Curso GitHub

Sección 1 aprendizaje:

* Qué son (y para qué sirven) los sistemas de control de versiones y cómo pueden ayudar a nuestro flujo de desarrollo
  + Nos ayudan a mantener un historial de cambios;
  + Nos ayudan a tener control sobre cada cambio en el código;
  + Nos ayudan a que un cambio de una persona no influya en el cambio realizado por otra;
  + Etc.
* Qué es Git y cómo instalarlo
* Que con el comando git init pudimos crear un repositorio Git;
* Cómo analizar el estado de nuestro repositorio usando el comando git status.

También aprendí a como se usa bien el comando “ cd ” que no me estaba saliendo bien su uso pero se logro

Sección 2 Aprendizaje:

Solo para aclarar un poco, ya que entenderemos mejor cómo funciona Git durante el curso, aquí hay algunas definiciones interesantes:

* HEAD: Estado actual de nuestro código, es decir, donde nos colocó Git
* Working tree: Lugar donde los archivos realmente están siendo almacenados
* index: Lugar donde Git almacena lo que será *commiteado*, es decir, la ubicación entre el *working tree* y el repositorio de Git en sí.

Como vimos en el último video, podemos ver el historial de cambios de nuestro proyecto de manera muy sencilla, usando el comando git log.

Aunque es fácil, este comando es muy poderoso. Ejecuta git log --help y ve algunas de las posibles opciones. Para obtener algunos ejemplos más fáciles de entender, puedes buscar sobre git log o echar un vistazo a este link: <https://devhints.io/git-log>.

Commitear

Deberíamos generar un *commit* siempre que nuestra base de código esté en un estado que nos gustaría recordar. Nunca deberíamos tener *commits* de código que no funcionen, pero tampoco es interesante dejar el *commit* solo al final de una función.

Esta puede ser una discusión interminable y cada empresa o equipo puede seguir una estrategia diferente. ¡Estudien al respecto, comprendan qué tiene más sentido para ustedes y su equipo y sean felices! :-D

* Que un commit es la forma de guardar un estado o versión de nuestro código;
* Cómo agregar archivos para *commitear* con git add;
* Cómo *commitear* archivos usando el comando git commit;
* Cómo verificar el historial de *commits*, a través de git log y algunas de sus opciones:
  + git log --oneline
  + git log -p
  + git log --pretty="parámetros de formato"
* Cómo hacer que Git no monitoree archivos, a través de **.gitignore**
* Que no debemos realizar *commits*, es decir, guardar un estado, de nuestra aplicación que no esté funcionando.

Sección 3 Aprendizaje: