Homework: Git

Pre-requisitos

* Instalar GIT
* Crear una cuenta en [Github.com](https://www.github.com/).
* Si usas una computadora con Windows, puedes instalar la terminal de comandos *git-bash* haciendo click [acá](https://git-for-windows.github.io/).
* Si usas Mac, git viene pre-instalado y puedes acceder desde la terminal pulsando ⌘+Espacio y escribiendo "terminal". Puedes asegurarte de que git está instalado escribiendo which git en la terminal y pulsando Enter, si aparece la ruta de un archivo está todo correcto. En algunas ocasiones, podrías ser redireccionado a la descarga de la línea de comandos de Xcode, sigue las instrucciones.
* Si usas Linux, tal vez necesites instalar git accediendo a tu terminal e ingresado apt-get install git
* Si tienes problemas instalando Git, este recurso puede servir de ayuda: <https://git-scm.com/book/en/v2/Getting-Started-Installing-Git>

Comandos básicos de terminal

Usaremos la "terminal" o "línea de comandos" durante tu tiempo en Codo a Codo. Si estás en una computadora con Mac o Linux, deberías tener la terminal y git ya instalados. Si estás en una máquina con Windows, te recomendamos instalar ["git-bash"](https://git-for-windows.github.io/).

Dentro de nuestra terminal podemos: revisar nuestra estructura de archivos, añadir, eliminar y modificar archivos, y muchas cosas más. La terminal es una herramienta muy poderosa para los desarrolladores y vas a usarla mucho en tu carrera profesional. Al principio puede parecer aterrador, pero con el tiempo terminarás disfrutando su uso, así que es mejor acostumbrarse ahora. Para comenzar, aprenderemos los comandos "básicos":

"List": Nos permite ver el contenido de la carpeta donde nos encontramos.

$ ls

"Change Directory": Nos permite movernos a una nueva carpeta o "directorio"

$ cd [carpeta]

"Make Directory": Crea una nueva carpeta en el directorio donde nos encontramos actualmente.

$ mkdir [nombre de la carpeta]

"Touch" creará un nuevo archivo.

$ touch [nombre del archivo]

"Remove": Elimina permanentemente un archivo. ¡CUIDADO! Esto traspasa cualquier "papelera de reciclaje" que exista en el sistema y elimina los archivos PERMANENTEMENTE. Como nota, esto no eliminará carpetas, necesitamos un comando especial para eso que aprenderemos más adelante.

$ rm [nombre del archivo]

Ejercicio

En este ejercicio crearemos un nuevo repositorio en Github y a partir de la linea de comandos crearemos un nuevo archivo y lo agregaremos a nuestro nuevo repositorio.

Iniciaremos creando una carpeta desde la consola llamada: "CarpetaCodoACodo"

Para ello primero iremos al escritorio y abriremos git bash

Luego escribe:

mkdir CodoACodo

y presiona Enter. ¡Felicitaciones, has creado un directorio!

Crear un Repositorio

Entra a github.com y logueate con tu usuario. Luego en la parte de arriba a la derecha buscá el signo '+' y elegí 'new repository'.

En la nueva pantalla, vas a tener que seleccionar el nombre de tu nuevo repo. Crea un repo que se llame: repositorio-CodoACodo. Podes agregarle una descripción también, pero no es obligatoria.

Luego elegí la visibilidad del proyecto como público (para tener repos privados tenes que pagar el premium de github).

Por último haz clic en **Crear repositorio**.

Listo ya tenes tu propio repo. Ahora vas a poder clonarlo con las intrucciones de más abajo.

Clonar

Para poder trabajar en un proyecto debes clonarlo (descargarlo) a tu máquina local. Para ello, accede al repositorio que creaste en tu cuenta, y haz click en el botón de la esquina superior derecha que dice "Clonar o descargar". Un dropdown aparecerá y podemos clickear en el icono del clipboard para copiar la dirección del repo. O, si el repo esta vacio vas a ver en el medio de la pantalla la URL con la dirección de tu repositorio.

Una vez tengas la dirección copiada, abre una terminal en el escritorio y escribe lo siguiente:

$ cd CodoACodo

$ git clone [dirección copiada]

Esto descargará el repositorio y tendrás disponible una copia local guardada en tu máquina.

Comandos de git:

Durante estas lecciones interactuaremos con git a través de la terminal. En el futuro, podrías estar interesado en usar una interfaz visual/gráfica, pero durante estas lecciones necesitaremos usar la terminal para acceder a todas las funciones de git.

En este ejercicio, añadiremos un archivo a nuestro proyecto y crearemos un commit para cambiar la "memoria" de git.

A continuación, escribe lo siguiente en tu terminal:

$ cd CodoACodo

$ touch archivo-ejemplo.js

Esto añadirá un nuevo archivo llamado "archivo-ejemplo.js" a tu proyecto.

En este momento, podemos usar el comando de git "status", esto nos mostrará los cambios hechos en el repositorio local. Usa "status" si no estás seguro de que algo haya funcionado y necesitas saber qué pasó.

$ git status

Deberías ver algo parecido a estas líneas:

$ git status

On branch main

No commits yet

Untracked files:

(use "git add <file>..." to include in what will be committed)

archivo-ejemplo.js

nothing added to commit but untracked files present (use "git add" to track)

Esto nos dice que tenemos un archivo que ha cambiado, pero todavía no ha sido guardado en el historial de git.

Para hacer esto, usaremos el comando "add":

git add archivo-ejemplo.js

Esto agregó nuestros cambios al historial. Ahora, para guardar el historial, usaremos el comando "commit". Commit agarrará todos nuestros cambios y los guardará en el historial de git. Para futuras referencias, podrás añadir un mensaje acerca de los cambios hechos, esto facilitará ir atrás en el historial y encontrar los cambios que vos (o cualquiera) busque en el futuro en caso de necesitarlo. Siempre es una buena idea hacer una descripción clara y concisa de los cambios. Un *shorthand* para añadir un mensaje, es usar la bandera (flag) "-m" y escribir el mensaje a continuación usando comillas.

git commit -m 'Agregado nuevo archivo, archivo-ejemplo.js'

La terminal te preguntará tu usuario y contraseña (En el caso de la contraseña cuando la estemos escribiendo por consola, por cuestiones de seguridad no se verá lo que estamos escribiendo). git config --global user.email "[you@example.com](mailto:you@example.com)"

Escribimos estos datos y a continuación ingresamos nuevamente el commit

git commit -m 'Agregado nuevo archivo, archivo-ejemplo.js'

Ahora que guardamos nuestros cambios localmente, vamos a querer compartir esos cambios en Github. Para ello, usaremos "push":

git push

Counting objects: 5, done.

Delta compression using up to 8 threads.

Compressing objects: 100% (4/4), done.

Writing objects: 100% (5/5), 2.97 KiB | 0 bytes/s, done.

Total 5 (delta 1), reused 0 (delta 0)

remote: Resolving deltas: 100% (1/1), completed with 1 local object.

To git@github.com:[your username]/Precourse.git

cccc682..283b9dd master -> master

¡Felicitaciones, acabas de subir tu primer git commit! Verificalo en la WEB <https://github.com/>