Nivelación Html, Css, JS

React Js es una Biblioteca para desarrollo web, por lo tanto, debemos contar con conocimientos mínimos sobre los lenguajes que el navegador web interpreta (HTM, CSS, Js).

HTML:

HTML es un lenguaje de etiquetas, con el cual podremos armar la estructura para nuestras páginas web.

```
<!DOCTYPE html>
<html lang="es">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Curso React Codo a codo</title>
</head>

<body>

    <h1>Hola clase 1 😁</h1>

    <h2>hola esto es un subtitulo</h2>

</body>

</html>
```

HTML5:

Todo lo que se ve en Internet está programado con un código interno, y cuando ingresas a una web, a tu navegador le llega este código, y lo traduce de forma visual para que veas lo que el creador de la web ha diseñado que puedas ver. Este código es el que decide la estructura de una página web, colocando sus diferentes elementos en los puntos correspondientes, y un fallo dentro del mismo hará que no se vea bien.

Estos códigos se les llama lenguajes de programación, y el lenguaje que se utiliza en la World Wide Web, el ecosistema de páginas de Internet, es el HTML. Su nombre son las siglas de HyperText Markup Language, que significa literalmente Lenguaje de marcado de hipertexto. Es el estándar con el que están programadas todas las webs.

Pero Internet evoluciona, y el contenido que se sube a las webs también cambia con los años, lo que quiere decir que estos lenguajes pueden quedar obsoletos y necesitan ser actualizados. En 1999 se lanzó el estándar HTML4, y como imaginas, las páginas web ahora no

tienen nada que ver con lo que eran entonces en cuanto a sus contenidos, y por eso desde hace unos años se está implementando el nuevo estándar HTML5.

Por lo tanto, el HTML5 es la última versión del estándar HTML que se utiliza para crear las páginas web que estás visitando, e incorpora algunas novedades interesantes. Una de las notables, es darle cobertura a la reproducción de contenido multimedia, de forma que ya no tengas que ir a recursos de terceros como el obsoleto Flash Player.

¿Qué cambia en HTML5?

Este nuevo estándar no tiene una cantidad asombrosa de novedades, pero las que tiene son absolutamente revolucionarias. La más destacada es la posibilidad de añadir archivos multimedia a la web, como vídeos o audios, y que estos no tengas que insertarlos utilizando otros plug-ins. Todo está incluido dentro del código.

También se han añadido etiquetas que permiten crear animaciones en 2D, con una etiqueta de canvas y una API que permiten que puedas dibujar elementos en dos dimensiones y animarlos en la web. También se pueden añadir eventos para el teclado, ratón o mandos, que permiten poder utilizarlos para interactuar con una página.

Relacionado con lo anterior, también se pueden programar aplicaciones web en HTML5, lo que quiere decir que las páginas pueden ser apps, y no necesitarás instalar una app independiente en el PC o móvil, ya que podrás usarlo igual desde el navegador. También se pueden crear videojuegos con este método, lo que se complementa con poder utilizar teclado, ratón o mando.

También se han añadido opciones de geolocalización, de forma que una web puede detectar la ubicación de los usuarios que acceden a ella. Con ello, se pueden ofrecer opciones de idiomas dependiendo del país desde el que entras, o la posibilidad de derivarte a la página específica de un país.

Metadatos del Documento

Los Metadatos son elementos HTML que muestran información sobre la propia página web que los contiene.

Son usados por los buscadores para definir la información principal de nuestra web (temática, descripción), por lo que será muy importante que lo tengamos correctamente configurados.

Estos elementos meta se encontrarán en el head del código de la web.

<title>: define el título del documento, el cual se muestra en la barra de título del navegador o en las pestañas de página.

<link>: utilizada para enlazar JavaScript y CSS externos.

<meta>: con esta etiqueta definimos la codificación que tendrá nuestro archivo, los mismos pueden ser:

UTF-8

ANSI: es el formato estándar de codificación de archivos utilizados en el Bloc de notas.

¿Qué son los lenguajes de scripting?

Los lenguajes de scripting son una popular familia de lenguajes de programación que se pueden utilizar para satisfacer rápidamente las exigencias más comunes. Los lenguajes de scripting antiguos se utilizaban más bien para aplicaciones muy concretas o como lenguajes pegamento, es decir, para pegar sistemas ya existentes. Con la aparición de la World Wide Web, se establecieron una serie de lenguajes de scripting para la utilización en servidores web. Puesto que los lenguajes de scripting simplifican el procesamiento de texto, son perfectos para la creación dinámica de páginas HTML.

