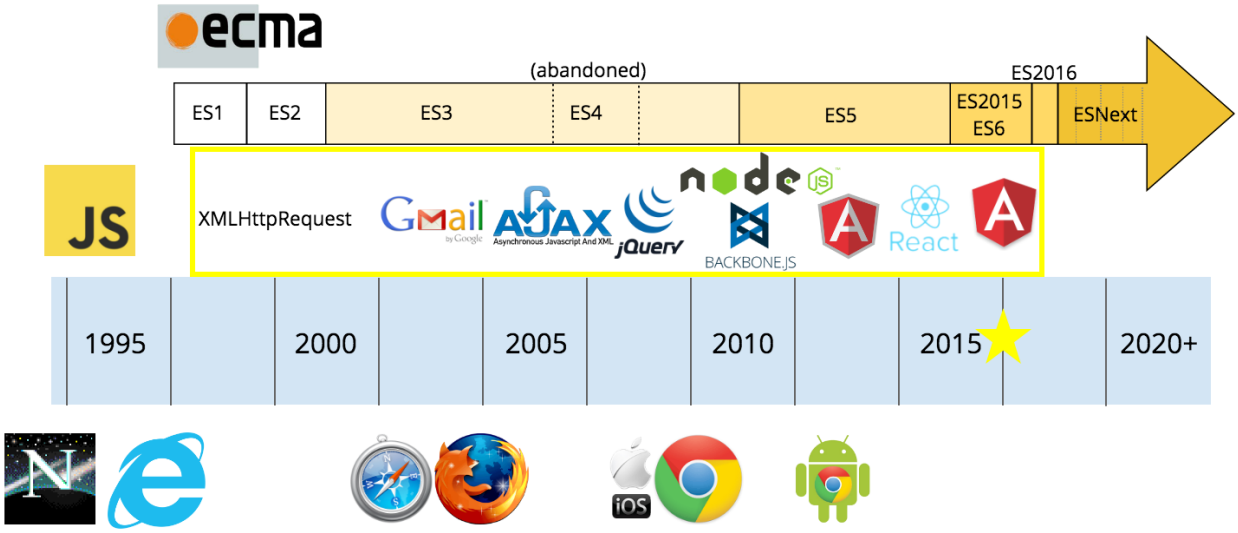
La norma ***ECMA-262*** es la que se relaciona con la estandarización del lenguaje.

Aunque **Microsoft y Netscape** se comprometieron con el estándar, lo cierto es que JavaScript y JScript continuaron añadiendo especificaciones al lenguaje.

Con la **versión 5 del lenguaje ECMAScript** (2009), con la aceptación de **HTML5** como estándar y, en especial con la **versión 5.1 de ECMAScript** (2011), el lenguaje se fue adoptando por todos los navegadores y finalmente Microsoft respetó el estándar en las últimas versiones de Internet Explorer y más aún con la aparición de Edge.

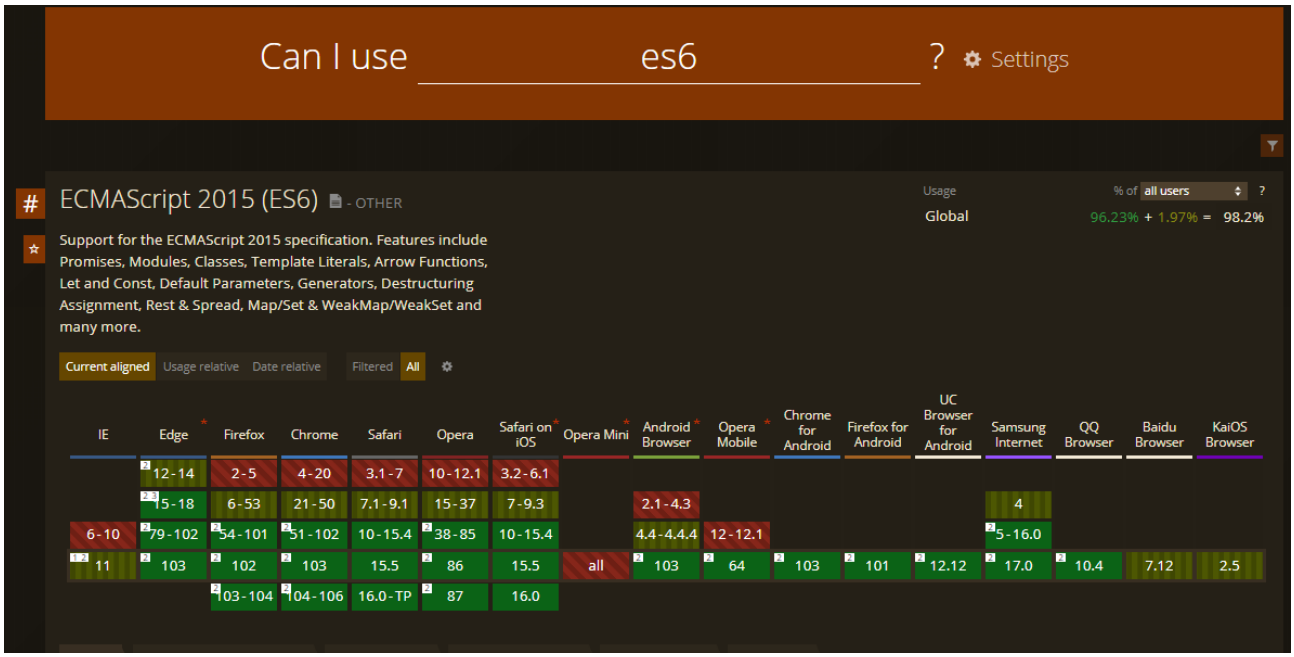
La influencia de los lenguajes preprocesados para JavaScript y las claras necesidades de mejora del lenguaje, dieron lugar a nuevos estándares:

