Centro Integrado de Formación Profesional

AVILÉS

Principado de Asturias

UNIDAD 1:
SELECCIÓN DE ARQUITECTURAS Y HERRAMIENTAS DE PROGRAMACIÓN

DESARROLLO WEB EN ENTORNO CLIENTE

2º CURSO

C.F.G.S. DISEÑO DE APLICACIONES WEB

REGISTRO DE CAMBIOS

Versión	Fecha	Estado	Resumen de cambios
1.0	16/09/2024	Aprobado	Primera versión

ÍNDICE

UNIDAD 1: SELECCIÓN DE ARQUITECTURAS Y HERRAMIENTAS DE PROGRAMACIÓN

1.1 DESARROLLO DE SOFTWARE

1.1.1 DESARROLLO DE APLICACIONES WEB

Muchas de las discusiones sobre diseño web o desarrollo web son confusas, ya que la expresión varía considerablemente. Mientras que la mayoría de la gente tiene algún tipo de noción sobre lo que significa diseño Web, no es fácil definirlo con exactitud.

Algunos componentes como diseño gráfico o programación forman parte de esa discusión, pero su importancia en la construcción de webs varía de persona a persona y de web a web. Hay quien considera la creación y organización de contenido -o más formalmente, la arquitectura de la información- como el aspecto más importante del diseño web. Otros factores como la facilidad de uso, el valor y funcionalidad del sitio web en la organización, su funcionalidad, accesibilidad, publicidad, etc. también forman una parte muy activa hoy en día sobre lo que se considera diseño web.

El desarrollo web sigue estando muy influenciado por múltiples campos como el de las nuevas tecnologías, los avances científicos, el diseño gráfico, la programación, las redes, el diseño de interfaces (medio o forma a través de la cuál un usuario se comunica con el ordenador) de usuario, la usabilidad y una variedad de múltiples recursos. Por lo tanto, el desarrollo web es realmente un campo multidisciplinar.

Se distinguen cinco áreas que cubren la mayor parte de las facetas del Diseño Web:

- Contenido: incluye la forma y organización del contenido del sitio. Esto puede abarcar desde cómo se escribe el texto hasta cómo está organizado, presentado y estructurado usando un lenguaje de marcas como HTML.
- Visual: hace referencia a la plantilla empleada en un sitio web. Esta plantilla generalmente se genera usando HTML y/o CSS y puede incluir elementos gráficos para decoración o para navegación. El aspecto visual es el aspecto más obvio del diseño web, pero no es la única disciplina o la más importante.
- Tecnología: aunque muchas de las tecnologías web como HTML o CSS entran dentro de esta categoría, la tecnología en este contexto generalmente hace referencia a los diferentes tipos de elementos interactivos de un sitio web, generalmente aquellos construidos empleando técnicas de programación.
- Distribución: la velocidad y fiabilidad con la que un sitio web se distribuye en Internet o en una red interna corporativa está relacionado con el hardware/software utilizado y el tipo de arquitectura de red utilizada en la conexión.
- Propósito: la razón por la que un sitio web existe, generalmente está relacionada con algún aspecto de tipo económico. Por lo tanto, este elemento debería considerarse en todas las decisiones que se tomen en las diferentes áreas.

El porcentaje de influencia de cada una de estas áreas en un sitio web, puede variar dependiendo del tipo de sitio que se está construyendo. Una página personal generalmente no tiene las consideraciones económicas que tendría unaweb que va a vender productos en Internet.

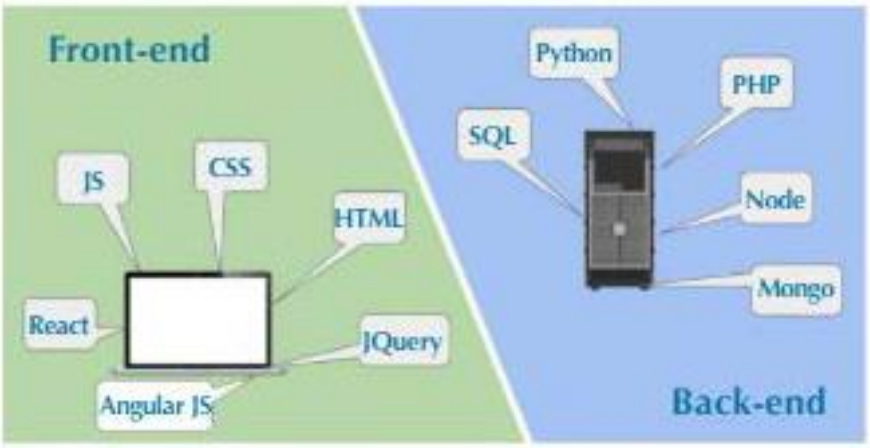
1.1.2 TIPOS DE APLICACIONES

Poniendo el foco en lo que sería el desarrollo web en entorno cliente, se pueden distinguir varios tipos de aplicaciones, cada una con características específicas que influyen en la selección de arquitecturas y herramientas de programación.

