Módulo 1 - Tema 1

Git & GitHub

Basilio David Gómez Fernández

Índice

1. ¿Por qué necesitas un control de versiones?

2. Bienvenidos a Git

Por aquí te dejo tu biblia durante esta semana.

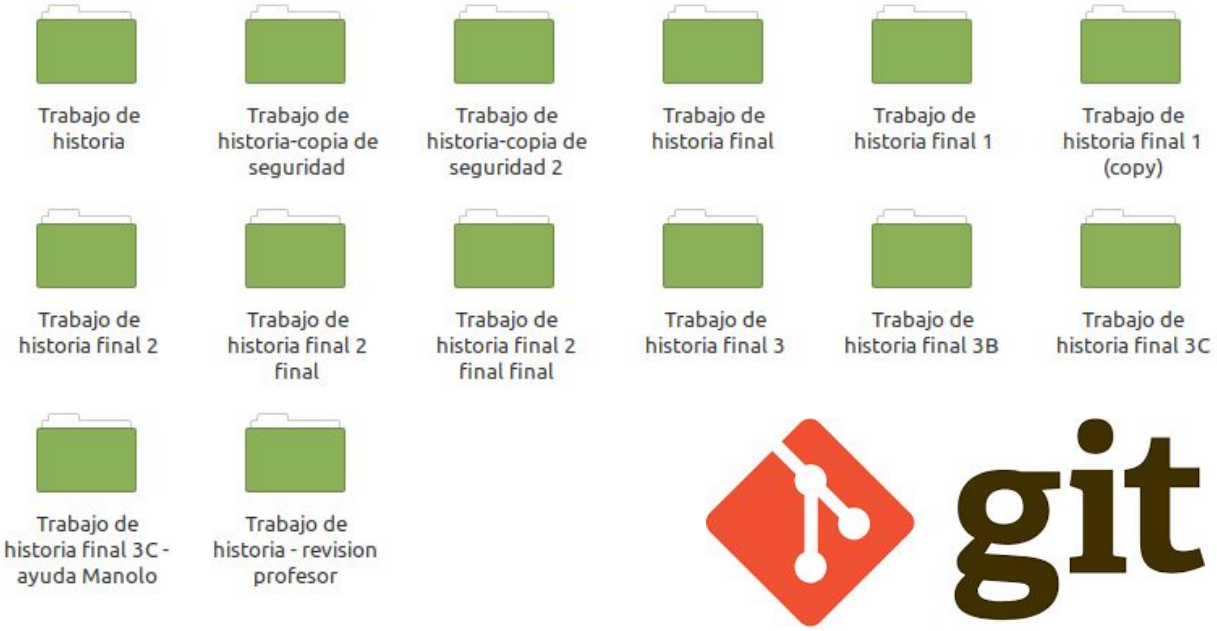
(y el resto de tu vida como desarrollador)

https://git-scm.com/docs

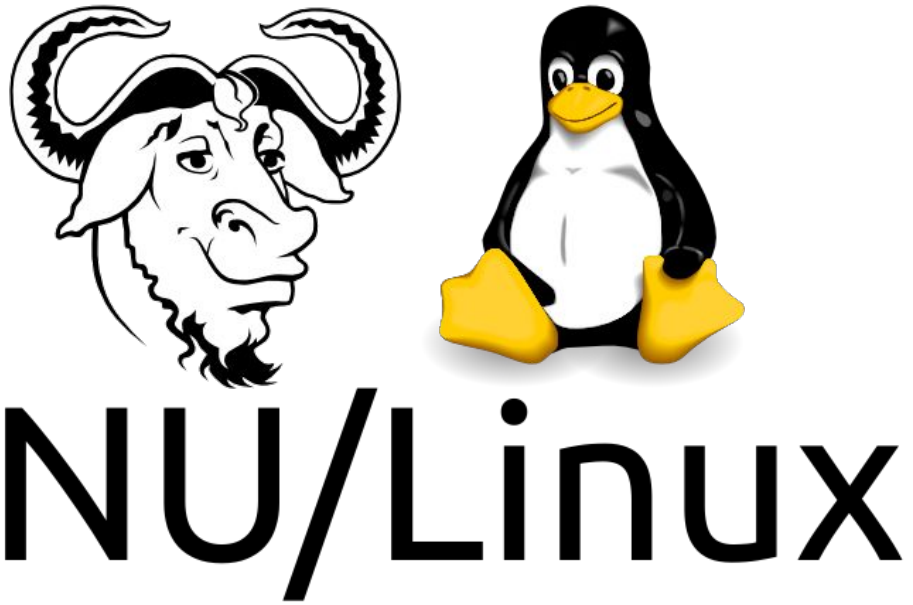
3. ¿Qué es el control de versiones?

Los programas de control de versiones son herramientas para gestionar las versiones de nuestro código de forma ordenada.

3. ¿Qué es el control de versiones?



4. ¿Cómo usamos git?

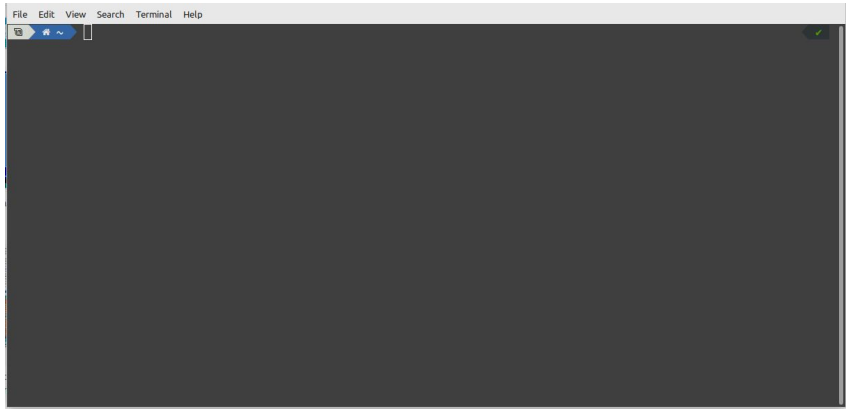


4.5. Qué y Por qué: Comandos

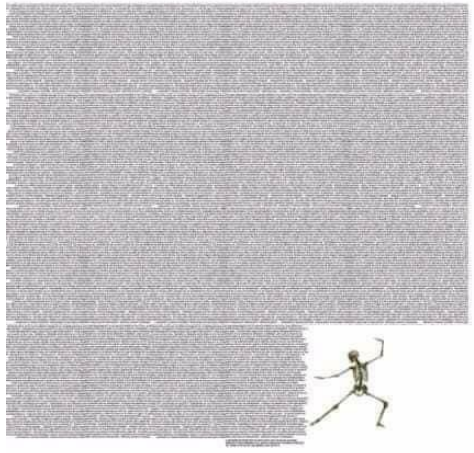
```
Leer archivo
Entrar en una carpeta
Editar un fichero
Descargar archivo de internet
```

4.5. Qué y Por qué: Comandos

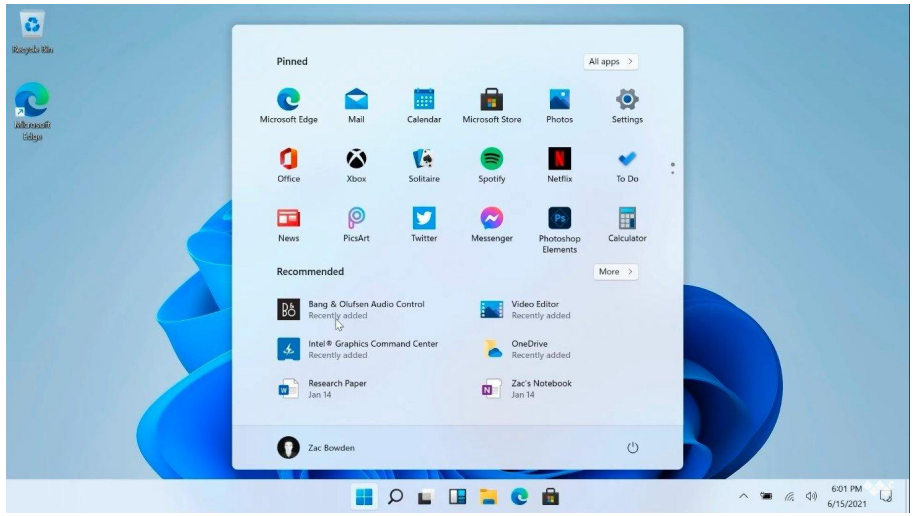
Interfaz de texto



Ventajas:



Interfaz gráfica



Ventajas:

✅ Es bonita

4.5 Qué y Por qué: Comandos

```
comando ...[--option1/--option2] [<argumento 1> ...<argumento N>]
```

`...` -> Puede haber uno o más

`[ ]` -> Puede haber cero o uno

`< >` -> Tiene que ser añadido por el usuario

`/` -> Puede incluir una o ambas opciones

4.5 Qué y Por qué: Comandos

Comandos

1. `mkdir <carpeta>` -> Crea una nueva carpeta
2. `cd <carpeta>` -> Muévete entre carpetas
3. `ls [<carpeta>]` -> Muestra todos los ficheros y carpetas en sistemas Linux
4. `dir [<carpeta>]` -> Muestra todos los ficheros y carpetas en sistemas Windows
5. `nano <fichero>` -> Crea y edita ficheros
6. `cat <fichero>` -> Muestra el contenido de un fichero
7. `rm [-r] <fichero/carpeta>` -> Borra ficheros y carpetas
8. `code .` -> Abre el visual studio code en la carpeta actual

4.5 Qué y Por qué: Comandos

Teoría: Ruta Relativa vs Ruta Absoluta

Ruta relativa es la que empieza por: `.`

Mientras que la ruta absoluta es la que empieza por: `C:\` o `/`

Teoría: Moverte a la carpeta anterior

La carpeta anterior se llama: `..`

CODE SPACE ACADEMY

4.5 Qué y Por qué: Comandos

Ejercicio

Crea una carpeta

que contenga el siguiente

árbol de directorios y llena

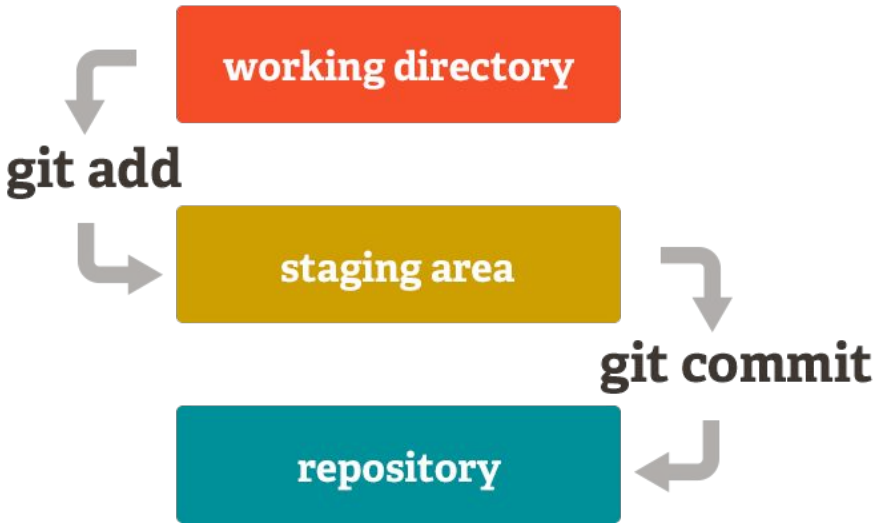
los archivos con el contenido

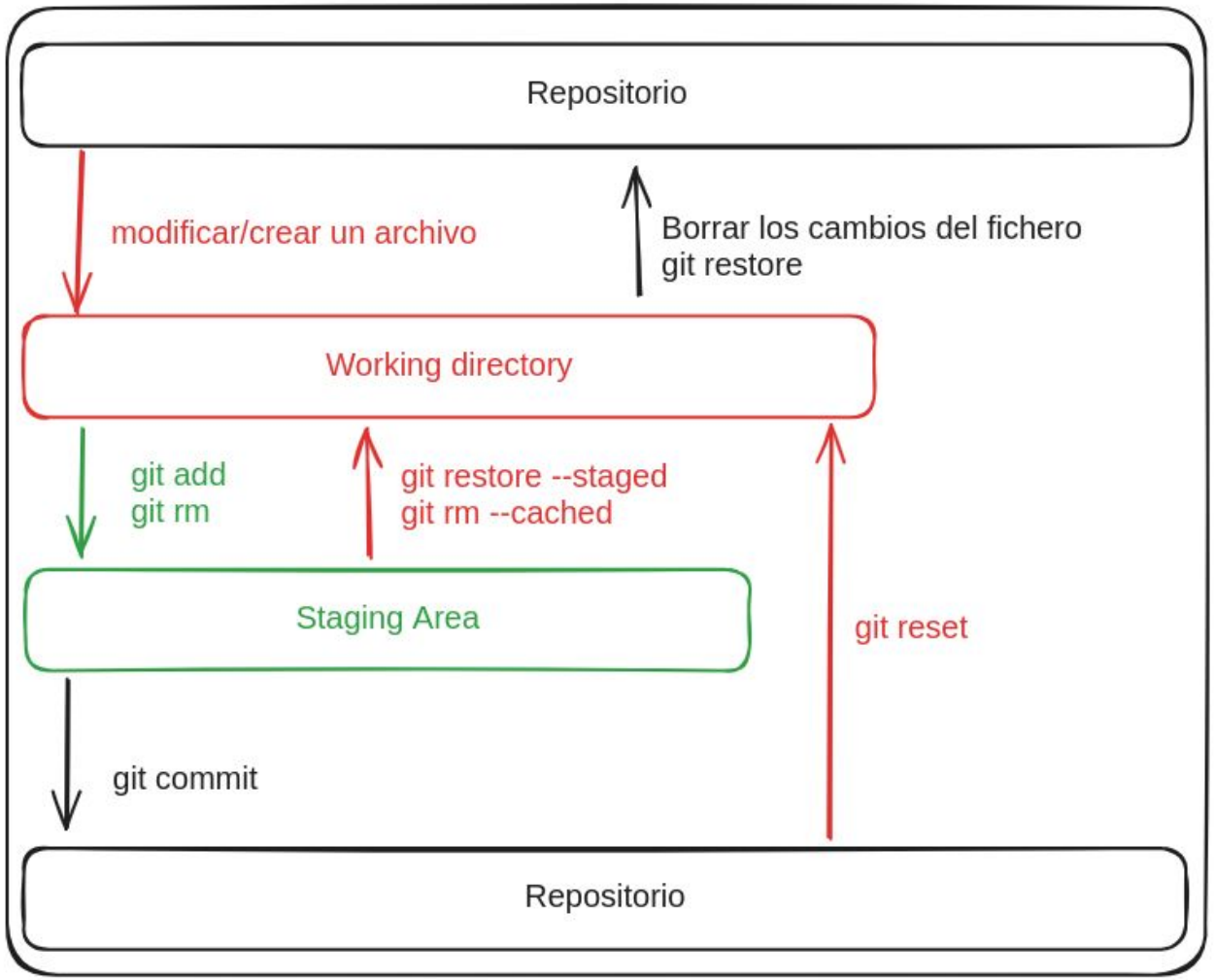
correspondiente usando solo

comandos

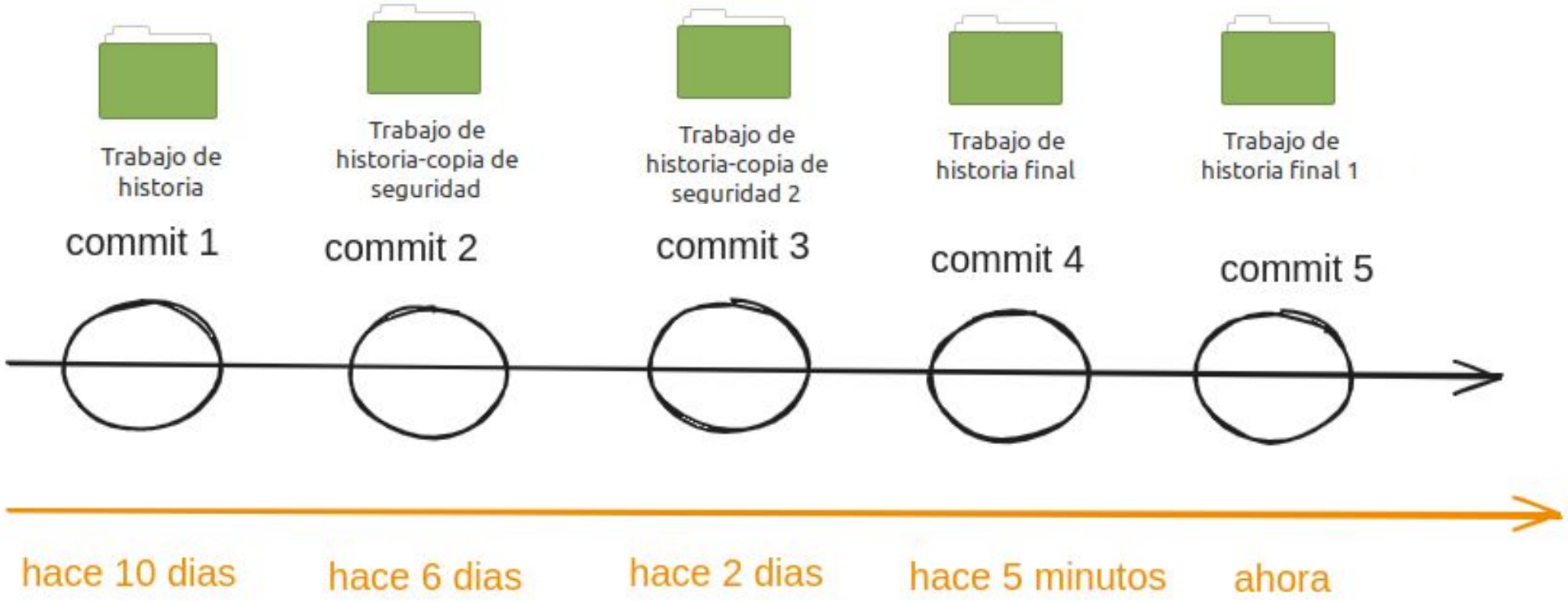
```
.
├── git
│   └── instructions.txt
├── informatica
│   └── index.html
├── linux
│   ├── distribuciones-de-linux.md
│   ├── linux-mint
│   └── ubuntu
└── pc

6 directories, 3 files
❯ cat git/instructions.txt
En este curso aprenderemos todo lo basico de git%
❯ cat informatica/index.html
<html>
  <head>
    Informatica
  </head>
  <body>
    <h1>Informatica</h1>
  </body>
</html>
❯ cat linux/distribuciones-de-linux.md
# Las distribuciones son para Linux lo mismo que es el coche para el motor

# Las distribuciones son el coche y linux es el motor que llevan
```

5. Domina el arte del commit





5. Domina el arte del commit



5. Domina el arte del commit

Comandos

1. `git init` **->** Inicializa un repositorio de Git en la carpeta donde te encuentras
2. `git add <fichero/carpeta>` **->** Añade un archivo al área de staging
3. `git status` **->** Mira el estado de tus cambios
4. `git commit ...[-m <commit message>/--ammend]` **->** Guarda toda la información del staging area en el repositorio
5. `git log [--oneline]` **->** Obtén información de todos los commits

5. Domina el arte del commit

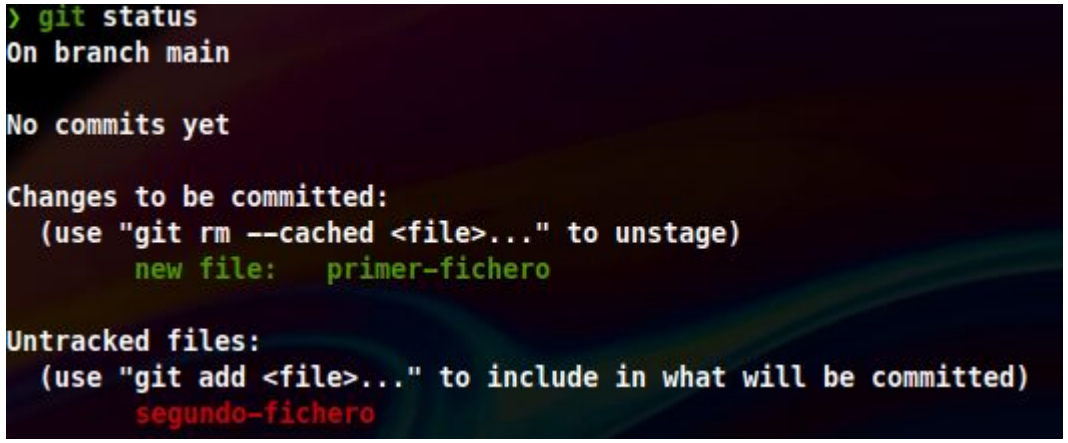
Comandos

1. `git restore [--staged] <fichero>` -> Restaura un fichero.
2. `git rm [--cached] <fichero>` -> Elimina un fichero.
3. `git reset [--soft/--mixed/--hard] <commit>/HEAD^/HEAD~N` ->

   Elimina un commit.

5. Domina el arte del commit

1. Crea en una carpeta llamada ‘CodeSpace’ un repositorio git.
2. Haz un comit de un fichero y que el nombre del commit sea 'Primer commit'.
3. Haz 3 commits y obten toda la información que puedas de ellos.
4. Edita un fichero, haz `git add`, vuelve a editarlo y realiza un commit, ¿Qué ha pasado? Explícalo!
5. Explica todo lo que puedas sobre la foto de la derecha.



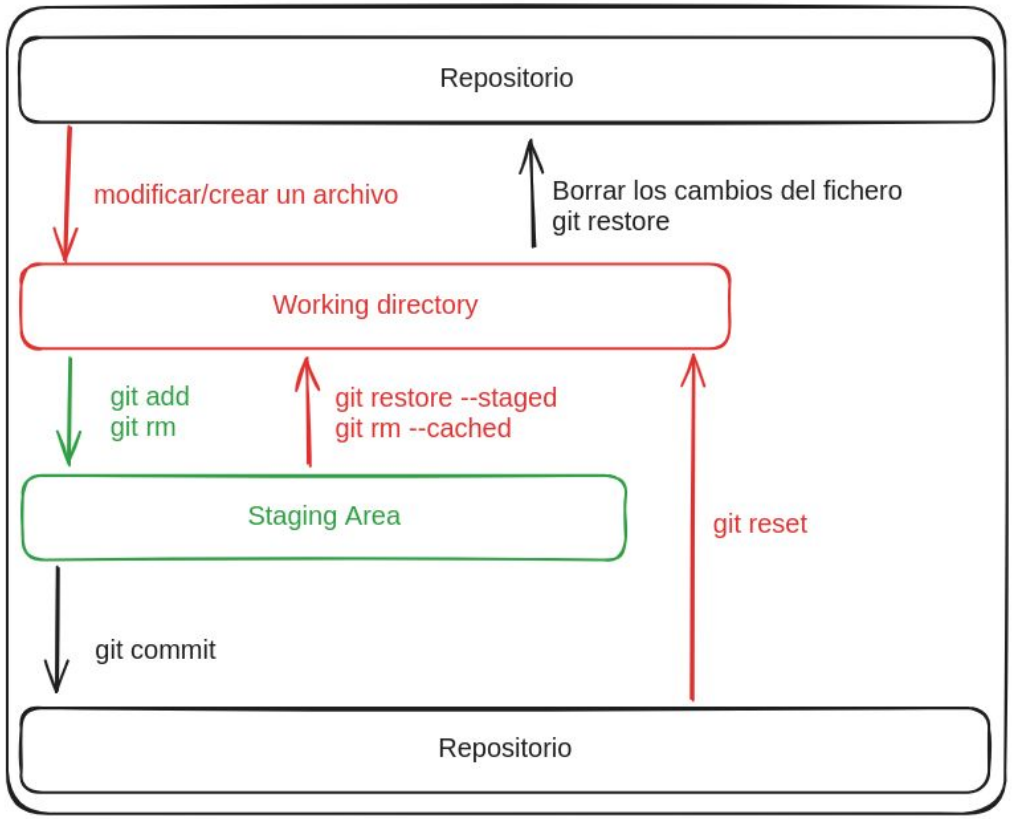
5. Domina el arte del commit

Para empezar crea un repositorio git con 3 commits y 3 ficheros y realiza los siguientes ejercicios en orden

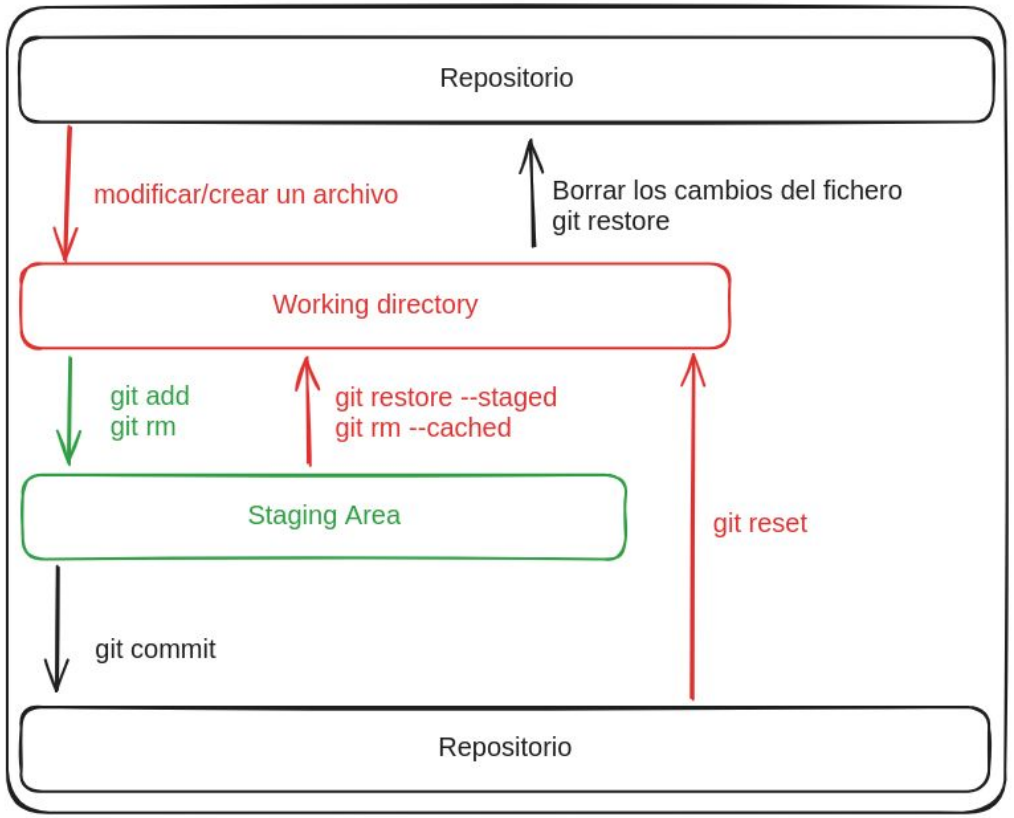
1. Edita un fichero y descarta los cambios que acabas de hacer usando git.
2. Edita y añade un fichero al área de staging y saca el contenido del área de staging.
3. Descarta los cambios de un commit.
4. Descarta los últimos 2 commits.
5. Descarta los cambios de un commit y borra su contenido.

6. Tu primera rama

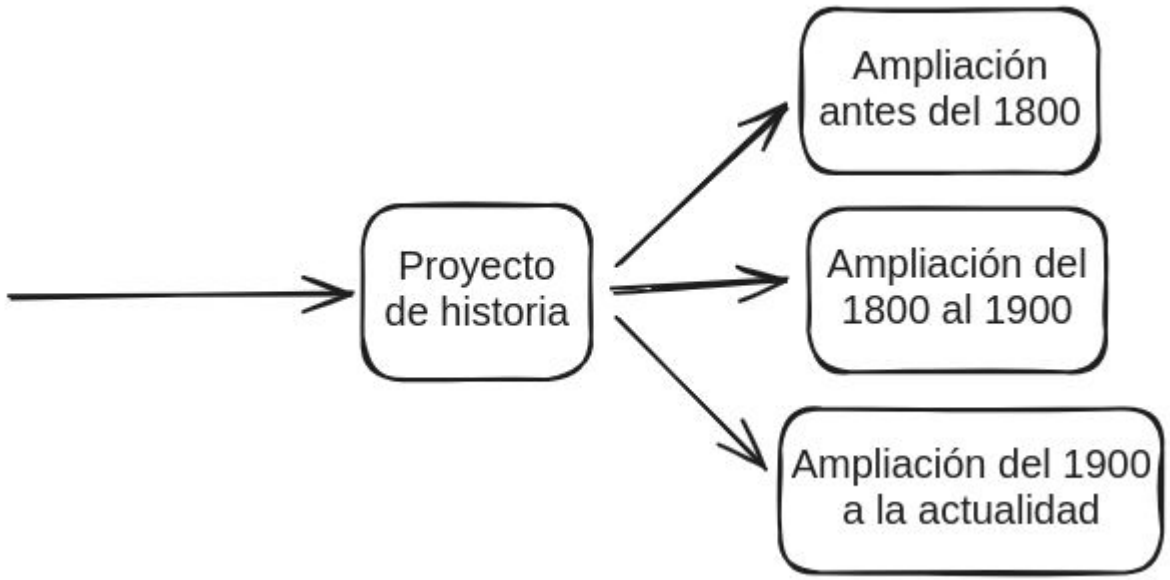
Rama: Master



Rama: Develop



6. Tu primera rama



6. Tu primera rama

Comandos

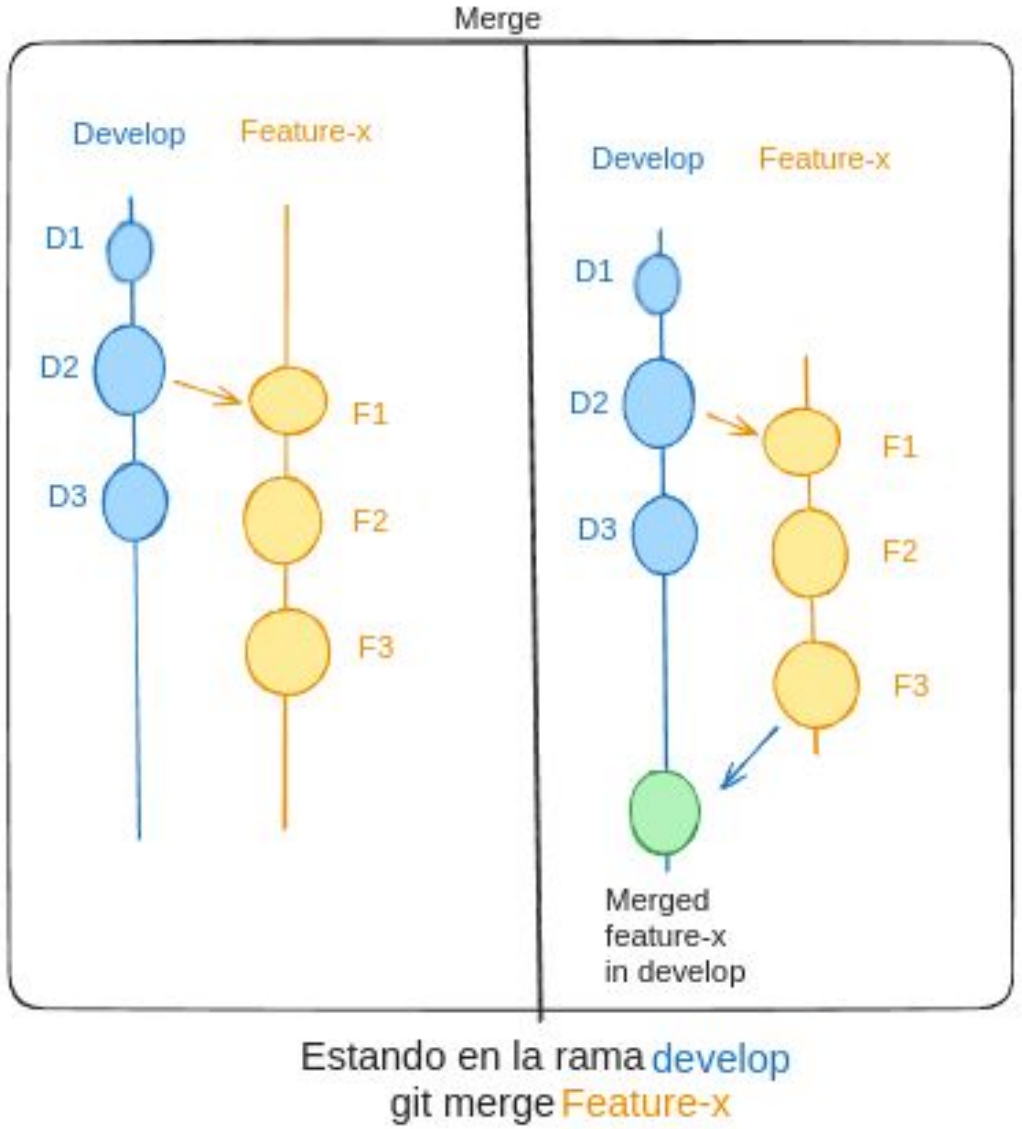
1. `git switch [-c] <branch>` **->** Crea y muévete entre ramas.
2. `git checkout [-b] <branch/commit>` **->** Crea y muévete entre ramas y commits.
3. `git branch [-d/-D] [<branch>]` **->** Crea, lista y maneja ramas.
4. `git log [--all/--graph]` **->** Obtén información de todos los commits de todas las ramas.