Hoy en día, los lenguajes de scripting suponen un tercio de los lenguajes de programación más utilizados en todo el mundo. JavaScript es prácticamente el único lenguaje de scripting que se ejecuta en el navegador en el lado del cliente, pero también los lenguajes del lado del servidor PHP, Python, Ruby y Perl son lenguajes de scripting.

<script>: define tanto un script interno como un enlace hacia un script externo. El lenguaje de programación es JavaScript

CSS

CSS son las siglas en inglés para «hojas de estilo en cascada» (Cascading Style Sheets). Básicamente, es un lenguaje que maneja el diseño y presentación de las páginas web, es decir, cómo lucen cuando un usuario las visita. Funciona junto con el lenguaje HTML que se encarga del contenido básico de las páginas.

Se les denomina hojas de estilo «en cascada» porque puedes tener varias hojas y una de ellas con las propiedades heredadas (o «en cascada») de otras.

Para muchas personas una simple plantilla de blog es suficiente. Sin embargo, cuando quieras personalizar la apariencia de un sitio

necesitarás implementar CSS que, en conjunto con un buen CMS, te ayudará a potenciar el alcance de tu contenido.

¿Para qué sirve CSS?

Con CSS puedes crear reglas para decirle a tu sitio web cómo quieres mostrar la información y puedes guardar los comandos para elementos de estilo (como fuentes, colores, tamaños, etc.) separados de los que configuran el contenido.

A través de esta herramienta puedes crear formatos específicos útiles para comunicar tus ideas y crear experiencias agradables para los usuarios del sitio web.

Diferencia entre HTML y CSS

HTML es un lenguaje de programación utilizado para dar estructura al contenido de un sitio web. Sus siglas en inglés significan «lenguaje de marcas de hipertexto» (HyperText Markup Language), y hacen referencia al código que define el significado de las instrucciones dadas a una plataforma computacional.

Estas instrucciones representan todos los enlaces (o hipertextos) que vinculan los contenidos de un sitio, por lo que HTML es la base de cualquier página web. En este lenguaje es posible incluir toda la información referente al contenido un sitio, así como las imágenes, audios y estilos del mismo; sin embargo, su uso para estas tareas conlleva una mayor complejidad en el código fuente.

Para hacer más eficiente el uso de HTML, se han diseñado otros lenguajes computacionales para facilitar la gestión de los datos relacionados con el diseño visual de las plataformas. CSS es uno de los lenguajes más importantes utilizados para ordenar las instrucciones referentes a la apariencia de un sitio y presentar los contenidos de una página de forma atractiva.

De este modo, HTML se emplea para estructurar el contenido de un sitio, mientras que CSS se usa para estructurar su presentación.

DOM

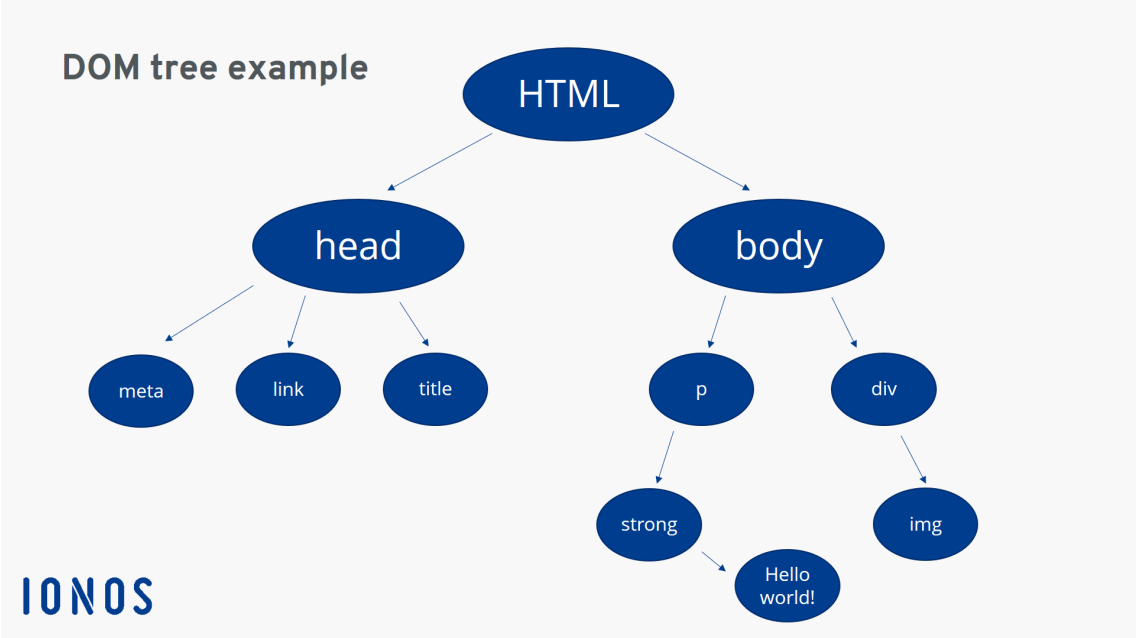
DOM es una abreviatura de Document Object Model. En español podríamos traducirlo por Modelo de Objeto de Documento, aunque en a veces nos hemos referido al DOM habitualmente con el nombre de jerarquía de objetos del navegador, porque realmente es una estructura jerárquica donde existen varios objetos y unos dependen de otros.

