LEZIONE 1/12

Per creare un file readme locale andare sulla linea di comando, scrivere la cartella di riferimento ad esempio programmazione ad oggetti e selezionare prova. Per trovare la cartella scriverò:

cd Desktop, **cd** programmazione ad oggetti, **cd** prova

Copiando la prima riga da GITHUB posso creare un file README locale. Comando da eseguire:

echo "# prova3" >> README.md

Poi con **git init** creo una repository vuota in locale;

Vogliamo eseguire un primo commit dove salviamo il file readme : dobbiamo spostare questo file nella cosiddetta "staging area" (un'area intermedia che sta tra la mia struttura del file locale e il commit) .

Eseguo a questo punto la funzione git **add README.md**; se voglio aggiungere più file tutti insieme posso semplicemente scrivere **git add.** e verranno ignorati i file specificati da **gitignore**(?).

Facciamo un prova con **gitignore**. Creiamo un file chiamandolo "ignorami" eseguendo il comando **echo "prova" >> ignorami.txt,** dopodichè eseguo la funzione:

"ignorami.txt" >> .gitignore

A questo punto eseguiamo il comando **git add .** , seguito da **git status;** otteniamo così la staging area. Da qui possiamo osservare il file readme e non il file **"**ignorami" che era stato contrassegnato con **gitignore.**

/* operazione che ci permette di ignorare delle cartelle: le tipologie di cartelle che dobbiamo evitare di condividere sono quelle che contengono dati sensibili e cartelle che contengono informazioni pesanti come ad esempio i data set

Git mi serve per gestire il codice.

Differenza tra GIT e GITHUB:

-GIT: qualcosa che può essere usato anche senza GITHUB; sistema che viene usato per gestire versioni, per lavorare insieme ad un progetto e così via.

- GITHUB: qualcosa che viene usato solo con GIT, gestisce i server remoti, una sorta di hosting per i progetti GIT che permette di avere repository pubbliche e tante altre funzioni…

(Se i comandi di GIT non vanno automaticamente bisogna inserire GIT tra le variabili di stato(?))

GITHUB ci serve per creare la repository, cioè come se avessimo creato una cartella su Google drive.

L'operazione di pushing è un'operazione di trasferimento che non c'entra nulla con GIT.

Quindi **git -add** mi aggiunge i file alla staging area, che è un'area temporanea che posso vedere con il comando **git status**(che mi fa vedere tutto ciò che è stato salvato in modo temporaneo e non ancora in un commit). Nel momento in cui eseguiamo il comando commit è come se salvassimo le modifiche effettuate.

Il prossimo step è eseguire il commit con il comanando **git commit -m "creato il file readme";** la linea di comando mi chiederà un messaggio, che mi serve per riconoscere che cosa ho salvato lì, perché ogni commit ha il suo identificativo univoco, che è una stringa alfanumerica.; in questo esempio il messaggio da scrivere sarà "Creato il file readme". Posso scrivere nuovamente **status** per osservare le modifiche.

È possibile aggiungere un file precedentemente contrassegnato da **gitignore** in un secondo momento? Si, andando ad eliminare tale assegnazione.

Facendo riferimento allo schema visto a lezione possiamo dire che:

Le modifiche fatte fino a questo punto sono locali e indipendenti da GITHUB. Con **git add**. andiamo a salvare questi file nella staging area (tutto ciò che abbiamo fatto fino ad ora è in locale). Poi con il comando **git commit.** salvo queste modifiche della staging area; questo commit fa parte di una time line di commit. Quando vado ad eseguire il push questo commit viene salvato in remoto, che nel nostro caso è GITHUB. Cosa significa avere una time line? Significa che nel primo commit ho i primi file che ho salvato, nel secondo commit non ho una copia dei file modificati, ma le differenze con quello precedente; cioè se ad esempio ho inserito una nuova riga in un file, nel secondo commit ho solo la riga nuova, analogamente dall'altra parte dello schema. Quindi non andiamo a ricopiare di volta in volta il codice, ma semplicemente le differenze tra i vari commit. Questo mi permette di andare indietro nel tempo e controllare ciò che avevamo fatto precedentemente. Ogni commit ha un suo codice alfanumerico (commento) che ci permette di distinguerli.

Il comando **Git reset head** (?) ci permette di tornare ad un commit precedente. (solo accennato in questo caso non è stato utilizzato!)

A questo punto creiamo il branch principale che si chiama main con il comando:

git branch -M main (la M grande indica che si tratta della time line principale)

Il prossimo step è andare a salvare in remoto, specificando dove si trova la repository usando il comando **git .add origin,** fornendo così proprio un indirizzo dove poi vado a comunicare le mie push. Quindi eseguiamo:

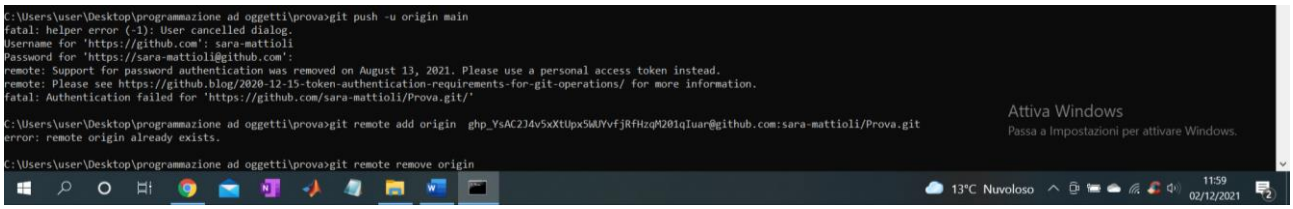
git remote add origin https://github.com/sara-mattioli/Prova.git

per poi eseguire il **push (git push -u origin main**), che è proprio l'operazione che mi va a salvare in remoto. Solo al primo push vado a specificare dove vado a salvare il tutto. Push vuol dire che vado a salvare su origin che è il nome che ho dato all'indirizzo e sul branch main; quindi con **origin** dico su che indirizzo vado a salvare e main su che branch vado a salvare. La **-u** nel comando sta ad indicare che mi chiederà di inserire le mie credenziali.
Nel momento in cui eseguo **git push -u origin main** e inserisco le credenziali, dalla linea di comando avrò un errore e mi verrà chiesto di utilizzare un token.

Usando il comando **git remote remove origin** cambio l'indirizzo e ci rimetto l'indirizzo https (chiede comunque di loggarsi). Se si apre una finestra possiamo chiuderla perché tanto richiede le credenziali anche dalla linea di comando.

Per creare un nuovo token: andiamo su GITHUB, settings, developer settings, personal access tokens e creiamo un nuovo token. Spuntare **repo** (a cosa serve un nuovo token?). Quando aggiungiamo il nostro indirizzo dobbiamo aggiungere anche il token.

Quando andiamo ad aggiungere l'indirizzo, prima di inserire l'indirizzo GITHUB dobbiamo aggiungere il token personale (va fatto solo la prima volta) nelle prossime si può utilizzare direttamente **git push**.



Quindi il comando da inserire sarà: **git remote add origin** token@github.com:sara-mattioli/Prova.git (Prova.git è il nome che ho dato alla mia repository in questo esempio).

Dopo aver inserito ciò avremo un altro errore.

Per risolvere rieseguiamo il comando **git remote remove origin** ed inseriamo il comando corretto per l'inserimento dell'indirizzo: **git remote add origin** https://token@github.com/sara-mattioli/Prova.git

A questo punto eseguendo il comando **git push origin main** salverò il commit su remoto; se ora andiamo su GITHUB troveremo i file salvati.

Modificando il codice su GITHUB effettuiamo un commit in remoto e non in locale, come fare per salvare le modifiche in locale? Posso usare il comando **git pull** dalla linea di comando.

Si utilizza il **merge** per unire i commit delle persone che lavorano al progetto; è come se avessimo due progetti differenti che poi vengono uniti attraverso il comando **merge**.

Il comando **git checkout** mi permette di spostarmi localmente su un altro branch , quando chiamo questo comando in locale non ho più il contenuto di main, ma avrò il contenuto di feature1 che è il nome che dò al branch; con **-b** creo il branch se non esiste, se già esiste non è necessario mettere **-b**.

Quando vado a richiamare **git merge** il branch dove mi trovo si va ad unire al main e viene eliminato. Quindi dopo aver richiamato il **git merge** devo richiamre **git checkout main** (main è facoltativo).