# Curso de LinkedIn: Como hacer que tu perfil trabaje por ti

# Clase 1: Que es git

# Para que sirve git:

# 

# Control de versión:

Git es un sistema de control de versiones.

En esta aula aprendimos:

* 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.

# Clase 2: Iniciando los trabajos

Git status = ver el estado del repositorio

Git add . = trackear todos los archivos de la carpeta

Git rm –cached <file> = dejar de trackear archive

Git commit -m “mensaje” = crear mensaje de los cambios

Al ejecutar el comando git status, recibimos información que puede no ser tan clara, especialmente cuando nos encontramos con términos como HEAD, working tree, index, etc.

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.

# Viendo el histórico:

Git log = hash (id del commit)

Git config –local = solo para este proyecto

Git config –global = todos los proyectos en la computadora

Git config user.name = tu nombre

Git config user.email = tu email

Git log –oneline = ver commits en una línea

Git log -p = ver cambios, Para salir :q

# Cuando hacer un commit:

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.

En esta aula aprendimos:

* 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.

# Clase 3: Compartiendo el trabajo

# Clase 4: Trabajando en equipo

# Clase 5: Manipulando versiones

# Clase 6: Generando entregas