

Práctica 3, GDV. MAP curso 22/23.

Trabando en repositorios distribuidos con Git.

1 Previos.

Vamos a trabajar con el SCV distribuido Git.

Para ello tienes que instalártelo primero: <https://git-scm.com/download> . También conviene que os descarguéis un cliente visual de Git de los que aparecen en esta página: <https://git-scm.com/downloads/guis> . En los laboratorios de la facultad están instalados:

- GitHub Desktop.
- Source Tree.
- Tortoise Git.

Esta práctica se realiza usando la consola Git Bash que proporciona Git. La práctica está redactada para realizar los comandos con consola. Simultáneamente, deberás realizar las mismas acciones con un cliente visual y razonar qué ventajas/inconvenientes ves a la operativa para cada una de las acciones que se te piden.

En esta práctica vais a trabajar por parejas, las mismas que las de la práctica anterior voluntaria de SVN (las parejas figuran en el campus virtual en la pestaña de Prácticas).

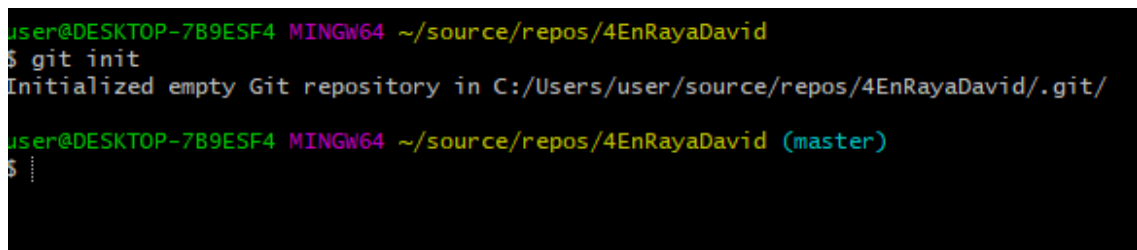
Uno de los miembros del equipo, va a crear un directorio de trabajo en local para el proyecto, donde va a comenzar una implementación de un aplicación o programa. En este punto, únicamente hay que crear una estructura general de proyecto y una visualización muy simple. Llama a tu directorio de trabajo GrupoXX (siendo XX vuestro número de grupo. Mirad la lista si no os acordáis).

2 Inicializar un repositorio (Estudiante A)

Abre la consola Git Bash, sitúate (usando la instrucción `cd`) hasta posicionarte en el directorio GrupoXX.

- Inicializa un repositorio:

```
$git init
```



```
user@DESKTOP-7B9ESF4 MINGW64 ~/source/repos/4EnRayaDavid
$ git init
Initialized empty Git repository in C:/Users/user/source/repos/4EnRayaDavid/.git/
user@DESKTOP-7B9ESF4 MINGW64 ~/source/repos/4EnRayaDavid (master)
$
```

Una vez hecho esto podemos ver que se ha creado un repositorio oculto llamado `.git`, que contiene la información sobre nuestro SCV. Al ser un repositorio distribuido, si nos liamos siempre podemos empezar desde el principio borrando el `.git` y comenzando con el estado inicial del proyecto.

.git	07/03/2022 14:09	Carpeta de archivos	
.vs	23/02/2022 18:13	Carpeta de archivos	
Debug	23/02/2022 18:15	Carpeta de archivos	
4EnRayaDavid.sln	23/02/2022 18:13	Visual Studio Solution	2 KB
4EnRayaDavid.vcxproj	23/02/2022 18:15	VC++ Project	8 KB
4EnRayaDavid.vcxproj.filters	23/02/2022 18:15	VC++ Project Filters File	1 KB
4EnRayaDavid.vcxproj.user	23/02/2022 18:13	Per-User Project Options File	1 KB
Source.cpp	23/02/2022 18:15	C++ Source	4 KB

- Ahora configura los datos de usuario y correo para este repositorio.

```
$git config user.name "Tu nombre completo"
$git config user.email tucorreo@ucm.es
```

```
user@DESKTOP-7B9ESF4 MINGW64 ~/source/repos/4EnRayaDavid (master)
$ git config user.name "pilarsancho"

user@DESKTOP-7B9ESF4 MINGW64 ~/source/repos/4EnRayaDavid (master)
$ git config user.email psancho@ucm.es

user@DESKTOP-7B9ESF4 MINGW64 ~/source/repos/4EnRayaDavid (master)
$ |
```

- Primer commit:

Ejecuta `$git status` (Copia lo que sale en la hoja de resultados).

```
user@DESKTOP-7B9ESF4 MINGW64 ~/source/repos/4EnRayaDavid (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    .vs/
    4EnRayaDavid.sln
    4EnRayaDavid.vcxproj
    4EnRayaDavid.vcxproj.filters
    4EnRayaDavid.vcxproj.user
    Debug/
    Source.cpp

nothing added to commit but untracked files present (use "git add" to track)
```

En estos momentos ninguno de los archivos que contiene el directorio está bajo sistema del control de versiones. Como primer paso, que añadirlo al índice (Staging Area):

```
$git add .
```

Copia los comandos y lo que aparece en consola en la Hoja de Resultados.

```
user@DESKTOP-7B9ESF4 MINGW64 ~/source/repos/4EnRayaDavid (master)
$ git add .
```

Comprueba que todos los archivos se han añadido al índice.

```
$git status
```

```
user@DESKTOP-7B9ESF4 MINGW64 ~/source/repos/4EnRayaDavid (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   .vs/4EnRayaDavid/v16/.suo
    new file:   .vs/4EnRayaDavid/v16/Browse.VC.db
    new file:   .vs/4EnRayaDavid/v16/ipch/AutoPCH/6ba61c127dd05cab/SOURCE.ipch
    new file:   4EnRayaDavid.sln
    new file:   4EnRayaDavid.vcxproj
    new file:   4EnRayaDavid.vcxproj.filters
    new file:   4EnRayaDavid.vcxproj.user
    new file:   Debug/4EnRayaDavid.exe
    new file:   Debug/4EnRayaDavid.exe.recipe
    new file:   Debug/4EnRayaDavid.ilc
    new file:   Debug/4EnRayaDavid.log
    new file:   Debug/4EnRayaDavid.pdb
    new file:   Debug/4EnRayaDavid.tlog/4EnRayaDavid.lastbuildstate
    new file:   Debug/4EnRayaDavid.tlog/CL.command.1.tlog
    new file:   Debug/4EnRayaDavid.tlog/CL.read.1.tlog
    new file:   Debug/4EnRayaDavid.tlog/CL.write.1.tlog
    new file:   Debug/4EnRayaDavid.tlog/link.command.1.tlog
    new file:   Debug/4EnRayaDavid.tlog/link.read.1.tlog
    new file:   Debug/4EnRayaDavid.tlog/link.write.1.tlog
    new file:   Debug/4EnRayaDavid.vcxproj.FileListAbsolute.txt
    new file:   Debug/Source.obj
    new file:   Debug/vc142.idb
    new file:   Debug/vc142.pdb
    new file:   Source.cpp
```

Realiza ahora tu primer commit:

```
$git commit -m "Archivos iniciales añadidos al índice del repositorio"
```

