AMARILLO - MICA
ROSA - SOFI
VERDE - VIVI
LILA - MANU

Diapositiva 1, 2 ---- Nos presentamos ------

Diapositiva 3 El tema designado para nuestro equipo es GIT y GitHub. Primero debemos comenzar con diferenciar ambos: Si bien tienen un nombre similar, Git es la tecnología que mantiene el sistema de versionado, mientras que GitHub es un proveedor del servicio.

GIT nos permite trabajar en grupo de forma colaborativa, comunicar cambios, manejar distintas versiones. Es por esto que es una de las más requeridas por el mercado a la hora de trabajar con control de versiones y de forma colaborativa. Además, es una herramienta que nos va a servir durante toda la cursada y en nuestro futuro ámbito laboral.
Por definición, GIT es un software de control de versiones, pensado para la eficiencia y confiabilidad del mantenimiento de aplicaciones cuando éstas tienen un gran número de archivos de código fuente. Su propósito es llevar registro de cambio de archivos y coordinar el trabajo que varias personas realizan.

Por otra parte, GitHub es el lugar en donde se irán almacenando los archivos de nuestro proyecto que se encuentra en la nube (precisamente, en GitHub) y a través del cual podremos hacer seguimiento de los mismos. Actualmente GitHub pertenece a Microsoft quien lo compró en 2018 por la cantidad de 7500 millones de dólares.

----------Explicamos de qué forma organizamos la presentación----------

Diapositiva 4

Para trabajar con GIt primero tenemos que ver que sistema operativo tenemos: si tu computador tiene sistema operativo Linux o MacOs lo más seguro es que ya venga instalado git. Si tu computador tiene sistema operativo Windows deberás descargar un programa llamado **Git Bash** que contiene todos los comandos para ejecutarlo.

Para verificarlo puedes abrir la terminal y escribir git --version y aparecerá por consola la versión de git instalada.

Diapositiva 5 ---- Introducción repositorios: tipos y conexión necesaria entre ambos.

Un repositorio es el lugar donde se irán almacenando los archivos de nuestro proyecto.

Hay dos tipos de repositorios:

-Repositorios remotos que son los que se alojan en GitHub.

- Repositorios locales que se alojan en nuestra PC.

Es necesario crear un vínculo entre ambos para poder mantener actualizados los archivos.

Diapositiva 6 Repositorio local

Primero debemos crear una carpeta en nuestra PC que será donde alojaremos nuestro proyecto. Esta carpeta será nuestro repositorio local. (Se podría crear también abriendo la terminal y usando los comandos)

Dentro de la carpeta, abrir una terminal y correr el comando “git init” Este comando inicializa un repositorio local en la carpeta del proyecto.

Diapositiva 7 Identidad

Luego de haber creado el repositorio hay que darle una identidad al mismo para que git pueda mantener un registro de quien realiza los cambios.

Con la terminal abierta en la carpeta que creamos el repositorio, en el caso del ejemplo dentro del escritorio en la carpeta “Informática” Corremos el comando “git config user.name "nombreDeUsuario" “ donde dentro de las comillas se debe escribir el nombre de usuario que utilizamos en GitHub. Para corroborar que se ha ejecutado utilizar el comando “git config user.name” que nos debería devolver el nombre anteriormente escrito.

Para configurar el email hay que seguir los mismo pasos, correr el comando “git config user.email "nombre@email.com" “ y entre comillas escribir el email que usamos en GitHub. Para corroborar que se ha ejecutado utilizar el comando “git config user.email” que nos debería devolver el email anteriormente escrito.

Diapositiva 8

Si nos quisiéramos ahorrar esos pasos por cada repositorio que creamos podemos configurarlo de manera global, para esto hay que agregarle a los comandos anteriores --global, quedando de esta forma

“git config --global user.name "nombreDeUsuario" “

“git config --global user.email "nombre@email.com" “

De esta forma no hará falta aclarar la identidad y quedarán guardados ambos.

Diapositiva 9 Repositorio Remoto

Una vez creada una cuenta en GitHub, en la parte superior derecha seleccionamos el + y elegimos la opción nuevo repositorio.

Luego le otorgamos el nombre CheckPointII

Diapositiva 10 - Vincular

Primero debemos vincular en la terminal con el comando “git remote add origin url de github” nuestro repositorio remoto con el local. Usando el comando “git remote -v” podemos verificar que git entendió a donde tiene que direccionarse.

Diapositiva 11 Clone

Utilizamos Git clone que nos permite crear una copia exacta en la computadora de todos los archivos de un repositorio remoto existente, seguido de ‘la URL del repositorio que quiero descargar asi: te ubicas en la carpeta donde quieres guardar los archivos, click izquierdo abrir terminal y escribes git clone “url…….” y este comando descarga todos los archivos en esta carpeta.

