Bloque 1 - Práctica 2 - Primer repositorio local Git con comandos

Con estos ejercicios configuramos el sistema para realizar el resto de prácticas, por lo que es importante que se realicen tanto en el ordenador del aula como en casa.

1. Abre el directorio LMSGI en VSCode y realiza el resto de ejercicios desde VSCode.

2. Abre un terminal interno, preferible tipo “Git Bash” o “CMD”, y realiza los siguientes ejercicios:
   - Crea un directorio de nombre `b1p2-nombrealumno`, recuerda
     `mkdir b1p2-nombrealumno`
   - Entra en el directorio creado, `cd b1p2-nombrealumno`
   - Muestra el contenido completo del directorio, utiliza `dir` si el terminal es CMD y `dir -a` si es Git Bash
   - Inicializa un repositorio local con `git init` y muestra el contenido, utiliza `dir` si el terminal es CMD y `dir -a` si es Git Bash, ¿qué ha cambiado?

3. Crea en el directorio `b1p2-nombrealumno` un fichero de texto, `ejemplo1.txt`, con una sóla línea con el texto “Primera línea”, no olvides grabarlo (Ctrol+s).

4. Realiza los siguientes ejercicios desde un terminal interno en VSCode con directorio activo `b1p2-nombrealumno`:

   - Utiliza `git status` para identificar el estado del fichero.
   - Identifica las tres áreas descritas en las diapositivas: Working Directory, Staging Area y Local Repo
   - Pasa todos los cambios realizados en Working Directory a Staging area, recuerda `git add .` y comprueba de nuevo el estado con `git status`
   - Haz un commit añadiendo una descripción, `git commit -m “C1”`
   - Ejecuta `git status` y `git log`

5. Añade al fichero `ejemplo1.txt` una segunda línea con el texto “`Segunda línea`” y crea un segundo fichero en el directorio `b1p2-nombrealumno` de nombre `ejemplo2.txt` con el texto “`Primera línea ejemplo2`”, no olvides grabar.

6. Desde el terminal, comprueba el estado, `git status`, y **realiza un nuevo commit, C2, que recoja estos cambios**. **Opcional** - Ejecuta `git log -p` y `git log --stat`

7. Desde el terminal:
   - Comprueba cuál es la rama activa, recuerda `git branch -avv`
   - Crea una nueva rama de nombre develop (`git branch develop`) y haz que sea la rama activa (`git checkout develop`)

8. Añade un tercer fichero en el directorio `b1p2-nombrealumno`, `ejemplo3.txt` con una línea de texto "Primera línea ejemplo3" y modifica el fichero `ejemplo1.txt` añadiendo una tercera línea con el texto "Tercera línea, añadida en el branch develop", no olvides grabar.

9. Desde el terminal, haz un commit con todos los cambios (`git add .` y `git commit -m"C3"`)y comprueba el estado (`git status`) y el log (`git log`). ¿En qué rama del repositorio se han almacenado los cambios?

10. Añade al fichero `ejemplo3.txt` una segunda línea con el texto "Segunda línea ejemplo3" y haz un **commit, C4, que recoja los cambios**.

11. **Opcional** - Compara los cambios realizados en los dos últimos commit, utiliza el comando `git diff id_commit id_commit` o `git diff HEAD~1 HEAD~2`

12. **Opcional** - Compara el último commit con la rama master, utiliza el comando `git diff master HEAD` o `git diff master develop`

13. Cambia al branch master (`git checkout master`) y haz un dir ¿está el fichero ejemplo3.txt? ¿qué contenido tiene el fichero ejemplo ejemplo1.txt?

14. Fusiona la rama master con la develop, recuerda, estando activa master ejecuta `git merge develop`, y comprueba de nuevo el contenido de `ejemplo1.txt` y si está el fichero `ejemplo3.txt`

15. **Opcional** - Cambia al commit con el mensaje C3, utiliza `git checkout id_commit 1DAM`

16. **Opcional** - Cambia a la rama develop (`git checkout develop`) y modifica el contenido del fichero ejemplo3.txt añadiendo una línea "Tercera línea ejemplo3" y ejecuta `git diff` ¿qué información muestra?

17. **Opcional** - Haz un commit que recoja los últimos cambios en la rama develop.

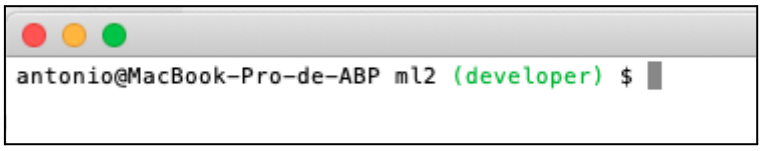
18. Prueba `git blame -e ejemplo1.txt` ¿Qué resultado obtienes?

19. **Opcional** - sigue los pasos explicados en las diapositivas para generar un conflicto al fusionar ramas y resuélvelo.

20. **Opcional** - Prueba los siguientes comandos git: `git rm`, `git mv`, `git diff`, `git reset --soft HEAD~1`, `git reset --mixed HEAD~1`, `git reset --hard HEAD~1`, `git revert commitid`, `git cherry-pick commitid`, ...

21. **Opcional -** Amplía información sobre git stash[1]. [git stash: Cómo guardar los cambios | Atlassian Git Tutorial]

22. **Opcional** - Configura el [terminal en MAC] o en [Windows (CMD)] para que muestre la rama activa cuando nos encontramos en un repositorio Git



[1] Área "stash" donde almacenar temporalmente una captura de los cambios sin enviarlos al repositorio. Está separada del directorio de trabajo (working directory), del área de preparación (staging area) y del repositorio.