Copia los comandos y lo que aparece en consola en la Hoja de Resultados y explica a continuación qué ha sucedido al ejecutar el commit.

```

user@DESKTOP-7B9ESF4 MINGW64 ~/source/repos/4EnRayaDavid (master)
$ git commit -m "Archivos iniciales añadidos al repositorio local"
[master (root-commit) c3b111d] Archivos iniciales añadidos al repositorio local
24 files changed, 360 insertions(+)
create mode 100644 .vs/4EnRayaDavid/v16/.suo
create mode 100644 .vs/4EnRayaDavid/v16/Browse.VC.db
create mode 100644 .vs/4EnRayaDavid/v16/ipch/AutoPCH/6ba61c127dd05cab/SOURCE.ipch
create mode 100644 4EnRayaDavid.sln
create mode 100644 4EnRayaDavid.vcxproj
create mode 100644 4EnRayaDavid.vcxproj.filters
create mode 100644 4EnRayaDavid.vcxproj.user
create mode 100644 Debug/4EnRayaDavid.exe
create mode 100644 Debug/4EnRayaDavid.exe.recipe
create mode 100644 Debug/4EnRayaDavid.ilc
create mode 100644 Debug/4EnRayaDavid.log
create mode 100644 Debug/4EnRayaDavid.pdb
create mode 100644 Debug/4EnRayaDavid.tlog/4EnRayaDavid.lastbuildstate
create mode 100644 Debug/4EnRayaDavid.tlog/CL.command.1.tlog
create mode 100644 Debug/4EnRayaDavid.tlog/CL.read.1.tlog
create mode 100644 Debug/4EnRayaDavid.tlog/CL.write.1.tlog
create mode 100644 Debug/4EnRayaDavid.tlog/link.command.1.tlog
create mode 100644 Debug/4EnRayaDavid.tlog/link.read.1.tlog
create mode 100644 Debug/4EnRayaDavid.tlog/link.write.1.tlog
create mode 100644 Debug/4EnRayaDavid.vcxproj.FileListAbsolute.txt
create mode 100644 Debug/Source.obj
create mode 100644 Debug/vc142.idb
create mode 100644 Debug/vc142.pdb
create mode 100644 Source.cpp

```

2.1 Ignorando archivos.

1. Abre el proyecto de tu copia de trabajo y ejecútalo.
2. Abre alguno de los archivos del proyecto y sin añadir nada, vuelve a cerrarlo.
3. Ejecuta un `$git status`

Copia los comandos y lo que aparece en consola en la Hoja de Resultados.

```

user@DESKTOP-7B9ESF4 MINGW64 ~/source/repos/4EnRayaDavid (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   .vs/4EnRayaDavid/v16/.suo
        modified:   .vs/4EnRayaDavid/v16/Browse.VC.db

no changes added to commit (use "git add" and/or "git commit -a")

```

Verás que aparecen archivos y carpetas para incorporar al repositorio. Todos estos archivos son *generados* por lo que no necesitas subirlos. En Git podemos añadir archivos `.gitignore` a cada carpeta. Estos archivos especifican qué archivos y carpetas no queremos que estén bajo el control de Git. GitHub te ofrece una lista oficial de los archivos `.gitignore` para muchos SO, entornos y lenguajes (<https://www.toptal.com/developers/gitignore>).

A continuación, dentro del directorio de trabajo:

1. Añade un archivo `.gitignore` con el contenido de los archivos que necesitan ser ignorados en tu proyecto (ATENCIÓN! EL ARCHIVO DEBE TENER EXTENSIÓN `.gitignore`).
2. Vuelve a ejecutar `$git status`

Copia los comandos y lo que aparece en consola en la Hoja de Resultados. Puedes comprobar que ahora únicamente aparece como cambio la incorporación del archivo .gitignore, por tanto:

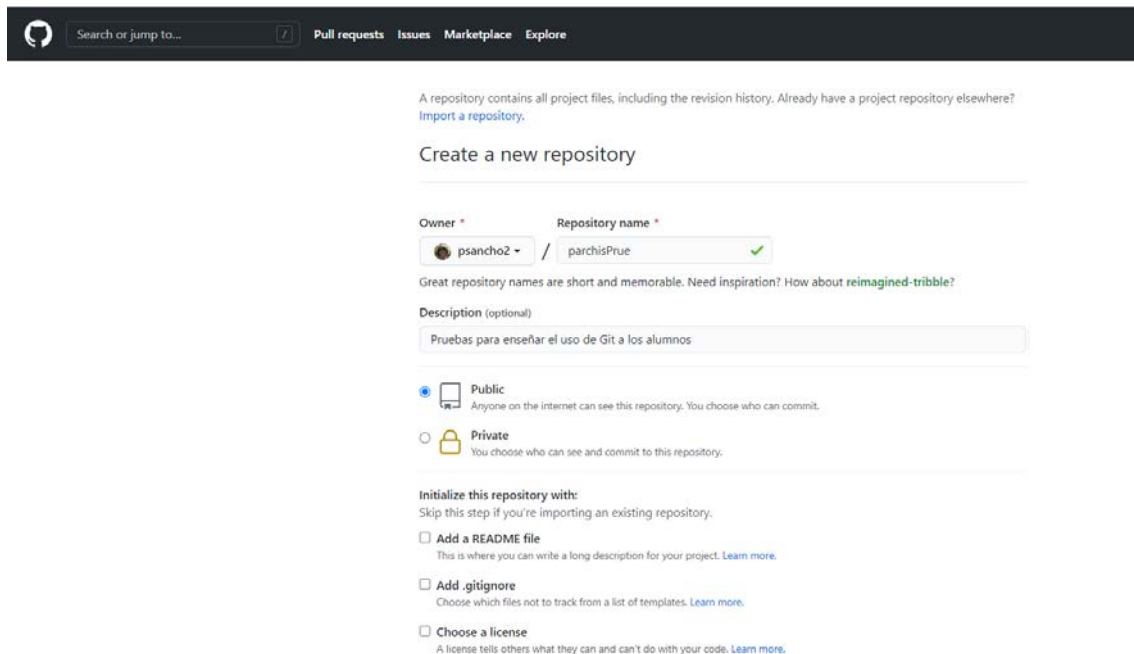
3. Añádelo al índice con `$git add .`
4. Ejecuta de nuevo un `$git status` y comprueba que ya no aparece.
5. Haz un `$git commit` con el mensaje “Añadido archivo gitignore”

Copia los comandos y lo que aparece en consola en la Hoja de Resultados.

3 Subida al repositorio remoto (Estudiante A)



Hasta ahora has estado trabajando en local. Tenías tu copia de trabajo y tu repositorio local, que no compartías con nadie.

Sube tu repositorio a un repositorio remoto en GitHub con el nombre GrupoXX en tu cuenta de GitHub. Atención, que primero tienes que crearte el repositorio en tu cuenta de GitHub.



A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Create a new repository

Owner ^{*}  psancho2 / Repository name ^{*} parchisPrue 

Great repository names are short and memorable. Need inspiration? How about [reimagined-tribble?](#)

Description (optional)
Pruebas para enseñar el uso de Git a los alumnos

☒ Public
Anyone on the Internet can see this repository. You choose who can commit.

☐ Private
You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.

☐ Add a README file
This is where you can write a long description for your project. [Learn more.](#)

☐ Add .gitignore
Choose which files not to track from a list of templates. [Learn more.](#)

☐ Choose a license
A license tells others what they can and can't do with your code. [Learn more.](#)

```
$git remote add origin <direccion del repositorio remoto>
```

```
$git branch -M main
```

```
$git push -u origin main
```

```

user@DESKTOP-7B9ESF4 MINGW64 /c/estudiante1/gitPruebas (master)
$ git remote add origin https://github.com/psancho2/MAP21_22.git

user@DESKTOP-7B9ESF4 MINGW64 /c/estudiante1/gitPruebas (master)
$ git branch -M main

user@DESKTOP-7B9ESF4 MINGW64 /c/estudiante1/gitPruebas (main)
$ git push -u origin main
Enumerating objects: 170, done.
Counting objects: 100% (170/170), done.
Delta compression using up to 8 threads
Compressing objects: 100% (115/115), done.
Writing objects: 100% (170/170), 88.61 MiB | 1.33 MiB/s, done.
Total 170 (delta 72), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (72/72), done.
To https://github.com/psancho2/MAP21_22.git
 * [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.