6. Tu primera rama

1. Crea una rama y muévete a ella en **un** comando.
2. Crea una rama y muévete a ella en **dos** comandos.
3. En un repositorio de tres commits, muévete al primero.
4. Borra una rama.
5. Cambiale el nombre a una rama.
6. Haz 1 commit en la rama **uno**, crea la rama **dos** y haz 2 commits en ella, muévete de nuevo a la rama **uno** y crea la rama **tres**, haciendo un commit mas, ¿Qué ha pasado? ¿Por qué la rama **tres** tiene los commits de la rama **uno** y no los de la **dos**?

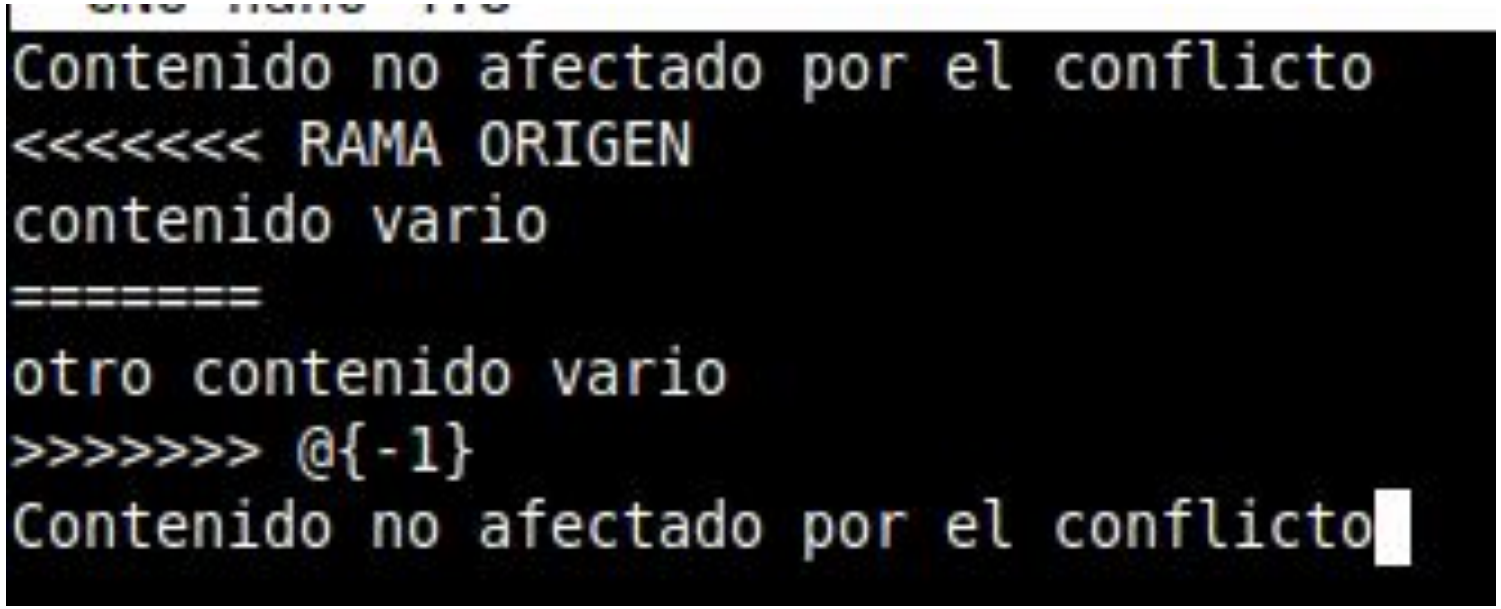
7. Unir ramas entre sí

Une los desarrollos



7. Unir ramas entre sí

Conflictos



7. Unir ramas entre sí

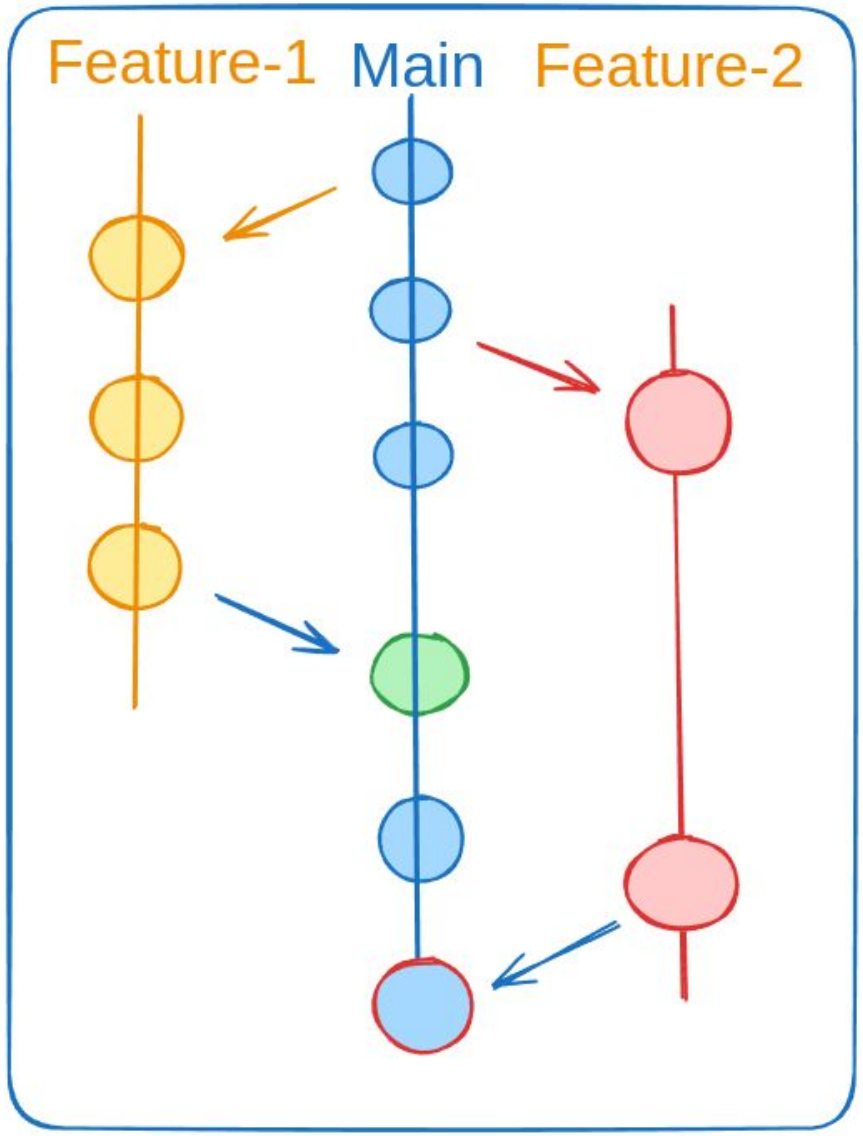
Comandos

1. `git merge [--abort] <branch>` -> Une dos ramas mezclando su contenido.

7. Unir ramas entre sí

Ejercicios

Consigue la siguiente estructura



8. No estás solo en la empresa

GitHub!

github
SOCIAL CODING

8. No estás solo en la empresa

Para configurar Github

```
$ ssh-keygen -t ed25519 -C "your_email@example.com"
```

(lo dejáis todo por defecto)

```
$ eval "$(ssh-agent -s)"
```

```
$ ssh-add ~/.ssh/id_ed25519
```

```
$ cat ~/.ssh/id_ed25519.pub
```

Si quieres investigar más sobre dónde salen estos comandos y que hacen

https://docs.github.com/en/authentication/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent

https://docs.github.com/en/authentication/connecting-to-github-with-ssh/adding-a-new-ssh-key-to-your-github-account?platform=linux

8. No estás solo en la empresa

Comandos

1. `git clone <repositorio> [<carpeta>]` **->** Guarda un repositorio remoto en tu máquina local.
2. `git push [<branch>]` **->** Sube los cambios locales de tu rama al remoto.
3. `git pull [<branch>]` **->** Descarga los cambios remotos a tu rama local.
4. `git remote add <alias> <url>` -> Añade un nuevo repositorio remoto.
5. `git remote remove <alias>` -> Elimina un repositorio Remoto.

8. No estás solo en la empresa

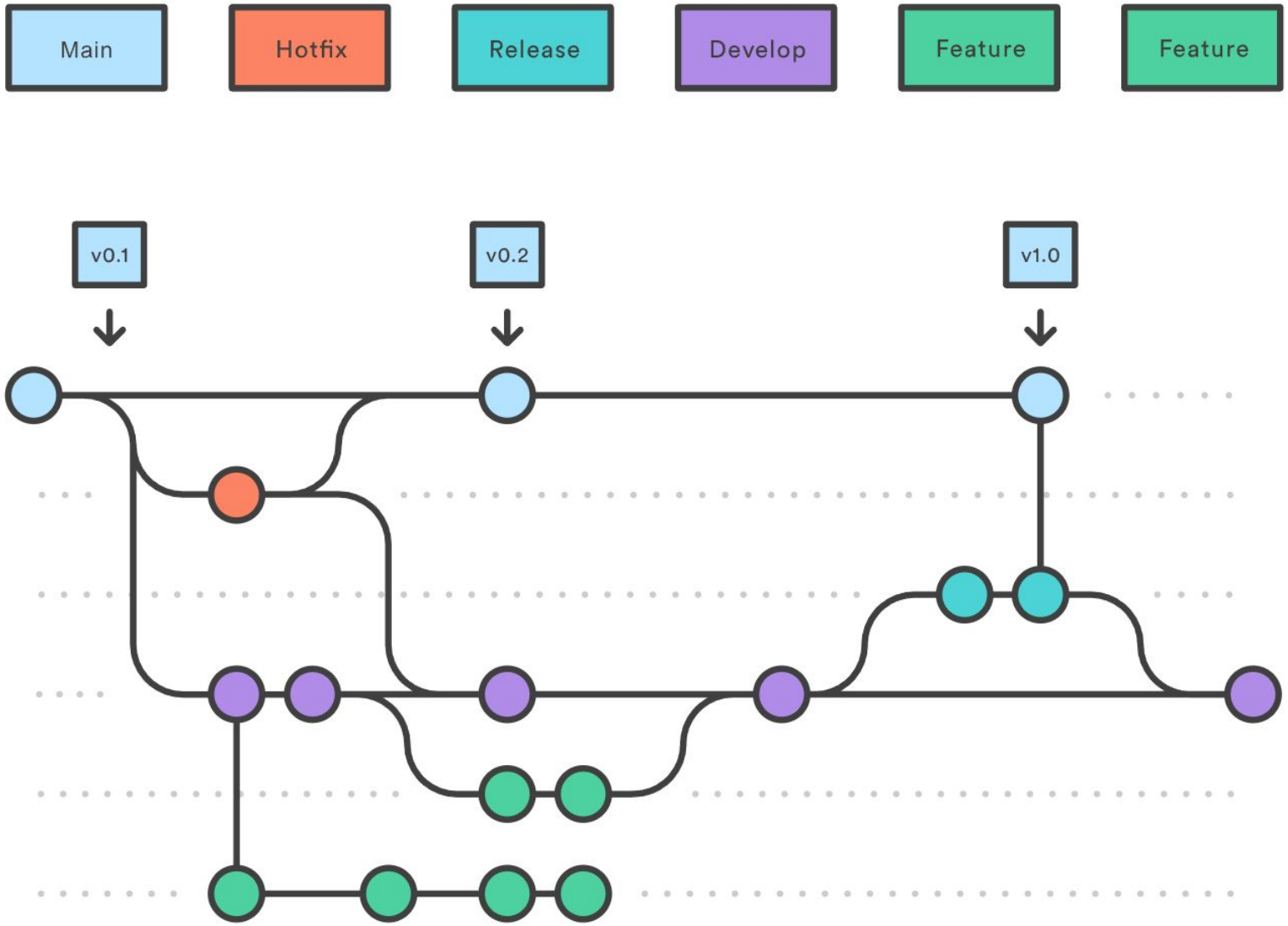
Ejercicios

1. Crea un repositorio en github llamado 8.1.repositorio y clónalo en tu máquina local con el nombre 8.1.repositorio-renombrado.
2. Crea un repositorio, clónalo en dos carpetas distintas (8.1.repositorio.1 y 8.1.repositorio.2 respectivamente), entra en la carpeta 8.1.repositorio.1, crea y edita el fichero primerFichero, tras esto, sube este fichero a github.

   Una vez realizado edita el mismo fichero en la carpeta 8.1.repositorio.2, haz un commit he intenta subirlo, ¿Qué ha pasado? ¿Cómo puedes solucionarlo?

9. Gitflow

Mantén un orden entre tus compañeros de trabajo



9. Gitflow

Teoría: Nombre de las ramas

1. Main/Master -> La rama que contiene el código de producción.
2. Develop -> La rama con los últimos cambios de los desarrolladores.
3. Feature-X -> Ramas con las nuevas funcionalidades que se están desarrollando.
4. Fix -> Ramas que arreglan código de develop que no funciona bien.
5. Hotfix -> Ramas que arreglan código imprescindible que no funciona bien.
6. Refactor -> Ramas que cambian el código sin cambiar la funcionalidad.
7. Release -> Rama que contiene código justo antes de ser subido a producción.

9. Gitflow

Ejercicios

1. Usa todos los conocimientos adquiridos en este curso para crear un historial de git como el de la foto de hace dos pestañas atrás

10. Trabaja en equipo

Sé el maestro de Github

10. Trabaja en equipo

Teoría: Pestañas de GitHub

1. Code -> Donde se encuentra todo el código subido, así como sus ramas, sus commits, descripción del proyecto, es la página principal de repositorio.
2. Issues -> Donde se encuentra todos los cambios propuestos y errores del código.
3. Pull requests (pr) -> Peticiones para unir dos ramas.
4. Actions -> Acciones que se ejecutan cuando pasa algo (se sube código, se mergea develop en main).
5. Projects -> Tabla con todos los cambios propuestos al proyecto.
6. Wiki -> Documentación del proyecto.

