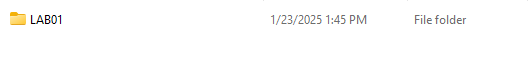
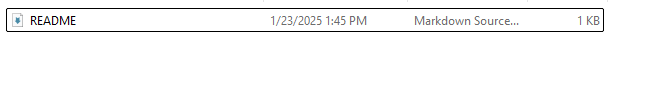
**BITACORA LABORATORIO 1**

## **PARTE I**

1. Crea un repositorio localmente.



1. Agrega un archivo de ejemplo al repositorio, el **README.md** puede ser una gran opción.

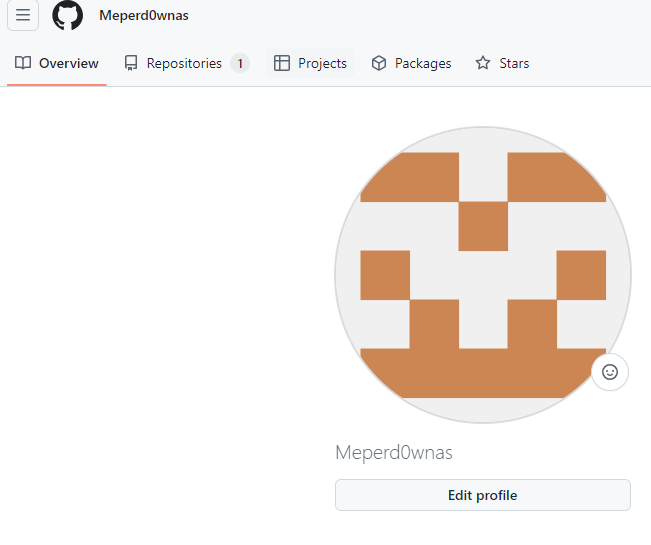


1. Averigua para qué sirve y como se usan estos comandos **git add** y **git commit -m “mensaje”**

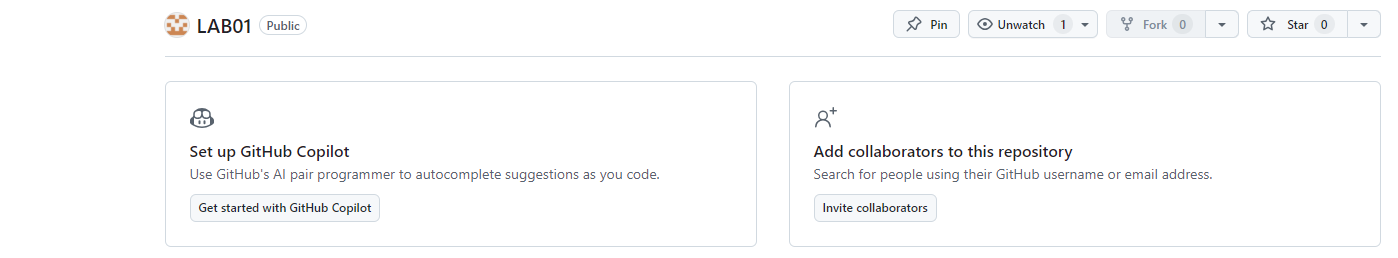
**guit add**: Mueve los cambios del directorio de trabajo al área del entorno de ensayo.

**git commit -m “mensaje”:** se utiliza en Git para guardar los cambios que has hecho en tu repositorio local.

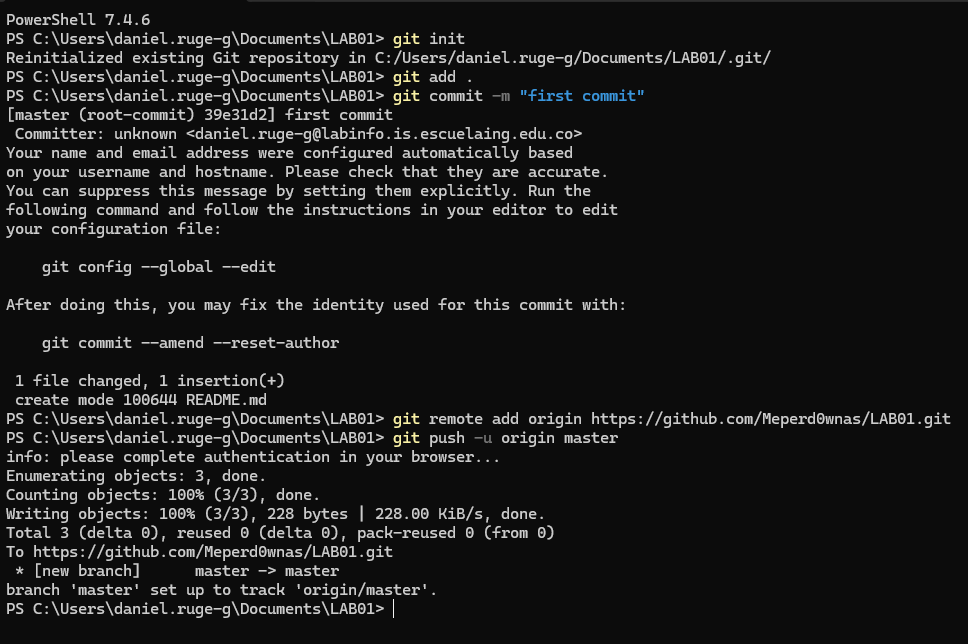
1. Abre una cuenta de github, si ya la tienes, enlazala con el correo institucional.



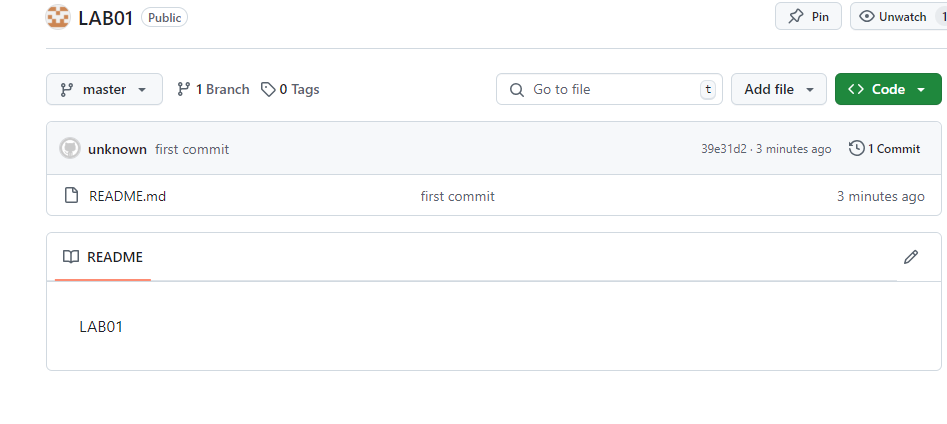
1. Crea un repositorio en blanco (vacío) e GitHub.



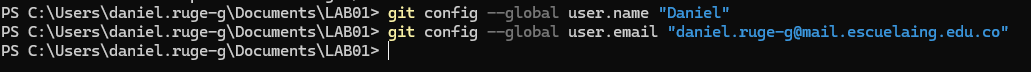
1. Configura el repositorio local con el repositorio remoto.



* 1. Sube los cambios, teniendo en cuenta lo que averiguaste en el punto 3 Utiliza los siguientes comandos en el directorio donde tienes tu proyecto, en este orden:



1. Configura el correo en git local de manera correcta [Configurar correo electrónico en GitHub](https://docs.github.com/es/account-and-profile/setting-up-and-managing-your-personal-account-on-github/managing-email-preferences/setting-your-commit-email-address)



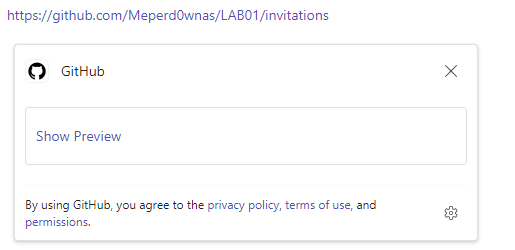
## **PARTE II**

1. Se escogen los roles para trabajar en equipo, una persona debe escoger ser "Owner" o Propietario del repositorio y la otra "Collaborator" o Colaborador en el repositorio.

Daniel Ricardo Ruge Gomez - Owner

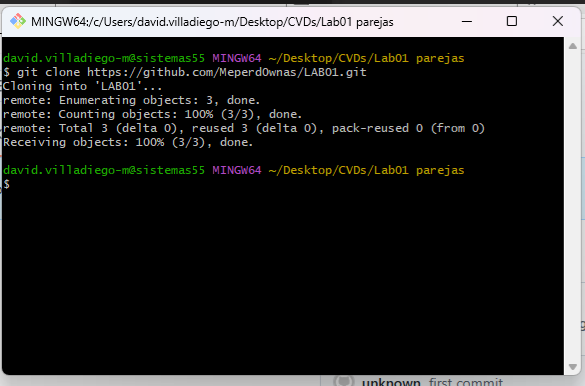
David Santiago Villadiego Médicis – Collaborator

1. El owner agrega al colaborador con permisos de escritura en el repositorio que creó en la parte 1
2. El owner le comparte la url via Teams al colaborador

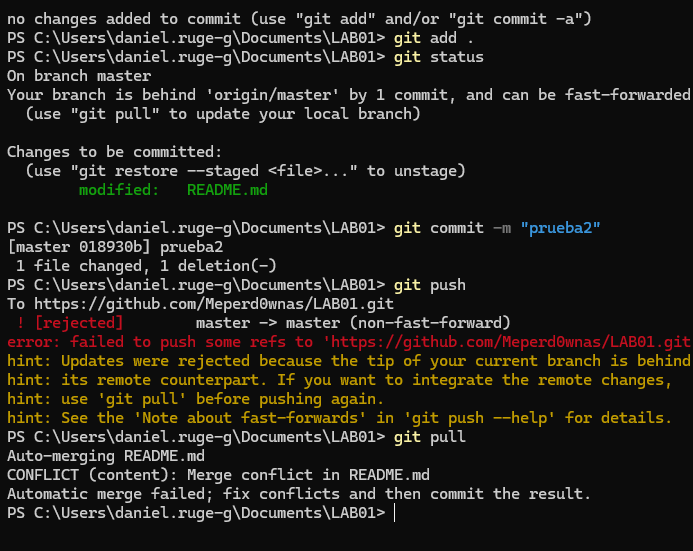


1. El colaborador acepta la invitación al repositorio

Desde la pantalla de David:



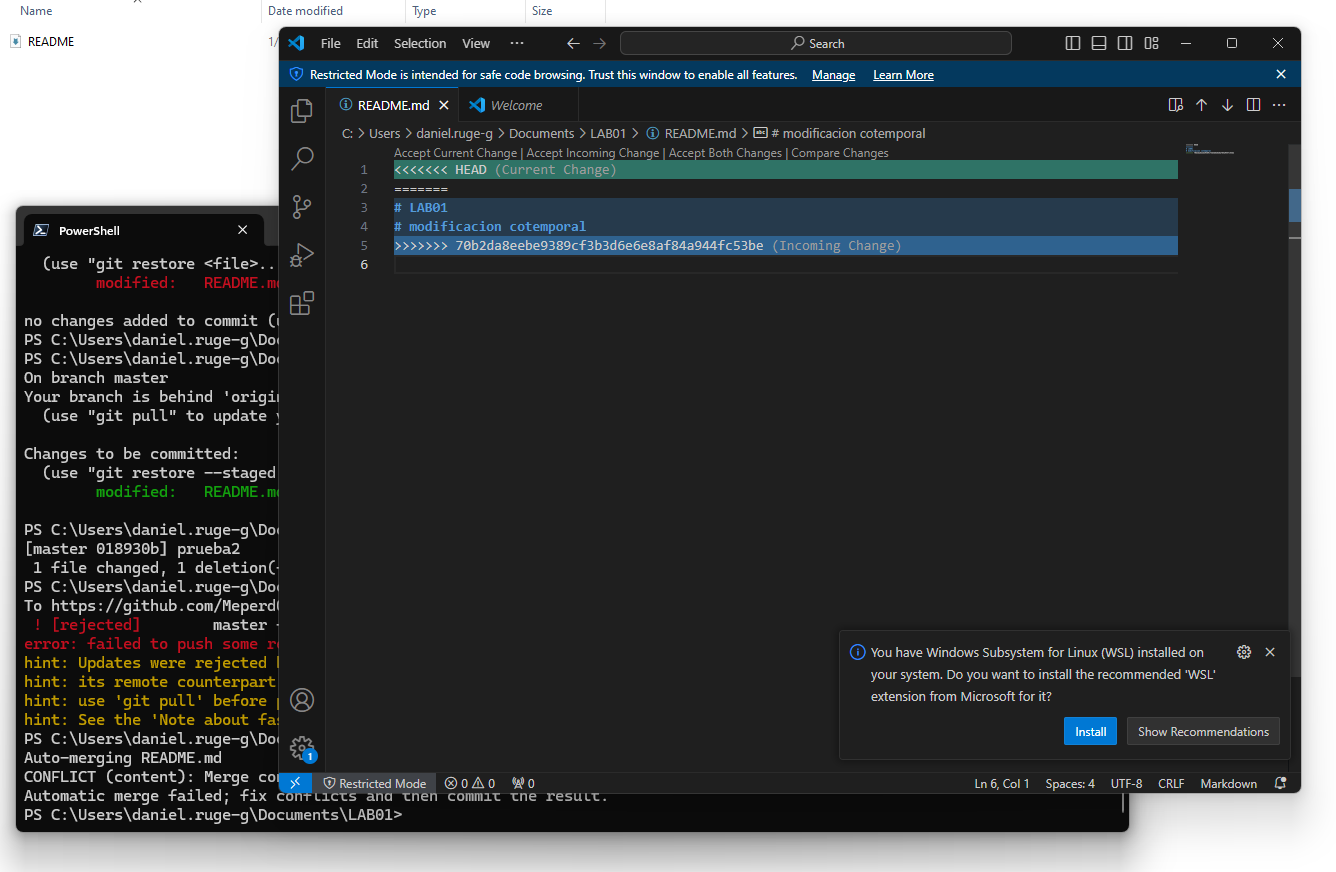
1. Owner y Colaborador editan el archivo README.md al mismo tiempo e intentan subir los cambios al mismo tiempo.

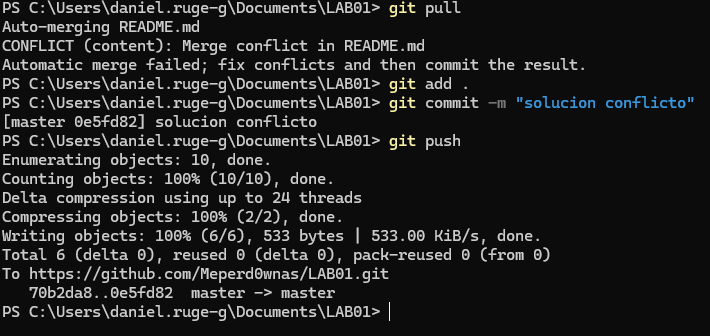


1. ¿Que sucedió?

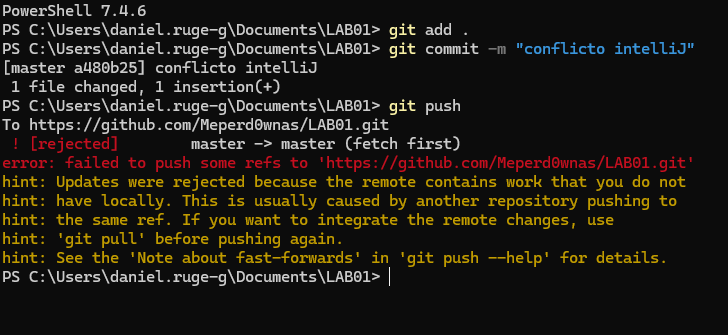
Al intentar subir los cambios al tiempo la consola nos tiro un mensaje de error

1. La persona que perdió la competencia de subir los cambios, tiene que resolver los conflictos, cúando haces pull de los cambios, los archivos tienen los símbolos <<< === y >>> (son normales en la resolución de conflictos), estos conflictos debes resolverlos manualmente. [Como resolver Conflictos GitHub](https://docs.github.com/es/enterprise-cloud@latest/pull-requests/collaborating-with-pull-requests/addressing-merge-conflicts/resolving-a-merge-conflict-on-github)





1. Volver a repetir un cambio sobre el README.md ambas personas al tiempo para volver a tener conflictos.



## **PARTE III**

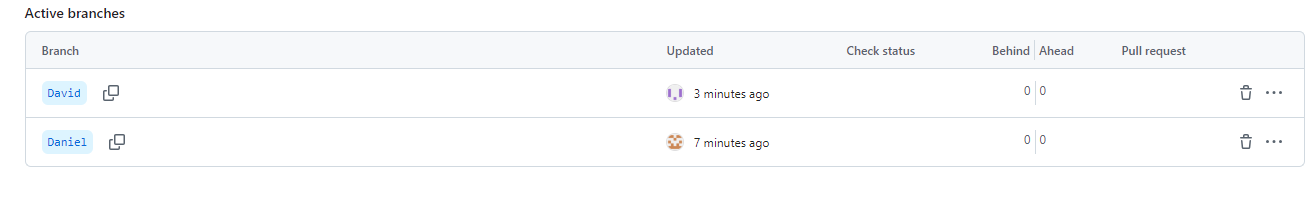
1. ¿Hay una mejor forma de trabajar con git para no tener conflictos?

Para evitar conflictos en Git, se debe trabajar siempre en ramas distintas a main y sincronizar frecuentemente con los cambios más recientes usando pull o merge. Se deben hacer commits pequeños y claros tras cada cambio importante y revisar el trabajo antes de subirlo. Se deben utilizar pull requests para integrar cambios revisados en el código principal, y se deben resolver conflictos de inmediato si ocurren. Finalmente, se debe mantener una buena comunicación con el equipo y definir un flujo de trabajo claro para todos.

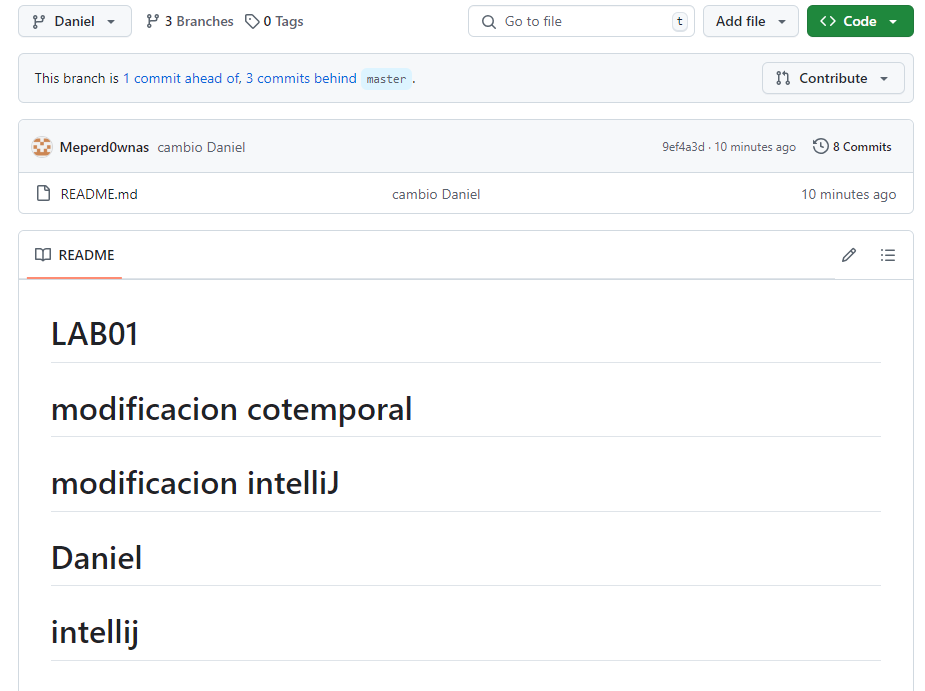
1. ¿Qué es y cómo funciona el **Pull Request**?

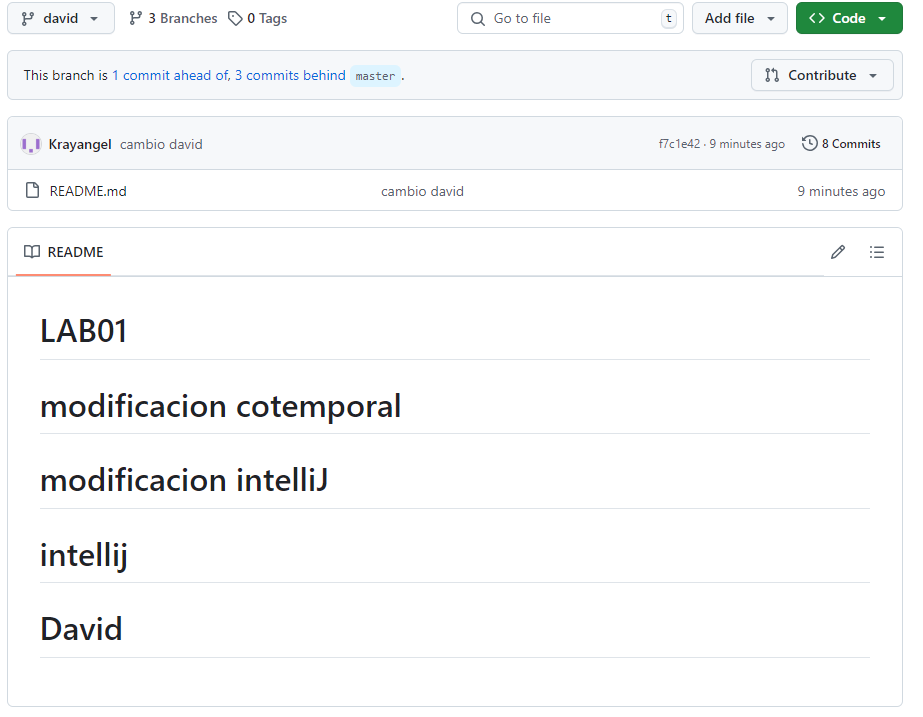
El Pull Request es básicamente una forma de pedir que los cambios que hiciste en tu rama se fusionen con otra, casi siempre con la principal. Es útil porque otros pueden revisar tu código, dejar comentarios o sugerencias antes de que se mezcle todo. Así se evita que algo roto termine en la versión principal.

1. Creen una rama cada uno y suban sus cambios



1. Tanto owner como colaborador hacen un cambio en el README.md y hacen un Pull Request (PR) a la rama main/master





1. Teniendo en cuenta la recomendación, mezclen los cambios a la rama main a través de PR con el check/review/approval del otro compañero (Cuando se hace merge se deberían borrar las ramas en github)

