Guía de ejercicios 5 - Terminal, Git y Github

 ¡Hola! Te damos la bienvenida a esta nueva guía de ejercicios.

¿En qué consiste esta guía?

En esta guía podrás realizar operaciones de navegación de directorios, usando los comandos básicos del terminal, para crear y manipular archivos y directorios.

Además, comprenderás que el terminal es una poderosa herramienta basada en una interfaz de texto que sirve para comunicarse directamente con un computador. Utiliza líneas de comandos para navegar por archivos y directorios, también se utiliza para interactuar con programas que no tienen interfaz gráfica.

Finalmente, también aprenderás a realizar con éxito las siguientes tareas:

- Ingresar al terminal.
- Utilizar el terminal para moverse entre directorios utilizando cd, ls y pwd.
- Inicializar un proyecto con git.
- Añadir archivos a git y confirmar cambios.
- Realizar un fork de un proyecto en github.
- Dividir el espacio de la página en columnas de distinto tamaño y que se ajusten a los distintos tamaños de dispositivo.

¡Vamos con todo!

¡Comencemos!

Inicialización de terminal

Para inicializar el terminal según los diferentes sistemas operativos utilizaremos los siguientes atajos:

- **En Linux:** Presiona `ctrl + alt + t`.
- **En Mac:** Presiona `⌘ + espacio`, busca por spotlight terminal.
- **En Windows:** Presiona `inicio(tecla de windows) + r`, escribe "cmd" en la caja de texto y presiona aceptar.

No obstante, considerando que no todas las terminales utilizan las mismas líneas de comando, para homologar los procesos que aprenderemos a hacer más adelante ocuparemos en windows un programa especial llamado git Bash, el cual se instala por defecto cuando instalas Git en tu sistema computador.

El árbol de directorios

Para movernos de directorio en el terminal es importante que conozcamos la estructura de directorios. Las carpetas de nuestro computador tienen una estructura del tipo árbol donde existen directorios y subdirectorios, esto quiere decir que el árbol de directorios comienza en la raíz y contiene ramas o directorios, al mismo tiempo que al interior de estos directorios pueden existir archivos u otros directorios.

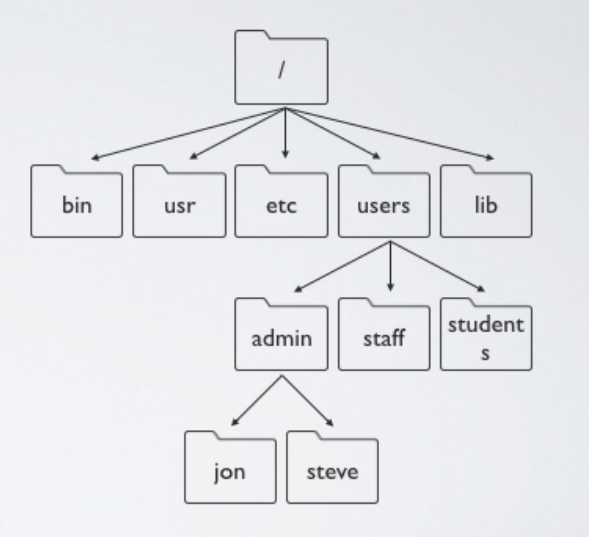

Imagen 1. Estructura de escritorio.
Fuente: Desafío Latam.

- A veces se utiliza el término técnico **nodo** para referirse a un archivo o un directorio en algún punto del árbol.
- El nodo principal de tu computador regularmente se identifica con el símbolo /, también llamado raíz.
- Todos los directorios y/o archivos de nuestro computador están dentro de este súper directorio general.
- La estructura de directorios puede cambiar dependiendo del sistema operativo con el cual estés trabajando, así que puede que en tu computador no veas los mismos directorios.

Conocer en qué directorio estamos (pwd)

Una de las cosas más importantes que necesitamos saber al trabajar con la terminal, es conocer en qué directorio estamos trabajando. Para ello, existe un comando que nos entregará esta información.

Escribe en tu terminal lo siguiente `pwd` y presiona enter.

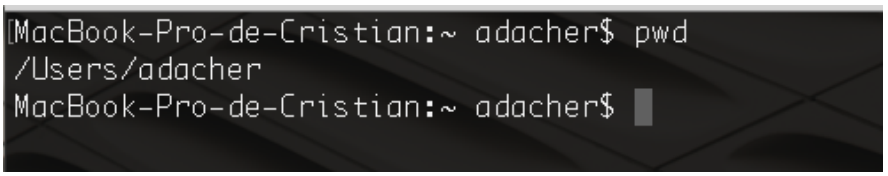

Imagen 2. Uso de `pwd`.
Fuente: Desafío Latam.

En tu consola se imprimirá la ruta en la cual estás posicionado, esto quiere decir que aparecerá como texto. Recuerda que siempre que escribamos un comando, necesitaremos presionar la tecla `enter` para que se ejecute.

Listar archivos (ls)

Observa los archivos o directorios que están en tu carpeta raíz o usuario, para ellos utilizaremos el comando `ls` de listar, este comando muestra una lista de los archivos y directorios contenidos en el directorio en el que se está ejecutando el comando.

