Comandos de utilidad.

Pwd -> Te muestra tu ubicación dentro de los directorios.

ls *name* -> Te muestra el contenido de los directorios *name.*

cd *name* -> Te traslada al directorio *name.*

cd.. ->Te devuelve al directorio anterior.

Como crear y borrar repositorio.

Para crear un repositorio desde la página inicial de GitHub podemos buscar el botón que esta en la parte superior derecha que se llama crear repositorio: *‘+’ -> ‘new repository’*Una vez seleccionado solo hay que rellenar unos pocos campos que darán información sobre el repositorio.

Para eliminar un repositorio desde la opción *‘setting’*

Comandos para GitHub local

Git ini -> este comando convierte la carpeta en la que se ejecute en un directorio de git *Copia local.*

Git add -> Al ejecutar este comando seguido del nombre de los archivos que quieres guardar se generara una copia que llamaremos *archivo provisional.*

Git reset -> Elimina el *archivo provisional* o alguno de los documentos almacenados en él.

Git commit -m “*name*” -> Almacena en el *archivo local* los documentos almacenados en el *archivo provisional* bajo un identificador *name,* que nos permite localizar los documentos una vez almacenados, y recuperarlos en caso de actualizar el archivo con información errónea.

Git restore <*fichero*> -> Nos permite restaurar un fichero almacenado a una versión anterior

Comandos para github nube

Git clone *URL ->* bajas de la *nube* al *archivo local* un clon del repositorio.

Git pull ->

Git fetch + git merge -> actualiza repositorio local respecto al de la nube.

Git push -> actualizas el repositorio de la nube con ficheros de *archivo local*

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

Guía 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 probablemente en qué consiste el control de versiones.

**Un control de versiones es un sistema que registra los cambios realizados en un archivo o conjunto de archivos a lo largo del tiempo, de modo que puedas recuperar versiones específicas más adelante.**

* Explicar los siguientes conceptos: repositorio local, copia local, repositorio remoto, *log* , conflicto.

**- Un repositorio de Git es un almacenamiento virtual de tu proyecto. Te permite guardar versiones del código a las que puedes acceder cuando lo necesites. Para crear un nuevo repositorio, usa el comando git init.**

**- Copia local: Es la copia que hacen los usuarios de un fichero sometido a control de versiones. El DIRECTORIO LOCAL (working directory o working tree) es el que contiene todas las copias locales.**

**- Los repositorios remotos son versiones de tu proyecto que están hospedadas en Internet o en cualquier otra red. Puedes tener varios de ellos, y en cada uno tendrás generalmente permisos de solo lectura o de lectura y escritura.**

**- Log: Registro de todos los cambios que se han producido en el repositorio. Es responsabilidad del cliente añadir información al log cuando se produce un cambio.**

**- Conflicto: Problema que surge cuando los clientes realizan cambios incompatibles entre sí.**

* Explicar los siguientes estados de un fichero: sin seguimiento, confirmado, modificado, preparado, ignorado.

**- Sin seguimiento: Un archivo que no se ha preparado o confirmado.**

**- Confirmado: archivos en el estado confirmado son archivos que se guardaron en el repositorio .git exitosamente.**

**Por lo tanto, un archivo confirmado es un archivo en el cual has registrado su versión preparada en el directorio (carpeta) Git.**

**- Modificado: Un archivo en el estado modificado es un archivo revisado – pero no acometido (sin registrar). En otras palabras, archivos en el estado modificado son archivos que has modificado, pero no le has instruido explícitamente a Git que controle.**

**- Preparado:  Archivos en la etapa preparado son archivos modificados que han sido seleccionados – en su estado (versión) actual – y están siendo preparados para ser guardados (acometidos) al repositorio .git durante la próxima instantánea de confirmación.**

**Una vez que el archivo está preparado implica que has explícitamente autorizado a Git que controle la versión de ese archivo.**

**- Ignorado: suelen ser artefactos de compilación y archivos generados por el equipo que pueden derivarse de tu fuente de repositorios o que no deberían confirmarse por algún otro motivo.**

* Explicar las siguientes operaciones: Clone, Add, Commit, Push, Pull, Fork y Pull Request.

