**Guia de Estudio para el Control de Versiones.**

Guia de Estudio de la parte de Gestión de la Documentación - Control de Versiones. Indica qué tenemos que saber y saber hacer en la asignatura de Proyectos de Sistemas Electrónicos.

**Conceptos mínimos que tenemos que saber sobre el control de versiones**

* Definir brevemente en qué consiste el control de versiones.
  + Sistema que registra los cambios que ha sufrido un fichero, de manera que se pueda recuperar cualquier versión pasada si fuera necesario compuesto normalmente de un **servidor y un cliente.**
* Interfaz de usuario gráfica, Diagrama, Texto, Chat o mensaje de texto

  Descripción generada automáticamenteExplicar los siguientes conceptos:
  + repositorio local (Local Repository): base de datos centralizado donde se guardan las distintas versiones de los ficheros sometidos a control de versiones
  + copia local: (working copy): Es la copia que hacen los usuarios de un fichero sometido a control de versiones. todas las copias locales son contenidas en el DIRECTORIO LOCAL (working directory/working tree/workspace).
  + repositorio remoto: versiones del proyecto almacenadas en Internet o en cualquier otra red.
  + *Log o historico:* Registro de todos los cambios que se han producido en el repositorio.
  + conflicto: Problema que surge cuando los clientes realizan cambios incompatibles entre sí.
* Diagrama

  Descripción generada automáticamenteExplicar los siguientes estados de un fichero: sin seguimiento, confirmado, modificado, preparado, ignorado.
* Explicar las siguientes operaciones:
  + Clone: Replica un repositorio entero con todo su historial de cambios y actualiza el directorio local
  + Add: Realiza una copia de un fichero modificado, poniéndola en la zona de preparación para poder ser confirmada.
  + Commit: Confirmar los ficheros preparados para su almacenamiento en el repositorio.
  + Push: Es la operación en la que se envían al repositorio centralizado (remoto) un commit o conjunto de commits, incluido una rama entera.
  + Pull: Es la operación en la que se actualiza el repositorio local y el directorio local con commits que provienen del repositorio remoto.
  + Fork: Clone que se hace dentro del mismo servidor. Por ejemplo, el repositorio original y el clonado ambos residen en (xej)GitHub…
  + Pull Request: (entre repositorios): Petición que hace el desarrollador de que los cambios hechos en su repositorio clonado mediante un FORK sean incorporados al repositorio original
* Traducir entre inglés y español la terminología de los tres puntos anteriores.
* Nombrar al menos dos servicios de repositorio remoto para el control de versiones.
  + GitHub y GuitLab
* Nombrar al menos un cliente gráfico (GUI) para el control de versiones.
  + Git Force

**Qué tenemos que saber hacer con Git (y GitHub)**

**En el intérprete de comandos de git-bash**

* Mostrar en qué directorio estamos.
  + pwd (Print working directory)
* Crear un directorio.
  + mkdir <dir>
* Cambiar de directorio.
  + cd <dir>
* Mostrar la lista de ficheros de un directorio.
  + ls [-l] [-a]
* Borrar un fichero.
  + rm <file>
* Cambiar (mover) un fichero de directorio.
  + mv <source> <dest>

**En control de versiones local**

* Crear un repositorio local en tu directorio de trabajo actual. También se creará una nueva rama principal
  + **git init**
* Preparar ficheros para ser confirmados en un repositorio local.
  + git add <file/files>
* Confirmar cambios en un repositorio local.
  + git comit
* Deshacer la operación de preparar.
  + git reset
* Deshacer la operación de confirmar.
  + git reset < --hard (Retrocedemos un commit y borramos los cambios) ó --soft (retrocedemos un commit pero no perdemos los cambios y se quedan almacenados para un futuro commit) HEAD~1>
* Identificar el estado de un fichero o ficheros en un repositorio local.
  + git status
* Descartar los cambios de un fichero de trabajo mediante la recuperación de una versión almacendada en el repositorio local.
  + git restore
* Crear una rama en un repositorio local.
  + git branch
* Cambiar de rama en la copia local.
  + git switch

**En control de versiones centralizado**

* Configurar git para que trabaje tras un proxy

***git config --global http.proxy***[***http://proxy.user:proxy.pass@proxy.name\_or\_ip:proxy.port***](http://proxy.user:proxy.pass@proxy.name_or_ip:proxy.port/) proxy.user es el usuario que tiene permiso para acceder al proxy. proxy.pass es la password de dicho usuario. proxy.name\_or\_ip es el nombre DNS o la dirección IP de la máquina que hace de proxy de internet. proxy.port es el puerto por el que se accede a comunicar con el proxy de internet. –global le indica a git que la configuración es propia del usuario y sirve para todos sus repositorios.

