

Máster en Full Stack Developer

Precurso

Índice

Tema 1. Uso del terminal	3
1.1. Introducción y objetivos	3
1.2. ¿Qué es un terminal?	3
1.3. Comandos básicos de terminal, tanto en Windows como en Linux/Mac	5
1.4. Alternativas a la terminal	9
1.5. Otras herramientas que vamos a necesitar durante el curso	11
Tema 2. Uso de GIT control de versiones	17
2.1. Introducción y objetivos	17
2.2. ¿Por qué es necesario tener un control de versiones?	18
2.3. Instalar Git	19
2.4. ¿En qué nos ayuda Git?	20
2.5. Características principales de Git	21
2.6. Interfaz gráfica para el uso de Git	22
2.7. Repositorio remoto	23
2.8. ¿Cómo funciona Git?	25
A fondo	29

Tema 1. Uso del terminal

1.1. Introducción y objetivos

La **terminal de comandos** se ha convertido en uno de los principales aliados de un **programador**, no solo porque nos ayuda a movernos por nuestros proyectos de una forma un poco más ordenada y sin necesidad de apartar nuestras manos del teclado, sino también porque nos permite **instalar** cualquier tipo de **framework a librería** dentro de nuestros **proyectos**. Lenguajes como PHP, JAVA, JAVASCRIPT hacen uso del terminal en todo momento para la creación de proyectos, instalación de dependencias y creación de elementos de desarrollo.

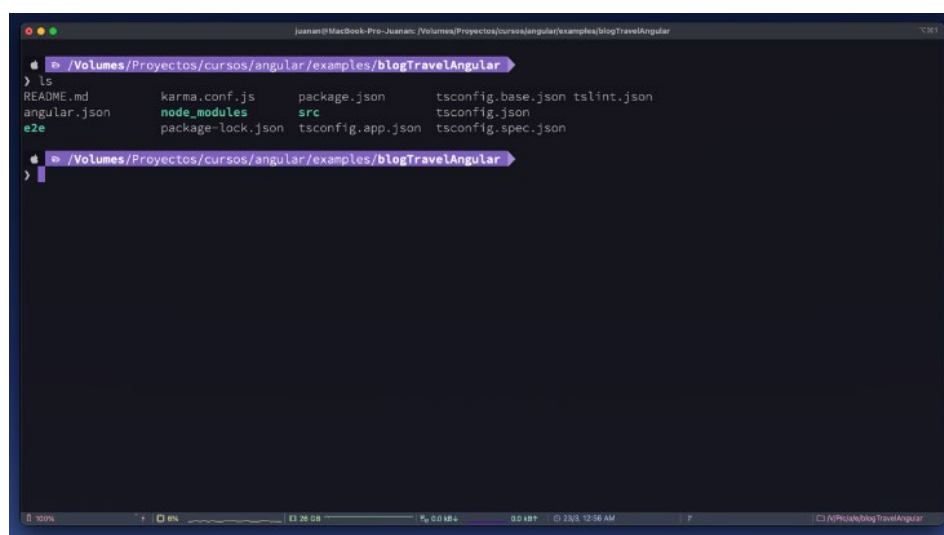
Desde hoy se va a convertir en tu nueva amiga si es que empiezas en el **mundo del desarrollo**, y este pequeño módulo te servirá para iniciarte en su uso y aprender lo básico para su manejo a esta herramienta fundamental para un desarrollador fullstack.

A lo largo de este módulo es fundamental que revises todo el material (texto, vídeos o enlaces), ya que te aportarán multitud de conocimientos que te serán muy útiles durante el curso.

1.2. ¿Qué es un terminal?

Un desarrollador tiene en la terminal un aliado que le ayudará en muchas de las tareas cotidianas, como instalar librerías, gestionar paquetes, gestión de dependencias, uso de control de versiones. Son multitud las tareas que un **desarrollador web** realiza a través de la terminal o CMD y perderle el miedo es vital para la profesión en la que estamos involucrados.

Existen muchas aplicaciones de las que un desarrollador utiliza en su día a día que no se encuentran en su **versión de escritorio**, si no que se ejecutan a través de una línea de comandos. El trabajo dentro de este tipo de interfaces es muy sencillo, el usuario lanza un comando y el ordenador nos responde dependiendo de la acción que vayamos a realizar.



```
juan@MacBook-Pro-Juanan: /Volumes/Proyectos/cursos/angular/examples/blogTravelAngular
> ls
README.md      karma.conf.js  package.json   tsconfig.base.json  tslint.json
angular.json    node_modules  src            tsconfig.json
e2e             package-lock.json  tsconfig.app.json  tsconfig.spec.json
```

Figura 1. Captura de la terminal. Fuente: elaboración propia.

Se pueden ejecutar **tareas sencillas**, pero también podemos accionar **complejos flujos de trabajo** que el **sistema operativo** no nos permite realizar a través de ventanas o botones predeterminados.

Muchos **frameworks de desarrollo**, como pueden ser Angular o React, incluyen una herramienta por la línea de comandos para poder **ejecutar sus acciones** más habituales. Comprender cómo funcionan estas herramientas agiliza y optimiza el día a día del programador.

Cada **sistema operativo** dispone de su propio terminal e incluso existen algunas distribuciones de Linux que únicamente permiten interactuar con ellas a través de la línea de comandos.

El uso de la terminal o línea de comandos difiere ligeramente si estas en entorno UNIX (Linux/Mac) o Windows, por eso vamos a ponerte una serie de **comandos básicos** para que te manejes dentro de la terminal y le pierdas el miedo a interactuar con ella. Te animo a que entres sin miedo y pruebes cada uno de los comandos para ver lo que ocurre.

1.3. Comandos básicos de terminal, tanto en Windows como en Linux/Mac

Comandos básicos en Linux/Mac

La terminal de comandos en Linux o Mac se suele llamar terminal y viene instalada por defecto en ambos sistemas operativos, para trabajar con ella simplemente tienes que buscarla dentro del listado de aplicaciones y ejecutarla.

Una vez abierta simplemente tendrás que colocar los comandos que te pongo aquí abajo dentro del terminal para empezar a usarlo como los profesionales.

- ▶ **PWD:** usa el comando `pwd` para encontrar la ruta del directorio (carpeta) de trabajo actual en el que te encuentras. El comando devolverá una ruta absoluta (completa).
- ▶ **CD:** para navegar por los archivos y directorios de Linux, usa el comando `cd`. Te pedirá la ruta completa o el nombre del directorio, dependiendo del directorio de trabajo actual en el que te encuentres.