**- El comando git clone se utiliza principalmente para apuntar a un repositorio existente y clonar o copiar dicho repositorio en un nuevo directorio, en otra ubicación**

**- El comando git add añade un cambio del directorio de trabajo en el entorno de ensayo. De este modo, indica a Git que quieres incluir actualizaciones en un archivo concreto en la próxima confirmación.**

**- El comando git commit captura una instantánea de los cambios preparados en ese momento del proyecto. Las instantáneas confirmadas pueden considerarse como versiones "seguras" de un proyecto: Git no las cambiará nunca a no ser que se lo pidas expresamente.**

**-** **El comando git push se usa para cargar contenido del repositorio local a un repositorio remoto. El envío es la forma de transferir confirmaciones desde tu repositorio local a un repositorio remoto.**

**- El comando git pull se emplea para extraer y descargar contenido desde un repositorio remoto y actualizar al instante el repositorio local para reflejar ese contenido.**

**- La función de fork en Git se define como una operación usual en el sistema de Git que se encarga de la creación de una copia de un repositorio en la cuenta de usuario. Cabe destacar que este repositorio copiado será igual al repositorio desde el que se realiza el fork en Git.**

**- Los pull requests son la forma de contribuir a un proyecto grupal o de código abierto. Básicamente un pull request es una petición para integrar nuestras propuestas o cambios de código a un proyecto.**

* Traduzca 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: Es una plataforma de desarrollo colaborativo de software, que alberga proyectos de código abierto en su mayoría aunque también ofrece servicios de pago para proyectos privados.**

**- GitLab: Es una herramienta de código abierto para el control de versiones y la gestión de proyectos. Esta herramienta está basada en Git, un sistema distribuido de control de versiones, y permite tanto el desarrollo colaborativo como la administración de proyectos.**

* Nombrar al menos un cliente gráfico (GUI) para el control de versiones.

**GitKraken, SourceTree, GitHub Desktop y TortoiseGit son algunas de las herramientas gráficas (GUI) para el control de versiones.**

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

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

* Mostrar en qué directorio estamos.

**pwd**: **nos muestra la carpeta actual en la que nos encontramos.**

* Crear un directorio.

**mkdir: nos permite crear carpetas, p. ej. mkdir *NuevaCarpeta***

* Cambiar de directorio.

**cd: nos permite cambiarnos de carpeta, p.ej. cd *NuevaCarpeta.***

* Mostrar la lista de ficheros de un directorio.

**ls: nos permite ver los archivos de la carpeta donde estamos actualmente.**

* Borrar un fichero.

**rm: Nos permite borrar un archivo o carpeta ej: rm NuevoArchivo.txt**

**rm -r ‘nombre de la carpeta’ :me permite eliminar la carpeta y los archivos dentro de ella de forma recursiva.**

* Cambiar (mover) un fichero de directorio.

**mv “el directorio de donde queremos mover/el nombre del archivo” “el directorio hacia donde lo queremos mover”: nos permite mover un archivo.**

**Mas comandos de comandos de git-bash:**

* **pwd: nos muestra la carpeta actual en la que nos encontramos.**
* **mkdir: nos permite crear carpetas, p. ej. mkdir *NuevaCarpeta***
* **touch: nos permite crear archivos nuevos, p.ej. touch *NuevoArchivo.txt***
* **cat: nos permite ver el contenido de un archivo, p.ej. cat *NuevoArchivo.txt***
* **cd: nos permite cambiarnos de carpeta, p.ej. cd *NuevaCarpeta.***
* **cd .. : nos permite regresar al directorio o carpeta anterior.**
* **cd o cd ~: nos lleva a la ruta del usuario.**
* **cd /c: nos vamos al disco C:/.**
* **cd -: nos lleva directamente al último directorio visitado.**
* **ls: nos permite ver los archivos de la carpeta donde estamos actualmente.**
* **ls -l: Ver todos los archivos como una lista en donde incluye el usuario, grupo, permisos sobre el archivo, tamaño, fecha y hora de creación.**
* **ls -lh: Muestra la misma información que ls-l pero con unidades de tamaño, es decir, kb o mb.**
* **ls-R: muestra el contenido de todos los sudirectorios de forma recursiva.**
* **ls -S Ordena los resultados por tamaño de archivo.**
* **rm: Nos permite borrar un archivo o carpeta ej: rm NuevoArchivo.txt**
* **rmdir “nombre del directorio”/: borrar un directorio: Solo funciona con directorios vacíos.**
* **rm -r ‘nombre de la carpeta’ :me permite eliminar la carpeta y los archivos dentro de ella de forma recursiva.**
* **cp “nombre del archivo que quremos copiar” “nombre del directorio a donde lo queremos copiar”: nos permite copiar un archivo.**
* **mv “el directorio de donde queremos mover/el nombre del archivo” “el directorio hacia donde lo queremos mover”: nos permite mover un archivo.**
* **clear: nos permite limpiar la pantalla.**
* **history: ver los últimos comandos que ejecutamos y un número especial con el que podemos volver a repetir el comando.**

**Control de versiones locales**

* Crear un repositorio local en nuestra máquina.