10. Trabaja en equipo

Ejercicio

1. En grupos, que uno de vosotros cree el repositorio, haga al resto del grupo y volver a hacer el gráfico del gitflow, pero esta vez repartirse las ramas, realizar issues y pull request (pr) por cada feature y hotfix

Contenido Extra

Contenido que se ha quedado fuera del curso por motivos de tiempo.

Aún así es muy recomendable que le echéis un vistazo tanto para vuestra estancia en el curso.

Como para vuestro futuro como desarrolladores.

E1. Muévete con eficacia

Atajos de teclado de VS Code

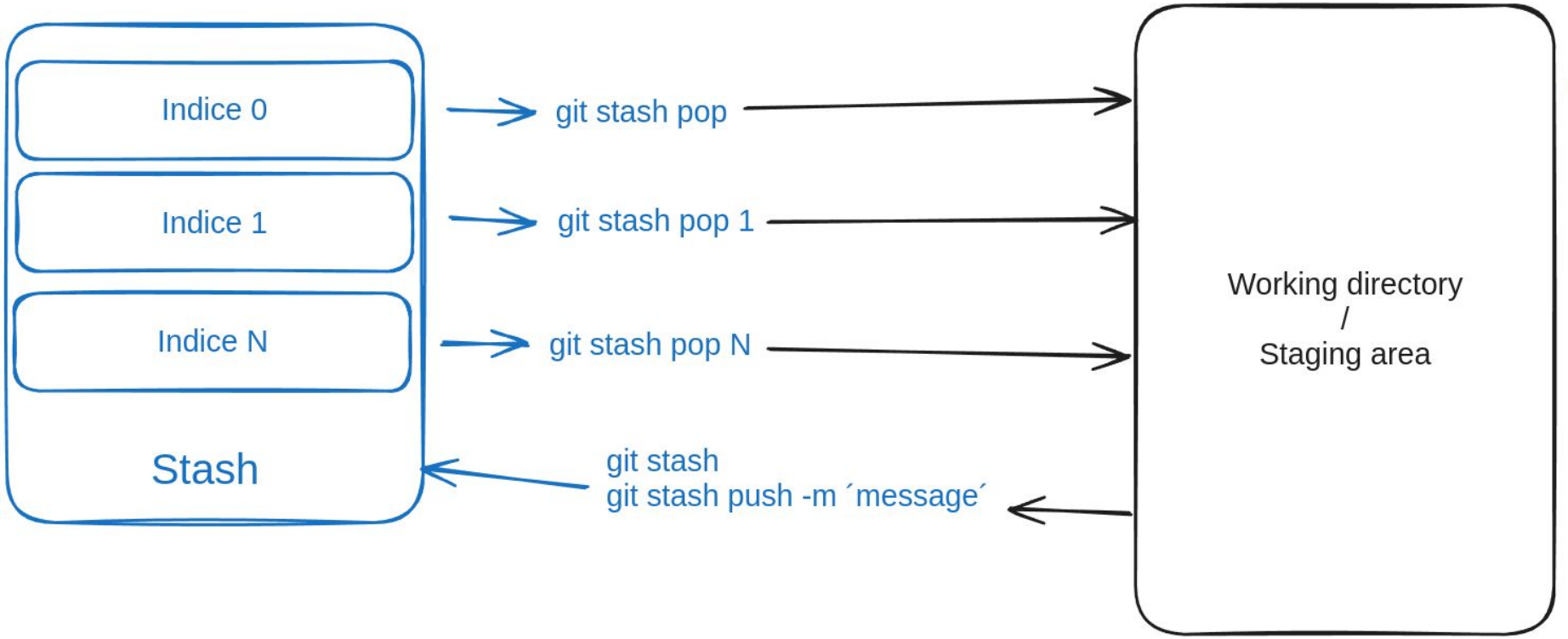
E1. Muévete con eficacia

Listado:

1. ctrl + , -> Abrir opciones.
2. ctrl + p -> Abrir fichero.
3. ctrl + shift + p -> Abrir lanzador de opciones.
4. alt + up/down -> mover una línea completa/ mover toda la selección.
5. ctrl + X -> cortar una línea completa.
6. ctrl + D -> multicursor buscando por el contenido seleccionado.
7. ctrl + shift + L -> multicursor con todas las referencias del elemento seleccionado.
8. ctrl + shift + F -> buscar en todo el proyecto el elemento seleccionado.
9. ctrl + B -> Alternar panel izquierdo.
10. ctrl + K & ctrl + C -> Comentar el código

E2. Guardar el código temporalmente

Ten varios desarrollos en la misma máquina



E2. Guardar el código temporalmente

Comandos

1. `git stash` -> Guardar temporalmente en el stack (LIFO).
2. `git stash list` -> Muestra todos los stash guardados.
3. `git stash push -m <mensaje>` -> Guarda temporalmente en el stack pero con un nombre.
4. `git stash pop [<index>]` -> Devuelve el código a tu área de trabajo.
5. `git stash drop [<index>]` -> Elimina un stash guardado.
6. `git stash show [<index>]` -> Muestra los cambios que contiene un stash.
7. `git stash clear` -> Borra todos los stashs.

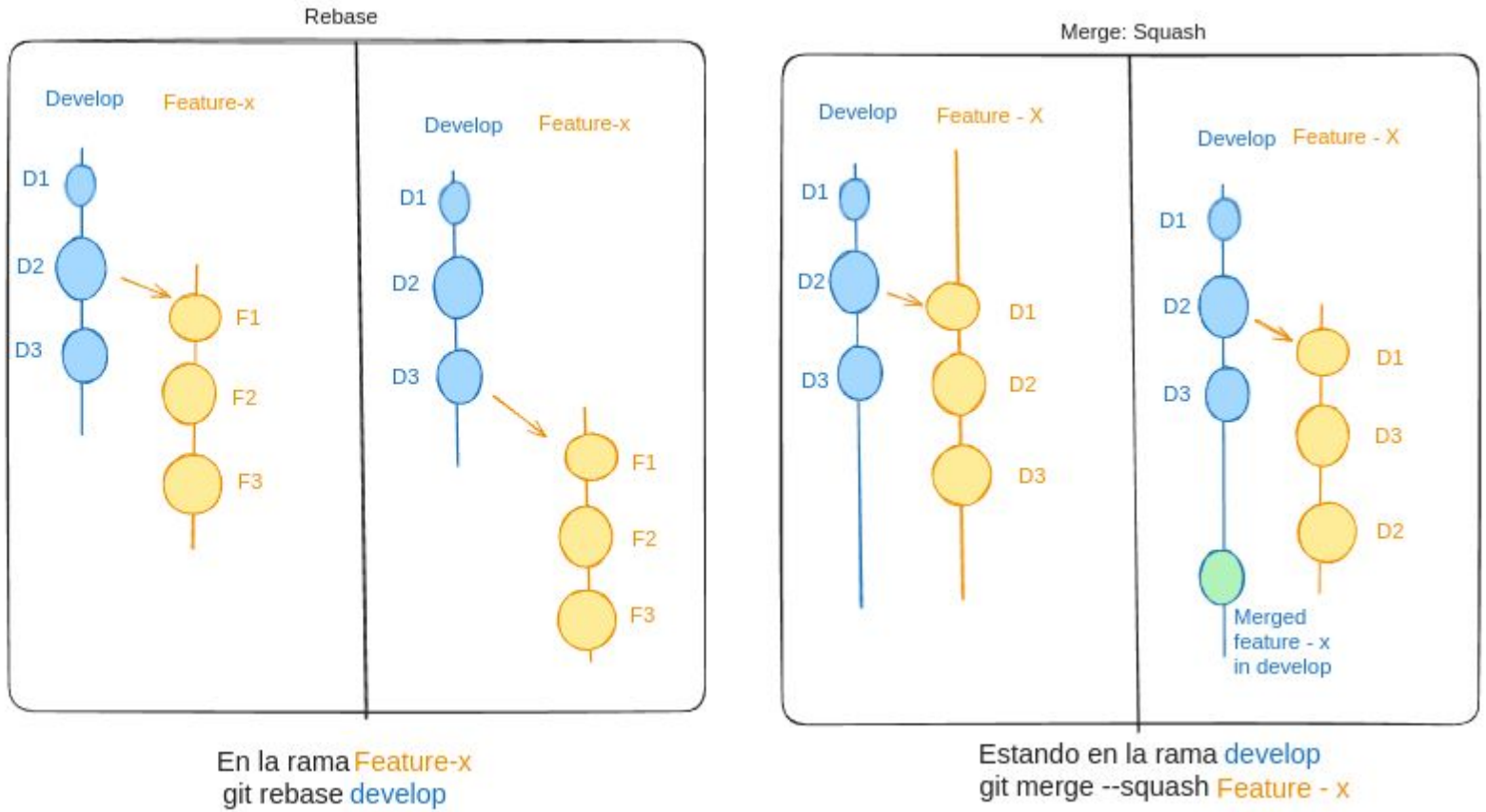
E2. Guardar el código temporalmente

Ejercicios

1. Crea tres ficheros, editalos y stashealos.
2. Crea 3 stashs y saca el segundo de ellos.
3. Ahora borra el de índice 1.
4. Borra todos los stashs con un solo comando.
5. Edita dos ficheros, uno déjalo en el área de staging y el otro en el working directory y solo guarda dentro del stash el que está en el área de staging.
6. ¿Cómo harías para ver el contenido de un stash?
7. Haz un stash en una rama y sacalo en otra rama.

E3. Unir ramas entre sí 2.0

Otra manera de unir ramas con resultados distintos para situaciones especiales.



E3. Unir ramas entre sí 2.0

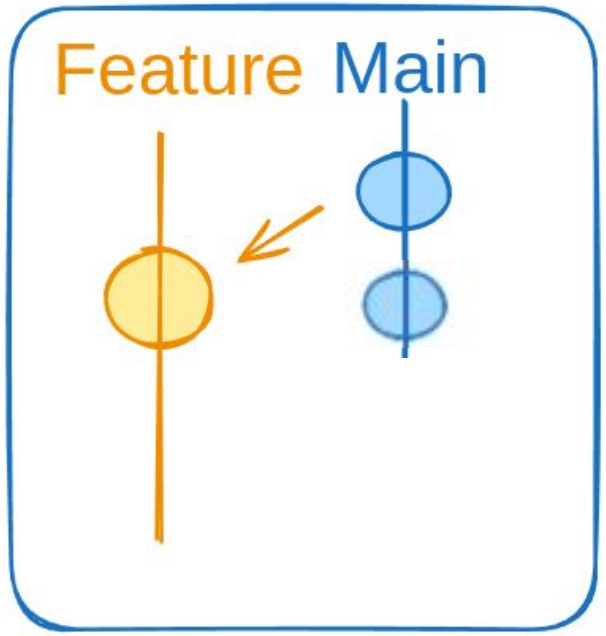
Comandos

1. `git rebase <branch>` -> Mueve el nacimiento de la rama origen al último commit de la rebasada.
2. `git merge --squash [branch]` -> Une dos ramas mezclando su contenido como un merge, pero no se apunta como que la rama objetivo muera en la rama origen.

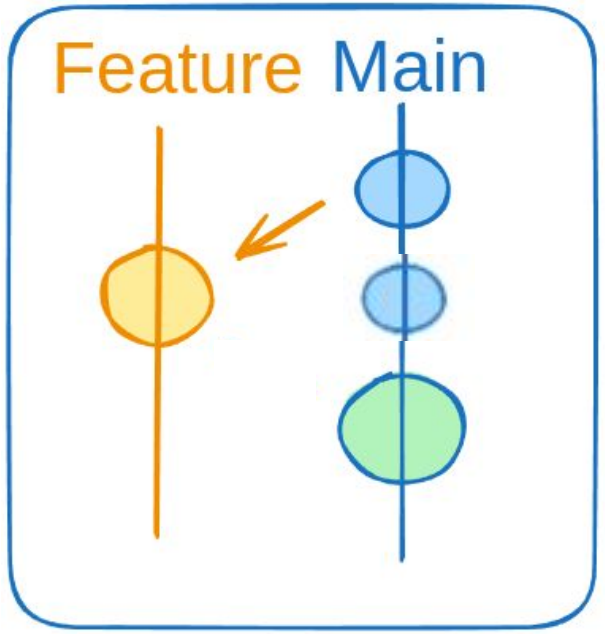
E3. Unir ramas entre sí 2.0

Ejercicios

Punto de partida



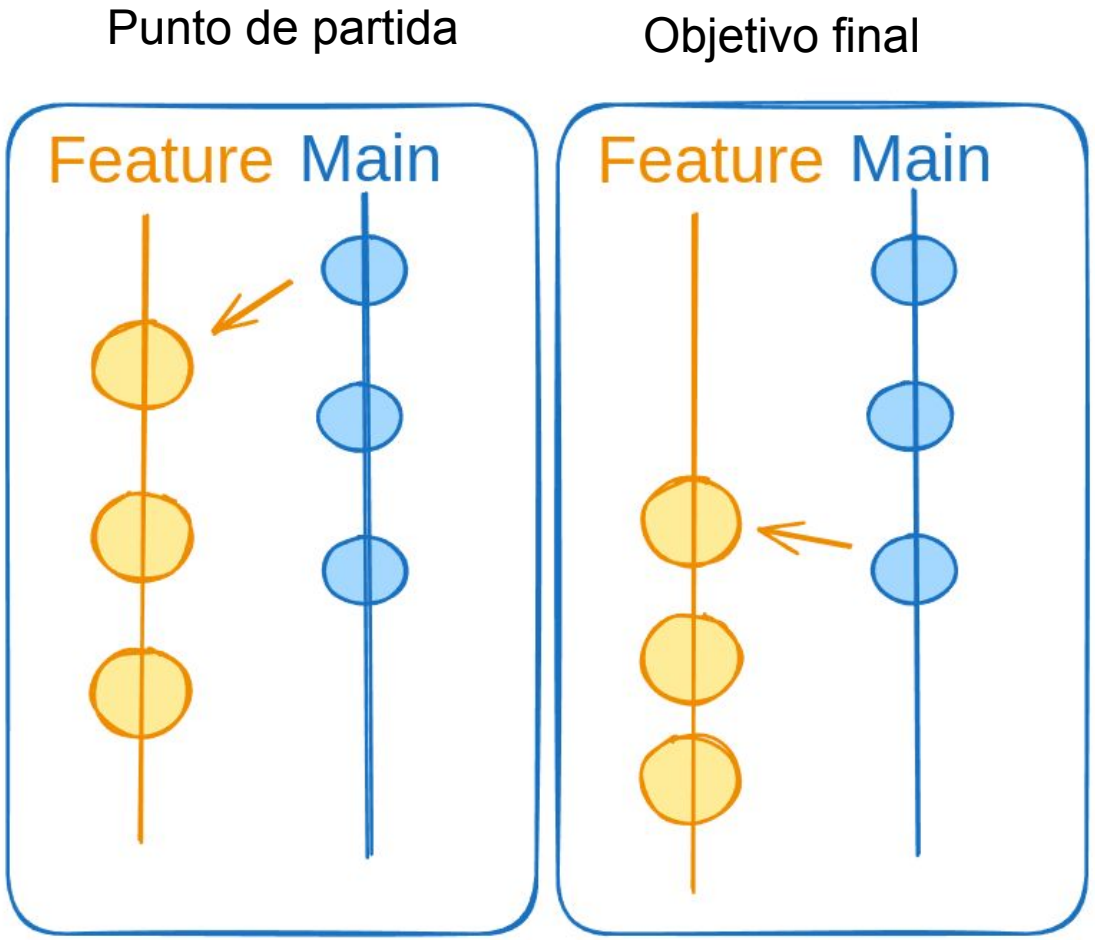
Objetivo final



E3. Unir ramas entre sí 2.0

Ejercicios

En main edita el fichero main.html y en feature el fichero feature.html



E4. V.0.0.1 - Etiquetas

Comandos

1. `git tag` -> Lista todos los tags en tu repositorio.
2. `git tag <name>` -> Crea una nueva tag con el nombre name.
3. `git tag -d <name>` -> Borra un tag.

DESCANSO

Basilio David Gómez Fernández