



Entorno | Intro

Después de esta lección podrás:

1. Utilizar tu ordenador para simular un entorno real de programación.
2. Guardar el progreso de tu código y compartirlo con otr@s desarrollador@s.
3. Utilizar Node.js para correr servidores escritos en JavaScript en nuestro ordenador.
4. Utilizar MongoDB para gestionar información en una base de datos.
5. Crear proyectos utilizando frameworks y librerías de última generación como Angular y React.

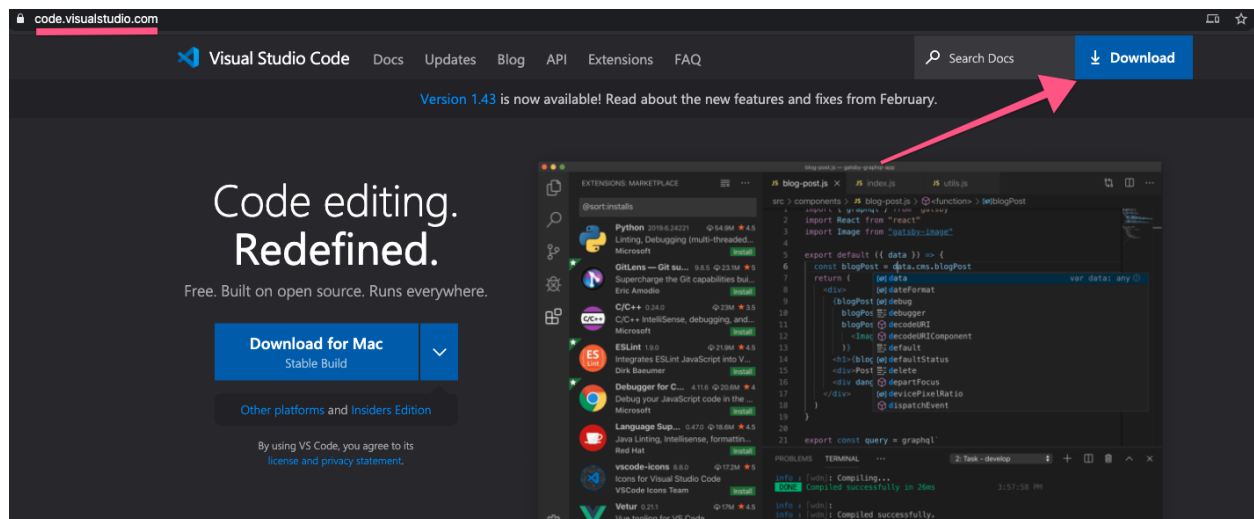
Editores de Código

¿Cómo instalar un IDE y cuál elegir?

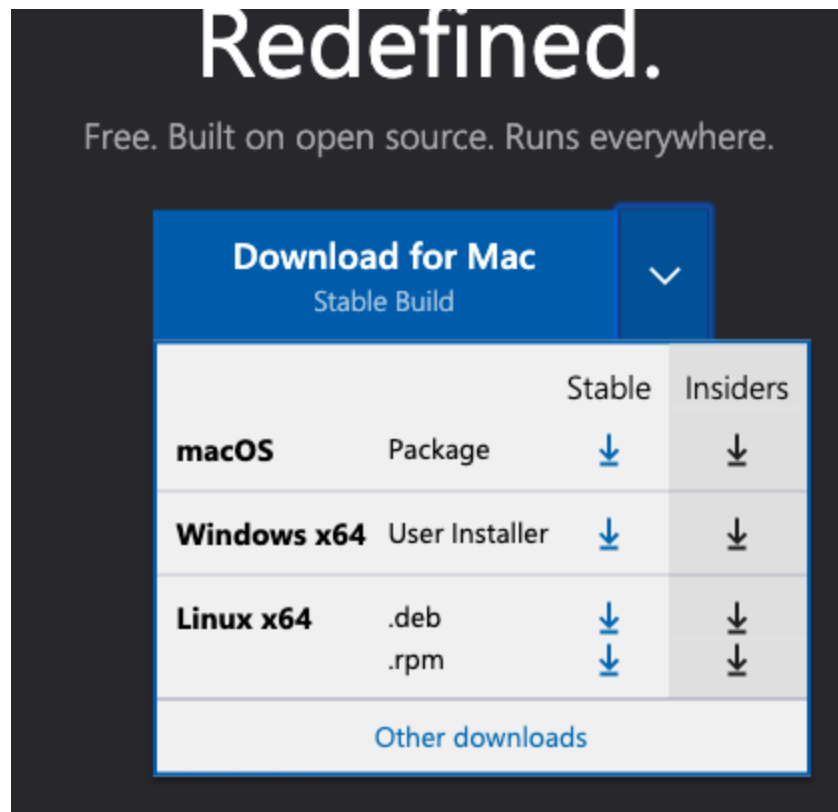
Un IDE (Entorno de Desarrollo Integrado) es, de forma general, una aplicación con la que escribiremos y desarrollaremos nuestro código a través de una interfaz gráfica para hacer más fácil nuestro trabajo.

Existen muchos IDEs, cada cual con su uso y lenguajes recomendados. En nuestro caso, utilizaremos **VSCode** (Visual Studio Code), desarrollado por Microsoft y especialmente útil e indicado para el Desarrollo de Aplicaciones Web con JavaScript.

Para instalarlo, iremos a la web oficial (<https://code.visualstudio.com/>) y descargamos el instalador correspondiente para nuestro **SO** (Sistema Operativo).

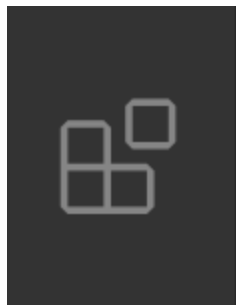


Recuerda que en Ubuntu y distribuciones de Linux usaremos instaladores **.deb** a ser posible para facilitar todo el proceso.



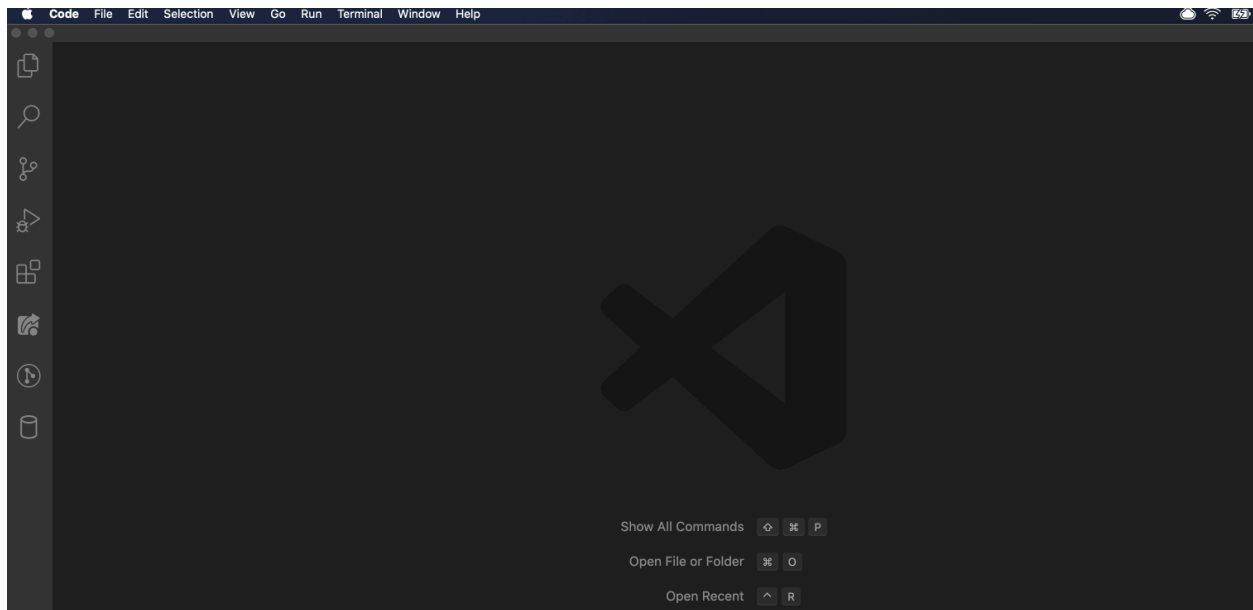
Ahora que tenemos nuestro IDE, vamos a instalar una serie de paquetes para hacernos la vida más fácil. Ya sabéis, cuando programamos, intentamos automatizar tareas que puedan ser repetitivas, así que toda extensión que haga esto una realidad será bienvenida 🚀

Para instalar una extensión clickamos sobre este símbolo en la barra lateral izquierda de VSCode:

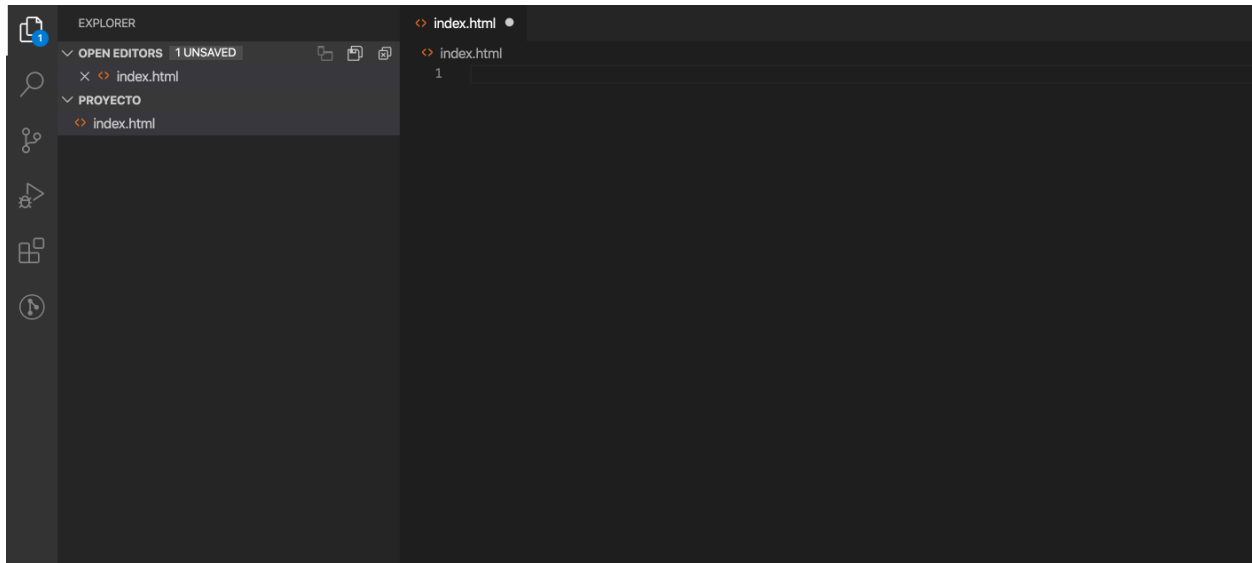


Una vez dentro, buscamos una extensión por su nombre y pulsamos en instalar (algunas necesitarán que cerremos y abramos de nuevo VSCode para funcionar).

GIF con el proceso a seguir para instalar una extensión:



Una vez instalada, conseguiremos funcionalidades adicionales en nuestro editor de código. Esta extensión que acabamos de instalar nos permite autocompletar bastante código en nuestros archivos **HTML**, como mostramos en el siguiente ejemplo:



Estas son las extensiones que usaremos durante toda la duración de los módulos relacionados con JavaScript en el curso, instálalas lo antes posible para que no te pille por sorpresa! 🤖

Para vigilar que escribimos código correctamente:

- Babel JavaScript
- ESLint
- TSLint
- Prettier - Code formatter
- Sass

Para autocompletar nuestro código cuando sea posible:

- HTML CSS Support
- HTML Snippets
- Auto Close Tag
- Auto Import
- ES7 React/Redux/GraphQL/React-Native snippets
- Angular Snippets (Version 9)

Para mejorar la interfaz visual:

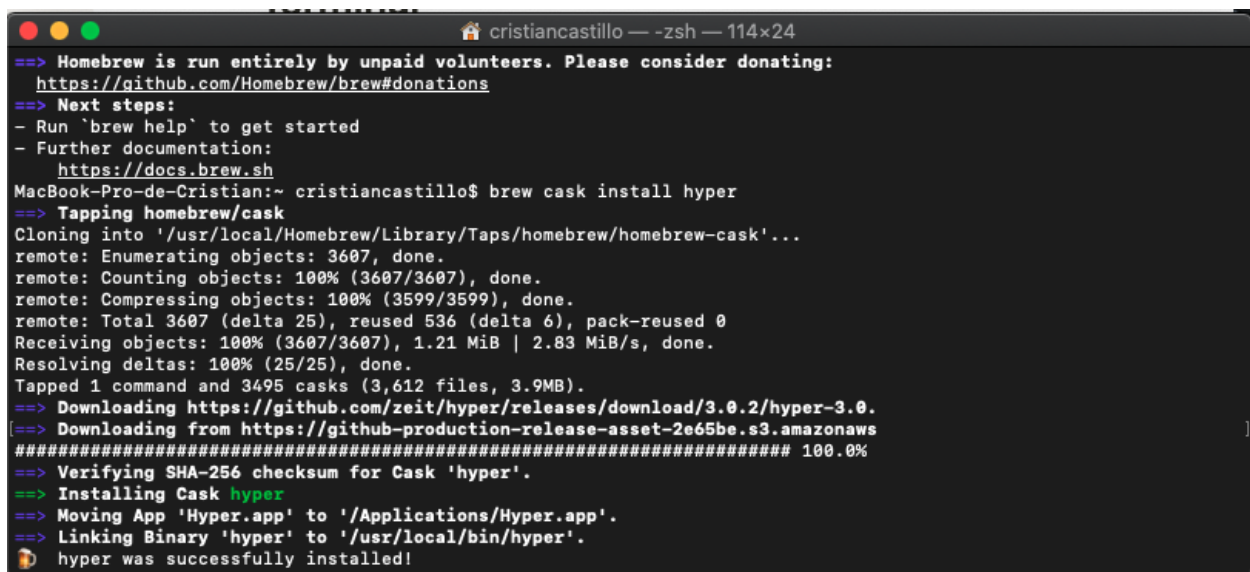
- Material Theme
- Material Theme Icons

Terminal

¿Qué es y para qué sirve una terminal?