**Git init**

* Preparar ficheros para ser confirmados en un repositorio local.

**Git add**

* Confirmar cambios en un repositorio local.
* Deshacer la operación de preparar.

**Git reset HEAD**

* Deshacer la operación de confirmar.

**Para salir o cambiar una confirmación específica, puedes usar  git checkout  y pasar el**[**SHA**](https://es.wikipedia.org/wiki/Secure_Hash_Algorithm)**de la confirmación en lugar de un nombre de rama.**

* 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 almacenada en el repositorio local.
* Crear una rama en un repositorio local.

**Git Branch**

* Cambiar de rama en la copia local.

**Para cambiar a una rama existente, puedes usar git checkout nuevamente (sin el indicador -b ) y pasar el nombre de la rama a la que desea cambiar.**

**Control de versiones centralizado**

* Configurar git para que funcione tras un proxy
* Replicar un repositorio remoto localmente en nuestra máquina.
* Replicar un repositorio local en un servidor remoto.
* Traer los cambios de un repositorio remoto a un repositorio local.
* Resolver los conflictos que se pueden producir al traerse estos cambios.
* Enviar los cambios de un repositorio local a uno remoto.
* Enviar una rama local al repositorio remoto.
* Incorporar a ramas locales cambios que se producen en el repositorio remoto.
* Realice una solicitud de extracción entre dos ramas de un repositorio remoto.

**En control de versiones distribuidas**

* Realice una solicitud de extracción entre dos repositorios que resultaron de un Fork.

**Cosas fuera de la guía**

Diferencia entre directorio y repositorio local:

* + **En resumen, podríamos decir que un directorio es un portal dónde aparecen un listado de repositorios o las obras derivadas de éstos, y un repositorio es una base de datos con documentos pertenecientes a una institución específica.**

**Chuleta de control de versiones con git**

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

* **Control de versiones**: El control de versiones es un sistema que registra los cambios realizados sobre un archivo o conjunto de archivos a lo largo del tiempo, de modo que puedas recuperar versiones específicas más adelante
* **Conceptos:**
  + **Repositorio local**: Es una 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. El **DIRECTORIO LOCAL (working directory/working tree/workspace)** es el que contiene todas las copias locales.
  + **Repositorio remoto**: (Remote repository/Repository)Es una base de datos centralizada donde se guardan las distintas versiones de los ficheros sometidos a control de versiones, y reside en el servidor centralizado.
  + **Histórico**: (LOG) Registro de todos los cambios que se han producido en el repositorio. Es responsabilidad del cliente añadir información al log cuando se produce un cambio. También llamado histórico.
  + **Conflicto**: Problema que surge cuando los clientes realizan cambios incompatibles entre sí.
* **Estados del fichero**:
  + **Sin seguimiento**: archivos que solo son locales y no existen en el repositorio remoto.
  + **Confirmado(commited file)**: Es el fichero que ya está almacenado en el repositorio. La última versión y la copia local son iguales.
  + **Modificado(modified file)**: Es el fichero modificado en la copia local y existe una diferencia entre la copia local y la última versión en el repositorio.
  + **Preparado**: (staged file)Es la copia del fichero modificado preparada para ser confirmada en la próxima operación de confirmación (COMMIT). La zona de preparación (staging area or index) contiene todos los ficheros preparados.
  + **Ignorado(ignored file)**: Es el fichero que está en el directorio local pero que deliberadamente no se somete a control de versiones.
* **Operaciones**:
  + **Clone**: (Clonar) Realiza una copia de un repositorio remoto en un directorio local.
  + **Add**: (Añadir) Realiza una copia de un fichero modificado, poniéndola en la zona de preparación para poder ser confirmada.
  + **Commit**: (Confirmar) Confirma los cambios en el repositorio local.
  + **Push**: (Enviar) Envía al repositorio remoto los cambios correspondientes a los commits realizados desde el último push.
  + **Pull**: (Traer) Descarga desde el repositorio remoto los archivos actualizados en commits que hayan realizado otros usuarios, y los integra (realiza un merge) con el repositorio local.
  + **Fork**: (Duplicado): Crear una copia de un repositorio. Esto es especialmente útil para modificar proyectos sin afectar al proyecto original.
  + **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.
* **Servicios de repositorio remoto**: GitHub, GitLab, BitBucket, etc.
* **Clientes gráficos para el control de versiones**: gitg, gitkraken, gitblade, sourcetree, etc.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente