**1 – Introducción a git.**

**Git** es un sistema de control de versiones que almacena la información como una fotografía de todos los estados por los que han pasado nuestros archivos desde su creación hasta que hemos terminado de trabajar con ellos.



Git es una especie de sistema de archivos con herramientas que nos permiten volver atrás en el tiempo a estados anteriores a nuestra versión actual. Es una especie de base de datos que va guardando todos los cambios siempre y cuando utilicemos los comandos correctos.

No se debe confundir git con github. Git trabaja en local, en nuestra máquina física. Github, Bitbucket o Gitlab son servidorers a los que subimos el trabajo local. Git nos permite tener la historia y sistema de archivos completo de nuestro proyecto en nuestro disco duro físico y para hacer esto no hace falta que hayamos trabajado con git + github, git + bitbucket etc…

Todos estos cambios están almacenados en un fichero .git que se crea dentro de nuestro proyecto cuando ejecutamos el comando **git init**. El fichero .git es invisible con un simple ls pero con un ls -la podemos verlo.

**2. Los tres estados de git: commited, modified y staged.**

Git tiene tres estados básicos en los que los archivos dentro de nuestro proyecto pueden estar: commited, modified y staged.

1. **commited** Significa que los datos están guardados de manera segura en la base de datos local. *Se ha ejecutado git add . + git commit “mensaje”*
2. **modified** Significa que algo ha cambiado en el proyecto, algún archivo ha sido modificado, añadido, modificado etc pero los cambios todavía no se han ***commiteado*** a la base de datos local. *No se han ejecutado ni git add . ni git commit “mensaje”*
3. **staged** Significa que un archivo ha sido marcado como modificado y este cambio va a verse reflejado en el siguiente commit. *Se ha ejecutado git add . pero no git commit.*

Los estados **commited**, **modified** y **staged** nos llevan a las tres grandes secciones de un proyecto de git: el **directorio git**, el **working tree** y la **staging area**.



1. **git directory** (repository )Es donde git almacena los metadatos y los objectos de la base de datos de nuestro proyecto. Es la parte más importante de git y es lo que se copia o clona cuando se ejecuta un ***git clone urlDelRepositorio***.
2. **working tree o working directory** La documentación de git define al **working tree** o **working directory** como un “***single checkout of one version of the project”****.* Los ficheros del **working tree** se sacan de la database almacenada en el directorio .git y se posicionan en el disco duro para que los veamos o los modifiquemos.
3. **staging area** Archivo generalmente contenido en nuestro directorio de git que guarda la información de lo que irá en el próximo commit.

**2.1 Git workflow**

1. Modificamos datos en el ***working tree***, es decir, en la *snapshot* particular del proyecto con el que estamos trabajando. Ejemplo: “Añadimos un nuevo fichero .java o una nueva función a un fichero”.
2. Pasamos a la **staging area** sólo aquellos cambios que queremos que sean parte de nuestro siguiente commit. Ejemplo: “En el paso anterior hemos añadido un nuevo fichero foo.java y una nueva función al fichero bar.java pero al final decidimos que no queremos guardar el fichero foo.java pero si los cambios introducidos al fichero bar.java, por lo que hacemos ejecutamos *git add bar.java*. Si quisiéramos guardar todos los cambios basta ejecutar **un git add .** , que guarda todos los cambios en todos los ficheros.”
3. Con git add se guardan los cambios a la staging area pero todavía no se han almacenado **definitivamente** en el repositorio de git o **git directory**, para hacer esto necesitamos ejecutar un **git commit -m “Mensaje deseado”.**

**git init**

Crea el repositorio en loca, inicializa git.

**git add .**

Añade los cambios, todos.

**git add archivo.extension**

Añade los cambios que se han realizado en el archivo al que hacemos mención.

**git checkout -- .**

Recupera los cambios que teníamos hasta el último commit EN LOCAL.