Selección de arquitecturas y herramientas de programación

INDICE

[1. Desarrollo Software 2](#_Toc161735845)

[2. Tecnologías del lado cliente 4](#_Toc161735846)

[3. Herramientas de desarrollo 6](#_Toc161735847)

[4. Framework Frontend 7](#_Toc161735848)

[Bibliografía: 7](#_Toc161735849)

[Apartado 1º: 7](#_Toc161735850)

[Apartado 2º: 7](#_Toc161735851)

[Apartado 3º: 7](#_Toc161735852)

[Apartado 4º: 7](#_Toc161735853)

# Desarrollo Software

El ciclo de vida del software es un periodo de lo más complejo, por el cual la propia aplicación ha de pasar por muchas etapas antes de poder llegar a su versión final.

Se debe de tener en cuenta, además, que el desarrollo del software es algo que se mantiene en el tiempo hasta su muerte o su fin de mantenimiento. Durante ese periodo de mantenimiento, se debe entender que el software agregará nuevas funcionalidades y que esas mismas actualizaciones requerirán de cambios en el proyecto y se tendría que replantear el desarrollo de la actualización desde el punto inicial al cual se enfrentó el nacimiento del mismo proyecto.

Por esto mismo existen metodologías de desarrollo de software (como el **SCRUM**) para un desarrollo mucho más sostenible y que permite un proceso que se adapte con mayor facilidad a sus problemas, errores y nuevas funcionalidades que se quieran agregar por parte del cliente.

Las etapas por las que ha de pasar el software antes de salir serían las siguientes:

* **Planificación:**

Esta primera etapa es el pie clave para el desarrollo de cualquier software que se intente crear. En esta parte, se trazan los objetivos y se valoran las necesidades y funcionalidades del proyecto.

También, se ha de tener en cuenta que este es el proceso de planificación y organización para las futuras fases del proyecto, por ello, si en este primer apartado flaquease en su realización el equipo del proyecto, el resto del mismo se vería truncado por una escasa solidez de sus propios cimientos.

Esta primera etapa es el periodo en el cual se va a tener en cuenta:

La **viabilidad** del propio proyecto, el análisis de los **riesgos** y contratiempos a los que se podrá enfrentar y la **planifica** del resto de fases con unas pautas bien planificadas y estructuradas.

* **Análisis:**

En la fase del análisis, se definirán las funciones que ejecutarán el programa y las características más esenciales del mismo.

En este proceso se calculan los costes del proyecto y se priorizan todas aquellas funcionalidades que sean elementales e indispensables para la app.

Gracias a lo cual, seremos capaces de comprender la envergadura del objetivo al que se intenta enfrentar el proyecto.

* **Diseño:**

El diseño es la parte donde se estudiaran todas las posibles alternativas en la estructura tanto de interfaz, como de navegación y de interfaz del propio proyecto.

Por ello, los puntos más importantes en el diseño serían:

La **estructuración** de la **BD** para facilitar la implementación del proyecto.

La **lógica del flujo** de datos trazando un diagrama de flujo de los propios datos, dando información visual detallada del proyecto. A parte del diagrama de flujo, también se pueden crear los mapas de navegación y los diagramas de clase para mayor detalle del funcionamiento de la aplicación.

Por último y, una de las partes más importantes, está la **interfaz de usuario** creando un diseño intuitivo y fácil de usar para cualquier tipo de usuario es una parte clave del mismo proyecto.

* **Desarrollo:**

Cuando se hubiese definido el diseño del software, el siguiente paso sería llevar el proceso creativo al plano de la creación y desarrollo. Por ello, en esta parte y con todas las herramientas de programación escogidas y los lenguajes de programación seleccionados, comienza el desarrollo de todas las funciones y vistas propuestas para la web.

* **Pruebas:**

Una vez se haya acabado el desarrollo del software, empieza el proceso de testeo y las pruebas de estrés contra la app. Ha de ser capaz de responder a estas pruebas que pueda satisfacer los objetivos marcados en la planificación. Algo de mencionar es que, durante el desarrollo de la aplicación, también se suele realizar el proceso de testeo, pero en este caso las pruebas más importantes y los cambios más notorios se realizan en el transcurso de esta misma fase.

* **Mantenimiento:**

Por decirlo de alguna manera, esta puede llegar a ser la fase final del proyecto en la cual lo único que es importante es la actualización del proyecto e intentar mantener sus funcionalidades y su código lo menos obsoleto posible, depurar y arreglar posibles errores que surjan en relación a la seguridad del mismo o, simplemente intentar mantener una estética y apariencia actualizada en cada momento.

# Tecnologías del lado cliente

A sabiendas que la programación en el plano del desarrollo web se puede llegar a dividir en 2 lados: el lado del **cliente** y el lado del **servidor** y, además de saber que, todo los scripts creados en el lado del cliente se ejecutarán en el mismo ordenador del usuario, en vez de un servidor, podemos diseminar las diferentes tecnologías o partes que son necesarias para el desarrollo en el cliente.

Las principales tecnologías empleadas en el lado del cliente serían:

**HTML**:

HTML (HyperText Markup Language) es el lenguaje de marcado estandarizado en toda la web. Esta tecnología se emplea con las etiquetas por defecto que ya tiene creadas para dar el cuerpo a la web estructurando el contenido en bloques y representado tanto texto con formatos diferentes, como imágenes, formularios,…

El documento HTML da posibilidad de referir o incluir los diferentes scripts que se hayan creado para la web a través de **JS** (u otro lenguaje de programación como **PHP**) y los estilos que configuran la apariencia y el diseño en el formato del **CSS**.

**CSS**:

Hablando del estilo, tenemos el segundo estándar y, por ende, la segunda tecnología más predominante en toda la web. Esta herramienta es el **CSS** (Cascading Style Sheets) y sirve para controlar el estilo, la apariencia y el diseño de los elementos definidos en el archivo **.html**.

Como su nombre indica, hay que tener en cuenta que los estilos siguen la corriente de una cascada, es decir, hay un orden y en función de cómo estén aplicados los estilos, se aplicarán y otros se sobrescribirán, por ello, también hay que tener cuidado en cómo se especifican los estilos (**imagen 2.1**)

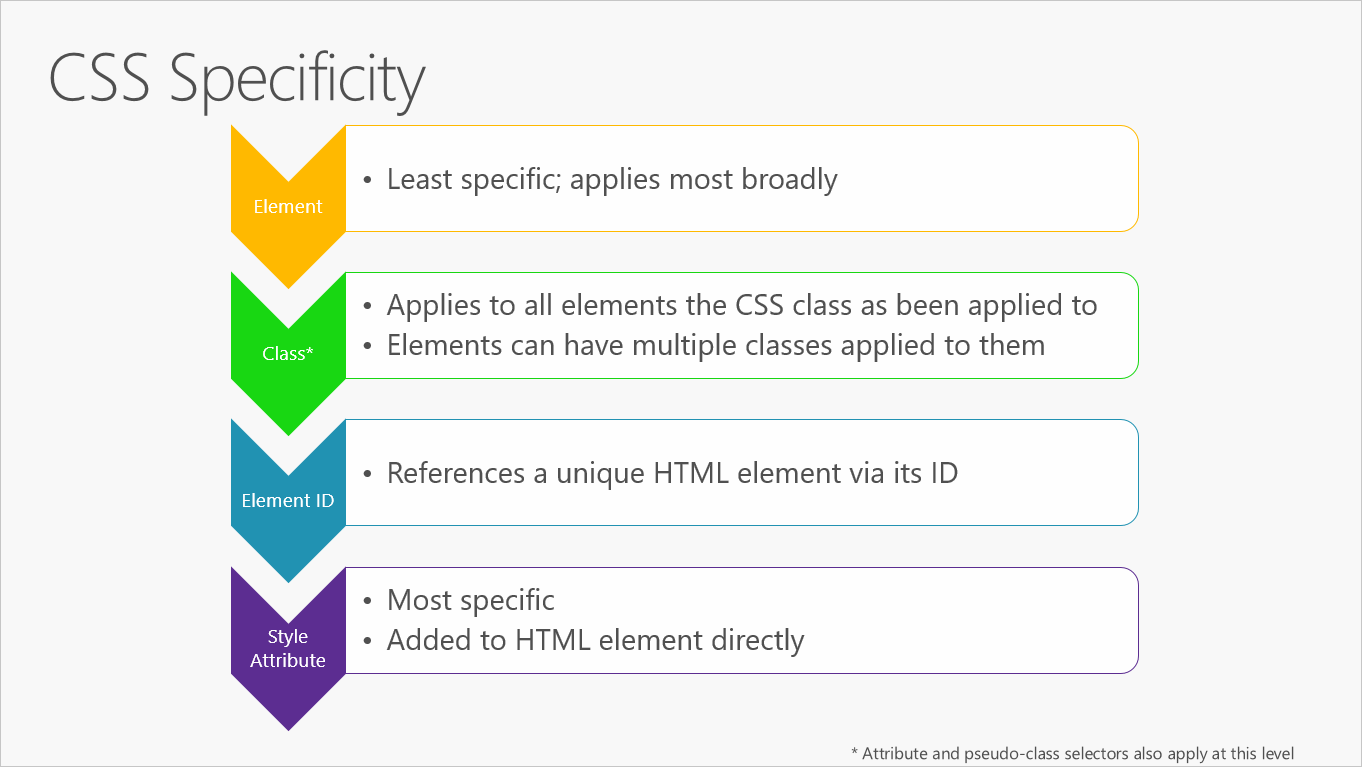


Imagen 2.1

**JS**:

Por último, la herramienta para la creación de los scripts de la página. JavaScript es un lenguaje interpretado y dinámico que se ha estandarizado como el lenguaje de programación de la web y su uso más común está en el empleo de eventos que modifiquen la vista de la página o eventos que permitan interactuar con el usuario sin necesidad de reclamar datos a un servidor.

Cuando se trabaja con este lenguaje de programación, lo más común es realizar tareas como:

* Seleccionar un elemento HTML y recuperar o actualizar su valor.
* Consultar datos en una API web.
* Enviar un comando a una API web (y responder a una devolución de llamada con su resultado).
* Realizar la validación.

**Otras tecnologías**:

Además de las tres ya mencionadas, en el desarrollo web existen una inmensa cantidad de herramientas utilizadas por los equipos de programación para, ya no solo facilitar el desarrollo, sino también para realizar funciones muy complejas sin necesidad de crearlas desde cero.

Herramientas muy conocidas serían los frameworks de CSS (SaSS, BootStrap, Tailwind,…) y frameworks de JS (Vue, Svelte, React, Angular,…). También existen otras herramientas como son **Ajax** y **JQeury**, pero las cuales no se van a definir, sino que en el apartado de los anexos y la bibliografía se dejaran ciertos documentos y blogs que explayen la descripción de cada una de estas herramientas. Otra cosa que cabe mencionar es el hecho del conocimiento base que se ha de tener de antemano de las principales tecnologías descritas para utilizar cualquiera de estas otras tecnologías, puesto que la base son estas y el resto no dejan de ser complementos que el día de mañana pueden quedarse obsoletos por la falta de mantenimiento, por ello, se ha de priorizar el aprendizaje en estas tres tecnologías antes que en cualquier otra por muy atractivas y sencillas que puedan parecer de aprender.

# Herramientas de desarrollo

Las herramientas de desarrollo que son más comunes dentro de la programación, ya sea fullstack, frontend o backend, suele ser un editor de código como lo es VSCode, Eclipse o netBeans. En mi caso personal, suelo preferir el VSCode por comodidad, familiaridad y por una vasta cantidad de extensiones que tienes disponible dentro de este editor.

Ahora si tuviésemos que concretar alguna que otra herramienta dentro de la programación fullStack tenemos ejemplos como lo son NPM y nodeJS, también esta BABEL y los diferentes frameworks de desarrollo como lo son Vue o React (solo se describirán los dos frameworks ya mencionados, hay muchos más)

Por defecto, el administrador de paquetes del lenguaje de programación de JavaScript es NPM. Este administrador te ayuda a encontrar cualquier paquete que se pueda necesitar en el programa que se desarrolle. A su vez, nodeJs sería el entorno de desarrollo que permite al programador escribir aplicaciones web desde el lado del servidor.

