Sistema de control de

Versiones: Git y GitHub

INTEGRANTES

01 Francis Muñoz Zampieri

02 Rodrigo Zambrana Martínez

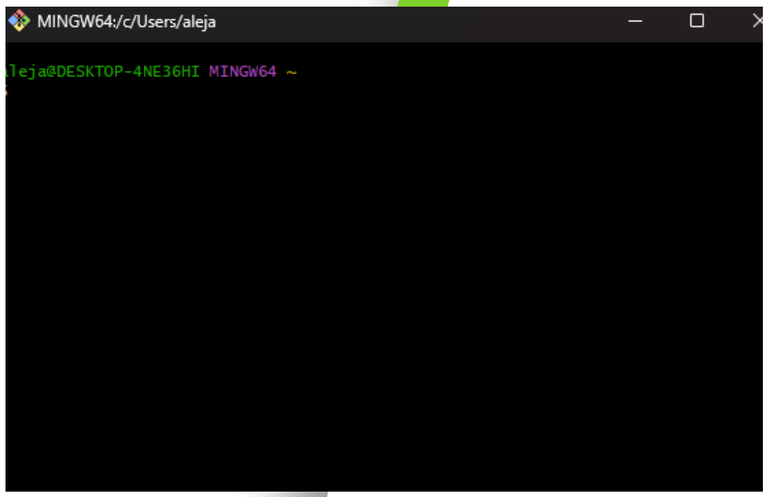
03 Alejandro Gut Angulo

INTRODUCCIÓN

Git y GitHub son herramientas clave para el control de versiones en proyectos de software. Git permite gestionar y registrar cambios en el código, mientras que GitHub ofrece una plataforma en línea para colaborar, compartir repositorios y mantener un historial organizado del trabajo en equipo.

¿Qué es Git?

Git es un sistema de control de versiones distribuido. Su objetivo principal es gestionar el historial de cambios de un proyecto, lo que permite que los desarrolladores trabajen en equipo sin perder el control de las distintas versiones del código.



Caracteristicas:

- Es un sistema distribuido: cada colaborador tiene una copia completa del repositorio.
- Permite crear y gestionar ramas para desarrollar funciones de forma independiente.
- Facilita el trabajo colaborativo y la integración de cambios.
- Guarda un historial completo de versiones de manera eficiente.

¿Qué es GitHub?

GitHub es una plataforma en la nube que almacena proyectos que utilizan Git. Se ha convertido en una red social para programadores, ya que no solo permite guardar repositorios remotos, sino también colaborar con otras personas en proyectos de software, ya sean públicos u privados.

Caracteristicas:

- Almacena repositorios en la nube para acceder desde cualquier lugar.
- Facilita la colaboración en proyectos open source y privados.
- Ofrece herramientas para revisar, organizar y automatizar el desarrollo.
- Funciona como una comunidad de desarrolladores, donde se pueden compartir proyectos y contribuir a otros.

¿Por qué es una tecnología emergente clave?

Facilita la colaboración remota y organizada.
Mejora la eficiencia en equipos de desarrollo.
Integrable con herramientas modernas como CI/CD y la nube.