- → ECMAScript 6, ES6, ó ECMAScript 2015: Estándar que agrega cambios muy significativos en la sintaxis de JavaScript y añade elementos avanzados de otros lenguajes.
- → ECMAScript 7, ES7 ó ECMAScript 2016: Estándar que incluye soporte para Array.prototype.includes y el operador exponencial.
- → ECMAScript 8, ES8 ó ECMAScript 2017: Incluye constructores async/await que funcionan con generadores y promesas.
- → ECMAScript 9, ES9 ó ECMAScript 2018: Incluye operadores rest/spread, iteración asíncrona, etc.
- → ECMAScript 10, ES10 ó ECMAScript 2019: Incluye Array.flat(), Array.flatMap(), errores en bloque try.. catch, etc.
- → ECMAScript 11, ES11 ó ECMAScript 2020: Incluye soporte para globalThis, Dynamic import(), BigInt, etc.
- → ECMAScript 12, ES12 ó ECMAScript 2011: Incluye soporte para cadenas con replaceAll, para promesas con any, amplía asignaciones lógicas, etc.
- → ECMAScript 13, ES13 ó ECMAScript 2022: Incluye soporte para Array.at(), Object.hasOwn()….
- → ES.Next: Así se llaman las nuevas versiones venideras en desarrollo cuyas características no son estables ni concretas puesto que se trata de propuestas que aún no han sido aceptadas por el organismo de estandarización.



Actualmente, todos los navegadores (que estén actualizados), trabajan perfectamente con todo el estándar ES2015 y con gran parte del estándar ES2016.

Para poder saber hasta qué punto una version de nuestro lenguaje JavaScript o CSS está implantado en los navegadores, podemos consultar la página web www.caniuse.com. De esta forma sabremos qué navegadores (y versiones) son compatibles con nuestro código.



5. FRAMEWORKS Y BASES DE DATOS

Podemos encontrar herramientas que nos **facilitan** la creación de sitios web: los **frameworks**.

Se trata de código ya escrito que **provee de funcionalidad** y es utilizado comúnmente en la mayoría de los proyectos.

Un framework puede ser expandido, es decir, podemos utilizar funciones que ya tiene el propio framework y modificar su código de acuerdo con nuestras necesidades.

Por ejemplo, todos los sitios web necesitan un sistema de autenticación de usuarios y la mayoría de los frameworks ya lo tienen implementado, por lo que podríamos hacer uso de esa funcionalidad sin necesidad de crearlo. Lo mismo ocurre con las consultas a las bases de datos…

Ventajas de usar un framework:

- Es un código probado y respaldado por una comunidad.
- Ahorro de tiempo en la parte de desarrollo.
- Normalmente es un código que sigue las buenas prácticas del desarrollo: bien comentado y escrito.
- Tiene una documentación de soporte y ayuda.

Inconvenientes de usar un framework:

- Aprender y seguir una sintaxis nueva.

Algunos son excesivamente densos en cuanto a funcionalidades por lo que, dependiendo del proyecto, deberemos elegir uno u otro.

Existen frameworks enfocados al frontend y otros al backend.

5.1. Frameworks FrontEnd

Frameworks orientados a la parte del cliente. Atendiendo a la tarea que queremos hacer, elegiremos uno u otro. Puede ser que varios framewoks sirvan para lo mismo, elegiremos según nuestro gusto.

Herramientas HTML: Sirven como punto de arranque para cualquier sitio web o aplicación web. Contienen código para crear elementos de la interfaz del sitio web de forma rápida con elementos de uso común como tablas, sliders, menús de navegación, etc… Como ejemplos HTML5 Boilerplate, Email-Boilerplate, HTML5 Bones.

Utilidades CSS: Existen frameworks enfocados a crear aplicaciones móviles, sitios web, otros para colores, tipografía, y otros que son todo en uno. Algunos ejemplos son BassCSS, TailwindCSS, Skeleton.

Utilidades + interfaz CSS: Otro grupo de frameworks. Con poco código HTML y CSS, contienen muchos elementos visuales para crear sitios web muy atractivos. Estos frameworks incluyen todo lo de los anteriores y además otros elementos de la interfaz como Sliders, Menús, animaciones, validaciones, formularios…. Como ejemplos Bootstrap, Semantic UI, Material Design Lite.

Frameworks Mobile: Contienen código y elementos de la interfaz especialmente diseñados para parecer elementos nativos de sistemas operativos para móvil como Android o IOS. Su finalidad es ser utilizados para aplicaciones móviles. Como ejemplo Ratchet (Crea elementos muy similares a los de IOS. Se hacen con HTML, CSS y JavaScript y simulan los elementos nativos.

Reset Frameworks: grupo de frameworks que reinician los elementos para que se vean igual en cada navegador: enlaces, botones…. Como ejemplo Eric Meyer Reset 2.0, Normalize o Sanitize.

Frameworks JavaScript: Existen algunos para escribir código que modifique e interactúa sobre el HTML y CSS como jQuery y otros para crear Interfaces Web y Aplicaciones Web como React, Vue.JS o Angular.

5.2. Frameworks Backend

Cada lenguaje de servidor tiene su propio framework o más de uno.

Cada uno tiene sus propias reglas, por lo que se facilita la documentación para ver cómo funciona cada uno.

Funcionalidades de los frameworks:

- Código para crear usuarios y login de administradores.
- Facilidad de creación de Templates HTML, CSS e integración de librerías JavaScript.
- Seguridad (validaciones).
- Facilidad para relacionar las tablas de la base de datos. Contienen herramientas que facilitan las consultas a la base de datos.
- Código ya escrito para realizar las consultas.
- Documentación

Frameworks para:

- Java: DropWizard, Grails, GWT, JHipster, JSF, Play Framework, Spring Boot, Spring MVC, Struts, Vaadin.
- PHP: Laravel, Code Igniter, Symfony, Zend, Yii 2, CakePHP, FuelPHP, Phalcon, Slim, Aura.

- Python: Django, Flask, TurboGears, Web2Py, Pyramid, Bottle, CherryPy, Sanic.
- Ruby: Ruby On Rails.
- C++: CppCMS, CivetWeb
- JavaScript: Express.

5.3. Bases de Datos

Es donde almacenaremos la información de nuestra aplicación web.

Guardaremos por ejemplo los usuarios y passwords (encriptadas, por ejemplo, con un hash), fechas de próximo pago. También datos de clientes, productos, pedidos, etc….

El lenguaje más utilizado en bases de datos es SQL para las bases de datos relacionales.

Ejemplos de bases de datos: MySQL, Oracle, MariaDB, PostgreSQL, MongoDB.

Con las bases de datos crearemos **aplicaciones CRUD**:

- C: create
- R: read
- U: update
- D: delete

6. HERRAMIENTAS PARA DESARROLLO WEB

Para poder realizar aplicaciones web vamos a necesitar unas herramientas.

Incluimos aquí los **editores de código**, ya sean editores de texto o IDEs, **ambientes de desarrollo** (incluyendo aquí servidor, lenguajes y bases de datos), **web hosting**, **navegadores**, **emuladores** (para poder probar nuestra aplicación en distintos dispositivos) y los **web depeloper tools** (que se incluyen en los navegadores y nos van a ayudar en el desarrollo).