La otra herramienta mencionada era Babel. Esta, aunque en ciertos lugares puedas encotrar que se la menciona como un compilador, en este caso sería más un transpilador, porque, mientras que en la compilación se traduce el lenguaje de alto nivel a uno más bajo que pueda comprender la máquina, en la transpilación se plantea su uso para traducir el código a una versión más antigua para que otros navegadores que no soporten las versiones modernas, puedan seguir ejecutando los scripts con esas versiones antiguas.

Ahora pasaremos a hablar de los diferentes frameworks que hay para JavaScript:

# Framework Frontend

Los framework más conocidos para el lenguaje de programación de JavaScript son los siguientes:

## React.js

Más conocida por ser una librería, es empleada por Facebook e Instagram y su desarrollo en un principio estuvo mantenido por Meta (Facebook) y utilizan la capacidad interactiva que proporciona esta librería, siendo una elección robusta y muy escalable.

Además tiene la capacidad de traer elementos reutilizables. Esto permite a los desarrolladores reutilizar los componentes en varias aplicaciones. Por último, destacar que ofrece el diseño Material-UI, inspirándose en el diseño de materiales de Google.

## Angular

Angular fue lanzado en 2009 por Google y es uno de los frameworks open source más avanzados. La vinculación **bidireccional** es la característica innovadora más intuitiva de Angular que permite la actualización en tiempo real de un sitio web desde diferentes dispositivos. Esto último quiere decir que, tanto en móvil como de web, se cambia de forma simultánea el desarrollo de la vista.

Por si fuera poco, la curva de aprendizaje es muy dada a facilitar su aprendizaje y posee un gran cariño por los ingenieros de software en el mundo entero.

## Vue.js

Framework que destaca por la sencillez de su API, y, sin embargo, los elementos dinámicos de interfaces web que se desarrollan con **Vue** son muy sofisticadas. La clave reside en la facilidad de uso.

También utiliza el paradigma de diseño **MVC** (como un ejemplo más cercano Laravel) que permite simplicidad en el diseño. Su principal característica es la libertad de utilizar módulos selectivos para necesidades particulares de desarrollo web.

Por si esto fuese poco, como otros frameworks de los que hemos hablado, utiliza enlace de datos bidireccional (mencionado anteriormente con Angular).

Además de los tres mencionados, hay que tener en cuenta que existen muchos otros, pero de un menor calibre o dimensiones, como pueden ser **Svelte.js**, **Ember.js**, **Backbone**.

Aunque se suele mencionar mucho con todos los framework, una característica que muchos de ellos mencionan es su facilidad por aprender dicha herramienta. Cosa que yo no desmiento, pero que no me demuestra ser una característica de mención, pues la facilidad de aprenderlo dependerá única y exclusivamente de la experiencia y el aprendizaje previo del programador.

# Bibliografía:

## Apartado 1º:

**Evotic** - Ciclo de vida del Software

<https://evotic.es/software-a-medida/ciclo-de-vida-del-software/>

## 

## Apartado 2º:

**Microsoft Learn** - Tecnologías web comunes del lado cliente

<https://learn.microsoft.com/es-es/dotnet/architecture/modern-web-apps-azure/common-client-side-web-technologies>

**manz.dev LenguajeJS** - Ajax: Peticiones HTTP

<https://lenguajejs.com/javascript/peticiones-http/ajax/>

**Jquery**:

<https://jquery.com/>

## Apartado 3º:

**freeCodeCamp** - Herramientas populares para el desarrollo del front end que deberías conocer

<https://www.freecodecamp.org/espanol/news/herramientas-populares-para-el-desarrollo-del-front-end/>

## 

## Apartado 4º:

**OpenWebinars** - Los 6 mejores frameworks de JavaScript

<https://openwebinars.net/blog/los-6-mejores-frameworks-javascript/>

**developer.mozilla** - Entendiendo los frameworks de JavaScript del lado cliente

<https://developer.mozilla.org/es/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks>