**¿Que es git?**

*Es un sistema de control de versiones especializado para proyectos de software.*

*Gratis*

*Open source => significa que podemos contribuir a su código fuente.*

*Fue creado el 2005 por linus Torvalds, creador del sistema Linux.*

**¿Por qué deberíamos usar git?**

*Vamos a poder tener todos los beneficios a la hora de trabajar, como:*

* *Copia de seguridad*
* *Facilitar el trabajo en equipo*
* *Recuperar versiones antiguas de nuestro proyecto*
* *Poder trabajar offline*

*Características especiales.*

*Podemos crear ramas para manejar entre ellas, unirlas borrarlas fácil y rápidamente.*

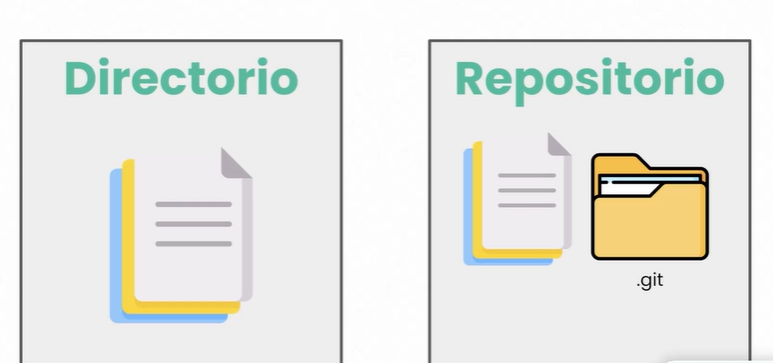
***Git permite trabajar offline***

***Git agrega cambios constantes***

***…………………………………………………………………………………***

***Repositorio***

*Un repositorio es un espacio de memoria en el que vive mi proyecto y su historial de versiones. En palabras mas fáciles, una carpeta que tenemos en la pc donde podemos guardar nuestro proyecto con sus versiones.*

**

*La diferencia es que tenemos una carpeta oculta llamada .git donde se almacena todos los cambios que tenemos de nuestro proyecto, podríamos verlo como una base de datos de historial de versiones.*

*La carpeta se crea automáticamente al inicializar nuestro repositorio.*

*Para subir nuestros cambios al repositorio remoto es ->* ***git push***

*Para extraer y descargar contenido desde un repositorio remoto es ->* ***git pull***

***Áreas de git***

*Área de trabajo -> código que estamos modificando*

*Área de staging -> cambios están listos para ser parte del repositorio*

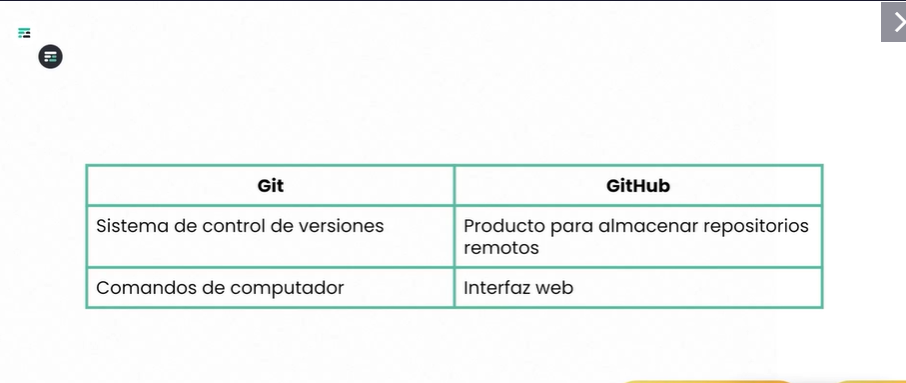
*Área de repositorio -> cambios que son parte del historial de versiones => nos almacena cambios de manera de fotos que podemos llamarlos* ***commit => es***

**

***¿Que es github?***

*Herramienta que nos permite almacenar nuestro repositorios remotos y gestionar nuestro proyecto de principio a fin.*

***Diferencias entre git y github***

**

***Comandos esenciales***

*Git init => genera un directorio llamado .git que incluye la configuración e historial*

*Git add => nos hace movernos de* ***área de trabajo***  *al* ***área de staging***

**

*A screenshot of a chat

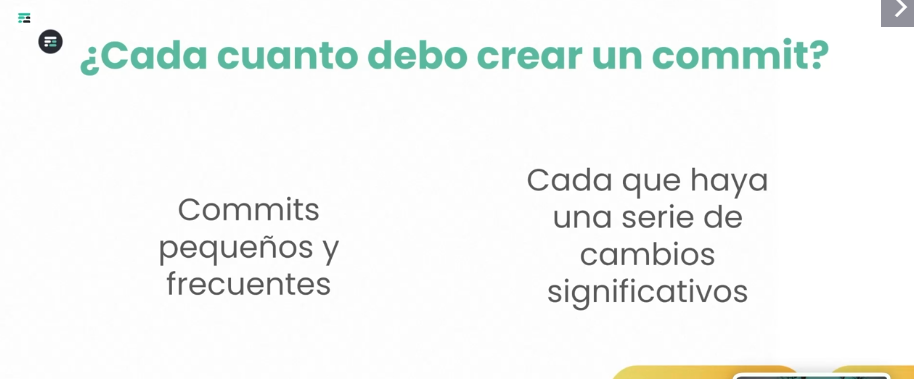
Description automatically generated*

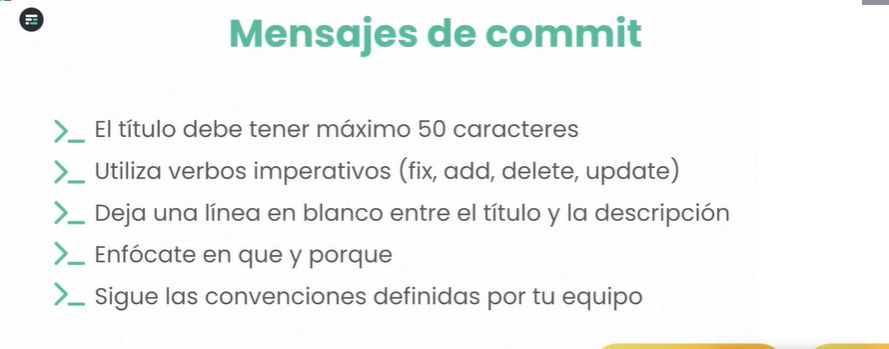
**

**

*Git commit => nos movemos del área de al* ***área de staging*** *a* ***área de repositorio***

***Buenas practicas***

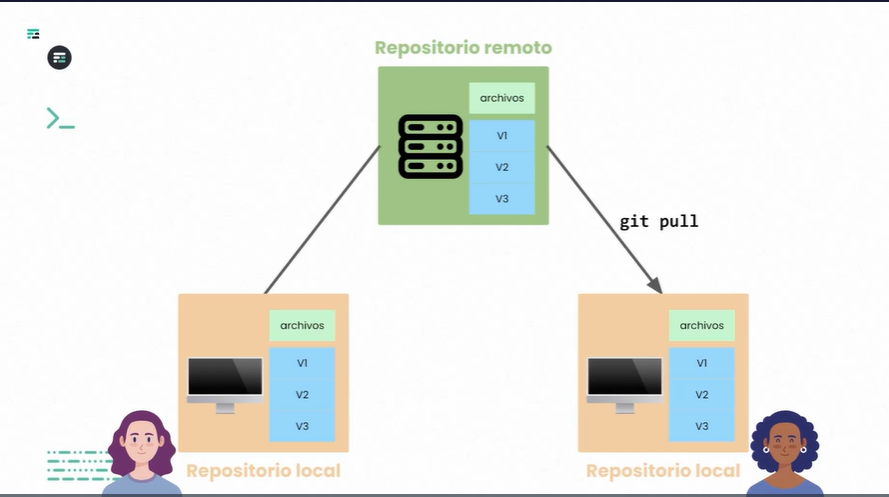
**

**

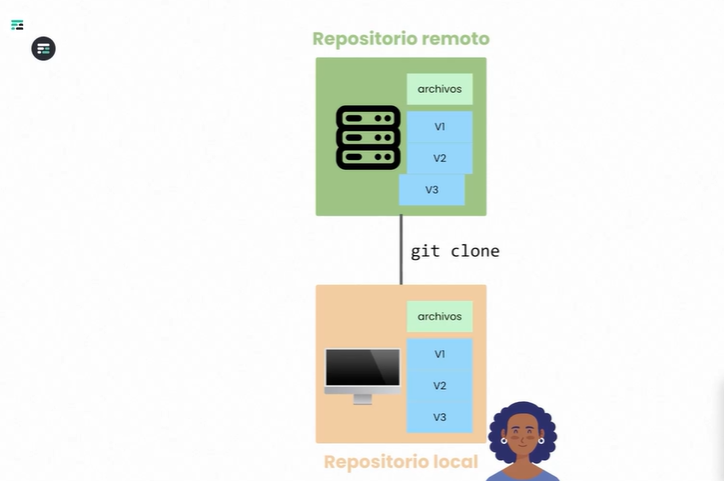
*Git pull and git push =>*

*Git push --> nos va servir para enviar los cambios de nuestro* ***repositorio local*** *al* ***repositorio remoto***

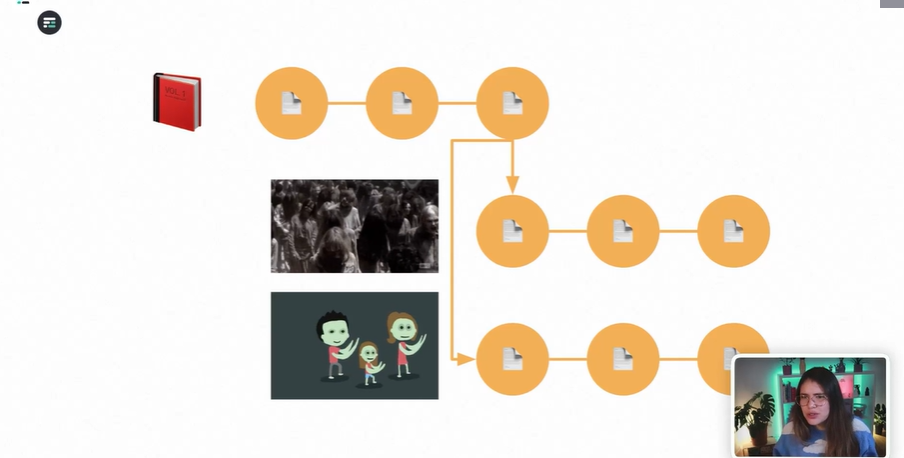
*Git pull --> nos va a servir para enviar los cambios de* ***nuestro repositorio remoto*** *al* ***repositorio local***

******

*Git clone => nos descarga una copia completa de nuestro repositorio remoto*

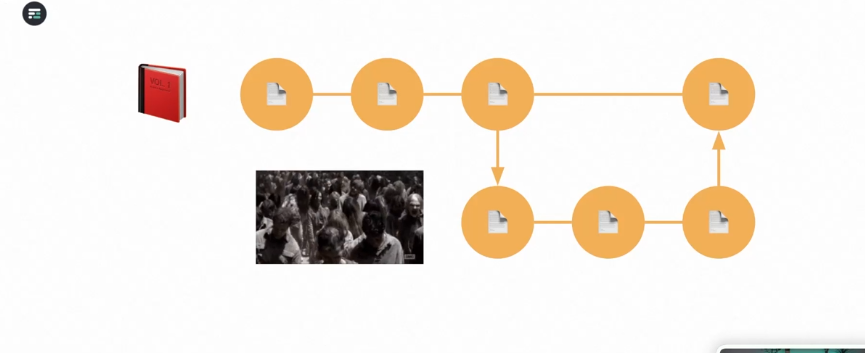
**

***¿Que son las ramas?***

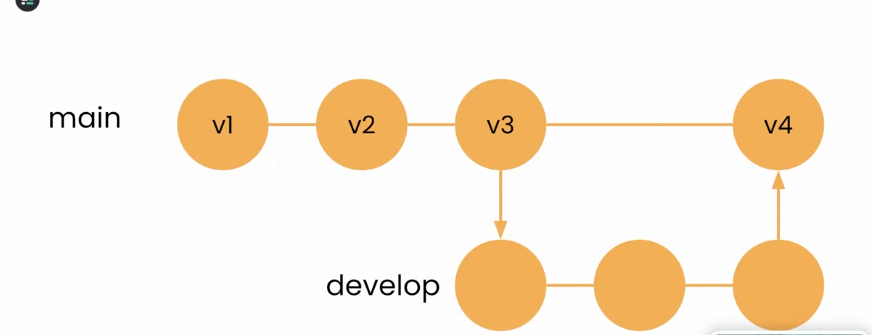
**

*Las ramas son como escribir un libro, en este caso de zombies. Donde tenemos una historia donde los zombies se comen a los humanos y otra línea ( rama ) donde los zombies bailan.*

*Ahora digamos que nos gusto la línea ( rama ) donde los zombies se comen a los humanos*

**

*Entonces unimos esa versión, es línea ( rama ) a nuestra historia principal.*

**

*A computer network with a telephone and laptops

Description automatically generated with medium confidence*

*A white and orange text

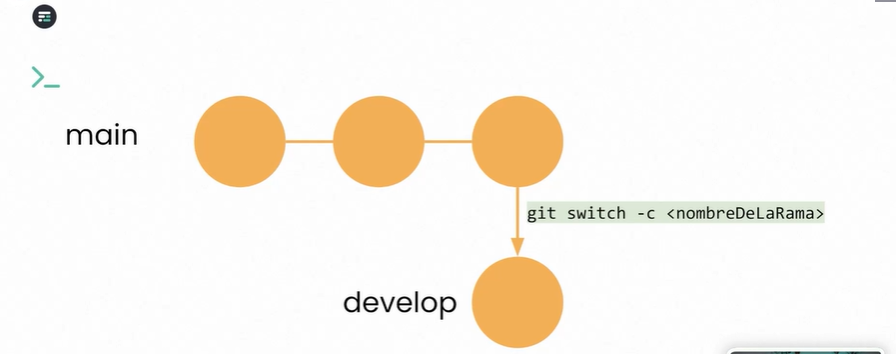
Description automatically generated with medium confidence*

***Crear ramas ( switch )***

*Git switch -c <<NombreDeLaRama>>*

*Dato extra: para mirar las ramas que tenemos podemos hacerlo con el comando*

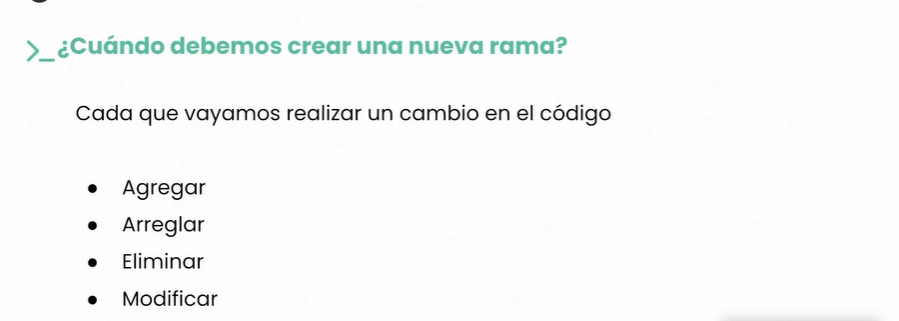
***Git branch***

**

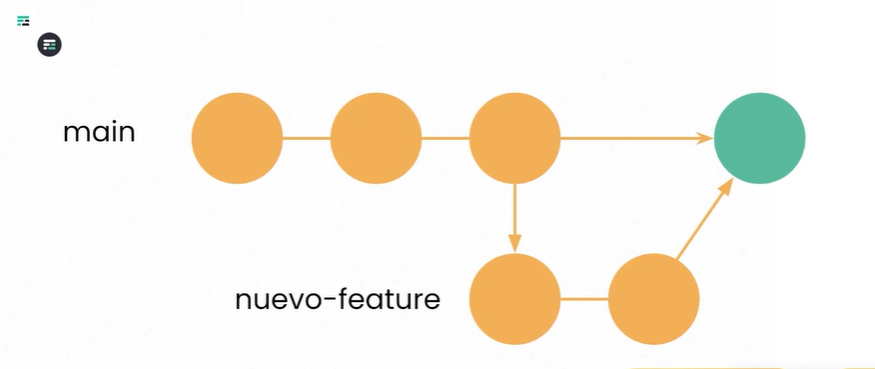
******

*A screenshot of a computer program

Description automatically generated*



***Unir ramas ( merge )***

**

*Git merge*

1. **git merge**: Este comando se usa para combinar los cambios de una rama (branch) en otra rama. Por ejemplo, si tienes una rama llamada "feature" donde has estado trabajando en una nueva funcionalidad y quieres incorporar esos cambios en la rama principal (por lo general, "master" o "main"), usarías **git merge** para fusionar los cambios de "feature" en la rama principal.

El proceso de fusión (**git merge**) crea un nuevo commit que representa la combinación de ambas ramas. Git automáticamente resuelve los conflictos si es posible, pero en caso de conflictos, te pedirá que los resuelvas manualmente antes de finalizar la fusión.

**Comandos útiles**

1. **git status**: Este comando te muestra el estado actual de tu repositorio Git. Te da información sobre qué archivos han sido modificados, qué archivos están preparados para ser confirmados (staged), y qué archivos aún no están bajo seguimiento de Git. Es una herramienta muy útil para entender el estado de tu trabajo antes de realizar una confirmación (commit).

*2.***git log**: Al ejecutar **git log**, Git te mostrará un registro de las confirmaciones (commits) que se han realizado en tu repositorio. Este registro incluye información como el autor de cada confirmación, la fecha y hora en que se realizó, y el mensaje asociado a la confirmación. Puedes utilizar opciones adicionales con **git log** para personalizar la salida, como limitar el número de confirmaciones mostradas o ver cambios específicos en un archivo.

1. **git diff**: El comando **git diff** se utiliza para mostrar las diferencias entre diferentes estados de tu proyecto. Puedes usarlo para ver las diferencias entre el directorio de trabajo actual y el área de preparación (staging area) utilizando **git diff**. También puedes usar **git diff** seguido de un hash de confirmación para ver las diferencias entre el estado actual y un estado específico en el pasado.

* **git status** te da una visión general del estado de tu trabajo.
* **git log** te muestra un registro de las confirmaciones realizadas en tu repositorio.
* **git diff** te permite ver diferencias entre diferentes estados de tu proyecto, lo que es útil para

***deshacer cambios:***

*git checkout =>para deshacer cambio, nos sirve para navegar en la historia de git entre ramas, commit o entre archivos.*