- Aplicaciones web estáticas: Contienen contenido fijo que no se va a modificar en respuesta las interacciones del usuario. En su concepción más purista, este tipo de aplicaciones solo constaría de HTML apoyado por CSS y un JavaScript muy básico. En cualquier caso, una web está sujeta a constantes cambios y rara vez no requiere algún tipo de contenido dinámico, por lo que se podría considerar una arquitectura en desuso.
- Aplicaciones Web dinámicas. Su contenido se actualizará en función de las interacciones del usuario pudiendo incluso sufrir modificaciones sin necesidad de recargar la página. Cualquier sitio de comercio electrónico o red social sigue este paradigma, el cual se apoya en frameworks como React o Angular, en lenguajes de servidor como PHP, C# en ASP.NET o Java JSP.
- Single Page Applications (SPA): Estas aplicaciones solo cargan una página HTML actualizando dinámicamente su contenido a medida que el usuario interactúa. Principalmente usan los frameworks ya citados.
- Progressive Web Apps (PWA): Son aplicaciones que usan capacidades modernas para comportarse como aplicaciones nativas. Principalmente su utilización se sitúa en las aplicaciones móviles y las web con experiencia de usuario mejorada.
- Aplicaciones híbridas: Combinan elementos de aplicaciones web y nativas permitiendo el despliegue en múltiples plataformas. Se usan en aquellas aplicaciones que necesitan acceder a funcionalidades nativas del dispositivo. Utilizan herramientas como Ionic o Cordova.
- Aplicaciones basadas en microservicios: Siguen lo que se conoce como SOA (Service Oriented Application), Aplicaciones Orientadas a Servicios. Se componen de pequeños servicios independientes que pueden comunicarse entre sí. Se basan en herramientas como Docker y Kubernetes a nivel de despliegue y como tecnologías de desarrollo, todas aquellas que permitan la creación de servicios web (Python, Node.js, .NET, Java, etc.)

1.2 MODELOS DE PROGRAMACIÓN EN ENTORNOS CLIENTE / SERVIDOR

Actualmente se suele separar el desarrollo de aplicaciones web en dos partes bien diferenciadas: back-end, que sería la parte no visible de la web como la que conecta con bases de datos o los propios scripts que se ejecutan en el servidor y front-end (o frontal), la cual corresponde a la parte visible de la web como las hojas de estilo, código HTML, scripts en el lado del cliente, etc.



1. Front-end y Back-end

En las empresas desarrolladoras de aplicaciones web, existen perfiles técnicos adecuados tanto al back-end como al front. Los del back-end se encargan de la parte del servidor como el acceso a bases de datos (MySQL, MariaDB, PostgreSQL, MongoDB, Oracle, SQL Server, etc.), creación de servicios, etc. Normalmente se utilizan lenguajes que permiten esa programación en servidor como PHP, Python, Ruby, C# y Java, aunque también se puede utilizar el mismo JavaScript, el cual históricamente ha estado asociado al front-end y como lenguaje cliente, pero que gracias a Node.JS también puede utilizarse en el back-end sin mayor problema.

En la parte del frontal prima la parte creativa y la originalidad ya que es un perfil más cercano al de un diseñador, aunque también se trabaje codificando. En este caso, se usan lenguajes como HTML, CSS, JavaScript y orientados al renderizado como JSP, ASP.NET. En la actualidad, Frameworks como Angular, React o VueJS están ganando terreno en esa parte cliente.

1.2.1 TECNOLOGÍAS Y LENGUAJES ASOCIADOS

1.2.1.1 Lenguajes de programación en entorno cliente

La programación web en el lado del cliente se basa en tres pilares fundamentales que se citan a continuación:

1. Lenguaje HTML. Es un lenguaje de marcado, no de programación. Define el contenido que va a tener el documento y será lo que finalmente leerá e interpretará el navegador. Su principal ventaja es que es capaz de ser visualizado e interpretado por cualquier navegador.