```

Esto copia todos los elementos del repositorio local al remoto. Haz una captura de pantalla del estado de tu repositorio remoto.

Explica qué crees que hace cada uno de estos tres comandos y por qué usamos los tres.

4 Clonación de un repositorio remoto

4.1 Estudiante B

Te vas ahora a sincronizar con el repositorio remoto que ha creado tu compañero. Para ello:

1. Crea una carpeta en tu ordenador.
2. Abre la consola de Git Bash y ve hasta la carpeta recién creada.
3. Ejecuta el siguiente comando:

```
$git clone <direccion url del repositorio remoto>
```

Copia los comandos y lo que aparece en consola en la Hoja de Resultados. Por ejemplo:

```

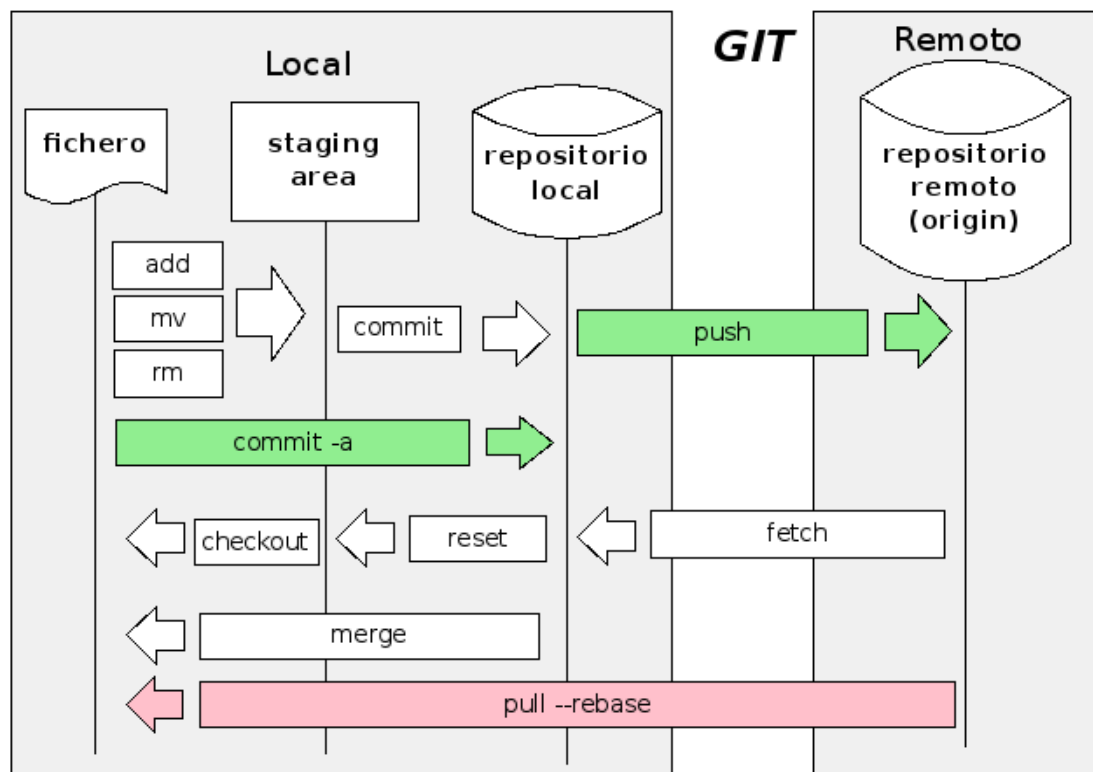
user@DESKTOP-7B9ESF4 MINGW64 /c/estudiante2/gitPruebas
$ git clone https://github.com/psancho2/MAP21_22
Cloning into 'MAP21_22'...
remote: Enumerating objects: 170, done.
remote: Counting objects: 100% (170/170), done.
remote: Compressing objects: 100% (43/43), done.
remote: Total 170 (delta 72), reused 170 (delta 72), pack-reused 0
Receiving objects: 100% (170/170), 88.61 MiB | 2.53 MiB/s, done.
Resolving deltas: 100% (72/72), done.
Updating files: 100% (101/101), done.