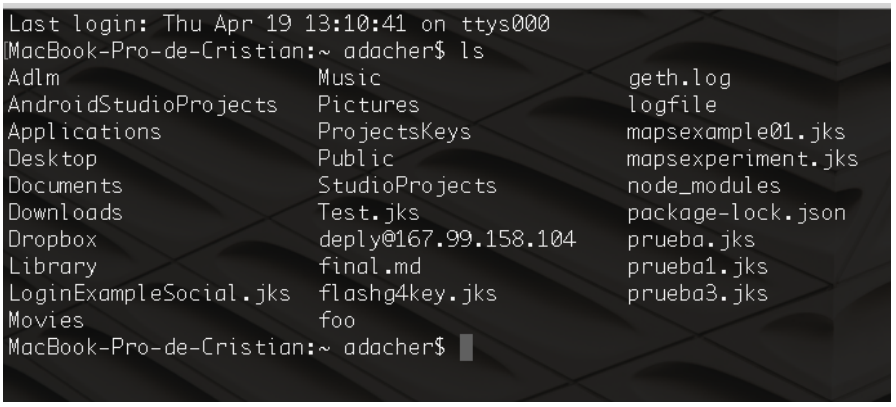

Imagen 3. Listar archivos.
Fuente: Desafío Latam.

Existe también una opción del comando `ls` que nos permitirá observar los archivos ocultos de un directorio. Para poder verlos debemos escribirlo así: `ls -a` el resultado será un listado de directorios y archivos ocultos que comienzan con un punto (.) antes de su nombre:

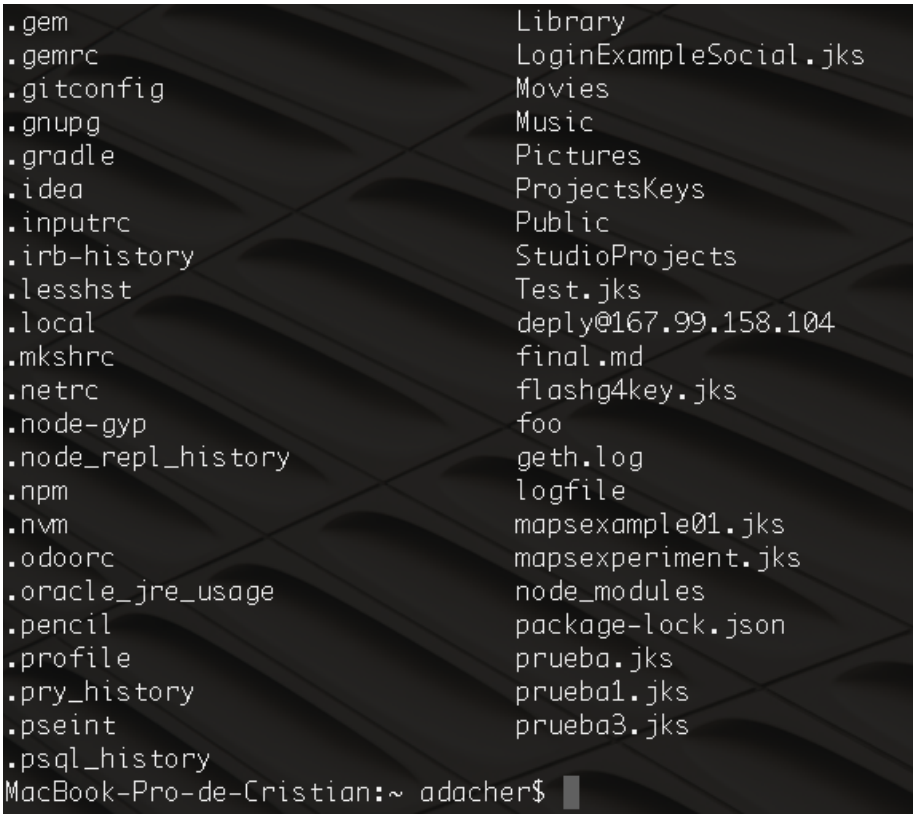

Imagen 4. Respuesta `ls -a`.
Fuente: Desafío Latam.

Comandos de navegación entre directorios (cd)

Para moverse de directorio utilizaremos el comando llamado `cd` lo que significa change directory.

Escribe cd / y presiona enter, en la terminal. Este comando te lleva a la raíz de tu usuario dentro del computador, lo que puedes comprobar escribiendo `pwd`. O sea, si quisiéramos volver al home de nuestro equipo, bastará con escribir `cd ~` y clickear enter en la terminal.

Si queremos recorrer e introducirnos en algún otro directorio con `cd` debemos escribir `cd` + la ruta al directorio que queremos llegar. Por ejemplo, accederemos a la carpeta `Desktop` o `Escritorio`, dependiendo de tu sistema operativo, escribiendo:

```
cd Desktop
```

Comprobaremos que estamos en la carpeta correcta escribiendo `pwd`.

Podemos, nuevamente, escribir `ls` para ver qué archivos o directorios hay dentro de esa carpeta Desktop e introducirnos en alguno de ellos con `cd`. Si ingresamos a un directorio y necesitamos volver atrás, podemos escribir lo siguiente: `cd ..` así seremos dirigidos hacia la carpeta contenedora.

Anatomía de un comando

Todos los comandos tienen un nombre que los distingue, por ejemplo `ls` y `pwd`, serían el nombre del comando. Algunos como `cd` además tienen uno o más argumentos, por ejemplo `cd carpeta`. En algunos comandos los argumentos son opcionales y en otros obligatorios.

Hay comandos que pueden recibir opciones, las cuales especificamos anteponiendo `-` o `--` al igual que con el comando `ls` que ya habíamos realizado, es decir, `ls -a`. En este caso el `-a` da la opción de ver archivos ocultos.

Importante

Linux es sensible a las mayúsculas. Esto implica que es distinto escribir `CD` o `cd`. OSX no lo es, pero tendremos esto en cuenta y para seguir la convención escribiremos todos los comandos en minúsculas.

Manejo de archivos y carpetas

A continuación, veremos algunos comandos útiles al trabajar con archivos y carpetas, que optimizarán nuestro tiempo una vez que adquiramos un poco de práctica.

Creación de archivos

Podemos crear un archivo nuevo directamente desde el terminal con `touch`. La forma de utilizarlo es `touch nombre_del_archivo.extension`, esto creará un nuevo archivo con el nombre que hayamos ingresado y la extensión que apunta al tipo de archivo.

```
touch archivo.extension
```

Creación de directorios

Para crear un directorio, se utiliza el comando `mkdir`, que significa make directory, de la siguiente manera:

```
mkdir directorio
```

Esto creará un nuevo directorio con el nombre que lo acompañe.

Copia de archivos

Para copiar archivos, se utiliza el comando `cp`, el nombre del comando, luego se añade el archivo que vamos a copiar, y finalmente la ruta donde queremos copiarlo, de la siguiente forma:

```
cp archivo.extension ruta_destino/archivo_nuevo.extension
```

Copia de directorios

Para copiar directorios, también se utiliza el comando `cp`, agregándole la opción `-r`, la carpeta a copiar y la carpeta de destino.

```
cp -r directorio_copiado directorio_destino
```

El -r indica que la copia es recursiva o sea que cada elemento dentro se copia y los que están dentro de un subdirectorio también.

Mover archivos y directorios

Para mover archivos, se utiliza el comando `mv` que significa move, el cual se utiliza de una forma muy similar que el anterior `cp`.

```
mv archivo.extension directorio_destino/
```

`mv` también nos permite renombrar el archivo que estemos moviendo, indicando el nuevo nombre en la ruta de destino.

Para mover un directorio utilizaremos la misma sintaxis que el comando `cp`:

```
mv directorio_origen directorio_destino
```

Borrar archivos

Otro comando importante que debemos conocer y manejar con mucho cuidado es `rm`, ya que con este comando podremos eliminar un archivo y con una opción en su sintaxis, también directorios completos.

Para eliminar un archivo utilizaremos el siguiente comando:

```
rm archivo.extension
```

¡Importante!

¡Ten mucho cuidado! Ya que los archivos eliminados de esta forma no van a parar en la papelera de reciclaje de tu computador.

Si por algún motivo escribimos el comando y no existe al interior del directorio, obtendremos la siguiente respuesta:

```
No such file or directory
```

Para eliminar un directorio completo, utilizaremos el mismo comando `rm` pero con una opción.

```
rm -r directorio
```

Esto eliminará por completo el directorio, por lo que debemos tener mucho cuidado al utilizar este comando.

Como ves, la terminal es una poderosa herramienta. Existen muchos comandos más que los que revisaremos, pero por el momento con lo aprendido ya puedes moverte libremente por los directorios, verificar en qué carpetas estás y además crear, copiar, mover y eliminar a través de la terminal.

Actividad 1: Manejando archivos y carpetas

Como vimos en las páginas anteriores, existen múltiples comandos que nos ayudarán a navegar a través de los archivos y directorios, facilitando nuestro trabajo. Para familiarizarnos con ellos, haremos un recorrido por los comandos que acabamos de revisar:

- ***Paso 1:*** crea un nuevo directorio con un comando. Para ello, dirígete a la carpeta raíz de tu computador. Si no estás en esta carpeta, ve a ella escribiendo `cd`.

Ahora escribe en la consola:

```
mkdir proyecto1
```

Esto creará un nuevo directorio llamado `proyecto1`.

- ***Paso 2:*** crea un archivo desde la terminal con el comando touch, con el nombre `index.html`:

```
touch index.html
```

Esto crea el archivo `index.html` en la ubicación actual, o sea la carpeta raíz del computador. Si utilizamos `ls` podremos ver el archivo creado.

- ***Paso 3:*** ahora que ya sabes crear un directorio y un archivo, copia el archivo `index.html` dentro del directorio que habías creado, o sea en la carpeta `proyecto1`, de la siguiente manera:

```
cp index.html /proyecto1/index.html
```

- ***Paso 4:*** utilizando este mismo comando, también puedes cambiarle el nombre a los archivos copiados. Vuelve al directorio anterior con `cd ..`, luego ocuparás `cp` nuevamente pero con un nuevo nombre de archivo.

```
cp index.html /proyecto1/index2.html
```

Si ingresas a `proyecto1` y listamos los archivos con `ls`, debería aparecer el archivo copiado con el nuevo nombre.

- ***Paso 5:*** si quieres copiar un archivo dentro de la misma carpeta donde se encuentra, solo debes escribir el nombre del archivo seguido del nombre que le quieres poner.

```
cp index.html index3.html
```

Resumen

- El terminal es una herramienta basada en una interfaz de texto que sirve para comunicarse directamente con un computador. Utiliza líneas de comandos para navegar por archivos y directorios, al mismo tiempo se utiliza para interactuar con programas que no tienen interfaz gráfica.

- Los directorios o carpetas de nuestro computador tienen una estructura del tipo árbol, es decir, comienzan en la raíz y contienen ramas o directorios, al mismo tiempo que al interior de estos directorios pueden existir archivos u otros directorios.

- Todos los comandos tienen un nombre que los distingue, por ejemplo `ls` y `pwd`, serían el nombre del comando.

Introducción a Git

Git es un sistema de control de versiones gratuito, muy útil y ampliamente utilizado en el desarrollo. Fue creado con la idea de ayudar a manejar proyectos sin importar su tamaño.

Es usado por grandes empresas de desarrollo, como podemos observar en su sitio web.

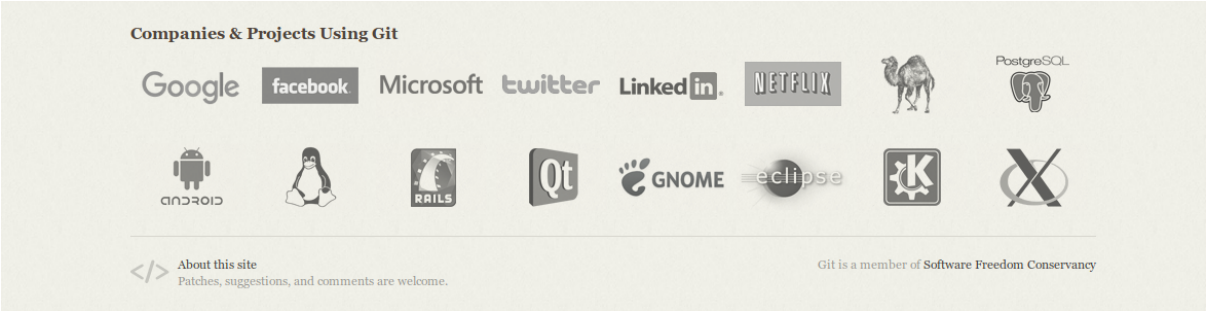

Imagen 5. Sitios que utilizan Git.
Fuente: Openwebinars.

Según la encuesta anual realizada por *stack overflow,* Git es el amplio líder de los sistemas de control de versiones, ocupando un 90% de las preferencias de los desarrolladores. (Fuente Stackoverflow).

Existen muchas razones para utilizarlo, ya que es un potente software de control de versiones y nos permitirá:

- Recuperar versiones anteriores de nuestro código;
- Recuperar archivos borrados;
- Ayudar a gestionar cambios realizados por otras personas;
- Administrar un proyecto donde trabajan múltiples desarrolladores.

Además, git nos permitirá subir nuestras páginas web a GitHub y crear un portafolio profesional como desarrollador.

Control de versiones

Para entender mejor qué es un sistema de control de versiones, imaginemos un editor de documento de texto como el proporcionado por **Google**, en el cual vamos añadiendo cambios y guardándolos. Si cerramos el programa solo tendremos los últimos cambios guardados, y utilizando git tendríamos acceso a todas las versiones guardadas, permitiendo incluso volver a una de ellas.

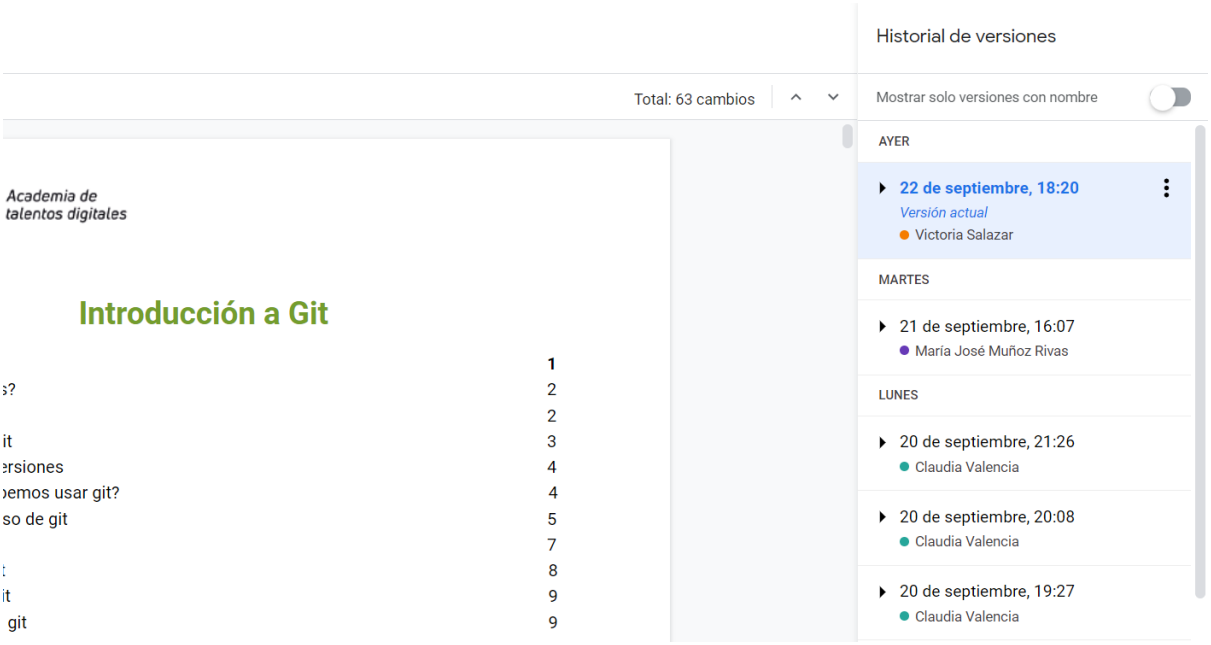

Imagen 6. Ejemplo de un control de versiones en Documentos de Google.
Fuente: Desafío Latam.

Git nos permite hacer lo mismo, pero con muchas ventajas adicionales.

¿Cuándo debemos usar Git?

La recomendación es usarlo **siempre** que trabajemos desarrollando código (o con documento en texto plano), ya que nos evitará realizar trabajo extra si ocurre algún problema como, por ejemplo, si borramos parte del código que pensábamos que no nos servía, pero luego nos dimos cuenta que sí.

Git también nos ayudará a hacer cambios en el sitio de forma ordenada sin poner en riesgo lo que ya está funcionando.

Durante esta experiencia utilizaremos git en uno de nuestros proyectos para manejar los cambios realizados con la finalidad de subir nuestro trabajo a una plataforma de colaboración o repositorio remoto, solo usando la terminal.

Formas de uso de Git

Existen distintas formas de trabajar con git. Algunos editores de texto traen incorporado formas automatizadas para usarlo, por ejemplo en Atom.

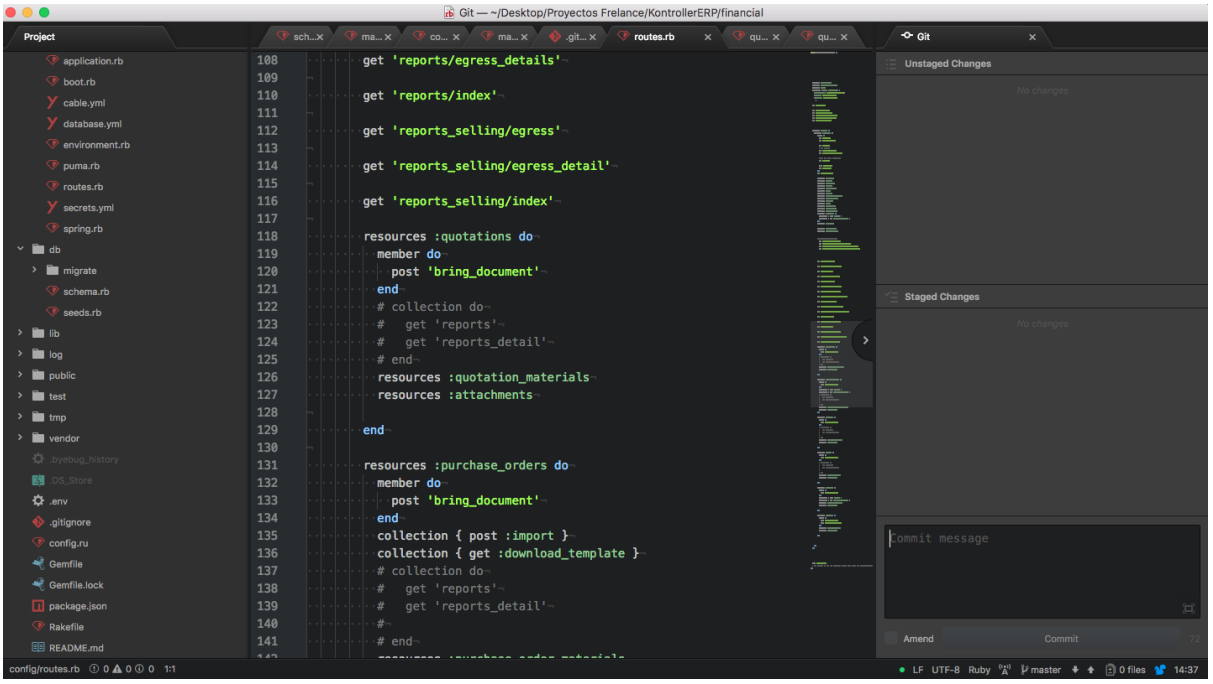

Imagen 7. Editor Atom.
Fuente: Desafío Latam.

También existen programas con interfaces gráficas como gitkraken o git Tower.

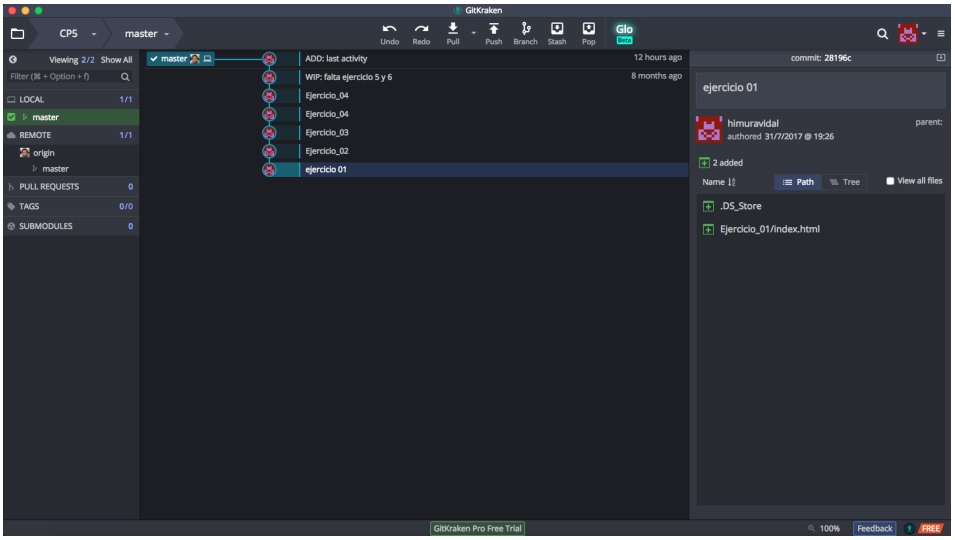

Imagen 8. Gitkraken.
Fuente: Desafío Latam.

Nosotros lo utilizaremos en nuestra terminal. Esto puede parecer a primera vista un poco más difícil, pero nos ayudará a entender mejor los conceptos más importantes.

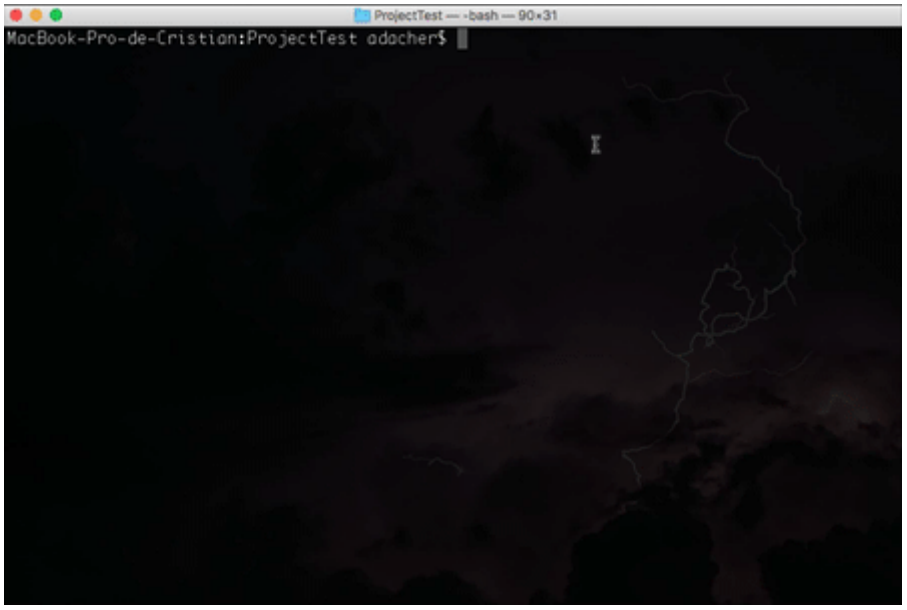

Imagen 9. Vista terminal.
Fuente: Desafío Latam.

Como has podido notar, Git es una herramienta ampliamente solicitada en el mundo del desarrollo, por lo tanto es bueno familiarizarse con ella.

Instalando Git

Vamos a instalar git en nuestro computador. Ahora veamos cuál necesitas dependiendo del sistema operativo que tengas (Mac/Linux, Windows).

Verificando si se encuentra instalado:

El primer paso es verificar si ya tenemos instalado git en nuestro sistema. Esto lo podemos realizar escribiendo el comando `git --version` en nuestra terminal.

Si está instalado, veremos que el terminal nos muestra algo como:

```
git version 2.14.3
```

En computadores con **OSX** es decir, computadores **Mac**, git viene instalado por defecto. Pero si por algún motivo no lo tienes, sigue los siguientes pasos:

1. Entra al sitio de git.
2. Descarga el archivo para **OSX**.
3. Ejecuta el archivo descargado y sigue los pasos del instalador.

En **Linux**, si no está instalado, debemos utilizar los siguientes comandos en la terminal.

```
sudo apt install git
```

Y esperar a que termine la instalación.

En **Windows**, si seguiste las instrucciones proporcionadas en la lectura para instalar el terminal, no deberías tener problemas. Pero, si por algún motivo no lo tienes, sigue los siguientes pasos:

1. Entra al sitio git/win.
2. Descarga el archivo dependiendo de tu versión del sistema operativo.
3. Ejecuta el archivo descargado y sigue los pasos del instalador.

Configurando Git

Ahora que ya tenemos **git** instalado, nuestro siguiente paso será configurarlo en nuestro equipo. Principalmente, lo que tenemos que configurar es nuestro usuario en git.

Para ello, utilizaremos los siguientes comandos:

```
git config --global user.name "Tu Nombre"
```

```
git config --global user.email tucorreo@mail.com
```

En el primer comando, debes ingresar tu nombre entre las comillas. Recuerda que este nombre será visible en cada interacción que realices.

Luego, debes ingresar tu correo electrónico, esta vez sin comillas, y también será un registro de las acciones que realices con git.

Una vez ingresados los comandos no veremos ninguna confirmación de la acción entonces puedes usar el comando.

```
git config --list
```

Y deberíamos ver dentro de la lista obtenida los siguientes dos elementos:

```
user.name=Nombre Apellido
```

```
user.email=micorreo@mail.com
```

Si ves este mensaje es porque lo lograste. Ahora que tienes instalado y configurado **git**, en el siguiente capítulo aprenderemos cómo añadirlo a nuestros proyectos.

Uso básico de git

Realizaremos una demostración de los pasos comunes para crear un proyecto con git y manejar cambios.

Inicializando git

Siempre que queramos trabajar con git, nuestro primer paso será escribir en la carpeta de proyecto lo siguiente:

```
git init
```

Todo lo que hace git lo realiza dentro de una carpeta oculta dentro del lugar donde fue inicializado. Si mostramos todos los archivos con `ls -a` podremos ver la carpeta `.git`. Todo ocurre de forma automática en el interior de este directorio.

> Es importante saber que la ejecución del comando `git init` solo lo debemos realizar una vez por proyecto.

Usando Git

El flujo básico de git consiste en agregar cambios y luego confirmarlos.

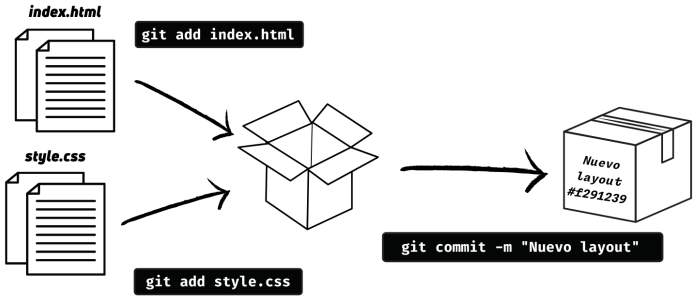

Imagen 10. Flujo básico de GIT.
Fuente: Desafío Latam.

La idea es similar al proceso de una mudanza, primero agregamos cosas (cambios) en una caja, y cuando estamos listos cerramos la caja con un mensaje.

Para guardar cambios en la caja utilizaremos `git add` cerrar la caja será lo mismo que confirmar los cambios, esto lo logramos con `git commit`

Los archivos tenemos que añadirlos para que git sepa que estos deben ser trackeados. Una vez que un archivo está trackeado y recibe alguna modificación pasará a estado modificado.

El siguiente diagrama nos muestra cómo cambia el estado del archivo al hacer distintas acciones.

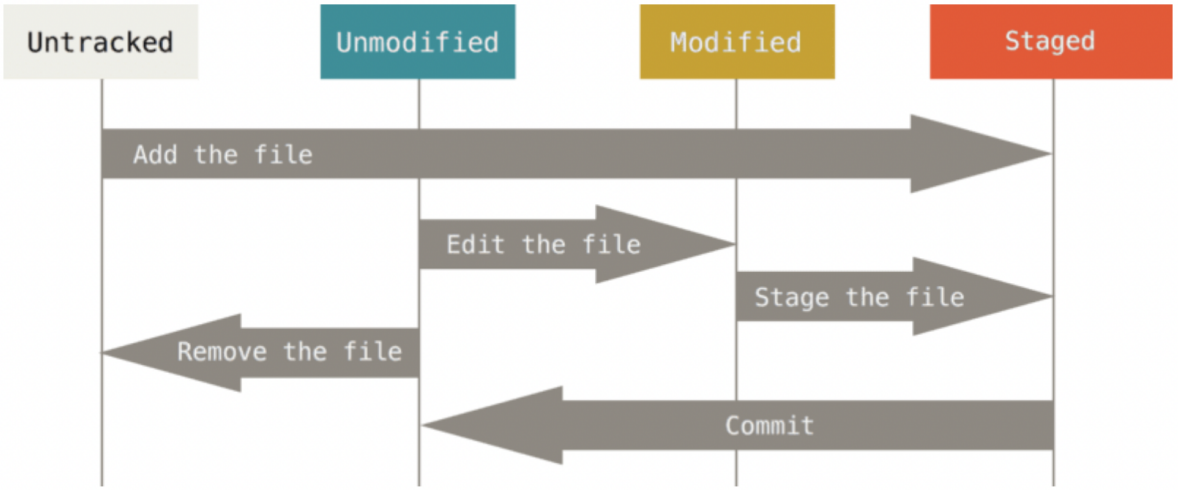

Imagen 11. Flujo básico de GIT.
Fuente: Documentación de GIT. https://git-scm.com/

Consideraciones:

1. Un archivo no trackeado pasa a staged (en preparación) via git add.
    a. Al principio de un proyecto cuando recién hacemos git init todos los archivos están en estado no trackeado.
    b. Al agregar un archivo con git add pasa a staged.
    c. Al confirmar (commit) pasa a modificado.
2. Un archivo no modificado pasa a modificado cuando lo editamos.
    a. Luego confirmamos los cambios con git commit y con esto vuelva a un estado no confirmado.
    b. Podemos ver el estado de los archivos con git status.
    c. Podemos ver todas las confirmaciones hechas con git log.

Actividad 2

- Crea una carpeta llamada actividad-con-git
- Dentro de la carpeta crea el archivo index.html
- Inicializa git con git init
- Agrega el archivo index.html con git add
- Verifica que haya sido agregado con git status
- Confirma el cambio con git commit -m "Mi primer commit"
- Revisa con git log que el commit se haya realizado correctamente.
- Revisa con git status que el estado del archivo haya cambiado.
- Si en git log el author aparece como unknown se debe a que te saltaste o te equivocaste un paso en el capítulo configurando Git de esta guía.

Introducción a GitHub

Existen varios tipos de repositorios remotos y empresas asociadas a proporcionarlos, las más usadas son: **GitHub**, Bitbucket y Gitlab. Todos funcionan de forma similar y lo importante es que soportan Git, permitiéndonos de esta forma gestionar con los mismos comandos desde nuestros computadores, independiente del servicio que utilicemos. Nosotros utilizaremos GitHub.

Como mencionamos, GitHub es un gestor de repositorios remotos, lo que quiere decir que podemos almacenar una copia de nuestro código en sus servidores. Así podemos trabajar colaborativamente y respaldar nuestro trabajo.

GitHub es gratis y no tiene restricciones de la cantidad de repositorios que podamos crear. Así que si no tienes tu cuenta de GitHub creada ve a crearla a GitHub.com.

¿Qué es un repositorio remoto? Los repositorios remotos son versiones de tu proyecto que se encuentran alojados en Internet o en algún punto de la red.

Configuración de GitHub

Una parte importante de conexión con cualquier servidor es la autenticación, es decir, el procedimiento informático que permite asegurar que un usuario es auténtico o quien dice ser.

Para generar y configurar las claves de autenticación debes **revisar la presentación generando claves ssh que puedes descargar desde la plataforma**. También si tienes alguna duda o problema con los pasos explicados en la presentación puedes revisar la documentación oficial.

Es necesario que sigas los pasos de la presentación antes de seguir la lectura para poder realizar las actividades.

Es necesario tener cuenta en github y hacer la configuración de las llaves para poder realizar las siguientes actividades.

Formas de trabajar en Github

En Github uno puede:

1. Trabajar en un proyecto donde uno está registrado como colaborador.
2. Realizar un fork de un proyecto.

Cuando uno crea un proyecto nuevo automáticamente queda como el administrador del proyecto y por lo tanto, es un colaborador y puede agregar colaboradores nuevos. Pero cuando uno quiere trabajar en un repositorio abierto de alguien más, entonces no es por defecto un colaborador (a menos que te agreguen, y la forma de trabajar en ese proyecto es realizando un fork.

Fork de un proyecto

Entendamos un fork como una bifurcación del proyecto, esto es tomar el original y poder realizar todas las modificaciones que queramos para nuestro propósito.

Actividad 3: Realizando nuestro primer fork

- Ingresa a: https://github.com/gsanchezd/fdsw-github
- Haz clic en la opción fork.
    - Serás redirigido a la bifurcación del proyecto en tu cuenta, es muy importante que no saltes este paso si no luego no podrás subir cambios.
- Haz clic en la opción code Descarga el proyecto utilizando git clone.
- Selecciona la opción ssh.
    - La opción https ya no funciona en github.
- Copia la dirección.
- En el terminal ingresa a tu carpeta de proyectos con cd proyectos.
    - Si no tienes puedes crear una con mkdir proyectos.
- Descarga el proyecto utilizando git clone direccion_copiada cv2.
- Carga la carpeta cv2 descargada en el editor de código.
- Cambia el header utilizando el editor de código.
- Agrega los cambios utilizando git add nombre_archivo
- Confirma los cambios utilizando git commit -m "primeros cambios"

- Sube los cambios utilizando git push origin main.
- Revisa en github que los cambios se hayan subido haciendo clic en el archivo modificado.
    - La fecha del último commit es también un buen indicador.

Esta actividad es parte del desafío, así que no la saltes.

Subiendo cambios con git push

En la actividad anterior utilizamos el comando git push, este se utiliza para subir los cambios al repositorio remoto. A continuación lo explicaremos.

Analizando git push

Las partes del comando git push son:

```
git push remote branch
```

Donde:

- **remote** es el punto de destino, este fue configurado automáticamente cuando hicimos un clon del proyecto, esta dirección también recibió un alias llamado origin, el cual es una estándar.
- **branch** es la rama del proyecto, las ramas son un elemento importante de Git de lo cual no hemos profundizado todavía. La rama por defecto en github es main.

Por lo tanto, al escribir git push origin main como lo hicimos en la actividad, estamos diciendo a git que suba el código al punto remoto configurado, el cual es nuestro repositorio en github y a la rama principal de este repositorio.

Por ahora, todas las actualizaciones cambios las realizaremos utilizando git push origin main.

Resumen

- GitHub es un gestor de repositorios remotos, lo que quiere decir que podemos almacenar una copia de nuestro código en sus servidores. Así podemos trabajar colaborativamente y respaldar nuestro trabajo.

- Las llaves, la pública y privada, son un medio por el cual podemos identificar nuestro equipo con un servidor o página específica, incluso sin tener que ingresar una contraseña.

- Fork es una acción que creará una bifurcación del repositorio, en otras palabras realizará una copia del proyecto completo en nuestra propia cuenta de GitHub.

- Podemos descargar un proyecto desde github utilizando git clone direccion_del_proyecto [nombre_del_proyecto]. Los corchetes indican que el argumento es opcional o sea podemos omitir el nombre del proyecto y en ese caso el nombre utilizado será el de github.

- Para subir los cambios al repositorio remoto debemos utilizar el comando `git push origin main`.

Ahora que terminaste la guía, resuelve el desafío disponible en la plataforma LMS.

¡Continúa aprendiendo y practicando!