2. Lenguaje CSS. Define la presentación del documento pudiendo catalogarse como un lenguaje de diseño gráfico, que no de programación. Su objetivo es que la web sea atractiva al usuario.
3. Lenguaje JavaScript. Lenguaje de programación cuya principal función es agregar contenido dinámico a las páginas web. Es preciso tener en cuenta que el ámbito del diseño web no es el único en el que se mueve este lenguaje, aunque sí es en el que más se utiliza.

1.2.1.2 Características de los lenguajes de script

En contraste con la concepción clásica de script como fragmento de código que realiza una secuencia de operaciones, actualmente, un conjunto de scripts puede conformar todo un sistema de información completo. A continuación, se enumeran algunas características de los lenguajes de script:

- Generalmente, los lenguajes de script son interpretados mientras que los de programación puros además pueden ser compilados.
- Los lenguajes de script se pueden incrustar dentro de otros lenguajes (p.e. JavaScript dentro de HTML).
- Precisamente, por su naturaleza de lenguajes interpretados, los scripts se ejecutan línea a línea.
- Los lenguajes de script no generan un ejecutable
- Los lenguajes de script han sido diseñados para ser fáciles de utilizar y programar.

1.2.1.3 Integración del código con las etiquetas HTML

El código de un programa escrito en JavaScript se incluye dentro de una página HTML entre las etiquetas <script> y </script >. Estas etiquetas pueden aparecer en la cabecera de una página (entre <head> y </head>) o en el cuerpo de la misma (entre <body> y </body>). El script estará compuesto por instrucciones que indicarán las acciones a realizar.

Cuando se incluye un script en la cabecera de una página HTML, éste suele estar estructurado en forma de funciones que se ejecutan sólo cuando se las invoca desde otro script o cuando se produce el evento al que están asociadas, mientras que cuando se hace en el cuerpo de una página HTML, las instrucciones que lo forman se ejecutan en el momento de la carga de la página en el navegador del cliente.

El primer ejemplo consistirá en el típico "Hola mundo". El script está formado por la expresión document.write("<h1 align='center'>Hola mundo</h1>"), que tiene como efecto escribir la cadena de caracteres entre comillas.

```
<html>
    <head>
        <title>Ejemplo de script en cuerpo de página</title>
    </head>
    <body>
        <script>
            document.write("<h1 align='center'>Hola mundo</h1>");
        </script>
    </body>
</html>
```

A continuación, se va a realizar el mismo ejemplo, pero usando una función (mensaje) en la cabecera de la página, a la cual se va a llamar en el momento de la carga de la página en el navegador. El resultado obtenido es el mismo que el del anterior.

```
<html>
    <head>
        <title>Ejemplo de script en encabezado de página</title>
        <script>
            function mensaje(){
                document.write("<h1 align='center'>Hola mundo</h1>");
            }
        </script>
    </head>
    <body>
        <script>mensaje()</script>
    </body>
</html>
```

Por supuesto, tener todo el código JavaScript mezclado con código HTML no parece una solución muy elegante especialmente cuando se dispone de Frameworks o bibliotecas externas. Además, muchos de las funciones requeridas por una web se necesitan en más de una página. Para estos casos, resulta más adecuado externalizar el código JavaScript en un archivo .js e incluirlo con una línea dedicada exclusivamente a esa referencia, tal como puede verse en el código a continuación:

```
<script src='Scripts/jquery.min.js'></script>
```

1.2.1.4 Mecanismos de ejecución de código en un navegador web

Como ya se ha visto anteriormente, es posible integrar JavaScript en código HTML, lo que hace que sea interpretado por el navegador. En este apartado, se verán además las formas con las que se puede comunicar con el usuario mediante código JavaScript.