6.1. Editores de texto e IDEs

Como **características principales** de un buen editor de código encontramos:

→ Coloreado de código: Dependiendo del lenguaje que estemos utilizando, utiliza colores determinados para los distintos elementos del lenguaje. Suele ser

personalizable y también podemos disponer de temas para elegir el color del entorno y del código.

→ Navegación avanzada: Algunos archivos son muy grandes y algunos editores de código permiten expandir y comprimir el código, ver mini mapas con porciones de código, o ir a una línea de código determinada a través del nombre de una función u objeto.

→ Búsqueda y reemplazo: En algunos editores se puede buscar en todos los archivos de un proyecto y tienen la posibilidad de reemplazo inteligente, por ejemplo, de una variable.

→ Autocorrección de escritura: Marcado en rojo de errores de código.

→ Uso de abreviaturas: Conocidos como Snippets, y son trozos de código que sabemos que vamos a usar a menudo y que las asociamos con una abreviatura. Por ejemplo, podemos escribir cl y que, directamente aparezca console.log (un comando muy usado en JavaScript).

→ Visualización del resultado: Algunos editores integran la posibilidad de lanzar el resultado de nuestro código en el navegador. Incluso, pueden crear un servidor web que permita poder examinar los cambios en la aplicación a la vez que los vamos escribiendo en el código.

→ Integración de herramientas: Se trata de herramientas como la depuración del código, acceso al terminal, etc.

→ Adición de extensiones y/o plugins: Para poder añadir funcionalidades.

En cuanto a los **tipos de editores de código** encontramos:

→ Editores de texto multipropósito: Editores de texto especializados en facilitar la escritura de código fuente para distintos lenguajes. Son capaces de adaptarse al tipo de código escrito. Por ejemplo, si escribimos código en JavaScript, el coloreado será diferente a si escribimos en PHP ya que es capaz de reconocer la sintaxis de los lenguajes. Son muy rápidos y ligeros. Sirven también para editar archivos de configuración de sistema o escribir en texto plano. Disponen, de base de menos herramientas que otros editores, pero tienen la posibilidad de aumentar sus funcionalidades instalando plugins y extensiones. Como ejemplos, **Sublime Text**, Notepadd++, TextMate y Geany (editor del sistema Ubuntu).

Sublime Text: Editor multiplataforma con alto nivel de personalización. Para habilitar el resaltado de sintaxis para el código ES6 y ReactJS, tiene el complemento Babel. Otros complementos importantes para el desarrollo de JavaScript son DocBlockr, JSFormat, entre otros.

→ Editores ligeros especializados en desarrollo web. La verdad es que la línea que separa estos de los anteriores es muy difusa, de hecho, a veces es, prácticamente imposible diferenciarlos. Su funcionamiento y características son similares. Disponen de más opciones de extensiones y plugins. Ya vienen preparados con herramientas y elementos que facilitan la escritura de código de aplicaciones web HTML, CSS y JavaScript. Como ejemplos **Visual Studio Code** de Microsoft, Atom de GitHub, Brackets de Adobe y Codekit y Coda, disponibles únicamente para entornos MacIntosh.

Brackets: editor de texto de código abierto para diseño web. Con una interfaz sencilla, permite a través de Google Chrome tener una vista previa dinámica, es decir, revisar los cambios en nuestro código sin necesidad de tener que guardar el texto. Se puede agregar extensiones a la aplicación para tener más características. El 01 de septiembre de 2021, Adobe deja de admitir Brackets, quedando descontinuado, aunque sigue como proyecto en github.

Visual Studio Code: Editor de código fuente gratuito de Microsoft. Ligero pero potente que se ejecuta en el escritorio y está disponible para Windows, Mac y Linux. Viene con soporte incorporado para JavaScript, TypeScript y Node.js y contiene extensiones para otros lenguajes (C++, C#, Python, PHP, entre otros). Ofrece el resaltado de sintaxis y autocompletado con IntelliSense, que proporciona terminaciones inteligentes basadas en tipos de variables, definiciones de funciones y módulos importados. Ofrece además la depuración del código directamente desde el editor.