Los objetos del DOM modelizan tanto la ventana del navegador como el historial, el documento o página web, y todos los elementos que pueda tener dentro la propia página, como párrafos, divisiones, tablas, formularios y sus campos, etc. A través del DOM se puede acceder, por medio de Javascript, a cualquiera de estos elementos, es decir a sus correspondientes objetos para alterar sus propiedades o invocar a sus métodos. Con todo, a través del DOM, queda disponible para los programadores de Javascript, cualquier elemento de la página, para modificarlos, suprimirlos, crear nuevos elementos y colocarlos en la página, etc.

El DOM está definido y administrado por el W3C, por lo que los distintos navegadores simplemente aplican las especificaciones del World Wide Web Consortium, para dar soporte al DOM en sus aplicaciones. El DOM no sólo permite modificar páginas web en HTML, sino también documentos XML.

A lo largo de la historia de los navegadores, se han ido aplicando en mayor o menor manera las características del DOM. A medida que se sucedían versiones de los navegadores también se iba dando un mayor soporte a las especificaciones del DOM, en lo que se han llamado los niveles del DOM. El primero que empezó a disponibilizar por medio de objetos los componentes de la página fue Netscape 2.0, que incorporaba lo que se llama el DOM nivel 0. Actualmente, la última especificación publicada es DOM nivel 4.

Es importante destacar ahora que, dado que los niveles del DOM cambian de versión a versión del navegador y que las especificaciones se han entendido de manera distinta por las distintas organizaciones creadoras de los navegadores, se ha producido un marco donde trabajar con los objetos de la página difiere de un navegador a otro.


DOM tree example
HTML
head
body
meta
link
title
p
div
strong
img
Hello world!
IONOS

Funcionamiento React

¿Cómo llega React a la performance que tanta fama le trae?

React.js está construido en torno a hacer funciones, que toman las actualizaciones de estado de la página y que se traduzcan en una representación virtual de la página resultante. Siempre que React es informado de un cambio de estado, vuelve a ejecutar esas funciones para determinar una nueva representación virtual de la página, a continuación, se traduce automáticamente ese resultado en los cambios del DOM necesarios para reflejar la nueva presentación de la página.

A primera vista, esto suena como que fuera más lento que el enfoque JavaScript habitual de actualización de cada elemento, según sea necesario. Detrás de escena, sin embargo, React.js hace justamente eso: tiene un algoritmo muy eficiente para determinar las diferencias entre la representación virtual de la página actual y la nueva. A partir de esas diferencias, hace el conjunto mínimo de cambios necesarios en el DOM.

Pues utiliza un concepto llamado el DOM virtual que hace selectivamente sub-árboles de los nodos sobre la base de cambios de estado, desarrollando esto, con la menor cantidad de manipulación DOM posible, con el fin de mantener los componentes actualizados, estructurando sus datos.

¿Cómo funciona el DOM virtual?

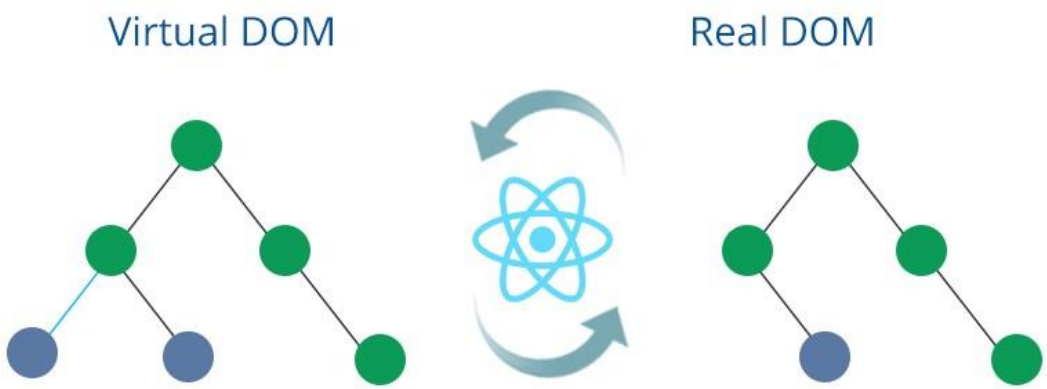
Imagina que tienes un objeto que es un modelo en torno a una persona. Tienes todas las propiedades relevantes de una persona que podría tener, y refleja el estado actual de la persona. Esto es básicamente lo que React hace con el DOM.

Ahora piensa, si tomamos ese objeto y le hacemos algunos cambios. Se ha añadido un bigote, unos bíceps y otros cambios. En React, cuando aplicamos estos cambios, dos cosas ocurren:

– En primer lugar, React ejecuta un algoritmo de “diffing”, que identifica lo que ha cambiado.

– El segundo paso es la reconciliación, donde se actualiza el DOM con los resultados de diff.

La que hace React, ante estos cambios, en lugar de tomar a la persona real y reconstruirla desde cero, sólo cambiaría la cara y los brazos. Esto significa que si usted tenía el texto en una entrada y una actualización se llevó a cabo, siempre y cuando nodo padre de la entrada no estaba programado para la actualización, el texto se quedaría sin ser cambiado.



Node JS

Node.js, es un entorno en tiempo de ejecución multiplataforma para la capa del servidor (en el lado del servidor) basado en JavaScript.

Node.js es un entorno controlado por eventos diseñado para crear aplicaciones escalables, permitiéndote establecer y gestionar múltiples conexiones al mismo tiempo. Gracias a esta característica, no tienes que preocuparte con el bloqueo de procesos, debido a que no hay bloqueos.

Características principales de Node.js

Node.js se ha hecho popular en los últimos años gracias a las siguientes características:

Velocidad: Node.js está construido sobre el motor de JavaScript V8 de Google Chrome, por eso su biblioteca es muy rápida en la ejecución de código.

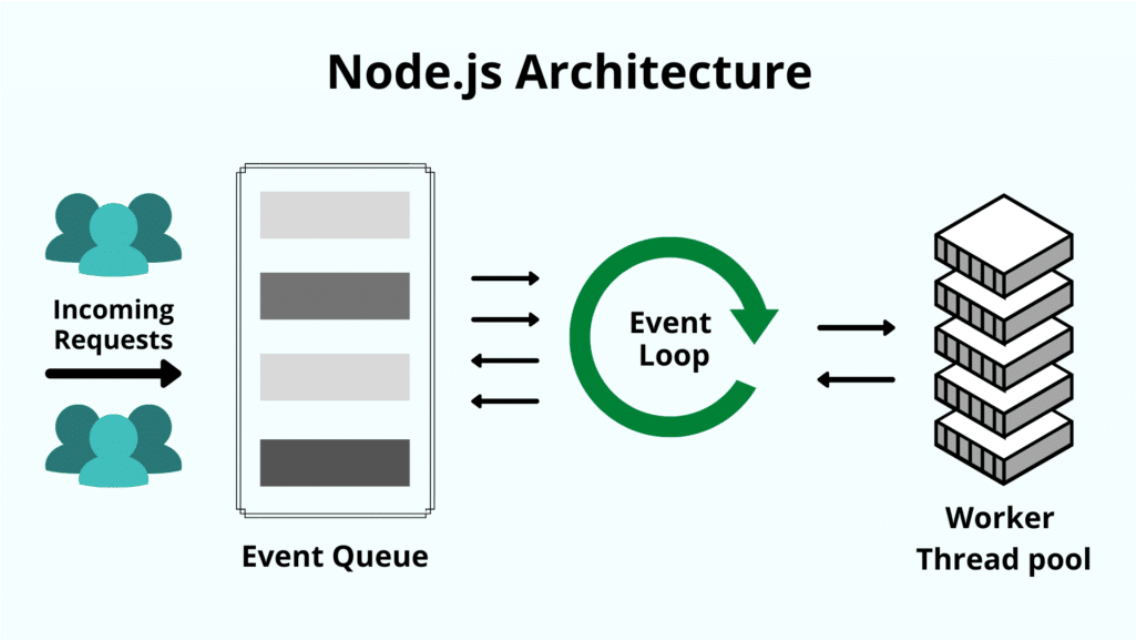
Sin búfer: Las aplicaciones de Node.js generan los datos en trozos (chunks), nunca los almacenan en búfer.

Asíncrono y controlado por eventos: Como hemos dicho anteriormente, las APIs de la biblioteca de Node.js son asíncronas, sin bloqueo. Un servidor basado en Node.js no espera que una API devuelva datos. El servidor pasa a la siguiente API después de llamarla, y un mecanismo de notificación de eventos ayuda al servidor a obtener una respuesta de la llamada a la API anterior.

Arquitectura de Node.js y su funcionamiento

Node.js utiliza la arquitectura «Single Threaded Event Loop» para manejar múltiples clientes al mismo tiempo. Para entender en qué se diferencia de otros tiempos de ejecución, tenemos que entender cómo se manejan los clientes concurrentes multihilo en lenguajes como Java.

En un modelo de solicitud-respuesta multihilo, varios clientes envían una solicitud y el servidor procesa cada una de ellas antes de devolver la respuesta. Sin embargo, se utilizan múltiples hilos para procesar las llamadas concurrentes. Estos hilos se definen en un pool de hilos, y cada vez que llega una petición, se asigna un hilo individual para manejarla.



¿Es Node.js un lenguaje de programación?

Node.js no es un lenguaje de programación. Más bien, es un entorno de ejecución que se utiliza para ejecutar JavaScript fuera del navegador.

Node.js tampoco es un framework (una plataforma para desarrollar aplicaciones de software). El tiempo de ejecución de Node.js se construye sobre un lenguaje de programación -en este caso, JavaScript- y ayuda a la ejecución de los propios frameworks.

En resumen, Node.js no es un lenguaje de programación ni un marco de trabajo; es un entorno para ellos.

Visual Studio Code

Visual Studio Code. Es un editor de código fuente desarrollado por Microsoft para Windows , Linux y macOS . Incluye soporte para depuración , control de Git integrado, resaltado de sintaxis , finalización de código inteligente , fragmentos de código y refactorización de código . También es personalizable, de modo que los usuarios pueden cambiar el tema del editor, los métodos abreviados de teclado y las preferencias. Es gratuito y de código abierto.

Visual Studio Code fue anunciado el 29 de abril de 2015 por Microsoft en la conferencia Build de 2015. Una versión preliminar fue lanzada poco después. El 18 de noviembre de 2015, Visual Studio Code fue lanzado bajo la licencia MIT y su código fuente fue publicado en GitHub . También se anunció el soporte de extensión. El 14 de abril de 2016, Visual Studio Code graduó la etapa de vista previa pública y se lanzó a la web.

El código combina la interfaz de usuario optimizada de un editor moderno con asistencia y navegación de código enriquecido y una experiencia de depuración integrada, sin la necesidad de un IDE completo. Visual Studio Code, cuenta con herramientas de Debug hasta opciones para actualización en tiempo real de nuestro código en la vista del navegador y compilación en vivo de los lenguajes que lo requieran (por ejemplo en el caso de SASS a CSS). Además de las extensiones, tendremos la posibilidad de optar por otros themes o bien configurarlo a nuestro gusto. Para modificar el esquema de colores y los iconos

Se puede utilizar como lenguajes de programación.

Visual Studio Code es una herramienta que tiene soporte nativo para gran variedad de lenguajes, entre ellos podemos destacar los principales del desarrollo Web: HTML, CSS, y JavaScript, entre otros.

Posibilidad de configurar la interfaz a nuestro gusto. De esta forma, podremos tener más de un código visible al mismo tiempo, las carpetas de nuestro proyecto y también acceso a la terminal o un detalle de problemas, entre otras posibilidades.

Existencia de una amplísima gama de temas o estilos visuales para Visual Studio Code, que hacen el trabajo con el software más agradable a la vista.

Goza de un soporte técnico formidable pues debido a su frecuente uso por la comunidad de desarrolladores, se puede encontrar fácilmente documentación y ayuda en foros y sitios relacionados.

GIT

Los sistemas de control de versiones son programas que tienen como objetivo controlar los cambios en el desarrollo de cualquier tipo de software, permitiendo conocer el estado actual de un proyecto, los cambios que se le han realizado a cualquiera de sus piezas, las personas que intervinieron en ellos, etc.

Para conocer uno de los sistemas de control de versiones existentes que en la actualidad se ha popularizado hasta convertirse casi en un standard, gracias al sitio Github. Se trata de Git, el sistema de control de versiones más conocido y usado actualmente, que es el motor de Github. Al terminar esta lectura entenderás que es Git y qué es Github, dos cosas distintas que a veces resultan confusas de entender por las personas que están dando sus primeros pasos en el mundo del desarrollo.

Necesidad de un control de versiones

El control de versiones es una de las tareas fundamentales para la administración de un proyecto de desarrollo de software en general. Surge de la necesidad de mantener y llevar control del código que vamos programando, conservando sus distintos estados. Es absolutamente necesario para el trabajo en equipo, pero resulta útil incluso a desarrolladores independientes.

Aunque trabajemos solos, sabemos más o menos cómo surge la necesidad de gestionar los cambio entre distintas versiones de un mismo código. Prueba de ello es que todos los programadores, más tarde o más temprano, se han visto en la necesidad de tener dos o más copias de un mismo archivo, para no perder su estado anterior cuando vamos a introducir diversos cambios. Para ir solucionando nuestro día a día habremos copiado un fichero, agregándole la fecha o un sufijo como "antiguo". Aunque quizás esta acción nos sirva para salir del paso, no es lo más cómodo ni mucho menos lo más práctico.

En cuanto a equipos de trabajo se refiere, todavía se hace más necesario disponer de un control de versiones. Seguro que la mayoría hemos experimentado las limitaciones y problemas en el flujo de trabajo cuando no se dispone de una herramienta como Git: marcar los cambios en archivos hechos por otros componentes del equipo, incapacidad de comparar de manera rápida dos códigos para saber los cambios que se introdujeron al pasar de uno a otro, etc.

Además, en todo proyecto surge la necesidad de trabajar en distintas ramas al mismo tiempo, introduciendo cambios a los programas, tanto en la rama de desarrollo como la que tenemos en producción. Teóricamente, las nuevas funcionalidades de tu aplicación las programarás dentro de la rama de desarrollo, pero constantemente tienes que estar resolviendo bugs, tanto en la rama de producción como en la de desarrollo.

Para aclarar, un sistema en producción se refiere a que está disponible para los usuarios finales. O sea, es una versión que está ya puesto en marcha y por lo tanto debería funcionar correctamente. Un sitio web, cuando lo estás creando, está en su etapa de desarrollo y cuando lo liberas en el dominio definitivo y lo pones a disposición de tu cliente, y/o los usuarios de Internet en general, se dice que está en producción.

Para facilitarnos el trabajo existen sistemas como Git, Subversion, CVS, etc. que sirven para controlar las versiones de un software y que deberían ser una obligatoriedad en cualquier desarrollo. Nos ayudan en muchos ámbitos fundamentales, como podrían ser:

- Comparar el código de un archivo, de modo que podamos ver las diferencias entre versiones

- Restaurar versiones antiguas
- Fusionar cambios entre distintas versiones
- Trabajar con distintas ramas de un proyecto, por ejemplo la de producción y desarrollo

En definitiva, con estos sistemas podemos crear y mantener repositorios de software que conservan todos los estados por el que va pasando la aplicación a lo largo del desarrollo del proyecto. Almacenan también las personas que enviaron los cambios, las ramas de desarrollo que fueron actualizadas o fusionadas, etc. Todo este mundo de utilidades para llevar el control del software resulta complejo en un principio, pero veremos que, a pesar de la complejidad, con Git podremos manejar los procesos de una manera bastante simple.

Alternativas y variantes de sistemas de control de versiones
Comenzaron a aparecer los sistemas de control del versionado del software allá por los años setenta, aunque al principio eran bastante elementales. Para hacerse una idea, en los primeros sistemas existía una restricción por la que sólo una persona podía estar a la vez tocando el mismo código. Es posible imaginarse que cosas semejantes provocaban retraso en los equipos de trabajo, por ello, a lo largo de los años fueron surgiendo nuevos sistemas de control de versiones, siempre evolucionando con el objetivo de resolver las necesidades de los equipos de desarrollo.

Existen principalmente dos tipos de variantes:

Sistemas centralizados: En estos sistemas hay un servidor que mantiene el repositorio y en el que cada programador mantiene en local únicamente aquellos archivos con los que está trabajando en un momento dado y se utiliza cuando se necesita conectar con el servidor donde está el código para poder trabajar y enviar cambios en el software que se está programando. Ese sistema centralizado es el único lugar donde está todo el código del proyecto de manera completa. Subversion o CVS son sistemas de control de versiones centralizados.

Sistemas distribuidos: En este tipo de sistemas cada uno de los integrantes del equipo mantiene una copia local del repositorio completo. Al disponer de un repositorio local, puedo hacer commit (enviar cambios al sistema de control de versiones) en local, sin necesidad de estar conectado a Internet o cualquier otra red. En cualquier momento y en cualquier sitio donde esté puedo hacer un commit. Es cierto que es local de momento, luego podrás compartirlo con otras personas, pero el hecho de tener un repositorio completo me facilita ser autónomo y poder trabajar en cualquier situación. Git es un sistema de control de versiones distribuido.

Git es un sistema de control de versiones distribuido. Con Git hacemos repositorios de software. GitHub es un servicio para hacer hosting de repositorios de software que se administra con Git. Digamos que en GitHub mantienes una copia de tus repositorios en la nube, que además puedes hacer disponible para otros desarrolladores.

Tanto sistemas distribuidos como centralizados tienen ventajas e inconvenientes comparativas entre los unos y los otros. Para no hacer muy larga la introducción, cabe mencionar que los sistemas un poco más antiguos como CVS o también Subversion, por sus características son un poco más lentos y pesados para la máquina que hace de servidor central. En los sistemas distribuidos no es necesario que exista un servidor central donde enviar los cambios, pero en caso de existir se requiere menor capacidad de procesamiento y gestión, ya que muchas de las operaciones para la gestión de versiones se hacen en local.

Es cierto que los sistemas de control de versiones distribuidos están más optimizados, principalmente debido al ser sistemas concebidos hace menos tiempo, pero no todo son ventajas.

Los sistemas centralizados permiten definir un número de versión en cada una de las etapas de un proyecto, mientras que en los distribuidos cada repositorio local podría tener diferentes números de versión. También en los centralizados existe un mayor control del desarrollo por parte del equipo.

De todos modos, en términos comparativos nos podemos quedar con la mayor ventaja de los sistemas distribuidos frente a los sistemas centralizados: La posibilidad de trabajar en cualquier momento y lugar, gracias a que siempre se mandan cambios al sistema de versionado en local, permite la autonomía en el desarrollo de cada uno de los componentes del equipo y la posibilidad de continuar trabajando aunque el servidor central de versiones del software se haya caído.

Sobre Git

Como ya hemos dicho, Git es un sistema de control de versiones distribuido. Git fue impulsado por Linus Torvalds y el equipo de desarrollo del Kernel de Linux. Ellos estaban usando otro sistema de control de versiones de código abierto, que ya por aquel entonces era distribuido. Todo iba bien hasta que los gestores de aquel sistema de control de versiones lo convirtieron en un software propietario. Lógicamente, no era compatible estar construyendo un sistema de código abierto, tan representativo como el núcleo de Linux, y estar pagando por usar un sistema de control de versiones propietario. Por ello, el mismo equipo de desarrollo del Kernel de Linux se tomó la tarea de construir desde cero un sistema de versionado de software, también distribuido, que aportase lo mejor de los sistemas existentes hasta el momento.

Así nació Git, un sistema de control de versiones de código abierto, relativamente nuevo que nos ofrece las mejores características en la actualidad, pero sin perder la sencillez y que a partir de entonces no ha parado de crecer y de ser usado por más desarrolladores en el mundo. A los programadores nos ha ayudado a ser más eficientes en nuestro trabajo, ya que ha universalizado las herramientas de control de versiones del software que hasta entonces no estaban tan popularizadas y tan al alcance del común de los desarrolladores.

Git es multiplataforma, por lo que puedes usarlo y crear repositorios locales en todos los sistemas operativos más comunes, Windows, Linux o Mac. Existen multitud de GUIs (Graphical User Interface o Interfaz de Usuario Gráfica) para trabajar con Git en ventanas gráficas, no obstante para el aprendizaje se recomienda usarlo con línea de comandos, de modo que puedas dominar el sistema desde su base, en lugar de estar aprendiendo a usar un programa determinado.

Para usar Git debes instalarlo en tu sistema. Hay unas instrucciones distintas dependiendo de tu sistema operativo, pero en realidad es muy sencillo. La página oficial de descargas está en gitscm.com, en donde encontrarás para descarga lo que se llama la "Git Bash" es decir, la interfaz de Git por línea de comandos. Tienes que descargar aquella versión adecuada para tu sistema (o si estás en Linux instalarla desde los repositorios dependiendo de tu distribución). Aparte de Windows, Linux o Mac, también hay versión para Solaris. La instalación es muy sencilla, lo que resulta un poco más complejo al principio es el uso de Git.

Conceptos rápidos sobre Git y GitHub
Antes de estudiar a fondo Git y comenzar a practicar, hay que cubrir algunas cuestiones que existen cuando se comienza a trabajar con un sistema de control de versiones. Se trata de algunas claves rápidas sobre las características de Git, el flujo de trabajo y algunas diferencias con GitHub.

En Git, cada desarrollador dispone de un repositorio completo instalado en su máquina; es algo inherente a los sistemas de control de versiones distribuidos, como vimos en el artículo sobre Qué son Git y GitHub. Es condición indispensable. Todos los cambios de nuestros archivos a lo largo del desarrollo los vamos a tener en local. Opcionalmente, esos cambios los enviaremos a repositorios remotos, como puede ser GitHub o cualquier otro.

Esto quiere decir que mi máquina tendrá todo lo que necesito para trabajar. Tendremos una copia del repositorio completo, cada uno de los archivos de todo el proyecto. Con el repositorio Git en local, luego decidiré a qué otros servidores o máquinas mando mis cambios.