A continuación, se verán varias opciones:

- Escribir en la consola del navegador usando **console.log()**. Con ella se puede mandar información a la consola, pero esta no es a priori accesible por los usuarios de un navegador salvo que utilicen las herramientas de depuración del navegador que se verán posteriormente. Por tanto, esta opción es muy útil para desarrollo. Un ejemplo de este tipo de script es el siguiente:

```
<script>
    console.log('Hola, mundo')
</script>
```

- Escribir en cualquier elemento HTML con innerHTML. Aunque se profundizará en este atributo posteriormente, se puede modificar el HTML subyacente de un elemento web. A continuación, un ejemplo de uso de esta carácterística.

```
<p id="parrafo"></p>
<script>
    document.getElementById("parrafo").innerHTML = 5 + 6;
</script>
```

- Generar directamente HTML mediante document.write(). Con este método se puede generar HTML incluyendo etiquetas.

```
<script>
    document.write("Hola, mundo");
</script>
```

1.2.1.5 Capacidades y limitaciones de ejecución.

En entorno cliente, un aspecto importante a tener en cuenta son las capacidades y limitaciones de ejecución de las aplicaciones.

Entre sus capacidades, se pueden distinguir las siguientes:

- Rendimiento: Que la aplicación funcione de forma fluida es importantísimo para tener una buena experiencia de usuario. Frameworks como React, Angular y Vue.js mejoran la experiencia del usuario al optimizar la velocidad de carga y ejecución.
- Compatibilidad: Es clave que las aplicaciones puedan funcionar en distintos navegadores y dispositivos. También aquí son claves los frameworks y herramientas modernas ya que se han adaptado a las nuevas tecnologías.
- Escalabilidad: A la escalabilidad de las aplicaciones ayudan arquitecturas como SPA (Single Page Application) ya que permiten aplicaciones escalables y de alto rendimiento.

En cuanto a las limitaciones, a continuación, se detallan las más significativas:

- Consumo de recursos: Hay que tener cuidado con algunas herramientas ya que pueden hacer un uso intensivo de memoria y CPU afectando el rendimiento en dispositivos menos potentes.
- Tamaño de la aplicación: El tamaño de los archivos JavaScript y CSS puede incrementar los tiempos de carga.

- Latencia: La cantidad de peticiones HTTP y la estructura de las APIs (RESTful,) influyen en la latencia, que equivale al tiempo de respuesta a una petición.
- Seguridad: Las aplicaciones web están expuestas a vulnerabilidades como XSS y CSRF, dependiendo de las herramientas y prácticas empleadas.

1.2.1.6 Compatibilidad con navegadores web

La compatibilidad con el navegador web es un aspecto fundamental en el desarrollo de aplicaciones web. Es importante considerar varios factores relacionados con dicha compatibilidad:

- Diversidad de navegadores. Es importante que las aplicaciones sean compatibles con los navegadores más utilizados como Chrome, Firefox, Edge, Safari y Opera. Existen herramientas como BrowserStack que sirven para probar la aplicación en distintos navegadores y versiones.
- Estándares web. La aplicación debería seguir los estándares web establecidos por el W3C (World Wide Web Consortium)
- Compatibilidad entre Versiones: Lo deseable es que las aplicaciones funcionen en varias versiones de los navegadores, incluyendo versiones más antiguas. Teniendo en cuenta que el JavaScript más moderno se basa en el estándar ECMAScript del año 2015, no se puede aspirar a llegar a navegadores completamente obsoletos, pero sí a versiones relativamente recientes. En cualquier caso, hay herramientas conocidas como "transpiladores" (tipos especiales de compilador que traducen de un código fuente a otro con un nivel de abstracción parecido) que permiten convertir ese código moderno a otro más antiguo. Un ejemplo de este tipo de herramientas es Babel. También se puede añadir soporte a funciones que no están disponibles en todos los navegadores utilizando lo que se conoce como Polyfills
- Inconsistencias: Diferentes navegadores pueden interpretar el código de manera diferente, lo que puede causar inconsistencias visuales y funcionales. Para evitar esto, es bueno ayudarse de bibliotecas como Bootstrap o Normalize.css
- Características Específicas de Navegadores: Algunos navegadores pueden tener características únicas que pueden aprovecharse, pero siempre se debe proporcionar una funcionalidad alternativa para otros navegadores. Para mitigar esto, se pueden usar herramientas de detección de características (feature detection) y provisión de soluciones alternativas (fallbacks).