**** Será este editor de código el que utilizaremos para el curso.**

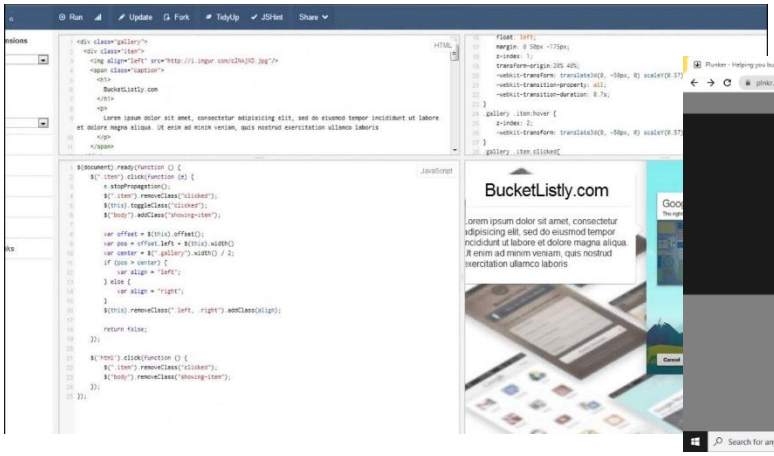
→ Entornos de desarrollo integrados (IDE): Software mucho más grande y que ofrecen todas las opciones que precisan los desarrolladores. Son entornos más pesados. Como ejemplos, WebStorm, Komodo, Aptana Studio, Eclipse (para Java), PhpStorm (para PHP), PyCharm (para Python), RybyMine (para Ruby), Xcode (para Mac) Visual Studio, Android Studio o Netbeans.

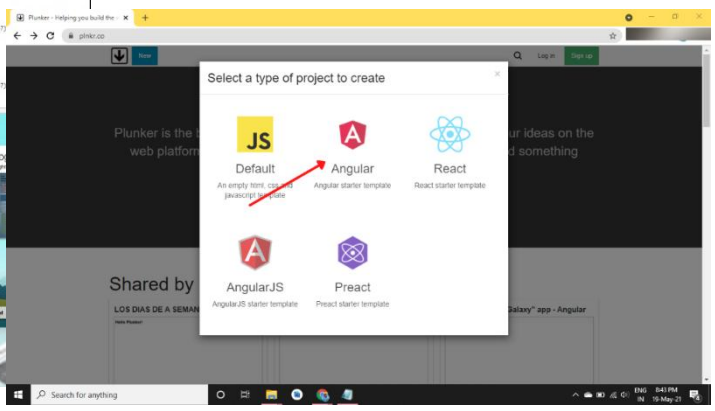
Netbeans: Para el desarrollo de escritorio, móviles y web de forma sencilla con Java, JavaScript, HTML5, PHP, C/C++ y más. IDE completamente gratuito, de código abierto. Proporciona resaltado de sintaxis, autocompletado y plegado de código. Dependiendo de lo que se va a desarrollar, puedes elegir la versión concreta. El soporte de jQuery está integrado en el editor. NetBeans 8.2 tiene soporte nuevo o mejorado para Node.js y Express, AngularJS, entre otros.

Visual Studio: Un IDE para Windows y Mac que te permite emular los distintos dispositivos en Windows y podemos crear aplicaciones para web, smartphones, etc…

Aptana Studio: Permite depurar el código JavaScript utilizando Firefox + Firebug. También se puede obtener una vista previa en el navegador integrado (soporta IE, Firefox y Safari). Se puede utilizar Aptana como un plugin con Eclipse o descargarse la versión autónoma de Aptana Studio.

→ Editores de código online: Aplicaciones web que permiten la escritura de código y permiten previsualizar el resultado al instante. Suelen incluir cierto espacio de almacenamiento para guardar el código en la nube y así acceder al mismo desde cualquier máquina. Como ejemplos, Fiddle y Plunker.





6.2. Ambientes de desarrollo

Cuando empezamos a crear un proyecto no lo subimos directamente a un hosting en la web, lo trabajamos en local.

Vamos a ver algunas herramientas que podremos instalar: Apache como servidor, MySQL como base de datos y PHP como lenguaje backend. Además incluyen administrador de bases de datos (phpMyAdmin).

Como ejemplos podemos destacar:

MAMP: Incluye Apache, Nginx, PHP y MySQL. Para Windows

WAMPServer: Incluye Apache, PHP y MySQL. Para Windows.

XAMPP: Incluye Apache, MariaDB, PHP, Perl, Tomcat. Multiplataforma.

Bitnami: Incluye imágenes de entornos de producción específicas a través de virtualización, orientadas a las distintas necesidades.

Vagrant / Vagrant Cloud: Simula entornos de producción a través de virtualización. Es multiplataforma.

