**Comandos para manejo de archivos y directorios:**

* **ls**: Nos permite listar los archivos y directorios que se encuentren dentro de la carpeta en la que estamos ubicados, podemos pasarle distintos parámetros a este comando:
* **-a:** Podemos ver los archivos ocultos.
* **-l:** Nos lista los contenidos mostrando sus permisos y propietarios.
* -**t:** Nos lista los contenidos según su fecha.
* **clear**: Nos limpia la pantalla.
* **pwd**: Nos retorna la ruta absoluta en la cual nos encontramos.
* **mkdir**: Crea una carpeta.
* **cd**: Nos mueve a alguna carpeta que le indiquemos, dentro de los archivos ocultos vimos que existe:
* **.:** Refiere a la carpeta en la cual estamos ubicados.
* **..:** Se refiere a la carpeta padre en la cual nos encontramos.
* **history**: Muestra el histórico de todos los comandos que hemos ejecutado.
* **touch**: Crea un archivo vacío con el nombre que le indiquemos.
* **nano**: Es un editor dentro de la consola, podemos abrir cualquier archivo y modificarlo.
* **mv**: Permite mover archivos entre distintas carpetas, solamente debemos indicarle el nombre del archivo y la ruta de destino.
* **rm:** Elimina únicamente un archivo, añadiendo el parámetro -rf podemos eliminar directorios también.

**Herramientas Básicas para la terminal:**

* **cat:** Permite visualizar un archivo completo en la terminal. Además, permite copiar archivos, agregando el signo > después de escribir el nombre del archivo original.
* **more:** Muestra por partes un archivo dentro de la terminal.
* **tail:** Muestra las últimas 10 líneas de cada archivo, se puede modificar pasándole el parámetro con el número de líneas -15.
* **open:** Abre un archivo con el programa que tengamos por defecto.

**Crea llaves SSH**

Las llaves SSH nos van a ayudar para autentificarnos con servidores. SSH utiliza criptografía asimétrica, o sea, tenemos dos llaves:

* Pública: la llave pública la podemos compartir por internet.
* Privada: debes tenerla en un sitio seguro y no debe ser compartida.

Tener una llave SSH nos permitirá una conexión fácil y segura con servidores, en el caso de la escuela de JavaScript nos va a servir para conectarnos con GitHub.

Para crear una llave SSH utilizamos el siguiente comando:

* ssh-keygen -t rsa -b 4096 -C llave, puede ser tu correo>
* RSA es el algoritmo para crear las llaves, tiene las ciclas de las personas que crearon el algoritmo (Rivest, Shamir y Adleman).
* -b indica la cantidad de bits que tendrá la llave



**¿Qué es NodeJS?**

Node es el entorno de ejecución que tenemos para JavaScript en el lado del servidor, está basado en el motor V8 de Google Chrome. Fue creado por Ryan Dahl en el 2009, es Open Source y multiplataforma. En esta clase vamos a aprender cómo instalarlo, cómo usarlo y cómo instalar paquetes usando npm.

**Cómo utilizar npm**

npm es el manejador de paquetes de Node con él podemos instalar dependencias a nuestro proyecto o instalar programas globalmente en nuestro sistema. A lo largo de este curso y de toda la Escuela de JavaScript npm será quien nos permita correr los proyectos e instalar nuestras dependencias.

**Instalación y configuración de VSCode**

Si la primera mejor amiga del programador es la línea de comandos, es momento de instalar y configurar el segundo mejor amigo del programador: el editor de código.

Existen multiples editores de código, para la escuela de JavaScript vamos a utilizar Visual Studio Code. Vamos a añadir diferentes plugins para VSCode:

Git Blame: va a mostrar el autor de la línea de código en la que estemos trabajando.

ESLint: es una herramienta de análisis de código estático para identificar patrones problemáticos encontrados en el código JavaScript, o sea, nuestro linter. Debemos instalar y configurar eslint para que siga el estilo de código que le indiquemos.

Color Highlight: resalta el color que estemos escribiendo.

SASS: es un preprocesador de CSS.