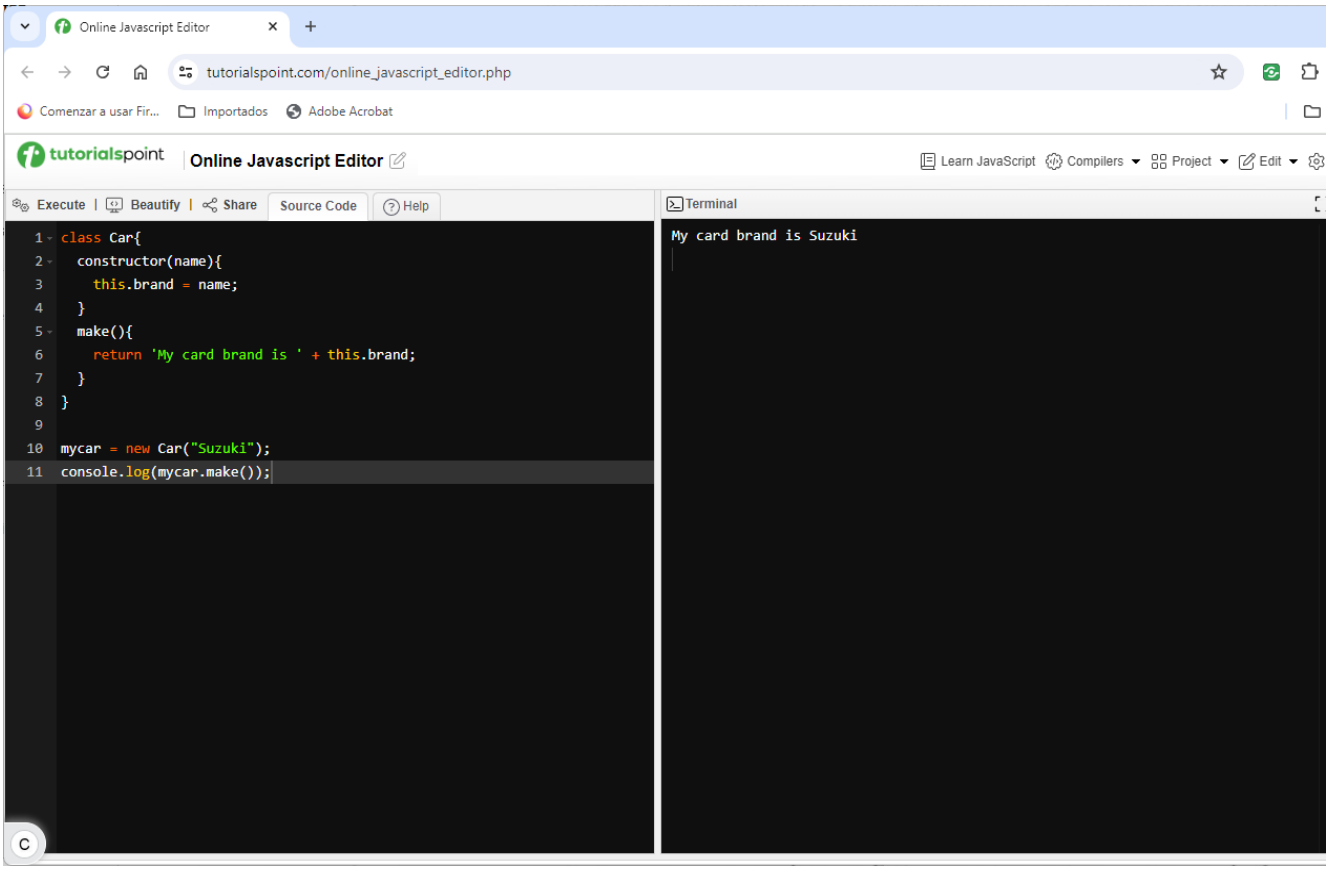
1.3 HERRAMIENTAS PARA EL DESARROLLO EN EL LADO CLIENTE

1.3.1 HERRAMIENTAS DE PROGRAMACIÓN

Existen muchas alternativas a la hora de elegir una herramienta de programación. Una de las ventajas de programar en JavaScript es que puede servir un simple editor de texto, aunque esto no es lo deseable cuando se desarrolla un proyecto. Lo normal es utilizar alternativas como entornos de programación online o editores avanzados complementados con un sistema de control de versiones.

1.3.1.1 Herramientas online

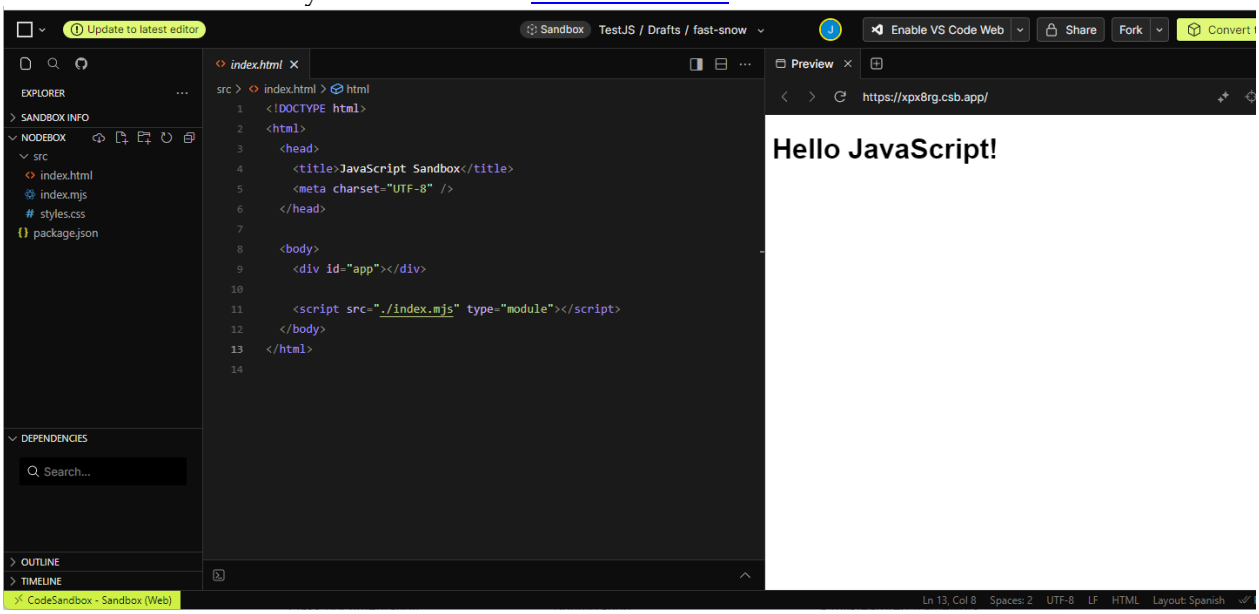
Las ventajas que ofrece un IDE (entorno integrado de desarrollo) online son evidente. Se puede ejecutar y probar código desde cualquier dispositivo simplemente con tener acceso a internet. Para esto, existen varias herramientas en el mercado. Una de las más interesantes es Coding Ground de Tutorialspoint.



2. Interfaz de Coding Ground

Con esta herramienta, no solo se pueden gestionar los distintos ficheros de un proyecto, sino que se pueden agregar ficheros externos.

Otra herramienta muy interesante es CodeSandbox



3. Interfaz de CodeSandbox

1.3.1.2 Utilización de IDE y sistemas de control de versiones

Para trabajar con JavaScript y tecnologías relacionadas, se pueden utilizar IDEs como Eclipse con extensiones como el JSDT (JavaScript Development Tools) o Visual Studio en proyectos .NET de tipo web. En cualquier caso, las mejores opciones parten de los editores avanzados como Visual Studio Code o Sublime Text, los cuales tienen funcionalidades que van más allá de la edición y coloreado de código acercándose a la potencia de lo que sería un entorno de desarrollo integrado.

En este módulo, se ha optado por trabajar principalmente con Visual Studio Code, aunque tanto código como extensiones son fácilmente adaptables a otros entornos. Entre sus ventajas están las siguientes:

- Es de código abierto, por lo que es completamente gratuito.
- Es modular. Se pueden deshabilitar ciertas funciones y añadir aquellas que sean del gusto del usuario.
- Dispone de un gestor de paquetes o extensiones.
- Se puede integrar con GitHub.
- Incluye un sistema muy potente de autocompletado.

Por ejemplo, para trabajar con JavaScript puede ser interesante instalar la extensión **JavaScript (ES6) code snippets** que contiene una serie de atajos de código en este lenguaje. La lista completa se puede consultar en la pestaña de Detalles de la propia extensión. Otra extensión interesante es **Live Server**, con la que se puede crear un miniservidor web de desarrollo para probar los cambios en el código pudiendo visualizarse en tiempo real.

Los sistemas de control de versiones (VCS – Version Control System) también llamados de control de código fuente (SCM – Source Content Management) son herramientas que permiten

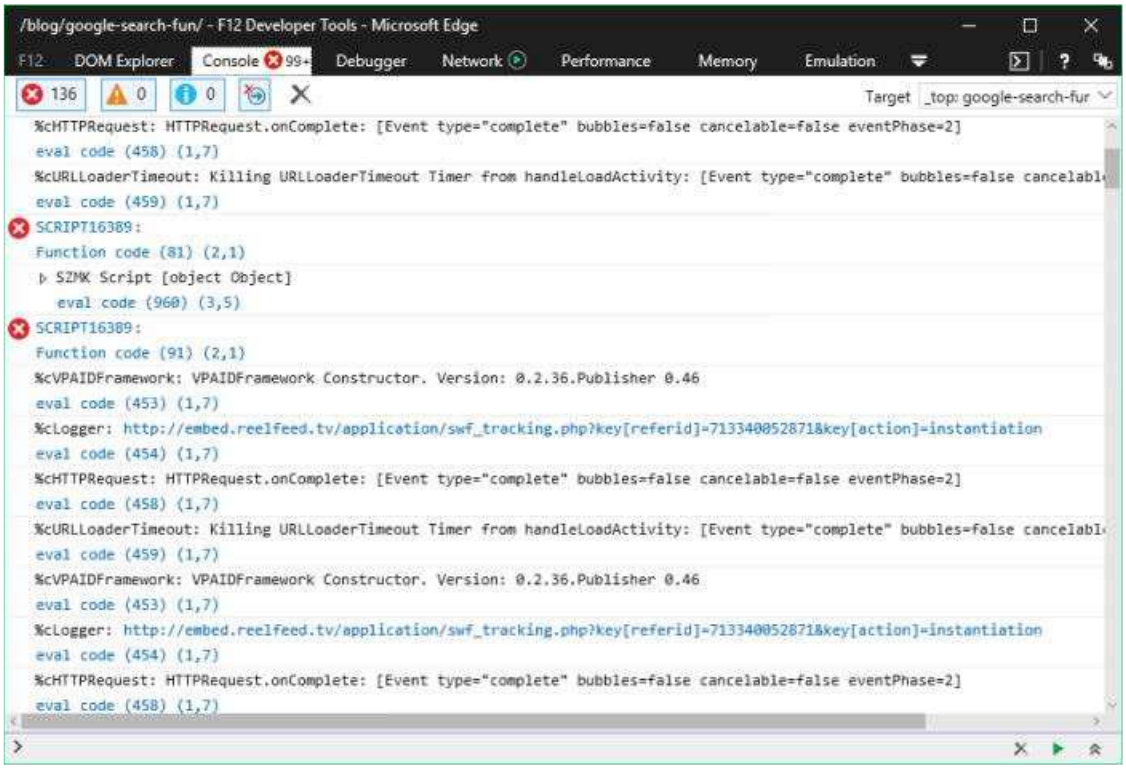
gestionar, compartir y mantener un historial de cambios de un código fuente. Es un sistema que permite seguir con precisión la evolución del contenido de los archivos. Este tipo de sistema se emplea mucho en el desarrollo informático, pero no está limitado a este, ya que cualquier actividad que utiliza archivos legibles podrá ser seguida por un VCS. Por ejemplo, los sistemas Wiki (como Wikipedia) utilizan un VCS. También es posible citar el conocido sitio de asistencia informática Stack Overflow, que sigue las versiones de las preguntas y respuestas.

Git es un sistema libre de gestión de versiones, lo que implica que no puede haber limitaciones contractuales sobre su uso. Sea cual sea el número de colaboradores, de proyectos o actualizaciones, Git siempre será gratuito. Una ventaja muy importante: Git es actualmente el sistema más popular y gana en popularidad cada año.

La potencia de Git reside en su carácter distribuido. Ello implica que varios desarrolladores estén en sus repositorios locales enviando y obteniendo datos de un repositorio remoto. Existen varias soluciones que ofrecen repositorios remotos. Las más conocidas son GitHub y BitBucket.

1.3.1.3 Herramientas del navegador para depuración de código

Los navegadores modernos incluyen una característica integrada que permite la depuración del código. Es gratuita siempre que lo sea el navegador y permite interactuar, evaluar, detectar errores y fallos de programación dentro de una página Web. Consta de distintas pestañas o apartados. Por ejemplo, si se trata de detectar vulnerabilidades en una web, es interesante acceder al apartado Network o Red.



4. Herramientas de desarrollo en Edge

Otro apartado interesante es el de la consola en el cual se puede ejecutar código JavaScript de forma directa. Cuando se lanza el comando console.log (aparecerá a lo largo del curso en varias ocasiones), su salida podrá verse en esta pestaña.

En el ámbito de la depuración, el análisis y tratamiento de errores es vital para mantener la integridad de la aplicación y, por ende, garantizar una experiencia de usuario fluida.

En los lenguajes de script se pueden distinguir los siguientes tipos de error:

1. Errores de sintaxis. Suceden cuando no se siguen las reglas sintácticas del lenguaje. Generalmente se detectan por el propio editor o depurador.
2. Errores en tiempo de ejecución. El código es sintácticamente correcto, pero intenta ejecutar una operación incorrecta durante la ejecución (por ejemplo, una división por cero).
3. Errores lógicos. El código está bien sintácticamente, hace operaciones correctas, pero no da el resultado esperado.

Existen distintas técnicas para tratar este tipo de errores, como, por ejemplo, el uso de bloques try…catch…finally o la validación de los datos de la aplicación. Además, es muy importante registrar todos los eventos relacionados con los posibles errores para poder analizarlos y mejorar el software.

1.3.2 BIBLIOTECAS Y FRAMEWORKS

1.3.2.1 Node.JS

Node.js es ahora mismo uno de los pilares del desarrollo de aplicaciones modernas. Es capaz de manejar operaciones asíncronas y resulta muy versátil de forma que se puede usar en múltiples plataformas. No es un framework al uso ni tampoco un lenguaje de programación; se trata de un entorno de ejecución para JavaScript del lado del servidor. Esto implica que se puede ejecutar código JavaScript fuera de un navegador web. Construido sobre el motor V8 de Google Chrome, NodeJS transforma el código JavaScript en un código de máquina eficiente y rápido, lo que permite construir aplicaciones de servidor altamente escalables.

Para instalarlo, basta con acudir a la web y descargar el paquete correspondiente al sistema operativo destino. Lo recomendable es instalarse la versión LTS (Long Time Support) que es la propuesta por defecto), ya que es la más estable. De todas formas, en la sección **Download** se pueden ver todas las opciones de descarga por sistema operativo y versión, tal como se observa en la figura.

5. Página de descarga de NodeJS

Para asegurarse que la instalación ha sido correcta, tal como también se ve en la figura, basta con llamar a la siguiente orden en consola:

```
npm -v
```

npm es el comando que lanza el NodeJS Package Manager, un gestor de paquetes propio de Node al estilo de apt u otros y que es la base para la instalación de una buena parte de frameworks JavaScript.

1.3.2.2 Angular

Angular fue creado y mantenido por Google. Su primera versión se denominó AngularJS por utilizar JavaScript como lenguaje soporte. En sucesivas versiones se eliminó el sufijo JS ya que en ellas se empezó a utilizar TypeScript, una evolución del lenguaje compatible con ECMAScript.

1.3.2.3 React

React es uno de los frameworks frontend más populares. Uno de los indicadores que resulta más relevante son las "estrellas" en GitHub y a fecha de junio de 2024 está en torno a las 225 mil. Además, el número de descargas semanales está sobre los 16 millones, lo que ya da una idea de su importancia en el mercado.

Desarrollado por el mismo equipo de Facebook, en la industria del desarrollo de software está siendo uno de los productos más punteros porque no solo se ciñe al ámbito web, sino que se pueden desarrollar aplicaciones móviles con React Native y de escritorio envolviéndolo en otro framework llamado Electron.

1.3.2.4 VueJS

Una de las características más relevantes de VueJS frente a otros frameworks es la ligereza y velocidad de ejecución. Al igual que React, usa un DOM virtual, lo que ofrece múltiples ventajas.

ÍNDICE DE FIGURAS

BIBLIOGRAFÍA – WEBGRAFÍA

Moreno Pérez, J.C. (2020) *Desarrollo web en entorno cliente*. Editorial Síntesis. 1ª Edición

Navarro Galdón, A. (2022) *React JS: La biblioteca de JS creada por Facebook*
https://www.udemy.com/course/react-js-inicia-en-el-mundo-de-los-frameworks-de-javascript/