Cuando hablamos de terminal, estamos hablando de esa pantalla negra con letras blancas que tanto impresiona y existe en todos los dispositivos.

Aquí tienes una muestra de lo que veremos en ella 🖥️:



```
cristiancastillo — zsh — 114x24

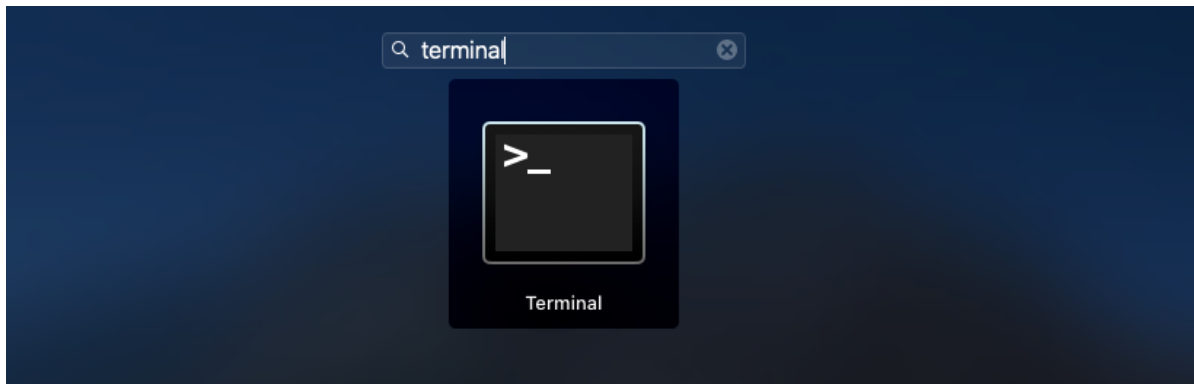
==> Homebrew is run entirely by unpaid volunteers. Please consider donating:
https://github.com/Homebrew/brew#donations
==> Next steps:
- Run 'brew help' to get started
- Further documentation:
  https://docs.brew.sh
MacBook-Pro-de-Cristian:~ cristiancastillo$ brew cask install hyper
==> Tapping homebrew/cask
Cloning into '/usr/local/Homebrew/Library/Taps/homebrew/homebrew-cask'...
remote: Enumerating objects: 3607, done.
remote: Counting objects: 100% (3607/3607), done.
remote: Compressing objects: 100% (3599/3599), done.
remote: Total 3607 (delta 25), reused 536 (delta 6), pack-reused 0
Receiving objects: 100% (3607/3607), 1.21 MiB | 2.83 MiB/s, done.
Resolving deltas: 100% (25/25), done.
Tapped 1 command and 3495 casks (3,612 files, 3.9MB).
==> Downloading https://github.com/zeit/hyper/releases/download/3.0.2/hyper-3.0.
[==> Downloading from https://github-production-release-asset-2e65be.s3.amazonaws
##### 100.0%
==> Verifying SHA-256 checksum for Cask 'hyper'.
==> Installing Cask hyper
==> Moving App 'Hyper.app' to '/Applications/Hyper.app'.
==> Linking Binary 'hyper' to '/usr/local/bin/hyper'.
🍺 hyper was successfully installed!
```

Pero, antes de tener nuestra ventana de la terminal hasta arriba de información, vamos a aprender a usarla y trabajar con ella.

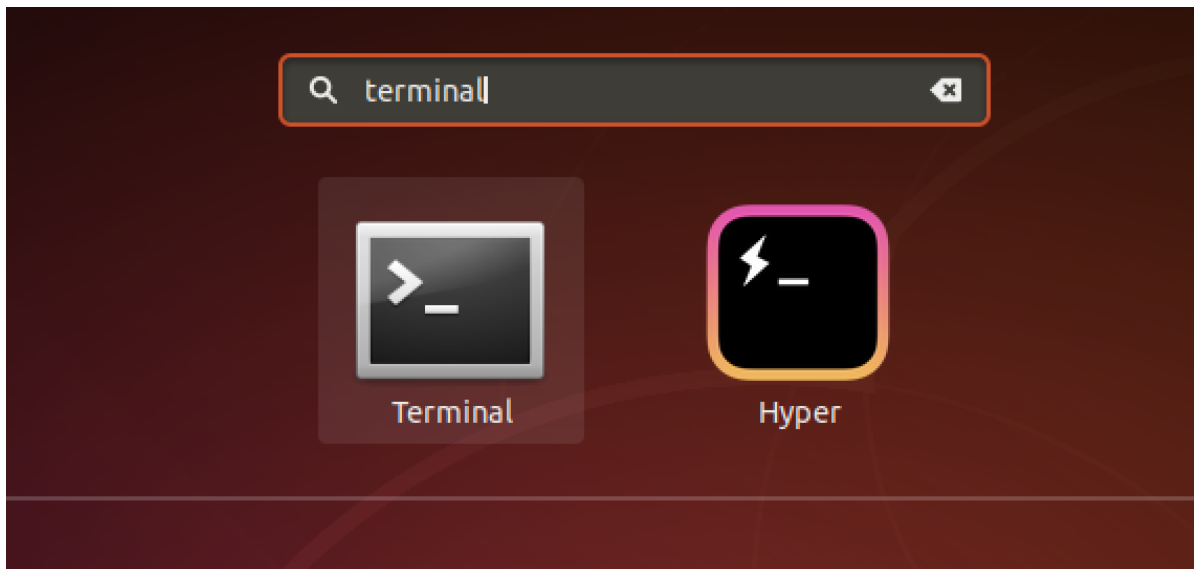
▼ Uso de la terminal en Mac y Ubuntu 🖥️

Para usar la terminal en Mac o Ubuntu, vamos a abrir el buscador de archivos y a escribir **terminal**, cuando aparezca el siguiente icono clickaremos sobre él para abrirla:

En Mac:



En Ubuntu:



- Hyper es otro tipo de terminal, en tu caso no aparecerá en la búsqueda, no te preocupes 😊

Una vez tenemos abierta la terminal, vamos a ver un par de comandos para movernos por nuestro entorno.

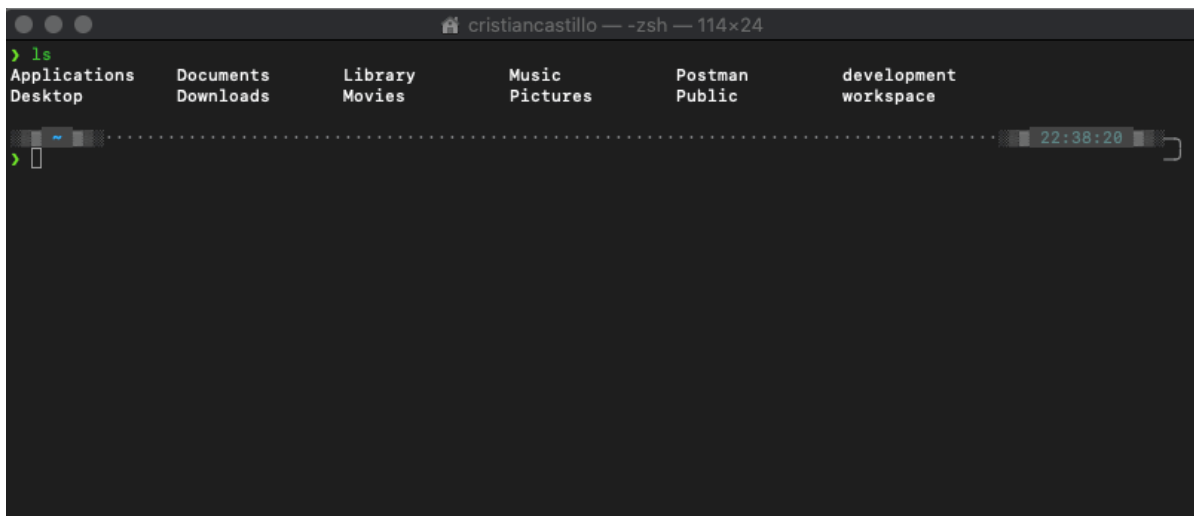
¿Dónde nos encontramos al abrir la terminal?

Nos encontraremos en la "base" de nuestro ordenador, en el punto de entrada de todas las carpetas importantes que contienen todo lo que tenemos en nuestro disco duro, es decir, donde tenemos las carpetas **Escritorio**, **Documentos**, **Descargas**...

Esta base se llamará `~` en nuestra terminal, y a cada carpeta, la llamaremos **directorio**.

¿Cómo puedo ver el contenido de un directorio?

Utilizaremos el comando **ls** directamente en la terminal:



Aquí en código en nuestra terminal, junto con su output:

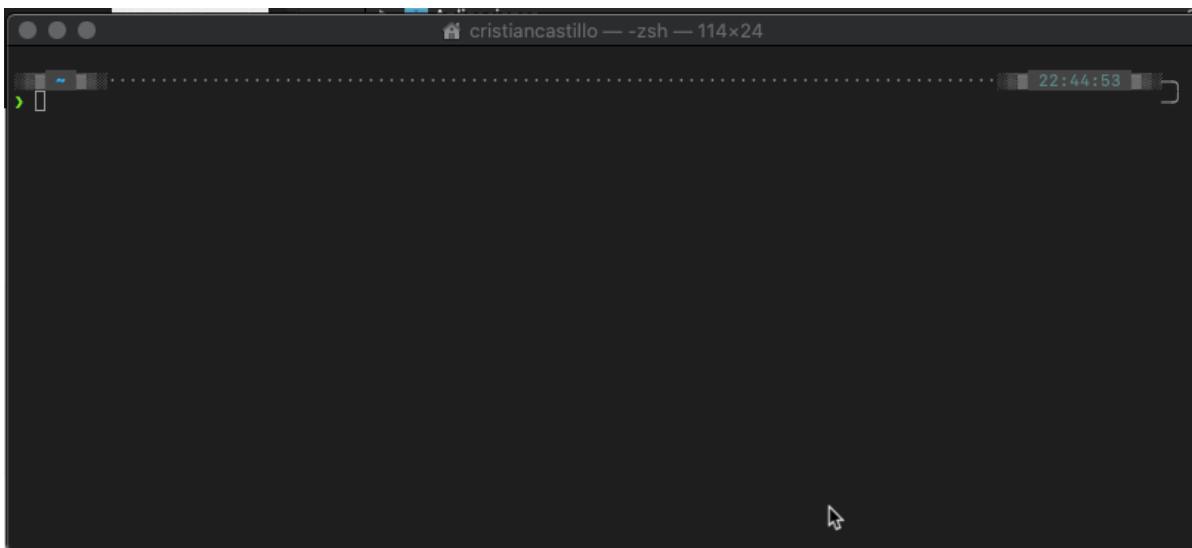

```
> ls
Applications  Documents Library   Music   Postman   development
Desktop       Downloads Movies    Pictures Public    workspace
```

¿Y cómo puedo acceder a una carpeta?

Como puedes ver en el resultado anterior (a lo que llamamos **log**) en la terminal, hay varias carpetas como Desktop o Documents.

Para acceder a ellas podemos usar el comando **cd** seguido del **nombre de la carpeta** como hacemos a continuación.

Ejemplo en Mac:



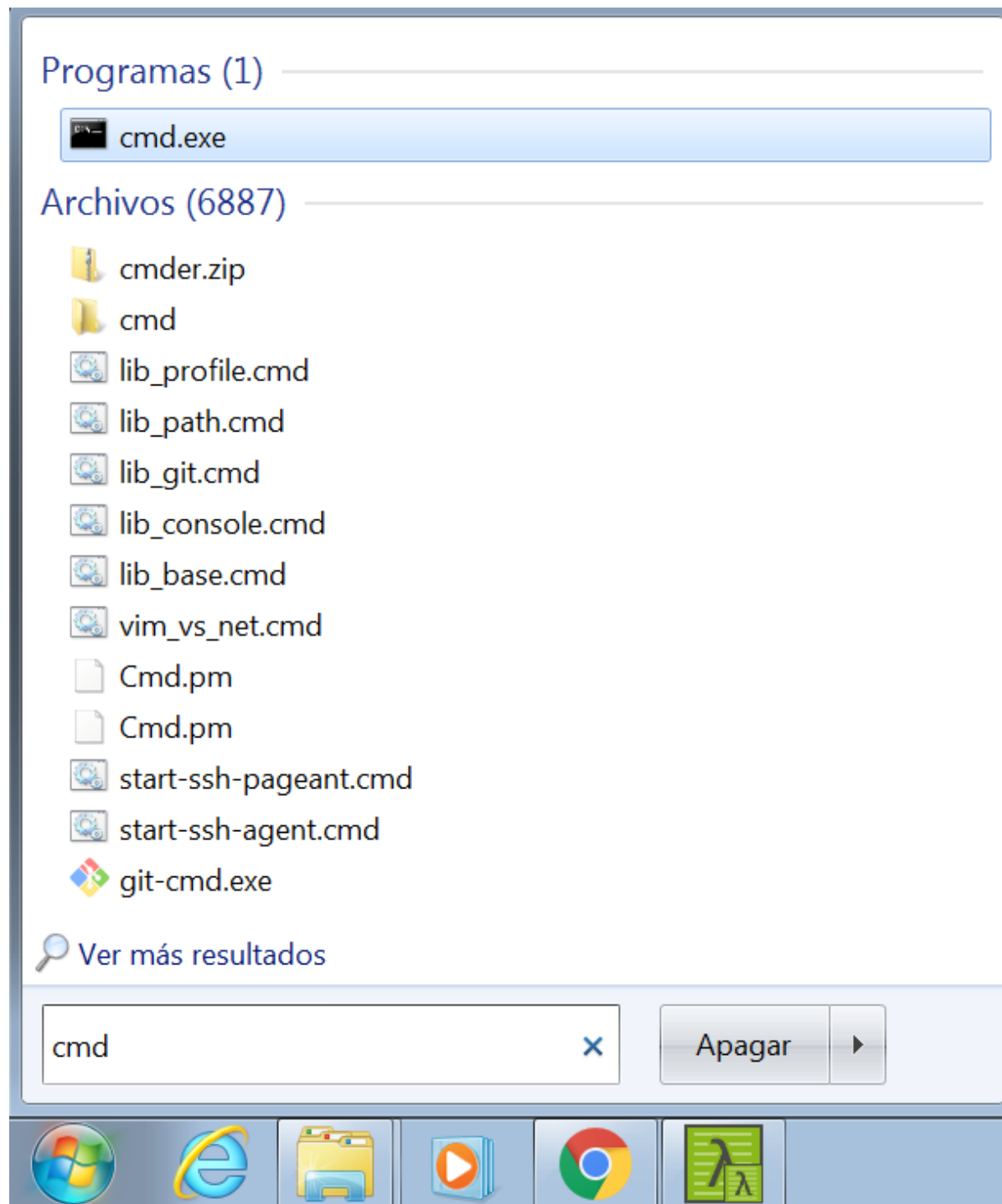
Ejemplo en Ubuntu:

```
cristian-CX62-7QL: ~/Desktop
File Edit View Search Terminal Help
@cristian-CX62-7QL:~$ cd Desktop/
@cristian-CX62-7QL:~/Desktop$ ls
archivo1.js  archivo2.js  archivo3.js  archivo4.js
@cristian-CX62-7QL:~/Desktop$
```

¡Ahora podemos ver todo lo que hay dentro de Desktop! Puedes probar a moverte por los archivos y carpetas de tu ordenador para ir mejorando tu soltura con la terminal 🚀

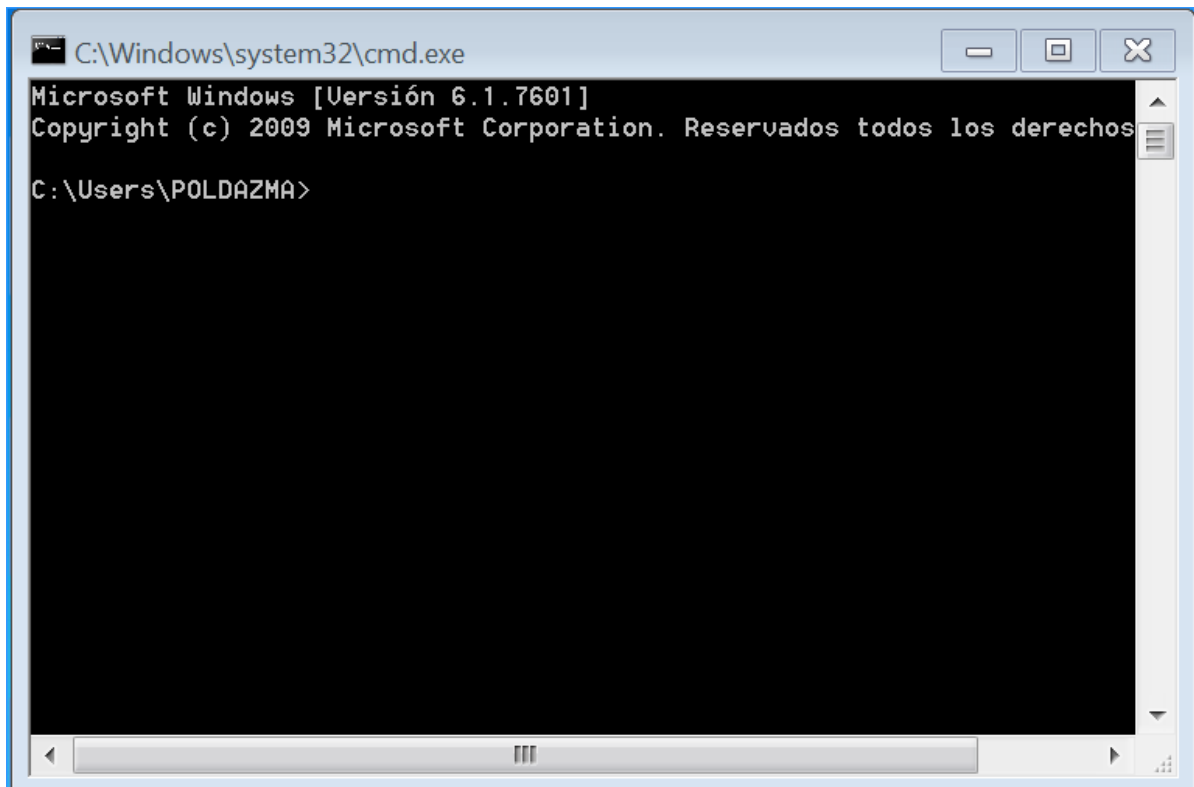
▼ Uso de la terminal en Windows

Para usar la terminal en Windows, vamos a abrir el buscador de archivos y a escribir **cmd**, cuando aparezca el siguiente icono clickaremos sobre él para abrirlo:



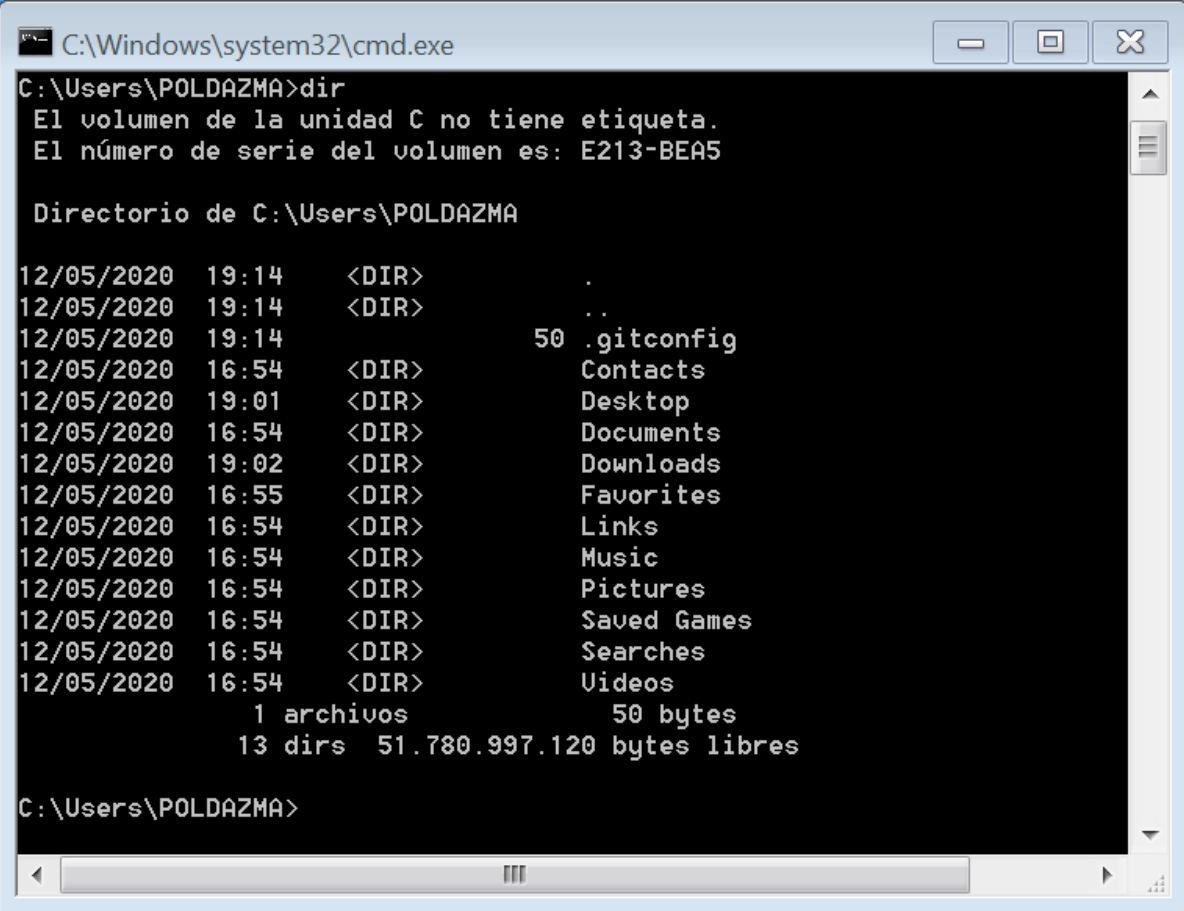
¿Dónde nos encontramos al abrir la terminal?

Nos encontraremos en la "base" de nuestro ordenador, en el punto de entrada de todas las carpetas importantes que contienen todo lo que tenemos en nuestro disco duro, es decir, en nuestra carpeta de usuario, donde tenemos las carpetas **Escritorio, Documentos, Descargas...**



¿Cómo puedo ver el contenido de un directorio?

Utilizaremos el comando **dir** directamente en la terminal:



```
C:\Windows\system32\cmd.exe

C:\Users\POLDAZMA>dir
El volumen de la unidad C no tiene etiqueta.
El número de serie del volumen es: E213-BEA5

Directorio de C:\Users\POLDAZMA

12/05/2020  19:14    <DIR>          .
12/05/2020  19:14    <DIR>          ..
12/05/2020  19:14             50  .gitconfig
12/05/2020  16:54    <DIR>          Contacts
12/05/2020  19:01    <DIR>          Desktop
12/05/2020  16:54    <DIR>          Documents
12/05/2020  19:02    <DIR>          Downloads
12/05/2020  16:55    <DIR>          Favorites
12/05/2020  16:54    <DIR>          Links
12/05/2020  16:54    <DIR>          Music
12/05/2020  16:54    <DIR>          Pictures
12/05/2020  16:54    <DIR>          Saved Games
12/05/2020  16:54    <DIR>          Searches
12/05/2020  16:54    <DIR>          Videos
                1 archivos             50 bytes
               13 dirs 51.780.997.120 bytes libres

C:\Users\POLDAZMA>
```

¿Y cómo puedo acceder a una carpeta?

Para acceder a una carpeta, por ejemplo "Desktop" vamos a usar el comando **cd** seguido del **nombre de la carpeta**, como hacemos a continuación:

```
C:\Users\POLDAZMA>cd Desktop
C:\Users\POLDAZMA\Desktop>
```

CMDER / Bash

Si alguno quiere tener una terminal más completa en Windows, puede explorar otras vías como:

- CMDER: <https://cmder.net/>
- Bash for Windows

¡Ahora podemos ver todo lo que hay dentro de Desktop! Puedes probar a moverte por los archivos y carpetas de tu ordenador para ir mejorando tu soltura con la terminal 🚀

Git y GitLab

¿Qué es y para qué instalar Git?

Cuando trabajamos con mucho código, ya sea por nuestra propia cuenta o en colaboración con más personas, tendremos problemas de sincronización en alguno momento.

¡Imaginad por un momento el quebradero de cabeza que es tener archivos y archivos que no coincidan en el ordenador de cada desarrollador del equipo! 🤯

Como esto ha sido algo común desde siempre, hubo que ponerle una solución, y para ello se usan los Controladores de Versiones, en nuestro caso el conocido **Git**.

Ya que tenemos unas sesiones específicamente indicadas para aprender **Git** y convertirnos en verdaderos *ninjas* del código, vamos a instalarlo y dejarlo preparado desde aquí.

Para instalar el software de Git en nuestros equipos, basta con descargarnos el **instalador** de la página oficial <https://git-scm.com/downloads> y ejecutarlo.

También existe una **guía** que nos puede servir de ayuda en <https://git-scm.com/book/en/v2/Getting-Started-Installing-Git> (y en caso de que necesitemos traducirlo, *Google Translate* es nuestro amigo).

Cuándo finalicéis la instalación, abrid un nuevo terminal y ejecutad los siguientes comandos para comprobar que la instalación ha sido correcta, y de paso configuraremos nuestro usuario (recordad que introduciremos nuestro nombre de usuario y el email con los que nos registremos en GitLab):

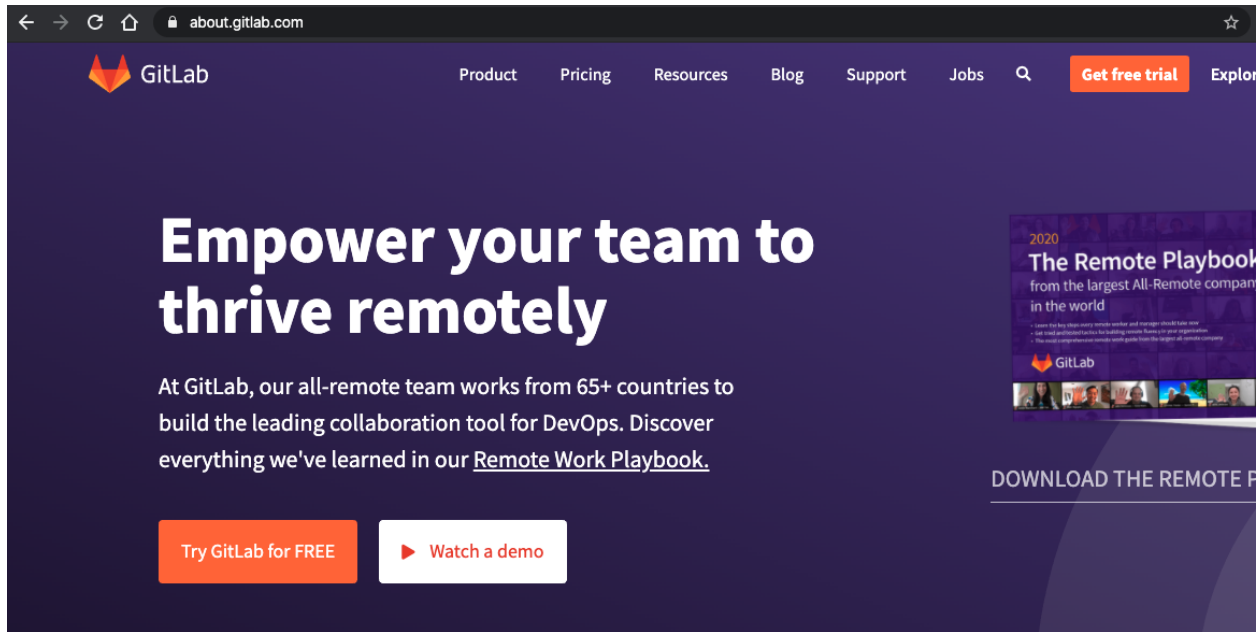
```
git --version
git config --global user.name "i-am-iron-man"
git config --global user.email "iron-man@avengers.com"
```

¡Vamos con GitLab!

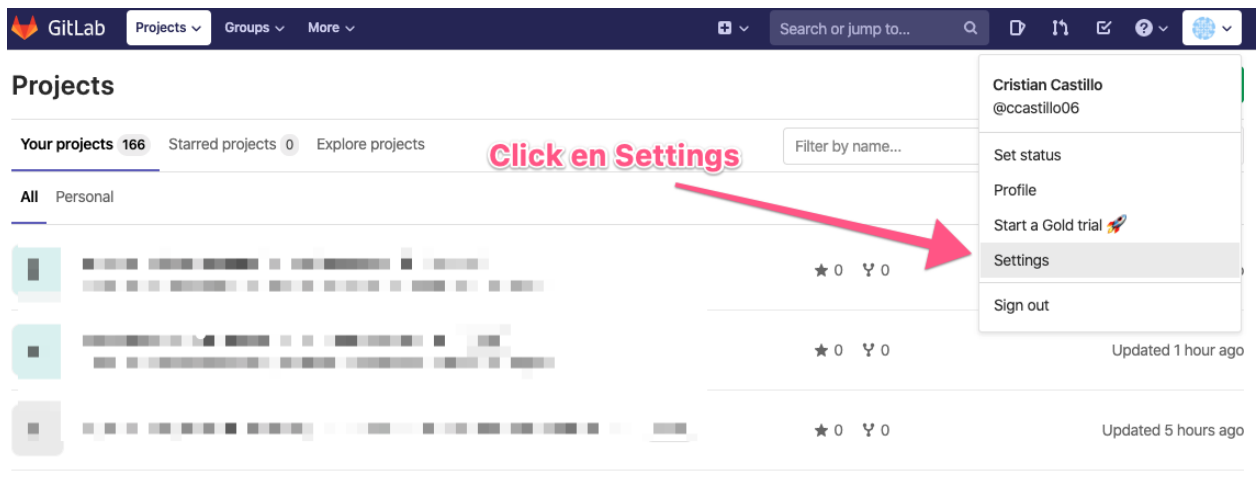
- **Si tienes problemas con esta sección y no eres capaz de crear las claves SSH, no te preocupes, lo volveremos a ver dentro del bootcamp y esto solamente es un adelanto. Con haber hecho la sección anterior tienes de sobra para poder completar los ejercicios del prework.**

¡NO TE QUEDES ATASCAD@ EN ESTA SECCIÓN, AVANZA CON EL PREWORK TODO LO QUE PUEDAS!

Tener nuestro código en el ordenador pero no compartirlo con nadie ni guardarlo en la nube no sirve de nada, ¿verdad? Para tener todo bien guardado y poderlo enviarlo a nuestros compis o profesores, vamos a registrarnos en **GitLab**.



Clickaremos en **Register** y crearemos una cuenta con nuestro email. Una vez creada, al entrar en Gitlab podremos ver una ventana de Proyectos, pero antes de nada iremos a **Ajustes o Settings** para configurar nuestra clave **SSH** y conectar nuestro ordenador con nuestra cuenta de **GitLab**.



Una vez dentro, iremos a la barra de navegación a la izquierda de la pantalla y entraremos en **SSH Keys**. Cuando estemos en ese panel de control veremos un cuadro donde podremos introducir nuestra clave, pero antes de nada, tenemos que generarla 🧑🏻💻🧑🏻💻

Pasos para crear una clave SSH:

- Abre una terminal e introduce el siguiente comando (cambia el email por el que has usado para registrarte en GitLab) y dale a **Enter** varias veces para aceptar las opciones por defecto:

```
ssh-keygen -t ed25519 -C "email@example.com"
```

- Cuando la hayamos generado, vamos a copiarla en el portapapeles, según el sistema operativo usaremos unos de los siguientes comandos:

¡Nota! Para poder usar el comando en Windows tendrás que instalar previamente Git Bash, que permitirá extender las funcionalidades de tu terminal: <https://git-scm.com/download/win>

→ **Selecciona la opción** `Use Git and optional Unix tools from the Command Prompt` **en el proceso de instalación cuando llegues a la ventana de configuración del** `PATH`.

→ **Selecciona “Use the OpenSSL library” para HTTP Transport.**

Para el resto de opciones de instalación usa las que tenemos por defecto.

Para macOS:

```
pbcopy < ~/.ssh/id_ed25519.pub
```

Para Linux/Ubuntu (instalaremos primero xclip y luego la copiaremos):

```
sudo apt-get install -y xclip
```

```
xclip -sel clip < ~/.ssh/id_ed25519.pub
```

Para Windows:

```
cat ~/.ssh/id_ed25519.pub | clip
```

- Por último, la pegamos en GitLab y guardamos. ¡Con esto deberíamos tener todo listo!

User Settings > SSH Keys

SSH Keys

SSH keys allow you to establish a secure connection between your computer and GitLab.

Add an SSH key

To add an SSH key you need to [generate one](#) or use an [existing key](#).

Key

Paste your public SSH key, which is usually contained in the file '~/.ssh/id_ed25519.pub' or '~/.ssh/id_rsa.pub' and begins with 'ssh-ed25519' or 'ssh-rsa'. Don't use your private SSH key.

1. Pegar Key



PEGAREMOS LA KEY AQUÍ! 📄

2. Ponerle nombre



Title

e.g. My MacBook key

Give your individual key a title

Expires at

dd/mm/aaaa

3. Guardar



Add key

¡Recuerda! Esta sección nos ayudará a facilitar todo el trabajo que hagamos en la sección de GIT, por lo que te recomendamos completarla antes de avanzar. Pero si tienes problemas con ello y no consigues hacer las claves o te parece muy complicado, pasa a la siguiente sesión porque lo veremos de nuevo dentro del bootcamp y solucionaremos tus dudas.

Node y Node Version Manager

¿Qué es y para que sirve Node?

Sabemos que JavaScript se usa para crear interacciones entre los usuarios y el navegador pero, ¿y si te dijésemos que también podemos usarlo para modificar archivos en nuestros ordenadores e incluso ejecutar código en un servidor? 🔥

Para estas nuevas formas de utilizar JavaScript, tenemos **Node**, un entorno que, una vez instalado, permite lanzar funciones de JavaScript utilizando un motor de ejecución llamado V8.

Antes de explicar cómo utilizarlo, vamos a instalarlo y dejarlo listo en nuestros dispositivos, así lo tendremos todo preparado para cuando lleguemos al módulo correspondiente.

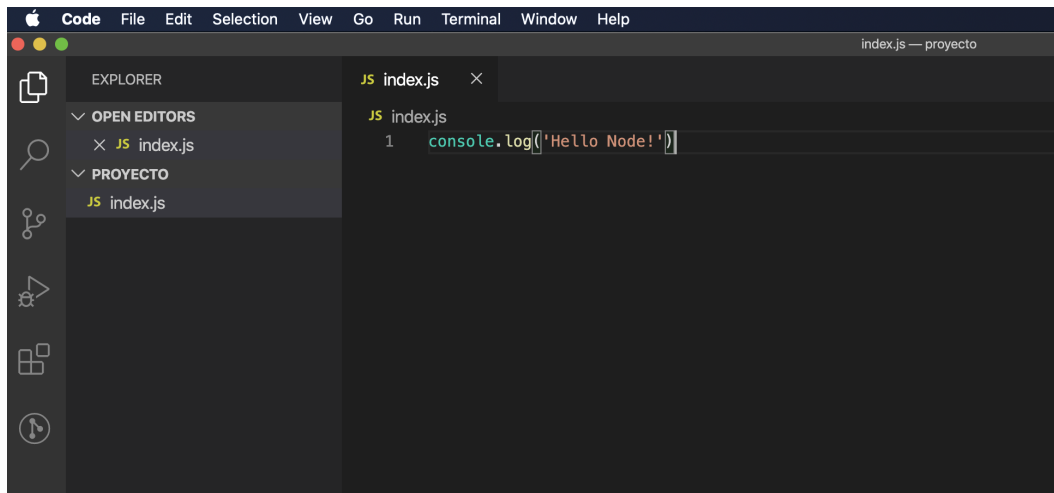
Basta con instalar su versión LTS desde el siguiente enlace

<https://nodejs.org/en/#home-downloadhead>

Con el paquete de Node, también se nos incluye el gestor de paquetes **NPM** para facilitar nuestro desarrollo y el uso de paquetes adicionales en JavaScript.

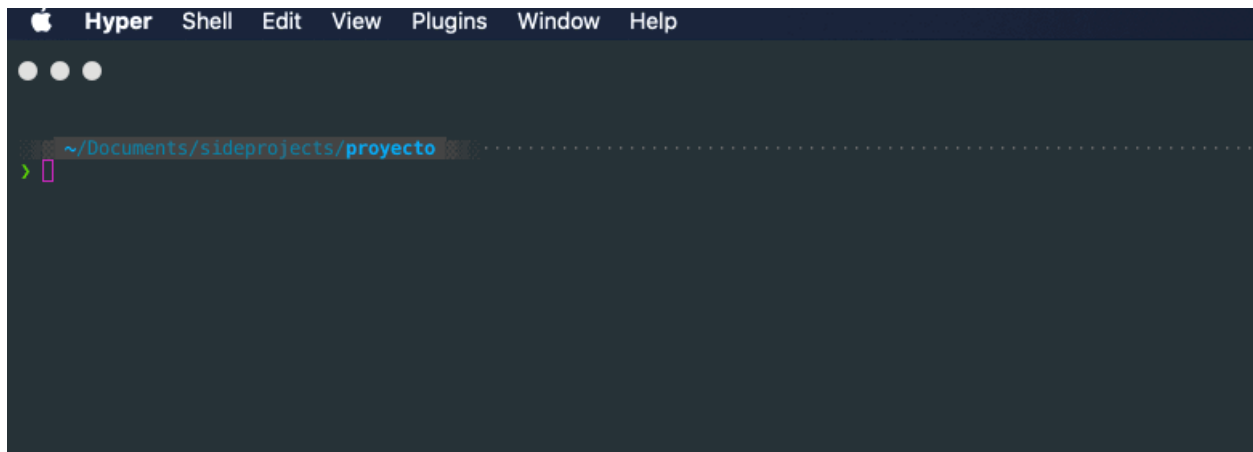
Probemos Node con un pequeño ejercicio

Una vez instalada la versión, vamos a probar si funciona bien. Para ello, crearemos un archivo `example.js` donde escribiremos `console.log('Hello Node!')` en una carpeta de nuestro ordenador, y desde la terminal nos moveremos al directorio de ese archivo.



Cuando estemos en ese directorio, usaremos el comando `node example.js` y deberíamos ver en la terminal el mensaje **Hello Node!**

GIF lanzando el comando en nuestra terminal:



Node Version Manager (BONUS) [Mac/Linux]

- ¡Esta sección es únicamente para Mac y Linux! Si tienes Windows, veremos esto en el bootcamp si es necesario, aunque recuerda, usar Ubuntu es tu

mejor opción para dedicarte al desarrollo y lo recomendamos encarecidamente.

Para los que utilicen maquinas en entornos Unix (Mac/Linux), podemos dar un paso más y en vez de instalar Node, tenemos **NVM**, una herramienta capaz de mover un entorno de Node en nuestro ordenador, ¡con la capacidad de cambiar y descargar versiones nuevas y antiguas al vuelo!

Aquí tenemos el repositorio de Github donde viene explicado como instalarlo:
<https://github.com/nvm-sh/nvm>

Usando este comando, lo instalaremos directamente y deberíamos poder usarlo sin problemas. Si surge algún imprevisto hablaremos con un mentor para que nos ayude, ya que la dificultad del problema que pueda aparecer será muy variable.

```
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.35.3/install.sh | bash
```

¡Vamos a probarlo!

- Abre una terminal y escribe:

```
nvm
```

- Si aparece contenido y ningún error, hemos instalado todo correctamente, debería verse algo parecido a esto:

Node Version Manager (v0.35.2)

Note: <version> refers to any version-like string nvm understands. This includes:

- full or partial version numbers, starting with an optional "v" (0.10, v0.1.2, v1)
- default (built-in) aliases: node, stable, unstable, iojs, system
- custom aliases you define with `nvm alias foo`

Any options that produce colorized output should respect the `--no-colors` option.

Usage:

nvm --help	Show this message
nvm --version	Print out the installed version of nvm
...	

- Ahora vamos a instalar una version de Node. Para ello lanzaremos el siguiente comando, que instalará la versión más reciente soportada y estable en nuestro dispositivo:

```
nvm install --lts
```

Con todo ello, podríamos trabajar con Node exactamente igual que instalándolo directamente, solo que con el potencial de usar diferentes versiones de Node, sin tener que andar instalándolas (sobreescribiendo la versión en nuestro ordenador).

¡Con esto tenemos todas las herramientas necesarias en nuestros dispositivos para completar el prework paso a paso! Igualmente durante el curso iremos instalando algunas cosillas más.

Seguimos aprendiendo en:

 [GIT | Intro](#)