6.3. Hosting

El hosting u hospedaje web nos va a permitir, una vez desarrollada la aplicación, tener un sitio web al que se pueda tener acceso desde cualquier parte del mundo.

Las compañías de hosting venden un espacio en sus servidores con acceso a internet para que podamos colocar nuestros archivos del proyecto, así como las bases de datos.

Hemos desarrollado nuestra aplicación y tenemos varios archivos, por ejemplo HTML, CSS, JavaScript, PHP y además una base de datos.

Contratamos un servidor, un hosting, y nos dan los accesos para colocar esos archivos en el servidor. Ahora ya podremos acceder a la página web a través de un usuario, una contraseña y una url.

Hay casos en los que las **empresas tienen sus propios servidores** como Facebook, pero otras **empresas alquilan su almacenamiento**, por ejemplo, en AWS (Amazon Web Service), donde están las películas de Netflix.

En algunos casos, algunos hostings permiten aumentar o disminuir los recursos dependiendo de las necesidades concretas.

También algunos hospedajes cuentan con herramientas de seguridad y respaldos.

Un hosting debe disponer de recursos tales como bases de datos ya instaladas, distintos lenguajes de programación, espacio, ancho de banda, memoria RAM, velocidad, correos electrónicos.

Acceso FTP, la forma más básica de acceso al servidor del hosting. Existen otras formas (SSH…).

Control Panel o cPanel para poder administrar tus dominios (www.comosellame.com), bases de datos, ver el espacio disponible, etc.

Seguridad de nuestras bases de datos.

7. NAVEGADORES WEB

Para el desarrollo de cualquier página web, debemos asegurar que el resultado que queremos es el adecuado con la mayor cantidad de navegadores posibles, especialmente los más populares o utilizados.

Hoy en día Google Chrome es el navegador más utilizado en el mundo. Además, debemos saber que existen otros navegadores como Microsoft Edge u Opera que se basan en el proyecto Chromium de Google.

Por otro lado, Apple Safari y Mozilla Firefox son los segundos y terceros en uso.

Para la realización de las prácticas y depuración, utilizaremos Google Chrome y/o Mozilla Firefox.

Basándonos en la página web statcounter, podemos ver el uso de los distintos navegadores en el último año (Junio 2021- Junio 2022).

En esta página también podremos encontrar los requisitos, características e instalación de los distintos navegadores.

A continuación, se detallan las características de algunos navegadores:

→ Internet Explorer: venía incluido con Windows 95 alcanzando una gran cuota de mercado. Navegador predeterminado en todos los equipos de Microsoft. Fueron surgiendo otras opciones que lo fueron relegando a un segundo plano por sus carencias respecto a la seguridad. Internet Explorer 11, después de muchos años, ha dejado de tener soporte desde el 17 de agosto de 2021.

→ Microsoft Edge: Aparece con Windows 10, reemplazo de Internet Explorer. Aporta una débil protección de datos aunque ofrece un filtro de pantalla que protege contra los ataques de malware y phishing. También tiene una página de inicio que se puede personalizar.

→ Google Chrome: nace en 2008 convirtiéndose en uno de los más populares y desbancando a Internet Explorer. Tiene un alto consumo de recursos, aunque destaca por su posibilidad de sincronización, rapidez y amplio portfolio de extensiones. Disponible la versión para desarrolladores, Chrome Canary (aún no es estable)

→ Mozilla Firefox: Nace en 2004. Es un navegador open source. Ofrece un excelente rendimiento y un alto número de extensiones, que le otorgan una gran versatilidad. Además, se actualiza con frecuencia. Es sólido y estable. En contrapartida puede ser un poco más lento de Chrome. Disponible la versión para desarrolladores, Firefox Developer.

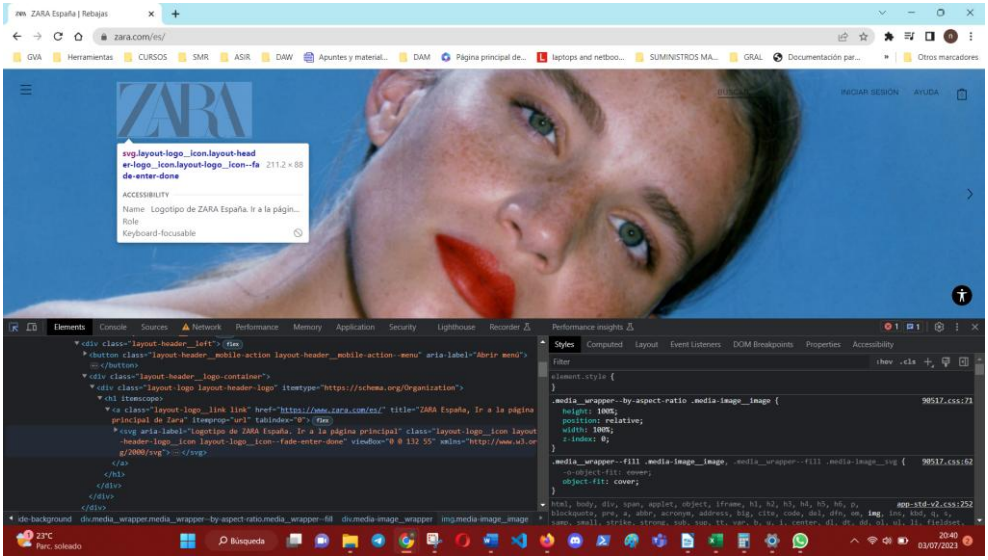
→ Safari: Navegador web exclusivo de Apple, lanzado en 2003. Ofrece una buena velocidad de navegación, es bastante intuitivo y la posibilidad de sincronización con todo el ecosistema de la marca. Ofrece además una muy buena protección de datos. Por otro lado, ofrece pocas actualizaciones y pocas extensiones. Su versión para desarrolladores es Safari Technology.

→ Opera: de origen noruego, lleva en el mercado desde 1995. Si bien, es de los más maduros, no es el más utilizado, siendo especialmente popular en África. La versión Opera 15 utiliza el mismo motor de búsqueda que Google Chrome, por lo que se pueden utilizar las mismas extensiones disponibles para el navegador de Google. Apunta un buen rendimiento sin un consumo excesivo de recursos. Ofrece monedero cripto. Como contrapartida, actualizaciones muy frecuentes.

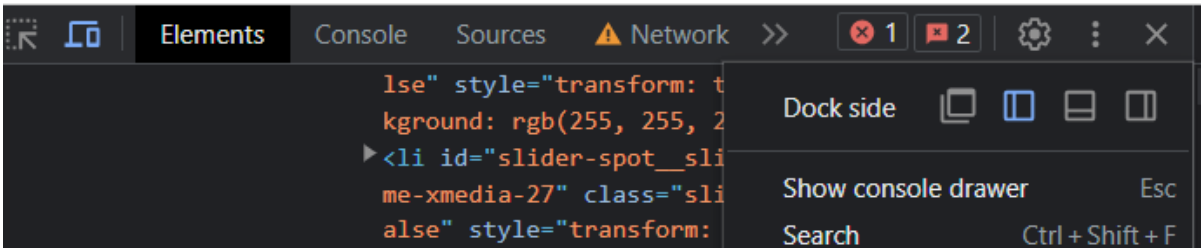
7.1. Web Developer Tools

En todos los navegadores podemos encontrar herramientas que nos van a ayudar en el desarrollo de la aplicación.

Con el botón derecho → inspeccionar → se abre la consola y podremos a través de distintas opciones, realizar pruebas o comprobaciones en nuestra aplicación web.

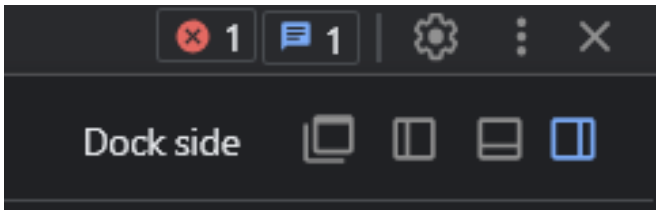


 → A través de este icono podremos seleccionar cualquier elemento de la página y podremos ver su posición en el html y sus estilos asociados. Los podremos modificar, pero no se guardarán los cambios.

