|  |
| --- |
| PARTE 1 (GESTION DE RAMAS Y FUSIONES BASICAS)    CONSIGNA 1 y 2: INICIALIZAR UN REPOSITORIO Y CREAR UN ARCHIVO LOCAL  Creo una carpeta donde va a ir mi repositorio (mkdir), luego me posiciono dentro de la carpeta (cd) e inicializo mi repositorio git (git init). Luego creo un archivo readme.md (echo) lo agrego al área de preparación de git (git add) y luego guardo y realizo los cambios en el repositorio local (git commit) |
| CONSIGNA 3 y 4: CREAR Y CAMBIAR A UNA RAMA NUEVA Y MODIFICAR EL ARCHIVO EN LA NUEVA RAMA  En la carpeta de mi proyecto, creo una nueva rama y me posiciono dentro de ella con el mismo comando (git checkout -b). Una vez dentro de la rama, creo un archivo readme (con echo), lo agrego al área de preparación (git add) y por ultimo guardo y realizo los cambios en el proyecto (git commit) |
| CONSIGNA 5: VOLVER A LA RAMA PRINCIPAL Y FUSIONAR  Me vuelvo a la rama principal del proyecto (con git checkout), fusiono la rama creada anteriormente con la rama principal del proyecto (git merge) para que los cambios se vean reflejados. Por último, hago que mi proyecto sea un repositorio en la nube de github (git remote add origin) y subo todo el contenido a github (git push) |
| PARTE 2 (MANEJO DE CONFLICTOS EN GIT)    CONSIGNA 6 Y 7: CREAR CONFLICTO POTENCIAL E INTENTAR FUSIONAR PARA GENERARLO  Trabajando con el mismo repositorio, dentro de la rama principal modifico el archivo readme (echo), luego lo agrego al área de preparación (git add), guardo y realizo el cambio en el proyecto (git add), subo el proyecto con sus cambios a github (git push), me posiciono en la rama secundaria (git checkout) y realizo la fusion con la rama principal (git merge).  NOTA: En este paso me debería dar un error, pero no me da error (no sé porque). De todas formas, sigo con los procedimientos para arreglar el problema que debería haber ocurrido |
| CONSIGNA 8 Y 9: RESOLVER EL CONFLICTO, FUSIONAR LA RAMA Y SUBIR LOS CAMBIOS  Abro el archivo readme de forma manual y lo edito para resolver el problema. Después de resolverlos, paso los cambios al área de preparación (git add) y realizo el registro y el cambio (git commit), me cambio a la rama principal (git checkout) y realizo la fusión con la rama secundaria (git merge). Por último, subo los cambios a github (git push) |

|  |
| --- |
| PARTE 3: COLABORACION EN GITHUB CON ORGANIZACIONES  CONSIGNA 10: CREAR UNA ORGANIZACIÓN EN GITHUB  Esta consigna fue realizada en grupo, desde el github de mi compañero creamos la organización en github (practica2Ramas) y luego creamos el repositorio dentro de la organización (ramas) |
| CONSIGNA 11: CREAR UN REPOSITORIO E INVITAR MIEMBROS  En el paso anterior ya habíamos creado la organización y junto con eso el repositorio, luego desde github nos envió invitaciones a la organización para agregarnos como miembros y luego nos cambio los permisos de members a owners |
| CONSIGNA 12 y 13: CLONAR EL REPOSITORIO A LA MAQUINA LOCAL Y COLABORAR EN EL REPOSITORIO  Después de crear la organización y el repositorio, clone el repositorio a mi computadora para trabajarlo de forma local (git clone), luego me posiciono dentro del repositorio (cd), creo una rama (para trabajar de forma local y paralela a mis compañeros) y me posiciono dentro de ella con el mismo comando (git checkout -b). Una vez en la rama, creo un archivo de texto (echo), luego agrego los cambios al área de preparación de mi rama (git add) y chequeo el estado de mi rama (git status) |
| CONSIGNA 14: REVISAR Y FUSIONAR CAMBIOS  Guardo los cambios en mi rama de organización (git commit), me posiciono en la rama principal del proyecto (git checkout), veo las ramas creadas solo para visualizar (git branch), por ultimo hago la fusión con la rama principal (git merge) y subo los cambios a github (git push) |
| CONSIGNA 14: REVISAR CAMBIOS  Reviso los cambios en github. En la primera imagen se pueden ver los commits hechos por mí y mis compañeros y en la segunda imagen se puede ver mi commit con los cambios y las ramas |
| CONSIGNA 15: INVESTIGAR MAS FUNCIONES  En esta parte voy a introducir otros comandos de git que son utilizados pero que no vimos en el practico   * *git log*: Muestra el historial de commits. * *git diff*: Muestra las diferencias entre los archivos en el área de trabajo y el área de preparación. * *git reset* <archivo>: Quita archivos del área de preparación. * *git rm* <archivo>: Elimina archivos del repositorio y del área de trabajo. * *git stash*: Guarda temporalmente los cambios no confirmados para limpiar el área de trabajo. * *git stash pop*: Aplica los cambios guardados en el stash y los elimina del stash. * *git remote*: Gestiona las conexiones a repositorios remotos. * *git fetch*: Descarga objetos y referencias desde otro repositorio. * *git rebase <rama>*: Reaplica commits en la parte superior de otra base. * *git tag <nombre\_etiqueta>*: Crea una nueva etiqueta en el commit actual. |