Para consultar el proxy configurado en git: ***git config --global --get http.proxy***

Para borrar el proxy configurado en git: ***git config --global --unset http.proxy***

* Replicar un repositorio remoto localmente en nuestra máquina.

Clonar un repositorio extrae una copia integral de todos los datos del mismo que GitHub.com tiene en ese momento, incluyendo todas las versiones para cada archivo y carpeta para el proyecto hacia nuestra maquina local. Para ello abriremos git-bash en el directorio que queramos clonar el repositorio e introducimos:

* + git clone <https://github.com/YOUR-USERNAME/YOUR-REPOSITORY>
* Repolicar un repositorio local en un servidor remoto.
  + 1. Creamos un repositorio en github.
    2. Convertimos nuestro proyecto local en un repositorio git, para ello abrimos git-bash en el directorio deseado e introducimos: *git init*
    3. Para añadir un repositorio remoto: *git remote add origin*[*https://github.com/GitHub\_UserName/Repository\_Name*](https://github.com/GitHub_UserName/Repository_Name)
    4. Creamos una nueva rama y nos cambiamos a ella: *git branch -M main*
    5. Añadimos los arcivos deseados: *git add nombre\_proyecto*
    6. Realizamos nuestro primer commit: *git commit -m "mensaje del commit"*
    7. Hacemos push de todas las ramas a la main: *git push -u origin main*.
* Traer los cambios de un repositorio remoto a un repositorio local.

En este caso tenemos dos posibles opciones en función de nuestras necesidades:

1. Clonar el repositorio con *git clone* (solo lo podemos hacer si es la primera vez).
2. En el caso de que ya tengamos un repositorio local clonado de uno remoto y queramos copiar los cambios del repo remoto a nuestro repo local para actualizarlo, haremos: *git pull origin BRANCH-NAME*

* Resolver los conflictos que se puedan producir al traerse estos cambios.

Puede haber solape entre lo que se cambió en el repo central y lo que nosotros hemos hecho en local, esto crea un conflicto al hacer el "pull". La solución:

1. Hacer: *git stash save* (stash nos permite guardar los cambios locales que no pasamos a commit en un universo medio paralelo)
2. Hacer ahora: *git pull*
3. Finalmente: *git stash pop*

* Enviar los cambios de un repositorio local a uno remoto.

Los pasos a realizar son tres:

1. *git add Archivo\_o\_Carpeta*
2. *git commit Archivo\_o\_Carpeta -m "Mensaje explicativo"*
3. *git push*

Es posible que al hacer un commit nos de un error por no estar identificados, en ese caso introduciremos:

* *git config --global user.name NombreDeUsuarioGit*
* *git config --global user.email NuestroCorreoGit@correo*

También podemos comprobar el estado del repo antes y despues de la operación con *git status*

* Enviar una rama local al repositorio remoto.

En este caso suponemos que hemos estado trabajando en una rama y llega el momento de subir nuestro trabajo al repo remoto, para esto, suponemos que ya se han hecho todos los adds y commits de manera local, entonces introducimos:

* *git push -u origin nombre\_de\_la\_rama*

Si queremos enviar todas las ramas de nuestro repo local al remoto:

* *git push --all -u*

En caso de que a rama no esté creada en el repo remoto todavía:

* *git push --set-upstream origin nombre\_rama*
* Incorporar a ramas locales cambios que se producen en el repositorio remoto.

Usaremos : git pull origin nombre\_rama\_remota

* Realizar un pull request entre dos ramas de un repositorio remoto.

Es necesario para esto que hayamos hecho cambios en las ramas: VAmos a github, entramos en el repositorio->code->branch->pull request

**En control de versiones distribuido**

* Realizar un pull request entre dos repositorios que resultaron de un Fork.