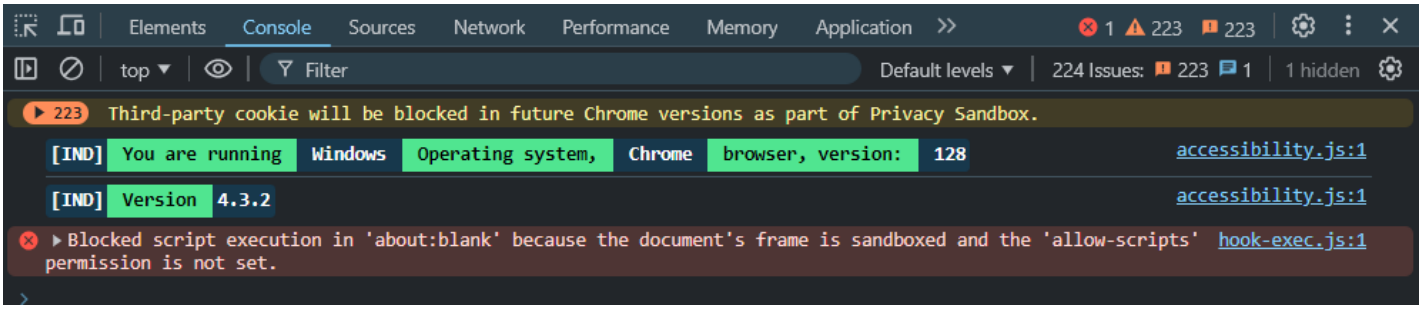


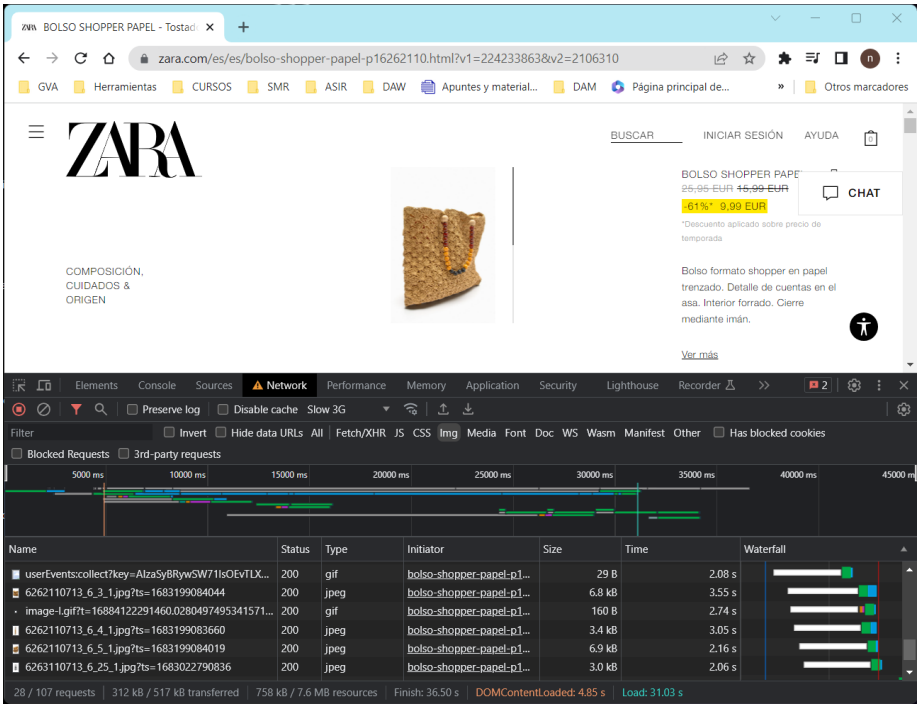
 → A través de este icono podremos ver cómo se comporta nuestra aplicación en distintos tamaños de dispositivos. Analizamos la característica de responsive. En el desplegable podremos poner directamente “Responsive” y utilizar las líneas laterales y oblicuas para redimensionar o bien elegir un dispositivo concreto.

 → A través de este icono podremos posicionar la pantalla de herramientas de desarrollo en una ventana aparte, a la izquierda, debajo o a la derecha de la página web.



 → Aquí podremos ver los errores, warnings e interactuar con la aplicación para detectar errores.





Network → En esta sección podemos ver las características de los elementos, por ejemplo las fotos: el peso, tiempo que tardan en cargarse… De esa forma, podemos controlar los tiempos de carga de los elementos.

Sources → En este apartado podemos inspeccionar los archivos de nuestro proyecto. Por ejemplo, queremos ver dónde está una funcionalidad concreta, un evento.

Podemos seleccionar el evento concreto que queremos inspeccionar, en este caso, el click del ratón.

Cuando hagamos click sobre cualquier funcionalidad de la página (añadir al carrito, eliminar un artículo…), iremos directamente al código que está ejecutando esa funcionalidad.

Lo vamos a ver en estado debugger, para poder ir siguiendo la traza del código.