Supongamos que estás en `/home/nombredeusuario/Documentos` y deseas ir a **Proyectos**, un subdirectorio de **Documentos**. Para hacerlo, simplemente escribe el siguiente comando: `cd Proyectos`.

Otra posibilidad es si quieres ir a un directorio completamente nuevo, por ejemplo, `/home/nombredeusuario/www`. En este caso, debes escribir `cd` seguido de la ruta absoluta del directorio: `cd /home/ nombredeusuario/www`.

Hay algunos atajos para ayudarte a navegar rápidamente:

- `cd` para ir directamente a la carpeta de inicio.
- `cd..` (con dos puntos) para ir un directorio hacia arriba.

Como nota al margen, el Shell de Linux distingue entre mayúsculas y minúsculas. Otro punto importante es que el Shell de Linux te autocompleta si escribes parte del directorio o archivo y le das a la tecla TAB te sugiere posibles nombres de archivos o carpetas que estén en ese directorio.

- **LS:** El comando `ls` se usa para listar (de ahí `ls`) el contenido de un directorio. Por defecto, este comando mostrará el contenido de tu directorio de trabajo actual.

Si deseas ver el contenido de otros directorios, escribe `ls` y luego la ruta del directorio. Por ejemplo, ingresa `ls/home/nombredeusuario/Documentos` para ver el contenido de Documentos.

Hay variaciones que puedes usar con el comando `ls`:

- `ls -R` también listará todos los archivos en los subdirectorios.
 - `ls -a` mostrará los archivos ocultos.
 - `ls -al` listará los archivos y directorios con información detallada como los permisos, el tamaño, el propietario, etc.
- **CP:** Usa el comando `cp` para copiar archivos del directorio actual a un directorio diferente. Por ejemplo, el comando `cp escenario.jpg`

/home/nombredeusuario/Imagenes crearía una copia de escenario.jpg (desde tu directorio actual) en el directorio de Imagenes.

- ▶ **MV:** El uso principal del comando `mv` es mover archivos. Los argumentos en `mv` son similares al comando `cp`. Debes escribir **mv**, el nombre del archivo y el directorio destino. Por ejemplo: `mv archivo.txt /home/nombredeusuario/Documentos`.
- ▶ **MKDIR:** Usa el comando `mkdir` para crear un nuevo directorio: si escribes `mkdir Sites`, creará un directorio llamado **Sites**.
- ▶ **RMDIR:** Si necesitas eliminar un directorio, usa el comando `rmdir`. Sin embargo, `rmdir` solo te permite eliminar directorios vacíos.
- ▶ **RM:** El comando `rm` se usa para eliminar directorios y el contenido dentro de ellos. Si solo deseas eliminar el directorio, como alternativa a `rmdir`, usa `rm -r`.

Nota: Ten mucho cuidado con este comando y verifica en qué directorio te encuentras. Este comando elimina todo y no se puede deshacer.

- ▶ **TOUCH:** El comando `touch` te permite crear un nuevo archivo en blanco a través de la línea de comando de Linux. Como ejemplo, ingresa `touch /home/nombredeusuario/www/index.html` para crear un archivo HTML titulado INDEX en el directorio `www`.
- ▶ **CLEAR:** Limpia la terminal para que nada te distraiga, sobre todo cuando has probado muchos comandos y tienes mucho resultado en pantalla.

Comandos básicos en Windows

En Windows tenemos dos terminales propias, la más sencilla es el CMD y luego tenemos el POWERSHELL que tiene algo más de potencia, aunque en muchos casos

no será necesario utilizarla, los comandos en Windows son ligeramente distintos que, en UNIX, aunque un poco más adelante en este capítulo se darán algunas alternativas por si quieres probarlas.

- ▶ **CD:** uno de los comandos más esenciales de la consola de Windows. Sirve para cambiar de directorio, utilizando `cd c:/directorio/subdirectorio` para ir al directorio o carpeta concreta que le digas, o `cd .` (con dos puntos y nótese que no hay espacio como en el comando de Linux) para salir de una carpeta e ir al nivel superior o carpeta donde estaba alojada.
- ▶ **DIR:** hace lo mismo que el comando `LS` en UNIX, lista el contenido del directorio o carpeta donde te encuentras, mostrando todas las subcarpetas o archivos que tiene. Con este comando podrás saber si el archivo que buscas está ahí o a qué subcarpeta navegar.
- ▶ **CLS:** limpia el CMD al igual que `CLEAR` en UNIX.
- ▶ **COPY:** a través del comando `copy rutaArchivo destinoArchivo`. Copia uno o más archivos en la dirección que tu elijas.
- ▶ **MOVE:** mueve a través de `move rutaArchivo destinoArchivo`. Traslada un archivo de una carpeta a otra sin necesidad de interfaz.
- ▶ **DEL:** borra el archivo que pongas después del comando.
- ▶ **MD:** crear una carpeta a través del comando `md nombredecarpeta`.
- ▶ **TYPE:** crear un archivo desde el propio CMD.

Con estos comandos te será mucho más fácil el manejo de fichero con la terminal, te recomiendo que los pruebes y vayas familiarizándote con ellos ya que muchos de ellos dentro de poco los estarás usando prácticamente a diario.

1.4. Alternativas a la terminal

Tanto en Windows, Linux y Mac hay alternativas a las **terminales por defecto** que añaden un extra a las mismas y que puedes instalar de forma gratuita en cualquier sistema voy a mostrarte tres de las más famosos y el **sistema operativo** donde las puedes encontrar. Si te apetece probar con alguna de ellas no dudes en hacerlo, en muchos casos supone una mejora sustancial respecto a la terminal por defecto de nuestro sistema operativo.

Alacritty - Linux

Se trata de un **terminal multiplataforma** que se puede instalar en los **tres sistemas operativos**. Está basado en comando UNIX con lo cual te permitirá tener en Windows el mismo tipo de terminal que en Linux y Mac y por ejemplo el que hay en la mayoría de los servidores que esta basados en LINUX.

Puedes encontrar la información en el siguiente enlace:

<https://github.com/alacritty/alacritty>

Puede ser una ventaja porque es un terminal que es tremendamente **rápida en la ejecución de comandos**. La instalación no es sencilla, pero tienes muchísimos tutoriales en internet que te explican como instalarla en los diferentes sistemas operativos, dentro del enlace que te he pasado anteriormente podrás encontrar las **instrucciones** para instalarlos en varios sistemas.

Iterm – Mac

En este caso estamos hablando de un **terminal exclusivo para Mac**, pero que te recomiendo que instales si usas este sistema operativo, es muchísimo más potente que la terminal por defecto de Mac y muy configurable.

Es un terminal basado en **comandos Unix** y en **Mac** se integra perfectamente. Su instalación es muy sencilla y **puedes personalizarla** en cuanto a colores y tipografía.

Tienes más información en este enlace: <https://iterm2.com/>

CMDder – Windows

Una terminal muy fácil de instalar y que te permite usar **un terminal UNIX dentro de Windows** de forma muy **fácil**, una vez lo instales podrás usar todos los comandos de UNIX en el terminal Windows sin problema.

Puedes encontrar toda la documentación para este terminal en el enlace que te adjunto: <https://cmder.net/> y en este video de YouTube que te adjunto puedes ver

como se instala y se usa dentro de Windows:

<https://www.youtube.com/watch?v=Qs0dG5b4xlc>

En el vídeo *Uso de terminal* vamos a realizar algunos ejemplos sencillos de como moverse por nuestro árbol de carpetas, crear directorios y archivos. Aprende poco a poco el uso del terminal.



Accede al vídeo

1.5. Otras herramientas que vamos a necesitar durante el curso

Navegadores

Quizá no es la herramienta preferida por los desarrolladores, pero sí es el elemento vital donde se despliegan, muestran y viven sus aplicaciones. El objetivo final de un desarrollo de software front end es la **visualización de una serie de interfaces** con posibilidad de **interacción**, de manera adecuada, dentro de toda la serie de navegadores que existen en los diferentes sistemas operativos.

No todos los navegadores permiten visualizar los **elementos gráficos** de la misma manera y no todos los navegadores van implementando los cambios producidos en las nuevas versiones de HTML, CSS o Javascript. Es por ello por lo que uno de los grandes trabajos que tiene el desarrollador de aplicaciones Front es el de adaptar sus programas para que existan el menor número de diferencias entre navegador y navegador.

Los navegadores más conocidos y por tanto aquellos en los que el desarrollador debe asegurar el funcionamiento de sus aplicaciones son: **Google Chrome, Mozilla Firefox, Opera, Safari y Microsoft Edge.**

La mayoría de ellos incluyen **herramientas de ayuda** para que el desarrollador pueda hacer pruebas sobre las aplicaciones que genera, así como para **ajustar los estilos** que se le aplican a sus interfaces o visualizadores para asegurar que el software no da problemas en cualquiera de los tamaños en los que se encuentra disponible.

De todas formas, como herramientas de desarrollo desde aquí te recomendamos que usen **Chrome o Firefox (existe una versión especial para desarrolladores)**. Te damos

los enlaces de descarga para que puedas acceder rápidamente a usarlo e instalarlos en u sistema.

Google Chrome <https://www.google.com/intl/es-es/chrome/>

Firefox <https://www.mozilla.org/es-ES/firefox/new/>

Firefox developer edition <https://www.mozilla.org/es-ES/firefox/developer/>

Editores de texto

Un alto porcentaje del tiempo empleado por cualquier tipo de desarrollador es gastado usando algún **editor de texto**. El código generado se debe escribir dentro de una herramienta que nos permita centrarnos en los problemas que necesitamos resolver con nuestro software y nos abstraiga de la estética y ordenación de nuestro código.

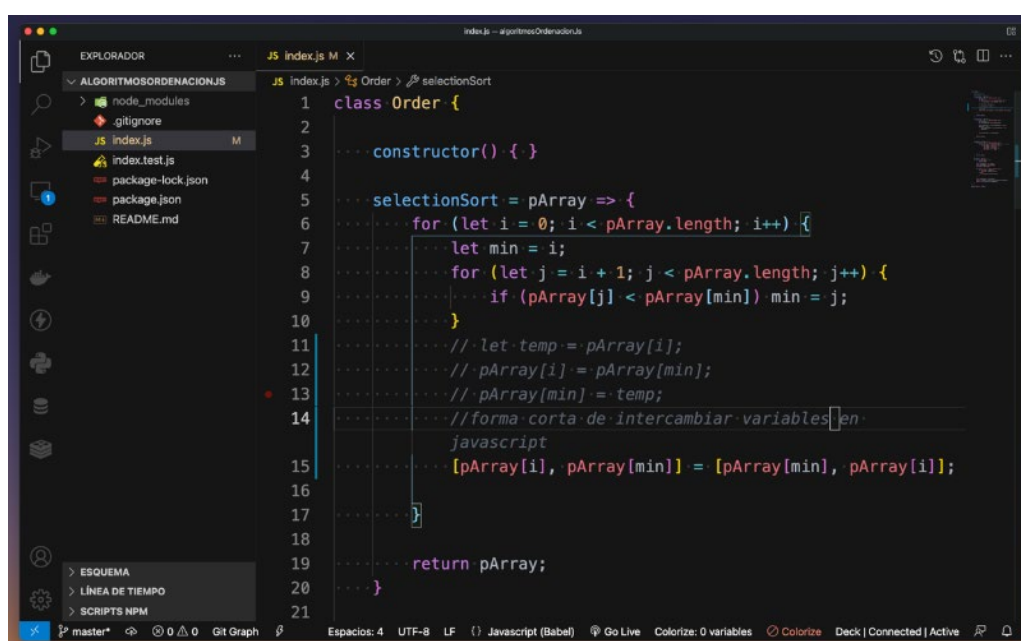


Figura 2. Captura de Visual Studio Code. Fuente: elaboración propia.

Existen numerosas **alternativas** en cuanto a editores de texto se refieren, cada persona se adapta mejor a una solución o a otro. Incluso cabe la posibilidad de que

la elección de nuestro editor de texto favorito se base en el lenguaje o en la tecnología que estemos utilizando.

La mayoría de los editores utilizados en el ámbito del desarrollo incluyen a su vez, herramientas o plugins alternativos que nos permiten darle **aspectos más personalizados** a los entornos de trabajo con los que vamos a desarrollar.

Alguno de los editores de texto que mejor se adaptan al perfil de Desarrollador Web son: **Sublime Text, Atom, Visual Studio Code...**

En este curso nosotros hemos elegido **Visual Studio Code** porque es el editor que mejor ha evolucionado los últimos dentro de los editores del mercado, además es **software libre** así que no tendrás que pagar por el para usarlo.

Su **sistema de extensiones** es muy completo y es muy fácil de aumentar las funcionalidades que tiene a través de plugins y extensiones de terceros.

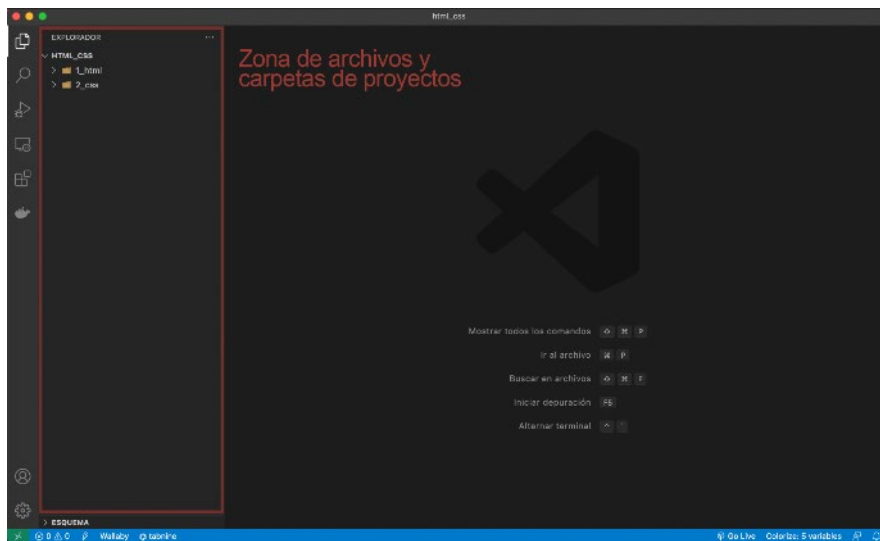
¿Cómo instalo estas Visual Studio Code?

Puedes acceder a la descarga totalmente gratuita de Visual Studio Code desde su página oficial, está disponible para cualquier versión de sistema operativo con lo que no será un problema que tengas MACOS, LINUX o WINDOWS para poder trabajar con esta estupenda herramienta.

Para instalar Visual Studio Code:

<https://code.visualstudio.com/>

Una vez instalado toca familiarizarse con el nuevo interfaz y para ello, aunque durante el curso iremos comentando cosas sobre estas herramientas vamos a comentar ciertas partes del interfaz para que tengamos cierta idea de donde se localizan las cosas.



Captura de Visual Studio Code. Fuente: elaboración propia.

En la imagen anterior podemos ver el interfaz general que tiene Visual Studio Code, con la zona de archivos, esta barra lateral es donde podremos ver los archivos y carpetas de nuestro proyecto y podremos organizar toda la estructura de este.

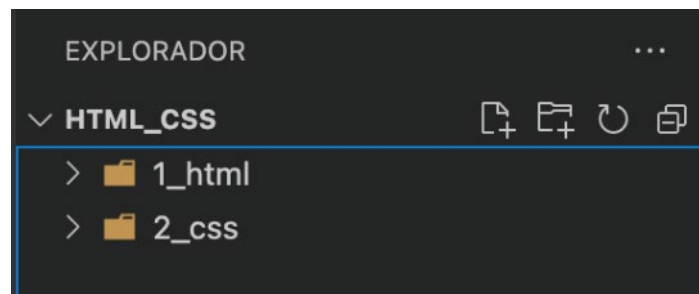


Figura 3. Captura de Visual Studio Code. Fuente: elaboración propia.

En la parte superior de esta barra encontramos los iconos de crear nuevo archivo, crear nueva carpeta, recargar el interfaz, nos serán muy útiles para poder crear una estructura, aunque estas acciones también son accesibles desde atajos de teclado y desde el menú general de Archivo que tiene VSC.

VSC se puede configurar de muchas formas, se le puede cambiar el aspecto visual a través de themes y la configuración de tamaño de fuente y archivos desde el menú de preferencias/usuario que está dentro del menú principal.

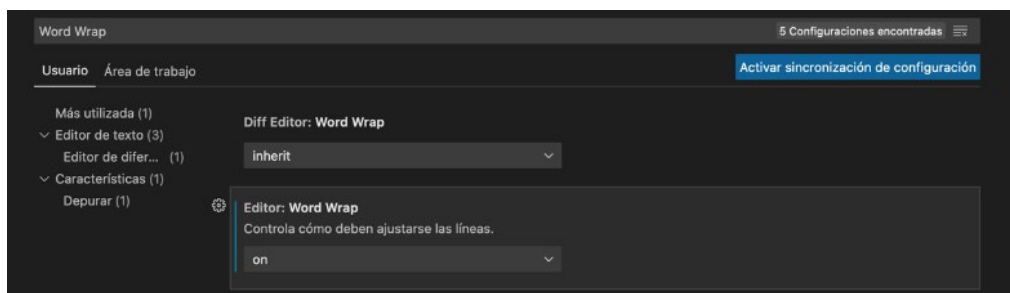


Figura 4. Captura de Visual Studio Code. Fuente: elaboración propia.

Una de las configuraciones más interesante de inicio para el VSC es WORD WRAP (la que está en la imagen anterior) que no genera scroll horizontal infinito que es muy molesto a la hora de programar.

Otra configuración muy interesante es Format On Save, que auto formatea el código cada vez que hagas un guardado del Archivo.

En este punto te recomiendo que tengas a mano siempre el atajo de teclado CTRL+S / CMD+S para guardar un archivo en cualquier momento.

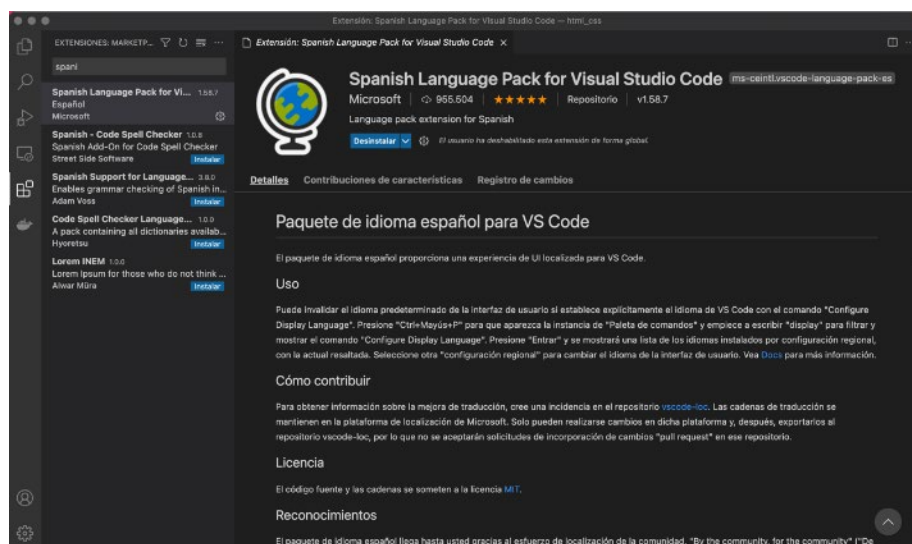


Figura 5. Paquete de idioma español para VS Code. Fuente: elaboración propia.