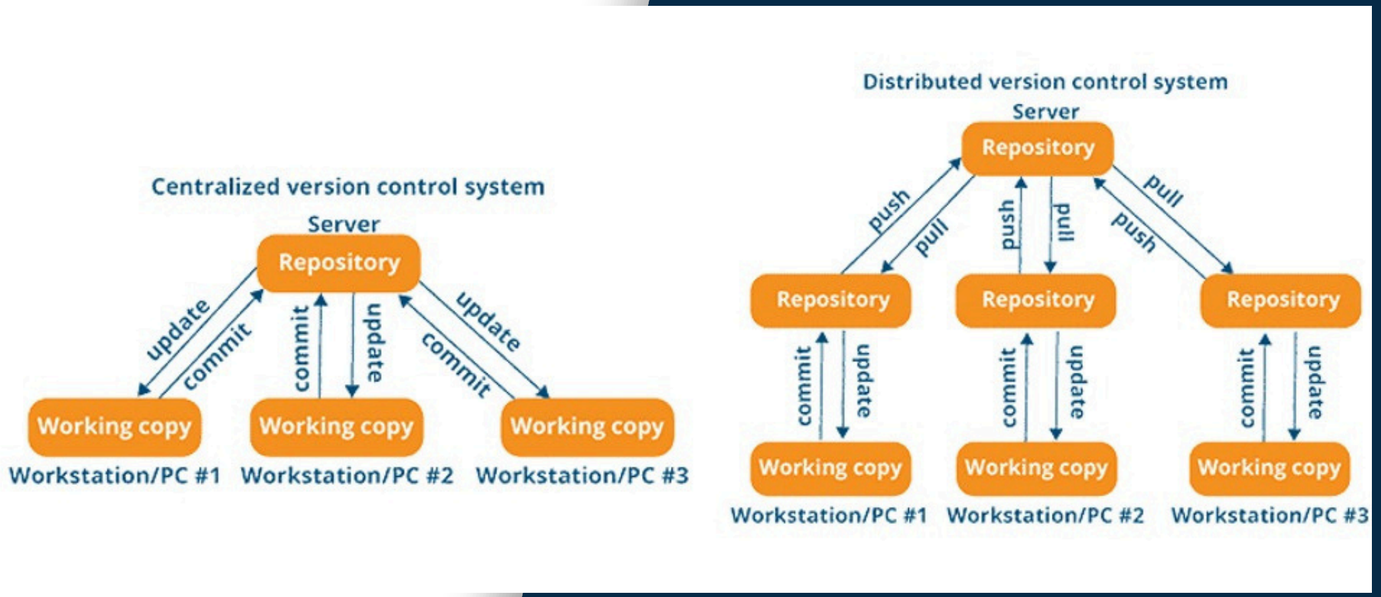
CI/CD son siglas que provienen del inglés:
CI → Continuous Integration (Integración Continua)
CD → Continuous Delivery / Continuous Deployment (Entrega / Despliegue Continuo)

Conceptos Clave

Sistema de Control de Versiones (VCS): Permite guardar cambios, recuperar versiones anteriores y colaborar sin conflictos.

Git: Sistema de control de versiones distribuido, cada desarrollador tiene copia completa del repositorio.

GitHub: Plataforma en la nube que centraliza repositorios y facilita colaboración mediante Pull Requests, Issues y revisión de código.

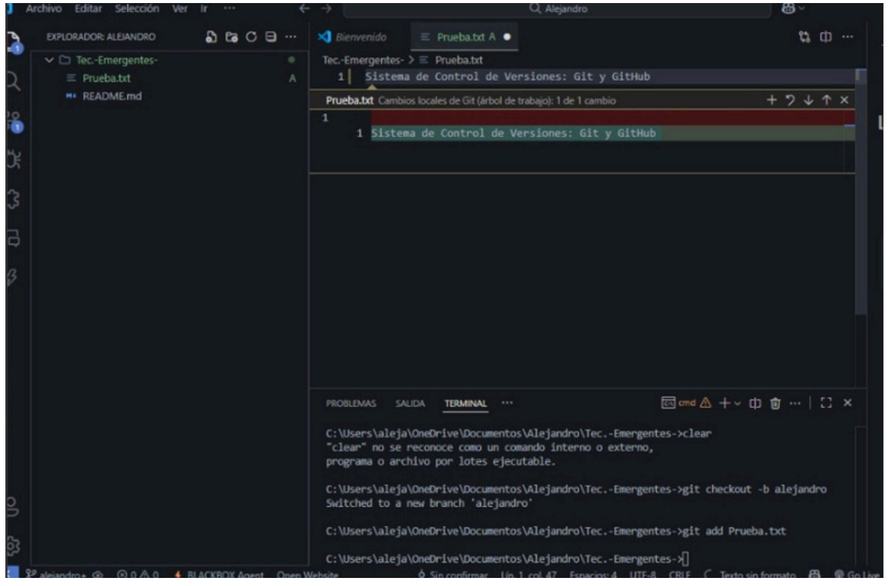


Comandos Principales de Git

commit → registrar cambios
branch → crear ramas
merge → combinar ramas
pull → traer cambios remotos
push → subir cambios locales al remoto

Herramientas utilizadas en el caso práctico

VS Code: editor de código
Git: control de versiones local
GitHub: repositorio remoto y colaboración



Desarrollo del caso práctico

01

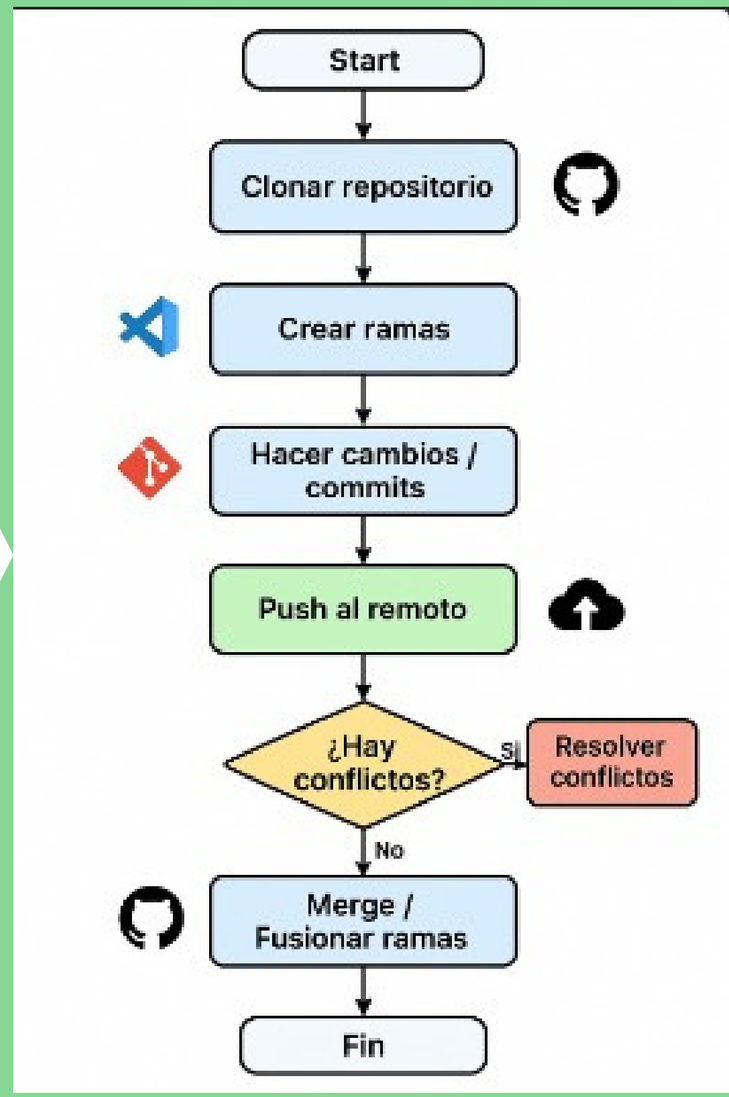
Problema a resolver:
Coordinar los aportes de todos los miembros del equipo de manera organizada y sin pérdida de información

02

Objetivo General
Implementar un flujo de trabajo colaborativo usando Git y GitHub

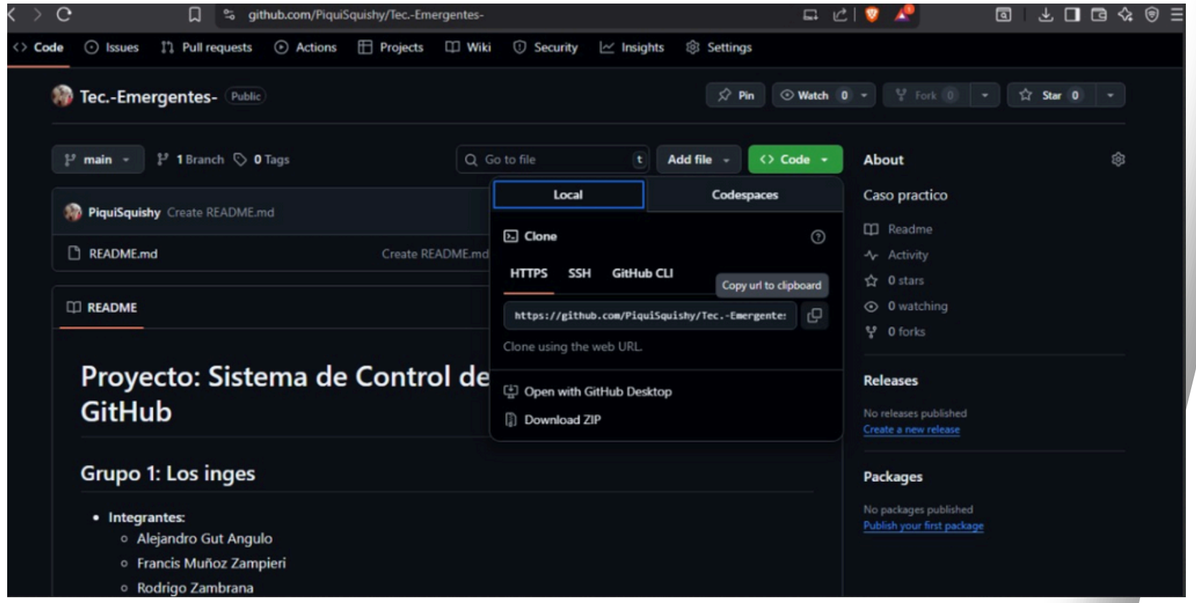
03

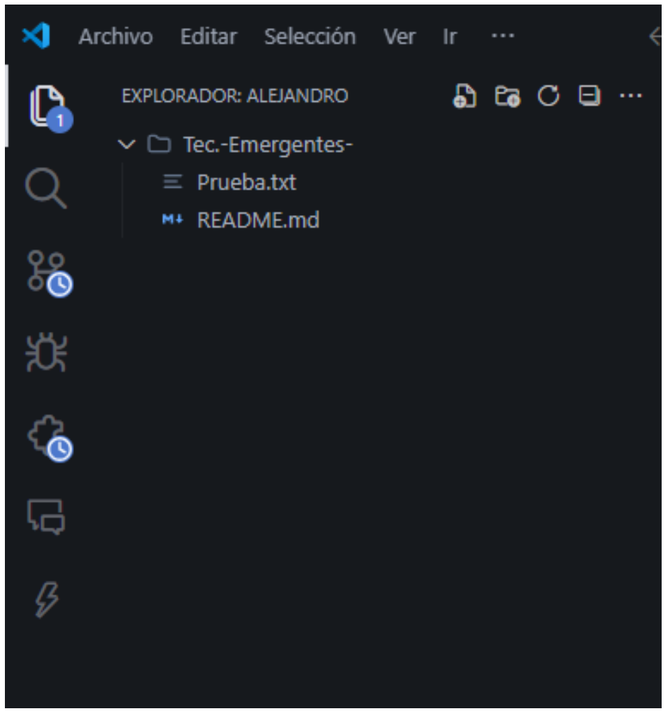
Objetivos Específico
Crear un repositorio central en GitHu
Cada miembro realiza cambios en su rama individua
Integrar los cambios mediante Pull Requests a la rama de desarrollo.



Desarrollo del caso práctico.

Paso 1: Una vez creado el repositorio, se clona el repsitorio en vscode






```
C:\Users\aleja\OneDrive\Documentos\Alejandro>git clone https://github.com/PiquiSquishy/Tec.-
Emergentes-.git
```

Paso 2: Crear ramas individuales (alejandro)

```
C:\Users\aleja\OneDrive\Documentos\Alejandro\Tec.-Emergentes->git checkout -b alejandro
Switched to a new branch 'alejandro'
```

Las ramas individuales en Git se crean para organizar el trabajo y evitar conflictos directos en el código principal.

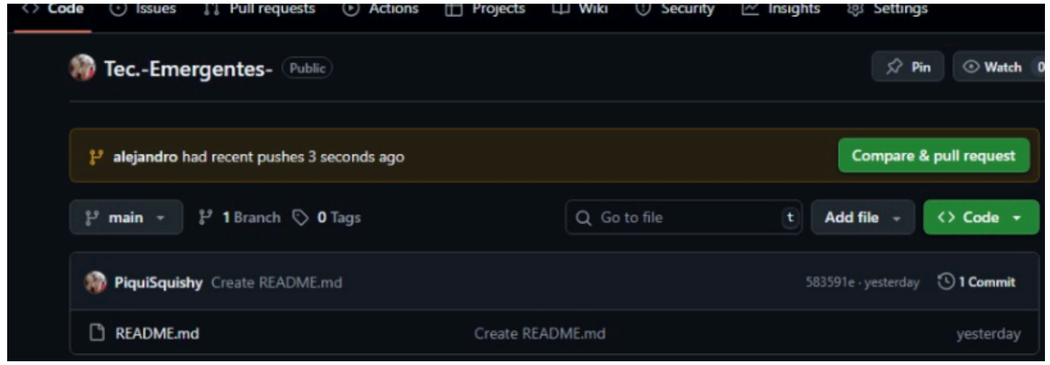
Paso 3: Editar archivos y hacer commits

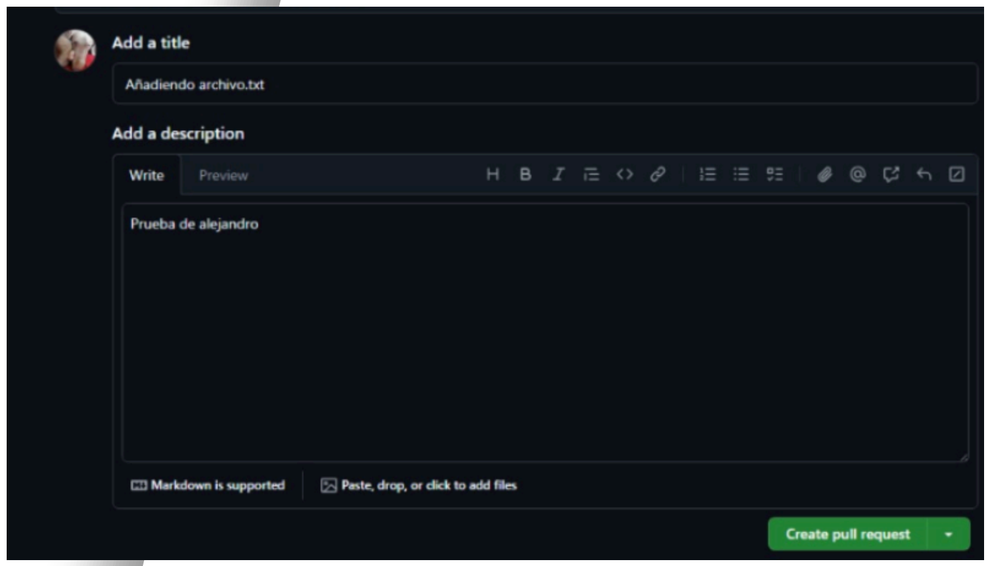
```
C:\Users\aleja\OneDrive\Documentos\Alejandro\Tec.-Emergentes->git commit -m "Añadiendo archivo.txt"
[alejandro f11a311] Añadiendo archivo.txt
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 Prueba.txt
```

Paso 4: Hacer push al remoto

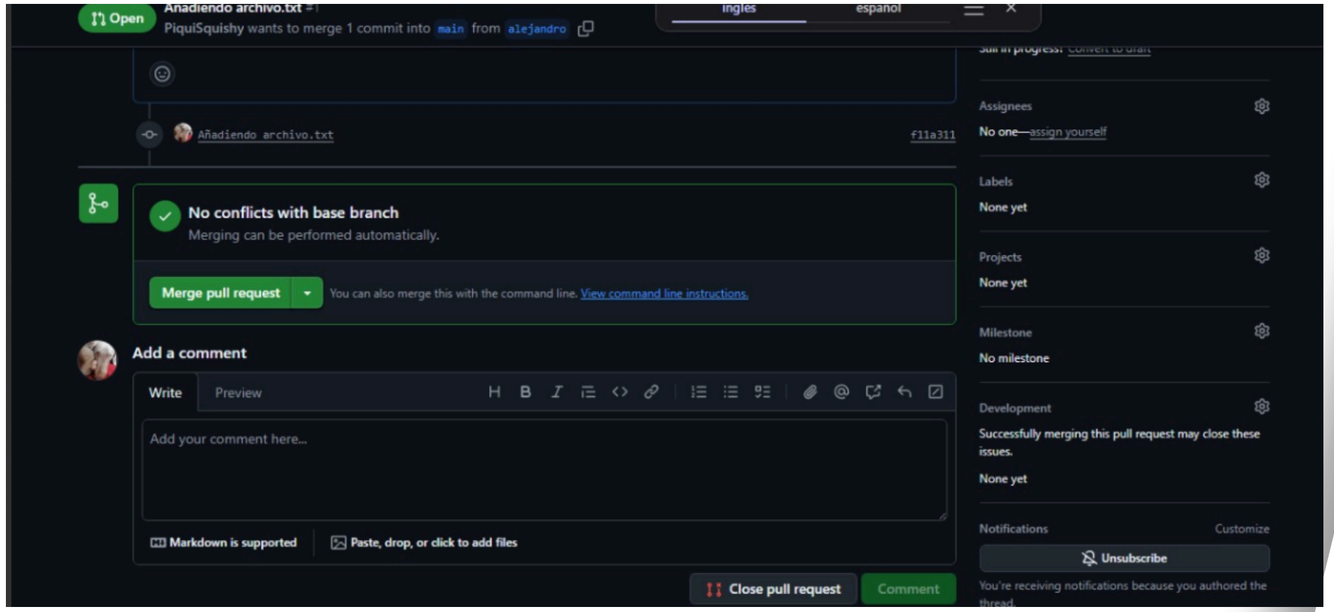
```
C:\Users\aleja\OneDrive\Documentos\Alejandro\Tec.-Emergentes->git push origin alejandro
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 297 bytes | 297.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'alejandro' on GitHub by visiting:
```

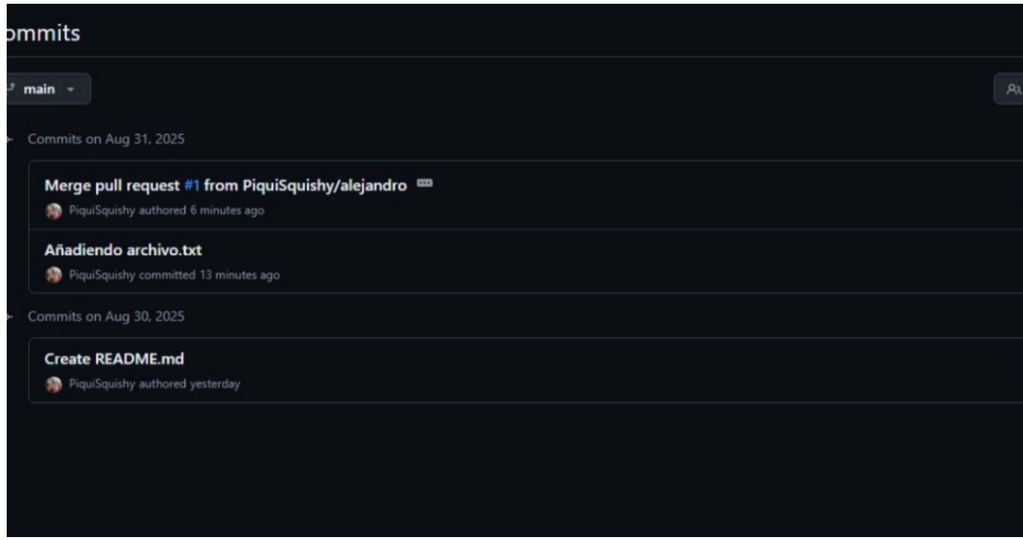
Paso 5: Crear Pull Request para integrar cambios



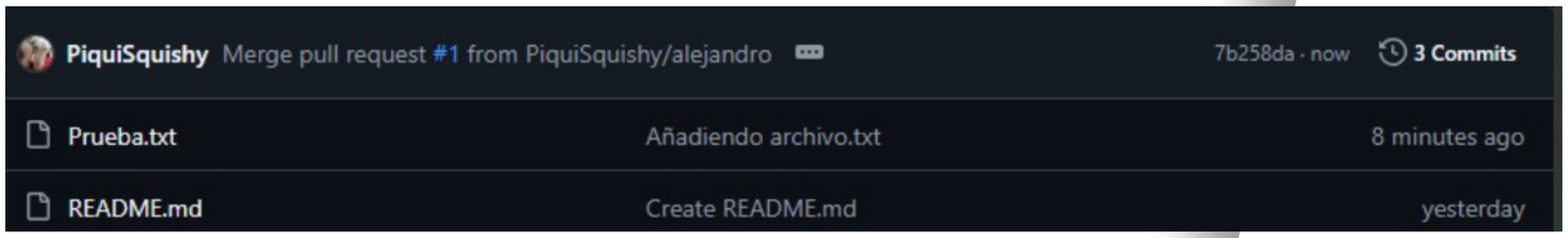


Paso 6: Resolver conflictos (si existen)





Paso 7: Fusionar los cambios en main



Conclusiones

La realización de este proyecto nos permitió comprender que Git y GitHub no son únicamente herramientas de control de versiones, sino una tecnología emergente clave en el desarrollo de software moderno. A través de su uso, experimentamos cómo la creación de repositorios, el trabajo con ramas, los commits descriptivos y la integración de cambios mediante pull requests facilitan la colaboración entre múltiples desarrolladores, incluso cuando se encuentran en diferentes lugares.