Hay que tener en cuentaque esta carpeta descargada ya está sincronizada con github y lo podemos comprobar escribiendo git remote -v.

Diapositiva 12 subiendo archivos

Con los comandos mencionados por el grupo anterior se pueden crear los archivos que se agregan a la carpeta como el mkdir (para crear carpeta) , cd (para ubicarnos en la carpeta) y touch (para crear el archivo) , o se puede realizar a con el mouse.

Diapositiva 13 Add. Commit -m y push

Todos los archivos creados se encuentran sin seguimiento por parte de git, por lo cual hay que correr los comando git add . o git add nombrearchivo , hacer un commit -m ”nombreCommit” .Antes de enviar tus archivos a github tendrás que revisar que todo este comitiado, para indicar a git que quieres subir una carpeta con archivos a la nube tienes que escribir en la terminal: “git push origin master “

Diapositiva 14 ---- Agregamos también más PUSH que realizó cada uno en su terminal.

Diapositiva 15

Si queremos actualizar los archivos del repositorio que descargamos anteriormente porque sufrieron cambios o actualizaciones con el comando “git pull” lo podrás hacer, debemos recordar que parado en la carpeta que descargaste se debe escribir

desde la terminal “git pull origin master” esto actualiza archivos modificados o nuevos del repositorio principal.

Entonces el primer comando a correr tiene que ser “GIT PULL origin master “ para asegurarse de tener la última versión del proyecto , luego seguir con git add . , git commit -m y finalmente subir con git push origin master.

Continuando el ejemplo, Emmanuel había clonado el repositorio y subido sus cambios al remoto, en este caso una carpeta con su nombre y un archivo dentro.

Para empezar a realizar nuevos cambios el primer comando que corrió fue el “git pull origin master “ para asegurarse de tener todas las actualizaciones del repositorio y

Diapositiva 16 :

ahí se descargaron las dos nuevas carpetas creadas por sus compañeras.

Diapositiva 17

Si quieres tener más información de los commits hechos hasta el momento podes correr el comando “git log”

Diapositiva 18

Una vez que todos clonaron el repositorio y subieron sus archivos a la nube, ahora en la página de github aparecen todos los archivos de la carpeta que seleccionamos con los commits realizados y estos tienen la descripción de la fecha y autor de los mismos.

Diapositiva 19 ---- Resolver conflictos

Con la posibilidad de que muchas personas trabajen con un proyecto en paralelo puede ocurrir que modifiquen el mismo archivo y es ahí cuando suelen aparecen los conflictos.

Un conflicto se genera cuando dos o más colaboradores modifican el mismo archivo. Entonces, al momento de hacer el ‘pull’ del repositorio, git intenta fusionar los archivos de forma automática si no lo logra puede ser porque, por ejemplo se modifica la misma línea de un archivo. Ahora es necesario resolver manualmente estos conflictos y para ello debemos ingresar a los archivos afectados y elegir cuál será el cambio que quedará definitivo.

Diapositiva 20

Por ejemplo estaban Mica y Viví trabajando juntas en un proyecto, ambas estaban modificando en la carpeta ForsterEmmanuel el archivo ejemplo.txt, Mica sube

primero sus modificaciones al repositorio remoto con los pasos para subir archivos git add . , git commit -m ”modifique el archivo de emmanuel”, y luego subiendo sus cambios con git push origin master.

Diapositiva 21

Cómo Mica fue la primera en hacer el push no tuvo problemas, al Viví ser la segunda se le presenta el conflicto cuando quiere hacer el push, entonces si por ejemplo estuviera trabajando en VSC le aparecerán las opciones que tendría de solucionar el conflicto,

Diapositiva 22 Entre las opciones, puede elegir “aceptar el cambio actual” en este caso el texto “Conflicto”, puede elegir “aceptar cambio entrante” que era lo que subió Mica en este caso “trabajando en error de GitHub con trabajos colaborativos” o puede “aceptar ambos cambios”.

Diapositiva 23

Antes de seleccionar como resolver el conflicto puede consultar la opción de comprar ambos cambios.

Diapositiva 24

Luego, volviendo al archivo Vivi decide cual cambio permanece, en este caso eligió mantener los cambios realizados por Mica.

Diapositiva 25

Una vez solucionado el problema se deben repetir los pasos para subir un archivo, es decir, realizar el add, el commit y un push.

Diapositiva 26---------- BONUS TRACK----------

GITHUB desktop