```

Una vez hecho esto, configura tu nombre de usuario y tu email.

5 Ciclo de trabajo

Recuerda el flujo de trabajo Git en todo momento:



5.1 Gestión de Cambios.

Estudiante A

1. Crea un archivo `README.md` en el directorio de trabajo. Con un editor de texto, escribe el contenido descriptivo del proyecto en este archivo. P.e:

```
#Resumen de la practica
```

```
Habituarnos al ciclo de trabajo con Git
```

2. Desde la consola Git Bash entra en el directorio de tu copia de trabajo.
3. Ejecuta un

```
$git status
```

Y copia en la hoja de resultados lo que obtienes.

4. Añádelo primero al stage y luego al repo local (pega aquí comandos y resultados).
5. Ejecuta de nuevo un `$git status`
6. Ahora sube todo al repositorio remoto.

Copia todo lo que has obtenido en consola en la hoja de resultados.

Estudiante B

1. Descarga el contenido del repositorio remoto.
2. Comprueba que el archivo `README.md` está y tiene el contenido añadido por el Estudiante A.

Haz captura de pantalla de todos los pasos y cópialas en la hoja de resultados.

5.2 Logs

Estudiante A

1. Ejecuta el comando `$git log`.
2. Hay otro `git log`, más interesante que nos va a ser más útil a lo largo de la práctica porque aporta mucha información sobre el funcionamiento de git:

```
$git log --pretty=format:"%h %ad | %s%d [%an]" --graph --date=short --all
```

Haz captura de pantalla de cada uno de ellos pasos y copia todo lo que te ha aparecido en consola en la hoja de resultados. Explica qué significa cada cosa paso a paso.

3. Prueba el comando que ha propuesto Luis Cabello en el foro de clase:

```
git config --global alias.graph "log --graph --abbrev-commit --decorate --format=format:'%C(bold blue)%h%C(reset) - %C(bold green)(%ar)%C(reset) %C(white)%s%C(reset) %C(dim white)- %an%C(reset)%C(bold yellow)%d%C(reset)' -all"
```

Haz captura de pantalla de lo que obtienes y comenta el resultado con relación al comando anterior.

5.3 Cread un conflicto y resuelto.

En este punto se os pide crear conflictos:

- Tipo 1. Se modifica la copia de trabajo y no se suben los cambios al repositorio local. A continuación se intentan bajar los cambios de la copia en el remoto (`$git pull`).
- Tipo 2. Se modifican simultáneamente el mismo archivo (por ejemplo en README.md) en ambas copias de trabajo y se cumple con el ciclo de trabajo correctamente. Documentad todo el proceso (el conflicto generado y su resolución) copiando las capturas de pantalla de todos los pasos en la hoja de resultados.
- Tipo 3. Se modifica la copia local (sin haber pulleado previamente), se suben los cambios al repo local y se intentan pushear al remoto.
- Modificad ahora archivos diferentes de ambas copias de trabajo subid los cambios al remoto (cumpliendo correctamente con el ciclo de trabajo).

Documentad todos los pasos y explicad cómo ha actuado Git en esta situación y cuál ha sido vuestra intervención para resolver el conflicto..

6 Recuperando versiones anteriores.

1. Modifica uno de los archivos de tu proyecto, súbelo al stage y de ahí lo commiteas al local.
2. Vuelve a modificar el mismo archivo... imagina que en estas nuevas modificaciones te has hecho un lío con el código que no sabes cómo subsanar y decides que te va dar menos trabajo recuperar el último archivo válido. Puedes recuperar el archivo commiteado en el paso 1 mediante el comando.

```
$git checkout main [nombre_fichero]
```