Otro punto muy importante de VSC es que su funcionalidad básica (que de por si es estupenda) se puede ampliar a través de extensiones, que puedes buscar e instalar

desde su Marketplace, como puedes ver en la imagen superior es el icono de la barra que son cuatro cuadrados (uno de ellos ligeramente desplazado).

Desde allí podrás instalar nuevas funcionalidades como el traductor de español (imagen superior) de VSC, linters de código, ayudas o extensiones. Durante el curso verás que te vamos recomendando algunas para que las instales, pero sería interesante que fueras investigando tú, por tu cuenta.

Si quieres saber más sobre este interesante editor hay muchos tutoriales y videos en YouTube que hablan sobre él y que son muy interesantes que les eches un vistazo. Te recomiendo muchísimo que vayas a nuestra última sección titulada «A fondo» y dediques tiempo visualizar y probar todos los enlaces con documentación interesante que creo que deberías ver, tanto para el usuario de la terminal, como del editor de código Visual Studio Code.

Tema 2. Uso de GIT control de versiones

2.1. Introducción y objetivos

En este tema vamos a hablar de otra herramienta muy importante para nosotros los desarrolladores y que vamos a usar durante todo el curso. Esta herramienta es **Git**, es un software que se instala en nuestro equipo y que nos permite realizar **versionados de nuestro código** sin necesidad de crear copias de carpetas con infinitas versiones.

Te recomiendo revisar todo el contenido de este tema, tanto del contenido de texto como de vídeo, así como los enlaces que te propongo en todas las secciones. Git es un contenido extenso y dominarlo requiere de años de trabajo con él. Pero durante el curso se convertirá en una herramienta fundamental, no solo para hacer un control de versiones si no por ejemplo para descargarse el repositorio de código del profesor y poder tener en tiempo real los ejercicios que vayas haciendo con cada uno de los profesores.

También es una excelente herramienta para compartir código y gestionar tareas en entornos de trabajo.

Aprenderemos a crear **nuestros propios repositorios** dentro de nuestro equipo y subirlos a un servidor remoto como Github que me permitirá tener una copia de todo mi diario de programación y poderla no solo descargar en otro equipo, sino poder compartirla con otras personas de mi entorno de trabajo o con el público en general si mi repositorio fuese público.

2.2. ¿Por qué es necesario tener un control de versiones?

En un equipo de trabajo multidisciplinar trabajan **multitud de perfiles** y, en ocasiones, todos deben modificar las mismas partes del código fuente de una aplicación. Es por ello por lo que se debe llevar un control de, tanto los cambios que se van realizando para poder documentarlos, como de quién es el encargado de realizar dichos cambios.

Aparte de facilitar el trabajo en equipo entre las múltiples áreas, nos permite llevar un **control de cambios** muy exhaustivo y así, en caso de error, siempre podemos recuperar versiones anteriores o más estables.

Git fue creado pensando en la **eficiencia** y la **confiabilidad** del mantenimiento de versiones. La herramienta que cuida el código de un proyecto debe ser segura y robusta para que no permita la pérdida de ninguna funcionalidad a través de cualquier tipo de error.

Es muy importante que el desarrollador sea capaz de adquirir de manera ágil la forma de organizarse con Git del equipo con el que va a trabajar ya que de ello depende las diferentes colaboraciones con sus compañeros y el buen funcionamiento del versionado. Por lo tanto, esta herramienta es, en la actualidad, una de las imprescindibles.

Git es una herramienta por terminal y, aunque desde aquí os animamos a que la uséis desde la terminal, creemos que la mejor forma de aprender en este caso es a través de un entorno gráfico.

Aun así, si alguien se anima, también os dejamos material para que podáis estudiar cómo se maneja Git desde terminal. Eso sí, antes de meterte con ello, repasa bien este capítulo para entender todos los conceptos y poderlos aplicar en la terminal.

Aquí te dejamos un recurso interesante donde se explica cómo se hacen los comandos más básicos de Git en un entorno de terminal.

Manual Esencial de Git: <https://www.freecodecamp.org/espanol/news/el-manual-esencial-de-git/>

2.3. Instalar Git

Para instalar Git hay muchas y muy variadas formas, pero aquí te ponemos algunas de ellas para ayudarte a iniciarte en esta estupenda herramienta.

Windows / macOS

- ▶ Ve a <https://git-scm.com/downloads> y elige el sistema operativo apropiado.
- ▶ La descarga debería comenzar automáticamente; si encuentra dificultades, descargue el instalador manualmente haciendo clic en el enlace correspondiente.
- ▶ Ejecute el instalador y seleccione las opciones propuestas durante la instalación.

Ubuntu (Linux sistemas DEBIAN con apt)

Para instalar el sistema de control de versiones de Git, use el terminal.

- ▶ Use la combinación de teclas **Ctrl-Alt-T** para abrir la terminal.
- ▶ Ejecutar comandos: `sudo apt update` (presione enter).

► `sudo apt install git` (nuevamente, presione enter).

Durante la instalación, el sistema le pedirá la contraseña de administrador.

Para verificar si has instalado GIT correctamente

Después de la instalación, verifique que todo esté instalado correctamente.

Abra la terminal (Linux/macOS) o el CMD (Windows) e ingrese el siguiente

`git --version` (git, espacio, menos, menos, versión).

Si la instalación salió bien, debería obtener una respuesta con la versión actual.

2.4. ¿En qué nos ayuda Git?

Cuando estamos trabajando en un proyecto, todos nuestros archivos son susceptibles de recibir **cambios** y en **desarrollo de software** estos cambios se pueden producir en varios ficheros y de forma continuada. Tener un control sobre lo que modificamos es vital ya que no podemos tener una copia de todos y cada uno de los archivos de un proyecto de desarrollo y cualquier cambio dentro de un archivo puede hacer que otro falle. El **control de versiones** y en este caso Git nos permite poder retroceder en la línea de tiempo de nuestro desarrollo para solucionar pequeños problemas y poder volver a una versión funcional de mi código.

La premisa no es evitar perder código, sino solucionar el problema para que siga funcionando.

Si esto lo hacemos nosotros solos puede ser un poco lioso al principio, pero aprender a manejarlo es vital sobre todo para trabajar con equipos donde hay mucha gente y donde tenemos que compartir y trabajar con varios archivos de código.

El **flujo de trabajo** dentro de un entorno de desarrollo consiste en que con el avance del tiempo es normal que se detecten errores y se desee volver a una instancia anterior, muchas veces lo que hace en este caso es hacer copias de respaldo, sin embargo, este enfoque es poco práctico en entornos con cambios constantes y al aumentar la cantidad de archivos y crece de manera sustancial cuando se trabaja en equipos grandes.

En este caso Git nos aporta una herramienta que esta todo el rato pendiente de los cambios que se producen en nuestro proyecto a nivel de línea esto hace que Git sea una herramienta muy dinámica para los desarrolladores que básicamente se pasan el día con fichero de texto de varias líneas. Para los ficheros pesados como las imágenes sin embargo hay otros sistemas más útiles como por ejemplo una herramienta de nube tipo DropBox a Wetransfer.

2.5. Características principales de Git

Vamos a ver algunas de las características que tiene Git como control de versiones.

- ▶ **Control de Versiones distribuido:** quiere decir que existe una copia del código tanto en local como en remoto.
- ▶ **Permite trabajar desconectado:** te puedes encontrar en alguna situación en el que no puedas tener conexión a internet, pero te permite acceder a todo el histórico del código desde cualquier sitio, por ejemplo, en un portátil.
- ▶ **Permite suspender trabajos:** durante los desarrollos siempre ocurren incidencias que te obligan a dejar lo que estás haciendo en este momento u poner a solucionar lo que no funciona.

- ▶ **Commits atómicos:** Git tiene una zona de stage para trabajar con distintos comits. GIT es muy fuerte si trabajamos con distintos commits para poder ir seleccionando tanto ficheros como partes de ficheros.
- ▶ **Creación de ramas:** cuando estamos trabajando en un proyecto siempre no surge la duda de si lo que estamos haciendo funcionará y entonces lo que hacemos es hacer un duplicado de nuestra carpeta por si acaso. Pues bien, esto ya no hará falta hacerlo porque GIT tiene un sistema de ramas que nos permite trabajar tranquilamente como si estuviésemos en otra carpeta. Cada rama se convierte en una copia de ese proyecto y cualquier error que se produzca en esa rama no afectará a las otras siempre y cuando no volquemos el contenido de una rama en otra sin asegurarnos que todo funciona.
- ▶ **Multiservidor:** puedes tener más de un repositorio. Puedes tener un repositorio de cliente y un repositorio de desarrollo en tu servidor y puedes pasar la información de un repositorio a otro. Resuelve también problemas legales (un defecto de software por que un tercero a tocado donde no debía, por ejemplo).

2.6. Interfaz gráfica para el uso de Git

Como hemos dicho Git es una herramienta que se maneja a nivel de terminal, pero también tiene multitud de aplicaciones que te permiten manejarla a nivel de interfaz gráfica, este tipo de aplicaciones que tiene un interfaz gráfico nos ayudan a entender cómo funciona Git y a cometer menos errores cuando estamos aprendiendo.

Nosotros vamos a usar como aplicación Gitkraken, es una herramienta multiplataforma que se puede usar tanto en Windows, como en Mac como en Linux. Es gratuita para uso personal, la única limitación que tiene es que no podemos crear repositorios remotos privados, pero en este caso para aprender a manejarlo en el curso nos es más que suficiente.

Podemos ver otras herramientas similares como puede ser como Sourcetree, Smartgit, Github Desktop. Puedes echarles un vistazo si quieres, incluso te animamos a que te las instales y las uses.

Para la instalación de Gitkraken te recomendamos que te leas la documentación oficial de la propia página donde te explica cómo se instala para cada sistema operativo: <https://support.gitkraken.com/how-to-install/>

Además, te recomendamos también que tengas esta página entre tus favoritos ya que aquí tienes todo lo necesario para el uso de Gitkraken y su aprendizaje.

2.7. Repositorio remoto

Tener un sitio donde alojar nuestros **proyectos** es fundamental ya que es la forma de compartirlos con más desarrolladores, enseñarlos en entrevistas técnicas de trabajo etc.

En nuestro caso la herramienta que vamos a usar es la que básicamente usan todos los desarrolladores: Github.

Es un portal web en el que te das de alta y generas tu área de trabajo, en esta área de trabajo, podrás crear diferentes repositorios, tanto públicos como privados (aunque recuerda que para usar Gitkraken es necesario que sean públicos, si no hay que pasar por caja).

La página web a la que tendrás que ir a darte de alta es <https://github.com/>

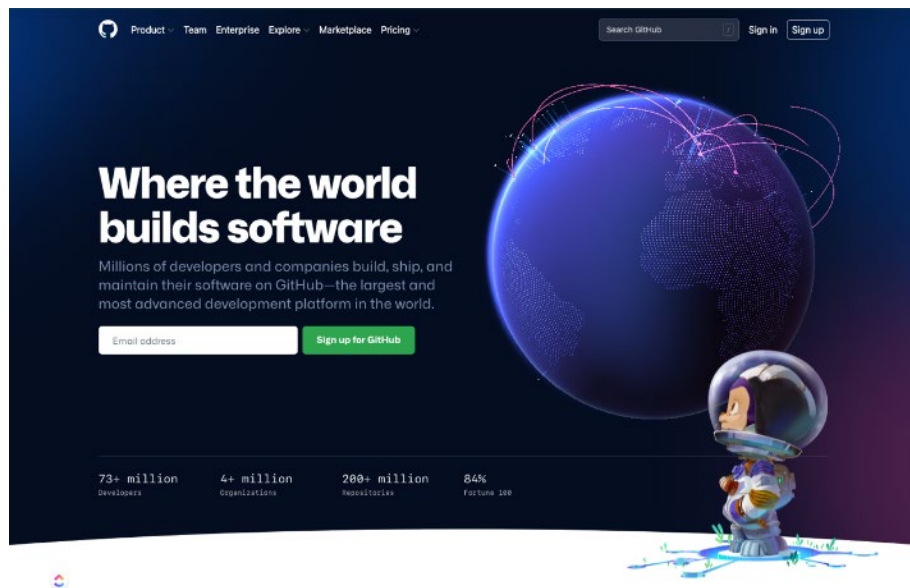


Figura 6. Captura de github. Fuente: elaboración propia.