**Plugins útiles para Visual Studio Code**:

* Git Blame: Permite ver qué persona hizo que línea de código.
* EsLint: Este plug-in además de instarlo, hay que hacer una configuración global en el sistema con el comando **npm install -g eslint**, para inicializar la configuración de eslint en nuestro proyecto hay que utilizar otro comando, **npx eslint –init.**
* Color Highlight: resalta el color que estemos escribiendo.
* SASS: es un preprocesador de CSS.

**¿Qué es EsLint?**

Es una herramienta que nos ayuda a verificar como estamos escribiendo nuestro código mostrando los errores que tenemos en el.

**¿Qué es Git, para qué se usa y qué resuelve?**

Git es un sistema de control de versiones que nos permite llevar un histórico sobre los cambios de nuestro proyecto, no es el único sistema de control de versiones, pero sí el más usado. Fue creado por Linus Torvalds. Git y GitHub no son lo mismo, uno es el sistema de control de versiones y el otro es la red social de programadores.

Los repositorios son una estructura de datos que almacenan información sobre archivos y directorios. Es el inicio de todo proyecto con Git, dentro de un repositorio encontraremos ramas, no son más que la duplicación de un objeto bajo un repositorio, permite trabajar en paralelo para al final unir los cambios.

**Cómo crear un repositorio, primer commit, reset y logs**

Aprende el flujo básico de Git: crear un repositorio, crear un commit y ver un histórico de los commits.

Para comenzar con un nuevo repositorio en Git debemos correr el siguiente comando:

* git init

Al correr el comando nuestra terminal nos va a mostrar que nos encontramos dentro de la rama master, la rama principal de todo proyecto en Git. Además, si ejecutamos ls -la veremos que hay una carpeta oculta llamada “.git”.

Todo cambio tiene varios estados dentro de Git:

* Sin seguimiento
* Sin cambios
* Con cambios
* En stagging

Para ver el estado del repositorio ejecutamos el comando git status. Podemos añadir un archivo con el comando git add <nombre del archivo>, una vez lo tenemos añadido podemos dar commit con el comando git commit -m <mensaje del commit>. Con git log podemos visualizar un histórico de los commits.

Dentro de Git es posible regresar entre commits con el comando git reset, tenemos dos opciones para regresar:

* --soft: vamos a movernos al commit que le indiquemos, sin eliminar los commits de por medio.
* --hard: nos movemos al commit que indiquemos y regresamos el repositorio al estado del commit, borrando todos los commits de por medio.

**Ramas, rebase y merge**

Recuerda que una rama es la duplicación de un objeto sobre el repositorio y nos va a permitir trabajar en paralelo para después unir los cambios a nuestro proyecto, en este caso a nuestra rama master, los comandos principales son:

* **git checkout -b develop**: según el commit en el cual ejecutemos este comando va a ser el punto en el cual se va a crear una rama idéntica, en este caso con el nombre de “develop”.
* **git merge develop**: va a añadir los commits a la rama master.
* **git rebase develop**: va a añadir los commits a la rama master unificando ambas ramas y conservando la historia de la misma.

**Github: configuración, repositorio remoto, push y pull**

GitHub es la red social de programadores, como todo buen programador necesitas desplegar tus proyectos desde la terminal, para ello debes añadir una llave SSH a tu cuenta de GitHub. Dentro de GitHub crear un repositorio es bastante rápido y sencillo, puedes elegir si va a ser un repositorio público o privado y el tipo de licencia que va a tener tu proyecto, si es un proyecto open source lo mejor es añadir una licencia MIT.

**¿Qué es README.md?**

El README es el archivo en el cual hacemos la descripción del proyecto, ya sea open source o privados es importante tener un buen README. Este archivo se escribe con formato markdown, el cual es un formato de escritura que permite la generación de contenido fácil y rápido, permite generar una salida (por lo general), en formato HTML sin necesidad de aprender a profundidad HTML. Es ampliamente utilizado por su facilidad de generar texto enriquecido.

<https://platzi.com/clases/1650-prework/21969-como-crear-un-buen-readmemd-y-sintaxis-de-markdown/>