3. Imagínate que esto no es suficiente, y que el commit anterior también lo quieres deshacer. Puedes recuperar cualquier versión commiteada al remoto mediante checkout, no sólo de un fichero, sino de todo el proyecto completo. Para ello, primero tienes que hacer un log y localizar el commit a partir del cual quieres restaurar tu copia local:


```
$git log --oneline
```

```
user@DESKTOP-7B9ESF4 MINGW64 ~/source/repos/_2023MAP/LightUp (main)
$ git log --oneline
5cbdb16 (HEAD -> main, origin/main) Merge branch 'main' of https://github.com/psancho2/AkariPuzzle Este merge es impres
cindible
41c8d7f quiero conflictos
c2fdffd main modificado usuario 2
36afa15 cambios en main y readme
98c4714 primer commit con el proyecto completo inicial

user@DESKTOP-7B9ESF4 MINGW64 ~/source/repos/_2023MAP/LightUp (main)
$
```

Este log te muestra un resumen de los últimos commit subidos. Localizamos el número de commit que es y hacemos su checkout:

```
$git checkout 36afa15 readme.txt
```

Restaura el readme.txt al estado en que estaba en ese commit. Para restaurar todos los archivos del proyecto basta con omitir el nombre del archivo.

Como al hacer esto has modificado la copia local, tendrás que meter los cambios en el stage y luego subirlos.

¡¡¡¡ATENCIÓN `git checkout` es un comando peligroso!!! Puede destruir todo el trabajo de tu copia local, que no es recuperable.

7 Trabajando con ramas

7.1 Crear una rama addColor

Estudiante A

En Git es muy común usar ramas para añadir nuevas funcionalidades o para arreglar errores, en lugar de continuar en la rama principal (main). El estudiante A va a crear una nueva rama para implementar una nueva funcionalidad a vuestro proyecto. Para ello:

- Creará la nueva rama.
- Te posicionarás en la rama para hacer las modificaciones y crear nuevo contenido:
 - Añade uno o varios nuevos ficheros para añadir nueva funcionalidad al juego (no tiene por qué ser complicada... lo que se te ocurra, cambiar los colores o lo que sea. En esta práctica se trata de acostumbrarse a trabajar con Git no de programar).
- Añadirás los cambios al índice y luego hará un commit a su repositorio local.
- Ejecutarás un log.

Como has añadido nueva funcionalidad al juego, vas a comentarlo también en el README.md, pero esta vez en la rama main.

- Cambiará a la rama main y comprobará que está posicionado en ella.
- Modificará el README.md añadiendo el texto:
 - Nueva funcionalidad añadida: customización de los colores del juego.
- Añade el README.md modificado al índice y haz un commit con el mensaje "Modificación del README.md"
- Ejecuta un log

```
$ git log --pretty=format:"%h %ad | %s%d [%an]" --graph --date=short --all
```

Copia todos los comandos y lo que te ha salido en consola y explica cuál es tu interpretación del último log que has obtenido.

Si compruebas ahora el directorio en el que estás verás que el fichero que has añadido antes no está... esto es porque estás en la rama main (los nuevos ficheros han sido añadido para la rama addColor).

Windows (C:) > estudiante1 > gitPruebas > 2021ExamenJunio1

Nombre	Fecha de modificación
.vs	07/03/2022 15:46
Debug	07/03/2022 15:46
2021ExamenJunio1.sln	10/06/2021 9:08
2021ExamenJunio1.vcxproj	08/03/2022 17:07
2021ExamenJunio1.vcxproj.filters	08/03/2022 17:07
2021ExamenJunio1.vcxproj.user	10/06/2021 9:08
Alejandro Gomez Ejercicio1 Examen.cpp	10/06/2021 9:58
matriz.txt	31/05/2021 3:32
matriz2.txt	31/05/2021 6:37

Si ahora cambias a tu rama, comprobarás que el readme.txt no tiene las modificaciones las modificaciones que hiciste (se hicieron en la rama main), pero sí están tus ficheros.

Windows (C:) > estudiante1 > gitPruebas > 2021ExamenJunio1

Nombre	Fecha de modificación
.vs	07/03/2022 15:46
Debug	07/03/2022 15:46
2021ExamenJunio1.sln	10/06/2021 9:08
2021ExamenJunio1.vcxproj	08/03/2022 17:12
2021ExamenJunio1.vcxproj.filters	08/03/2022 17:12
2021ExamenJunio1.vcxproj.user	10/06/2021 9:08
addColor.cpp	08/03/2022 17:12
addColor.h	08/03/2022 17:12
Alejandro Gomez Ejercicio1 Examen.cpp	10/06/2021 9:58
matriz.txt	31/05/2021 3:32
matriz2.txt	31/05/2021 6:37

En resumen: cuando creas una rama, los cambios realizados en la rama son independientes de los cambios realizados en la rama main.

Copia todos los resultados que has obtenido en la consola en la hoja de resultados.

Ahora vamos a fusionar las dos ramas:

- Vuelve a la rama main.
- Ejecuta la instrucción para fusionar las ramas (poniendo un comentario explicativo a la hora de hacer la fusión).
- Ejecuta un log para ver que todo es correcto.
- Envía todos estos cambios al repositorio remoto.

Copia todos los comandos y lo que te ha salido en consola y explica cuál es tu interpretación del último log que has obtenido.

Estudiante B

Vas a actualizar tu directorio local con el remoto para incorporar los cambios que ha hecho tu compañero.

A continuación ejecuta un

```
$git log --pretty=format: "%h %ad | %s%d [%an]" --graph --date=short --all
```

Y compara los resultados con el último hecho por tu compañero. Explica la ligera diferencia que hay entre ambos logs y razona su por qué.

7.2 Crear rama scrollPieces

Estudiante B

Ahora será el Estudiante B quien cree una rama porque quiere añadir una nueva funcionalidad para poder situar las piezas con el ratón en la posición correcta del tablero (recuerda que la finalidad de esta práctica no es implementar código, lo puedes dejar en crear el fichero y escribir dentro un comentario).

- Ejecuta el comando para ver que estás posicionado en la nueva rama creada.
- Crea tu nuevo fichero, añádelo al índice y haz un `commit` con el mensaje “Creada la funcionalidad para rotar piezas”
- Ahora modifica el README.md de la main añadiendo el texto:
 - Nueva funcionalidad para rotar piezas con el ratón.
- Añádelo al índice y sube los cambios con el `commit` con el correspondiente mensaje.
- Ejecuta un

```
$git log --pretty=format: "%h %ad | %s%d [%an]" --graph --date=short --all
```

- Sincroniza tu repositorio la rama activa (main) con el remoto.
- Ejecuta ahora la instrucción:

```
$git branch -a
```

Copia todos los comandos y resultado de la consola en la hoja de resultados. Explica qué está pasando.

Estudiante A

- Sincroniza tu repositorio con el remoto.
- Fusiona las ramas scrollPieces y master.
- Sube los cambios al remoto.

Copia todos los comandos y resultado de la consola en la hoja de resultados. Explica qué está pasando y sincroniza el repositorio completo.

7.3 Tags

Vamos a terminar creando un tag de nuestro repositorio. Esto es una versión con nombre (se suele hacer al final de un sprint para crear una release estática del repositorio a la que se pueda volver fácilmente).

Estudiante A

- Ejecuta el comando:

```
$git tag -a v1.0 -m "Versión con la parte obligatoria de la práctica"
```

- Ejecuta un log.
- Para ver lo que tenemos ejecuta:

```
$git show v1.0
```

- Súbelo al repositorio remoto.

Copia los resultados de consola en la hoja de resultados y explica con tus palabras qué ha hecho este proceso.

Estudiante B

- Sincronízate con el repositorio central.
- Visualiza los detalles del tag creado por tu compañero.

Copia los resultados de consola en la hoja de resultados.

8 Vuelve a repetir toda la práctica con aplicación de interfaz visual y comenta cada paso en relación con la consola.