El proceso de registro es muy sencillo, solo necesitarás una cuenta de correo y el registro es gratuito.

Una vez dentro tendrás acceso a crear tus repositorios, te enseño por ejemplo como tengo yo mi perfil personal donde puedo crear los repositorios que yo quiera.

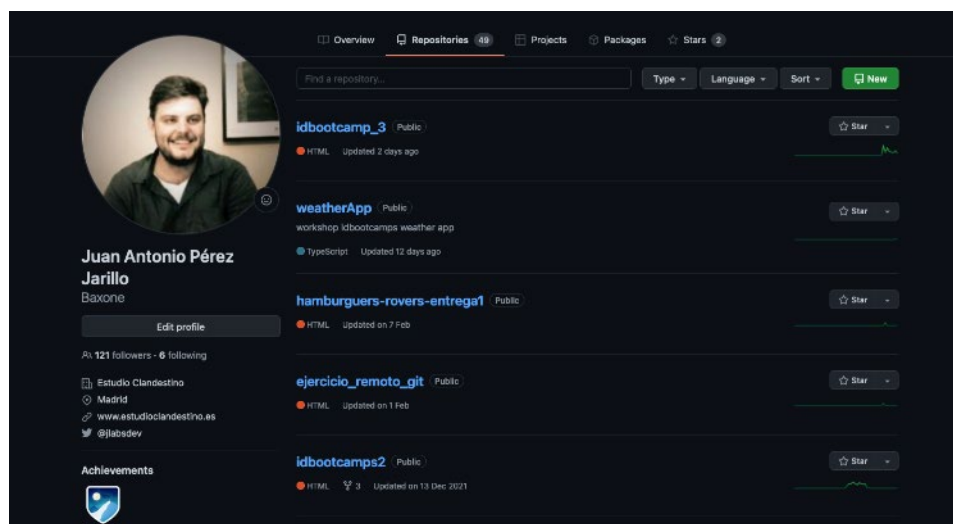


Figura 7. Captura de la github. Fuente: elaboración propia.

2.8. ¿Cómo funciona Git?

Áreas de trabajo en Git

Git principalmente trabaja con tres áreas de trabajo.

- ▶ Área de trabajo (working area): son los archivos del proyecto en el que estamos trabajando
- ▶ Área de staging: son los archivos que han sido modificados y que debemos añadir a nuestro control de versiones.
- ▶ Repositorio (file status): son los archivos y el diario que has ido llevando dentro del repositorio.

Durante el desarrollo de un proyecto tendremos que pasar archivos de un área a otra y cada proceso tendrá un nombre. Por ejemplo, pasar del Working Area a la Staging Area requerirá hacer un stage, y para pasar de Staging Area a Repository tendremos que hacer un commit.

La zona de stage es una zona temporal donde estarán el o los archivos que queramos añadir a nuestro control de versiones. Es el sitio en el que debemos tener nuestros archivos guardados para poderlos comitear y añadir a nuestro control de versiones.

Trabajo con ramas

El trabajo con ramas es quizás el punto más interesante que tiene Git ya que nos permite tener y trabajar con una copia de nuestro proyecto de forma fácil y segura. Dentro de un repositorio se pueden tener múltiples ramas, cada rama tiene su área de trabajo independiente, esto hace que todo lo que hagamos en esa rama no afecta a las otras, con lo que cualquier problema que surja no afectará al resto del equipo. Existen mucho flujo de trabajo a la hora de trabajar con ramas, pero quizás el más famoso sea Gitflow. Al menos es en el que se basan multitud de empresas. Es

fundamental tener un esquema de trabajo para funcionar con GIT porque si no es muy fácil que cualquier error que cometamos afecte al resto de nuestro desarrollo. Gitflow divide el ámbito de trabajo en 5 ramas.

- ▶ **Main:** Donde se tiene el código de la última versión estable.
- ▶ **Release:** Donde se concentran todos los cambios que estarán en la próxima versión y se realizan las pruebas finales.
- ▶ **Develop:** Donde se tiene el código experimental aquí es donde los desarrolladores van volcando sus **features**.
- ▶ **HotFix:** Donde se reparan errores detectados a posteriori (fuera de las pruebas).
- ▶ **Features:** Una rama independiente por cada característica que se le quiera agregar al software.

Visualizar cómo funciona Gitflow nos puede ayudar a comprender el porqué del uso de estas ramas.

Bajo estas líneas podemos ver lo que es un flujo de trabajo con ramas y como se pasan de unas a otras, dependiendo del trabajo que vallamos a realizar.

Desde **develop** se crear todas las **features** nuevas, son los trabajos nuevos en los que deben trabajar los desarrolladores.

Cuando en **develop** funciona todo correcto se crea una **release**, esta es la que se volcaría si todo estuviese correcto en **Main(master)**.

Si ocurre un error este se saca de **Main(master)** se corrige y se vuelca en **develop** y en **Main(master)** así como en las diferentes **features** que estén creadas.

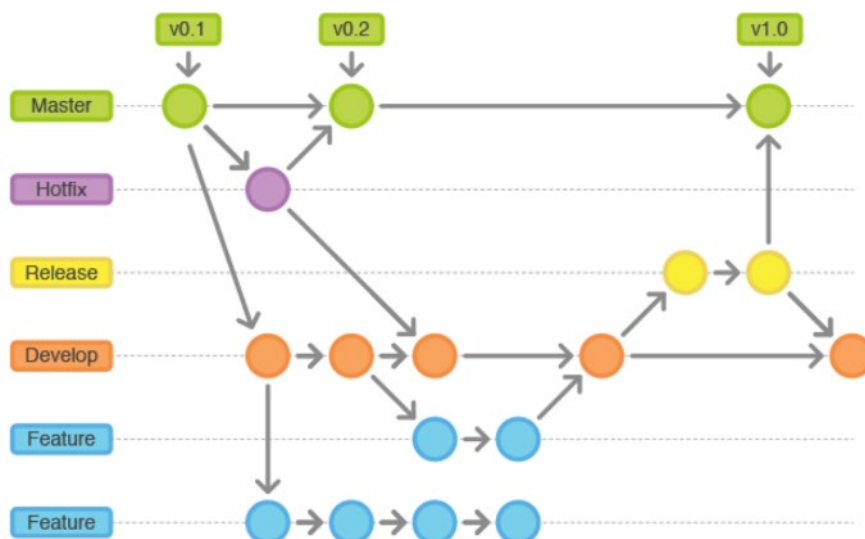


Figura 8. Gitflow. Fuente: Rodríguez, J. (s.f.). *Blog de Javier Rodríguez*. Recuperado de <https://www.javierrguez.com/>

Este proceso puede ser lioso al principio, pero no le tengas miedo por que cuanto más lo uses más fácil te será.

Vamos a mostrarte un video en el que puedes ver como aplicamos este patrón de GIT a un ejercicio de html, y donde vamos a aprovechar para explicarte cómo funciona Github y Gitkraken y cómo puedes enlazarlos.

Tomate tu tiempo y repite el video cuantas veces quieras, hasta que entiendas todo el proceso ya que esta herramienta va a ser fundamental para ti en tu día a día de programador.

Cuando acabes este video vete a la sección «A Fondo», donde encontrarás multitud de enlaces interesantes que te recomiendo mucho ver y comprender, serán el añadido perfecto a esta primera toma de contacto con Git.

En el vídeo *GIT control de versiones* vamos a realizar un pequeño tutorial de html y css, no se requieren conocimientos en cuanto a código simplemente límtate a copiar los fragmentos de código. Lo realmente interesante es como trabajaremos con GIT.

Explicaremos paso a paso todos los puntos para que puedas así entender cómo funciona un control de versiones.



Accede al vídeo

Uso de Visual Studio Code e instalación

VSCode	Ya.	Página	web	oficial.
https://www.tutorialesprogramacionya.com/herramientas/vscodeya/				

Cómo	instalar	Microsoft	Visual	Studio	Code.	Página	web	oficial.
https://support.academicsoftware.eu/hc/es/articles/360006916138-C%C3%B3mo-instalar-Microsoft-Visual-Studio-Code#:~:text=Paso%201%3A%20Ve%20a%20la,acepta%20el%20acuerdo%20de%20licencia.								

Tutorial bastante completo de cómo usar Visual Studio Code.

Vídeo de cómo usar los comandos básicos de una CMD en Windows

FalconMasters. (14 de agosto de 2014). *Cómo utilizar la Consola de Windows (Comandos básicos CMD)* [Vídeo]. YouTube. <https://www.youtube.com/watch?v=W6434nulBu8>

Vídeo de los comandos básicos en Windows con el terminal.

Vídeo de cómo usar los comandos básicos de UNIX valido para Linux y para Mac

Sys Beards. (29 de octubre de 2020). *Comandos básicos Linux, Ubuntu 20.04 y otras distros* [Vídeo]. YouTube. <https://www.youtube.com/watch?v=sqOJsgBE70I>

Vídeo muy interesante de algunos comandos de Linux en terminal pero que también te servirán para manejar una terminal de Mac.

Cómo instalar Terminal Linux en Windows CMDER

Contando Bits. (5 de noviembre de 2020). *Cómo instalar CMDER (Emulador de Terminal) – Comandos de Linux en Windows 10* [Vídeo]. YouTube. <https://www.youtube.com/watch?v=OJx9l7Wbf80>

Fazt Code. (8 de octubre de 2019). *Cmder – Emulador de Consola para Windows (Consola con características de Terminal Linux)* [Vídeo]. YouTube. <https://www.youtube.com/watch?v=Qs0dG5b4xlc&t=359s>

Dos videos muy interesantes en español donde te explica como configurar una terminal Linux en Windows y trucos para usarla, si no te gusta el CMD de Windows te recomiendo que le eches un vistazo a este vídeo.

Listado de comandos para Linux

Terminal Cheat Sheet. Página web oficial. <https://terminalcheatsheet.com/es/>

Lista de comandos básicos de Linux que deberías probar.

Command Promp Cheatsheet

Cs.columbia. (s.f.). *Windows Command Prompt Cheatsheet*. <http://www.cs.columbia.edu/~sedwards/classes/2017/1102-spring/Command%20Prompt%20Cheatsheet.pdf>

Pdf descargable con los comandos básicos de una Command Prompt en Windows.

Tutorial paso a paso de cómo funciona Git – Gitkraken – Github

Elc.github.io. (29 de abril de 2019). *Guía introductoria de Git con Interfaz Visual* [Vídeo]. YouTube. <https://elc.github.io/posts/git-guide-with-visual-interface/es/>

Imprescindible su lectura ya que explica de forma muy detallada cómo se realiza cada paso con Git y Gitkraken.

Como trabajar con un repositorio con Gitkraken

García, J. (26 de marzo de 2018). *Acceder y usar un repositorio con GitKraken* [Blog]. <https://ckolmos.blogspot.com/2018/03/acceder-y-usar-un-repositorio-con.html>

Una publicación muy interesante donde se explica el uso de GIT y gitkraken.

Tutorial para el manejo tanto por terminal como con Gitkraken

Pythones. (s.f.). *Git en remoto tutorial en consola + GitKraken*. <https://pythones.net/git-en-remoto-gitkraken-tutorial/>

Aquí puedes ver cómo funciona GIT tanto a nivel de consola como a nivel de interfaz con Gitkraken. Entenderás los procesos que se producen en cada uno de los momentos del control de versiones.

Vídeo tutorial de cómo funciona Gitkraken

Yo Androide. (26 de diciembre de 2019). *Tutorial completo Gitkraken desde cero – Gestor de Git gráfico* [Vídeo]. YouTube. <https://elc.github.io/posts/git-guide-with-visual-interface/es/>

Video explicativo de cómo gestionar un proyecto con gitkraken.

Aprende Git ahora! curso completo gratis desde cero

HolaMundo. (18 de febrero de 2022). *Aprende Git ahora! Curso completo GRATIS desde cero* [Video]. YouTube. <https://www.youtube.com/watch?v=VdGzPZ31ts8>

Si quieres profundizar en el uso de Git como control de versiones y sobre todo en el uso del terminal te recomiendo encarecidamente este video ya que te aportará el conocimiento necesario para el uso de Git a nivel de terminal.