GitHub es un hosting de repositorios Git, por tanto, el uso que le daremos es como repositorio remoto. Pero debe quedar claro que primero debo tener los cambios en el repositorio local y luego podré "empujar" esos cambios al repositorio remoto.

Por tanto, ya se ve la primera diferencia entre Git y GitHub: Git es la tecnología para hacer el control de versiones y GitHub simplemente es un hosting de repositorios Git, con una interfaz web que nos ofrece algunas utilidades basadas en el propio control de versiones Git.

En GitHub puedo tener repositorios diversos y si quiero trabajar con alguno de ellos, primero debo tenerlo en local. No necesitamos tener todos los repositorios que has publicado en GitHub en local, solo aquellos con los que vayamos a trabajar. En el momento que los tenemos en local podremos hacer cambios, almacenarlos en nuestro repositorio local y cuando lo juzguemos oportuno, enviarlo (empujar, push) a tantos servidores o repositorios remotos como queramos.

Nota: también al referirnos a GitHub hablamos como un repositorio remoto en general. Es decir, realmente lo que describimos de GitHub en líneas generales sirve para entender otros servicios de hosting de repositorios Git como Bitbucket.

Para concluir y tener más claro el flujo de trabajo con un control de versiones Git, imaginar un equipo de trabajo con varios componentes. Todos y cada uno de los desarrolladores deberán tener una copia completa de todo el repositorio de software que se está desarrollando. Luego el equipo decidirá a qué servidor con un repositorio remoto quiere enviar los cambios.

Cada desarrollador podrá enviar los cambios que tiene en local hacia el repositorio remoto y ese repositorio remoto lo usarán todos los componentes del equipo para sincronizarse y tener la versión más nueva del código en cada momento que lo deseen.

Este esquema podemos considerarlo como la base del trabajo con Git, aunque luego en la práctica hay que resolver algunos temas importantes.

Instalar Git
Tener Git instalado en local es condición indispensable para trabajar con el sistema de control de versiones. El proceso para instalar Git es bien sencillo porque no difiere de la instalación de cualquier otro software que hayas hecho.

Te tienes que descargar la versión de tu sistema operativo en la página oficial (o si usas Linux lo bajarás de los repositorios de software que usas habitualmente en tu distribución).

gitscm.com

Lo instalas como cualquier otro software. Si estás en Windows tendrás un asistente al que harás "siguiente, siguiente" hasta acabar el proceso. Puedes ver este vídeo que aclara algunos puntos sobre la instalación.

Para usuarios Windows, ¿Git Bash?
El único sitio donde puedes tener dudas es en el paso que te dice si quieres instalarlo como comando en la línea de comandos de tu consola o si simplemente quieres el "git bash".

Si lo instalas en la propia consola del Windows, la única ventaja es que lo tendrás disponible desde la ventana de línea de comandos de Windows y podrás hacer desde allí los comandos de Git. Si lo instalas solo en Git Bash no habrá más problemas, solo que cuando quieras usar Git tendrás que abrir la consola específica de Git que ellos llaman "git bash".

La manera más sencilla de saber si Git está instalado en tu sistema es a través del comando:

git version

Esto te mostrará en la pantalla el número de la versión de Git que tengas instalada. Si en Windows instalaste Git en consola (la ventana de línea de comandos), observarás que en cualquier consola de comandos de Windows tienes disponible Git. Si lo instalaste únicamente en el Git Bash, no tendrás ese comando y Git no está disponible desde la consola de Windows.

Git Bash es la línea de comandos de Git para Windows, que además te permite lanzar comandos de Linux básicos, "ls -l" para listar archivos, "mkdir" para crear directorios, etc. Es la opción más común para usar Git en Windows.

Es indiferente si instalas Git en la línea de comandos del Windows o si lo instalas solamente en el Git Bash. Simplemente escoge la que te sea más cómoda.

Primera configuración de Git, primeros comandos que debes lanzar
Antes que nada, inmediatamente después de instalar Git, lo primero que deberías hacer es lanzar un par de comandos de configuración.

git config global user.name "Tu nombre aquí"
git config global user.email "tu_email_aquí@example.com"

Con estos comandos indicas tu nombre de usuario (usas tu nombre y apellidos generalmente) y el email. Esta configuración sirve para que cuando hagas commits en el repositorio local, éstos se almacenen con la referencia a ti mismo, meramente informativa. Gracias a ello, más adelante cuando obtengas información de los cambios realizados en el los archivos del "repo" local, te va a aparecer como responsable de esos cambios a este usuario y correo que has indicado.

Video proporcionado por el profesor Alejandro Zapata:
https://www.youtube.com/watch?v=ptXiQwE535s&t=0s