

Sistema de control de Versiones: Git y GitHub

INTEGRANTES

01

Francis Muñoz Zampieri

02

Rodrigo Zambrana Martínez

03

Alejandro Gut Angulo



INTRODUCCIÓN

El desarrollo de software moderno se caracteriza por la colaboración entre múltiples desarrolladores, equipos distribuidos en distintas partes del mundo y la necesidad de entregar soluciones de manera rápida, segura y confiable.

En este contexto, el control de versiones se convierte en una herramienta fundamental. Permite gestionar cambios en el código, mantener un historial de modificaciones y coordinar esfuerzos en proyectos complejos.

Sistemas de Control de Versiones (VCS)

Un Sistema de Control de Versiones (VCS, por sus siglas en inglés: Version Control System) es una herramienta de software que permite gestionar los cambios realizados en archivos y proyectos a lo largo del tiempo.

Su función principal es registrar cada modificación con información sobre qué se cambió, quién lo hizo, cuándo y por qué.

Características principales de un VCS:

Historial de versiones: ·Un VCS guarda todas las modificaciones realizadas en un archivo en forma de revisiones o commits.

Colaboración: Permite que varias personas trabajen en el mismo proyecto sin sobrescribir el trabajo de los demás.

Recuperación: Si ocurre un error grave, puedes restaurar el proyecto a una versión anterior

Ramas de desarrollo: El VCS permite crear “ramas” (branches) para probar nuevas ideas sin afectar el código principal

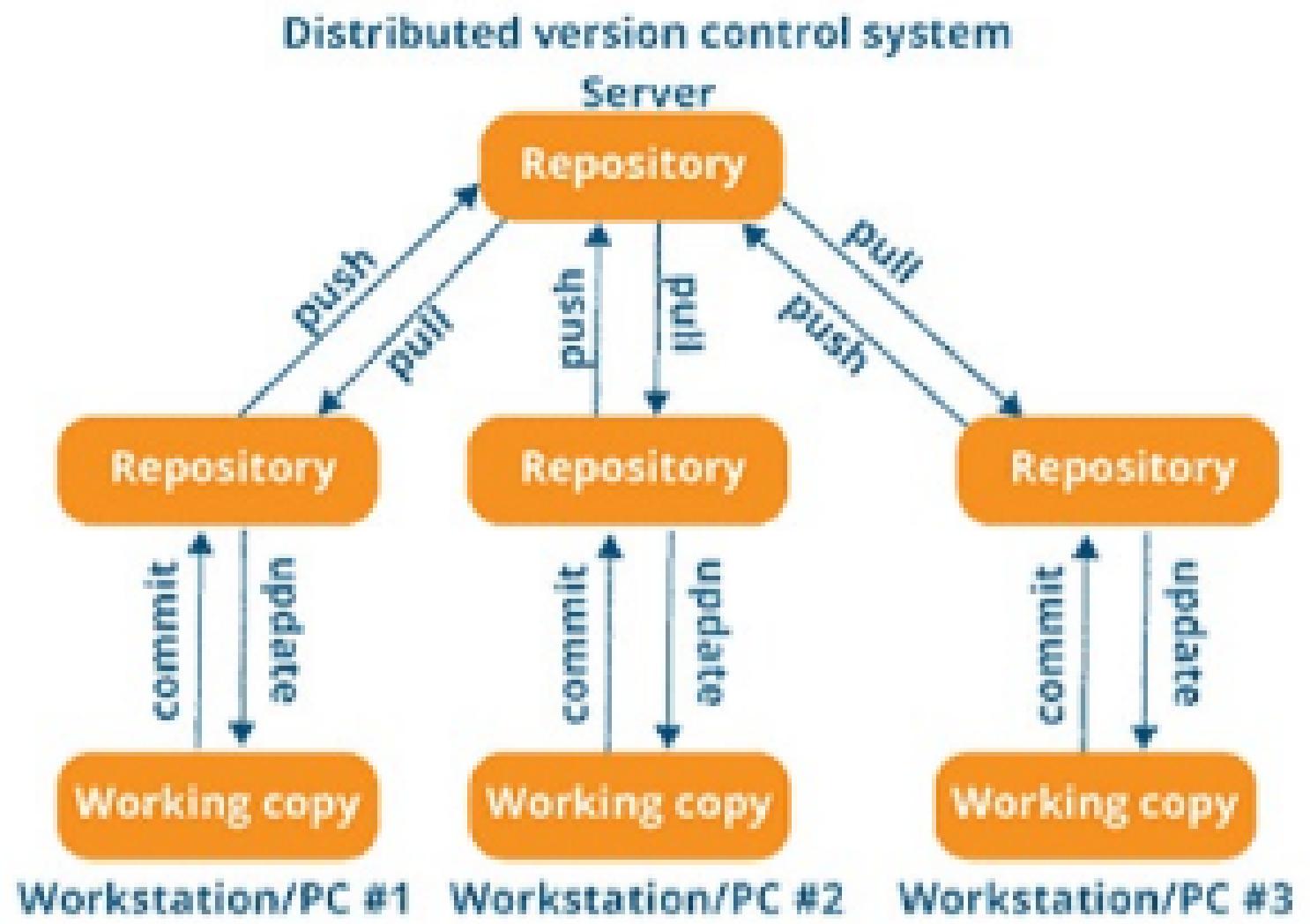
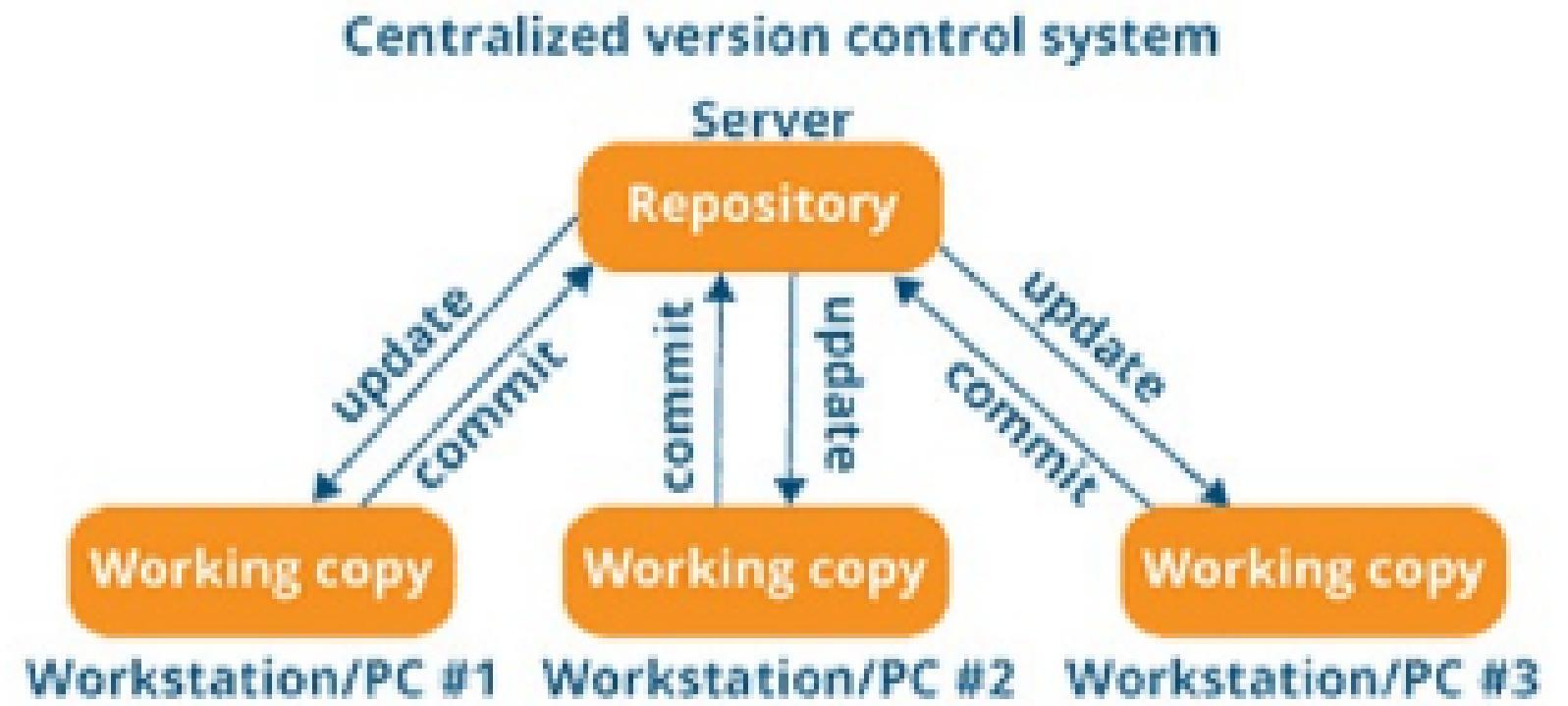
Tipos de VCS:

1. Sistemas de Control de Versiones Centralizados (CVCS):

En los sistemas centralizados, todo el código fuente y su historial se almacenan en un único servidor central al que los desarrolladores deben conectarse para acceder a los archivos o realizar cambios.

2. Sistemas de Control de Versiones Distribuidos (DVCS):

En los sistemas distribuidos, cada desarrollador mantiene una copia completa del repositorio, incluyendo todo el historial de cambios, lo que permite trabajar de manera independiente del servidor central.

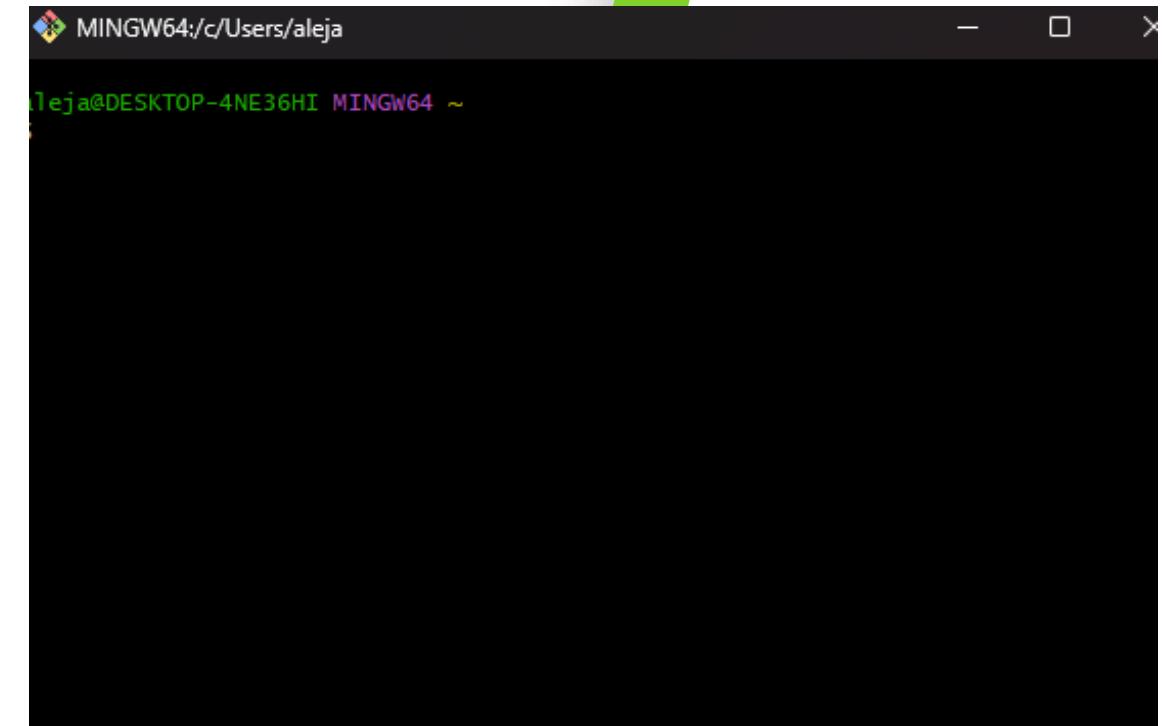


¿Qué es Git?

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.

Características:

- 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.

A screenshot of a terminal window titled "MINGW64:c/Users/aleja". The window shows a command-line interface with some text at the top and a black background for the main area. The title bar includes the path "c/Users/aleja" and the window title "MINGW64".

Conceptos basicos de Git:

Repository (Repository):

Es el lugar donde se almacena todo el proyecto, incluyendo archivos, carpetas y el historial de cambios.

Commit:

Es el acto de guardar un conjunto de cambios en el repositorio local. Cada commit tiene un identificador único (hash SHA) que garantiza su integridad.

Branch (Rama):

Las ramas permiten trabajar en diferentes líneas de desarrollo de forma aislada.

Merge (Fusión):

Es el proceso de combinar los cambios de una rama en otra.

Pull:

Comando que descarga cambios de un repositorio remoto y los integra en tu repositorio local.

Conceptos basicos de Git:

- **Push:**

Comando que envía tus commits locales al repositorio remoto, permitiendo que otros desarrolladores accedan a tus cambios.

- **Add:**

Comando que prepara los archivos modificados para el próximo commit. Es como seleccionar qué cambios quieres guardar en tu “fotograma” del proyecto.

Ventajas de usar Git

- Permite trabajar sin conexión y mantener un historial completo de cambios.
- Facilita la colaboración entre varios desarrolladores simultáneamente.
- Ofrece seguridad mediante hashes criptográficos que garantizan la integridad de los datos.
- Permite experimentar sin riesgo, gracias a las ramas y fusiones.
- Escala desde proyectos pequeños hasta grandes proyectos con miles de archivos y colaboradores.

¿Qué es GitHub?

GitHub es una plataforma en línea que potencia y extiende las funcionalidades de Git, facilitando la colaboración entre desarrolladores y equipos de cualquier tamaño. Mientras Git se centra en el control de versiones del código, GitHub agrega un conjunto de herramientas sociales y de gestión que permiten centralizar proyectos, organizar tareas y automatizar flujos de trabajo.

Git vs GitHub



VS



¿Qué es Git?

- Sistema de control de versiones distribuido
- Gestiona el historial de cambios de un proyecto
- Permite trabajar en equipo

¿Qué es GitHub?

- Plataforma en la nube
- Almacena proyectos que usan Git
- Collaboración open source y privado

Funciones más importantes:

Pull Requests (PRs):

Los pull requests son solicitudes que permiten a un desarrollador proponer cambios realizados en una rama para integrarlos a la rama principal del proyecto.

Issues:

Los issues son herramientas para gestionar errores, tareas pendientes y mejoras dentro del proyecto.

Wikis y documentación:

GitHub proporciona espacios para crear documentación y wikis dentro de cada proyecto.

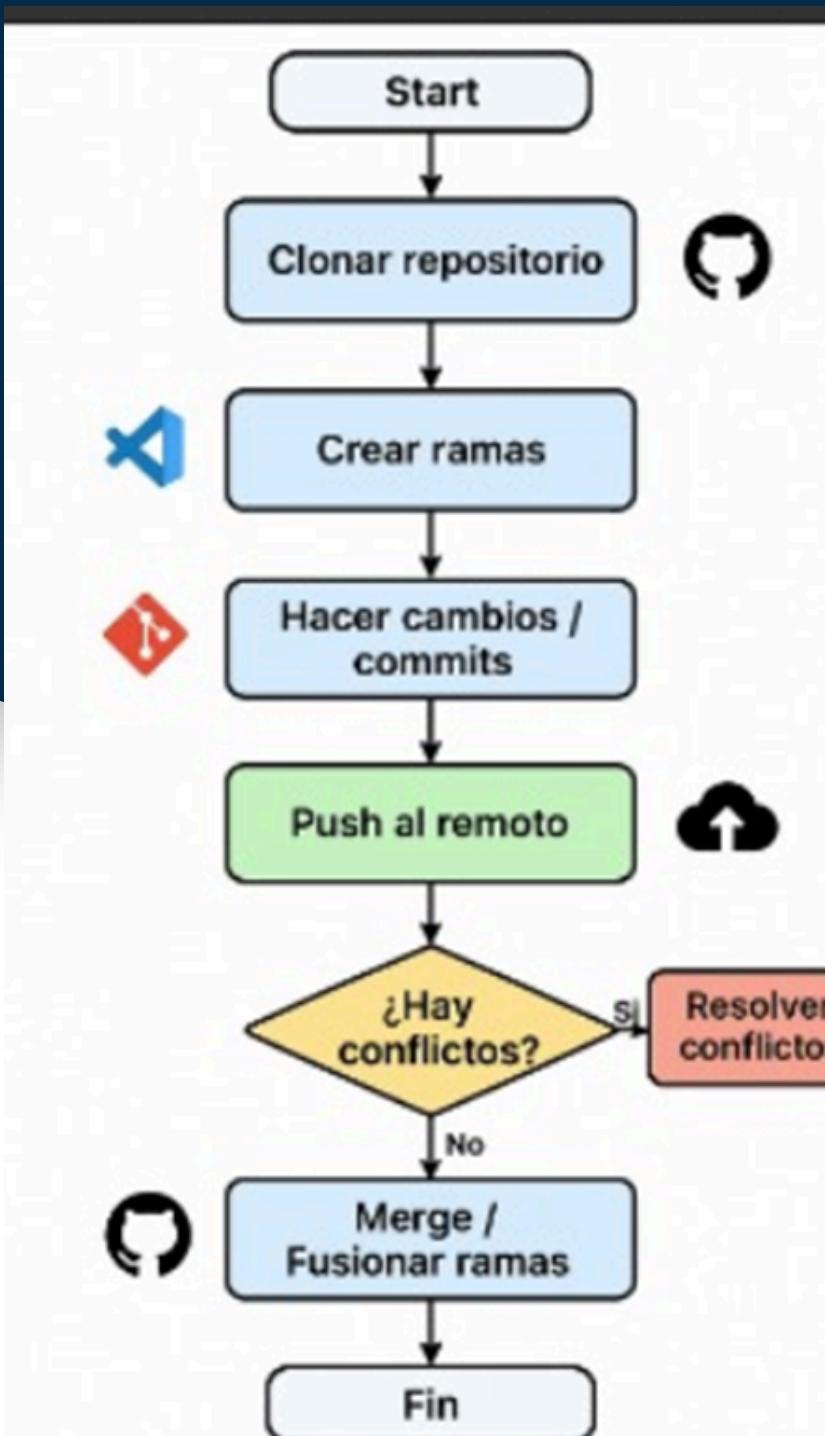
Funciones más importantes:

GitHub Actions:

GitHub Actions permite automatizar flujos de trabajo como pruebas de código, compilación de proyectos y despliegues a servidores o entornos de producción.

Control de permisos y roles:

GitHub permite definir quién puede ver, editar o administrar cada repositorio.



Git y GitHub como Tecnologías Emergentes

Git y GitHub son consideradas tecnologías emergentes debido a la forma en que han transformado radicalmente el desarrollo de software moderno, haciendo que la colaboración, la gestión de proyectos y la entrega continua sean más eficientes y accesibles para desarrolladores de todo el mundo.

Razones que destacan su relevancia como tecnologías emergentes:

Colaboración global:

Git y GitHub permiten que equipos distribuidos geográficamente trabajen de manera simultánea en los mismos proyectos.

Agilidad y soporte a metodologías modernas:

Estas herramientas se integran de manera natural con metodologías ágiles y prácticas DevOps.

Razones que destacan su relevancia como tecnologías emergentes:

Automatización y calidad de software:

GitHub ofrece integración con pipelines de CI/CD (Integración Continua y Despliegue Continuo) a través de GitHub Actions.

Ecosistema de comunidad:

GitHub se ha convertido en un espacio central para la colaboración en proyectos de código abierto.

Escalabilidad:

Tanto Git como GitHub son capaces de manejar proyectos de distintas magnitudes, desde aplicaciones pequeñas hasta sistemas distribuidos complejos.

Comandos Principales de Git

commit → registrar cambios

branch → crear ramas

merge → combinar ramas

pull → traer cambios remotos

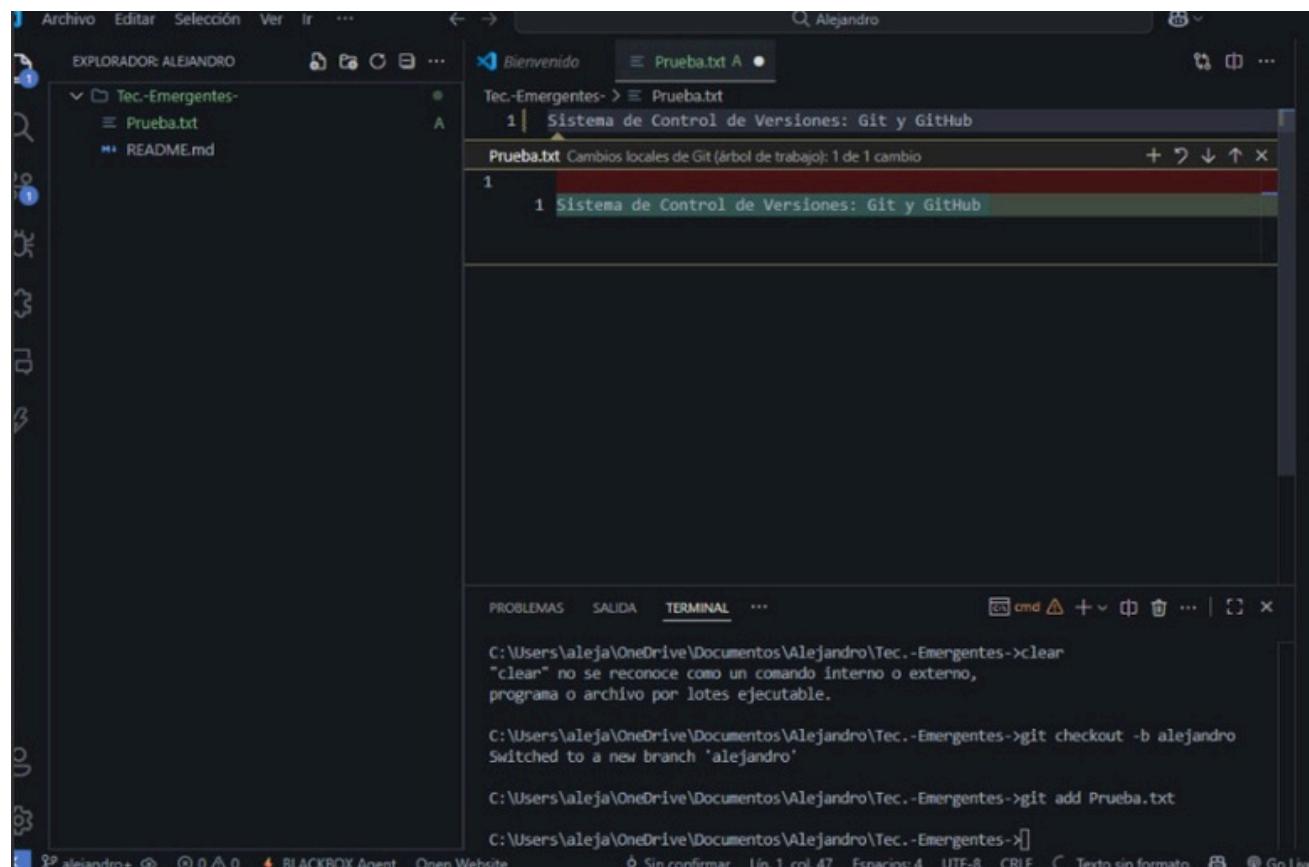
push → subir cambios locales al remoto

Herramientas utilizadas en el caso práctico

VS Code: editor de código

Git: control de versiones local

GitHub: repositorio remoto y colaboración



The screenshot shows the VS Code interface. On the left is the Explorer sidebar with a folder named 'Tec.-Emergentes-' containing 'Prueba.txt' and 'README.md'. The main area displays a commit history for 'Prueba.txt' under the heading 'Sistema de Control de Versiones: Git y GitHub'. The commit message is '1 Sistema de Control de Versiones: Git y GitHub'. Below this is a terminal window showing command-line interactions:

```
C:\Users\aleja\OneDrive\Documentos\Alejandro>Tec.-Emergentes->clear  
"clear" no se reconoce como un comando interno o externo,  
programa o archivo por lotes ejecutable.  
C:\Users\aleja\OneDrive\Documentos\Alejandro>git checkout -b alejandro  
Switched to a new branch 'alejandro'  
C:\Users\aleja\OneDrive\Documentos\Alejandro>git add Prueba.txt  
C:\Users\aleja\OneDrive\Documentos\Alejandro>[REDACTED]
```

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

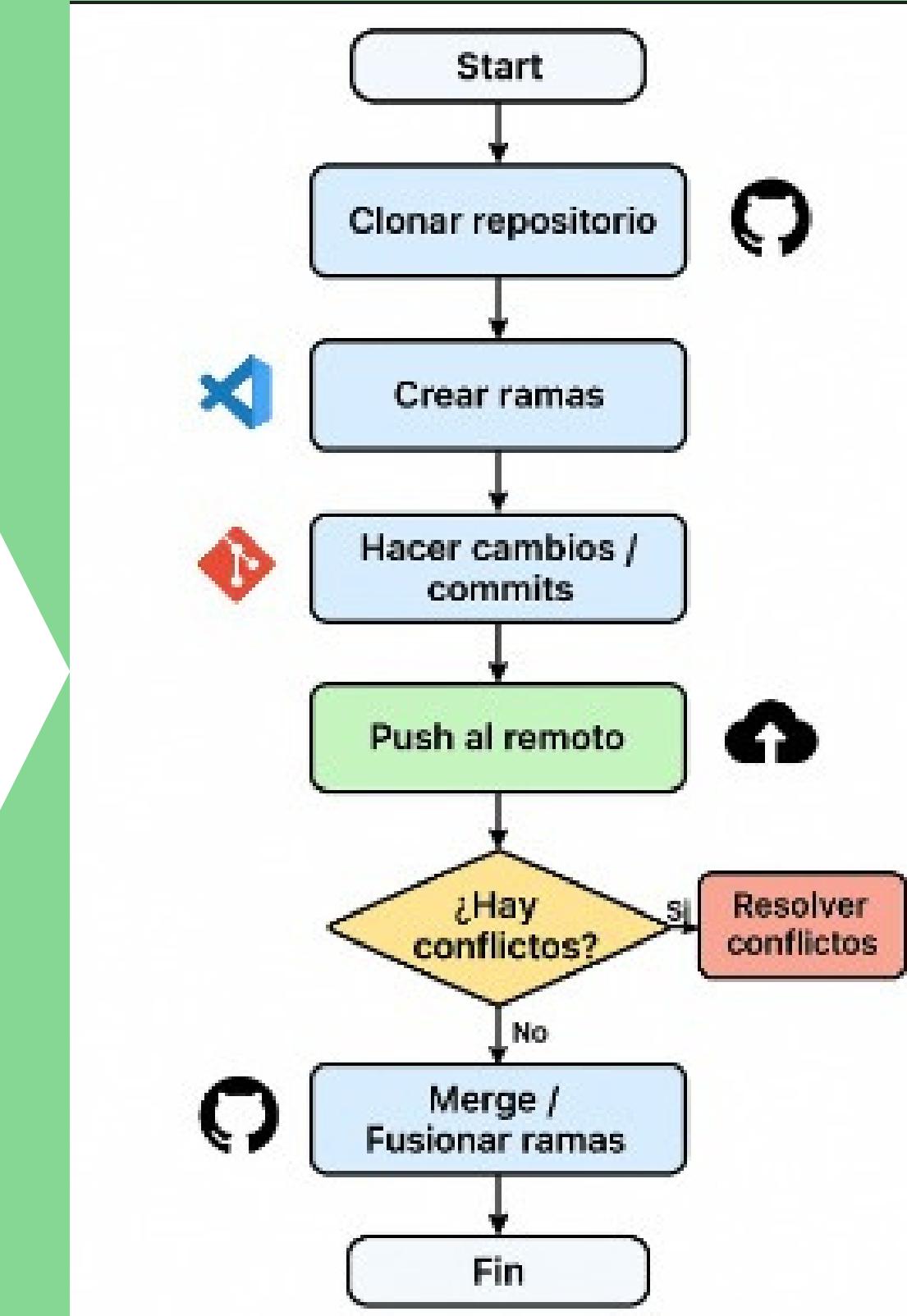
03

Objetivos Específico

Crear un repositorio central en GitHub

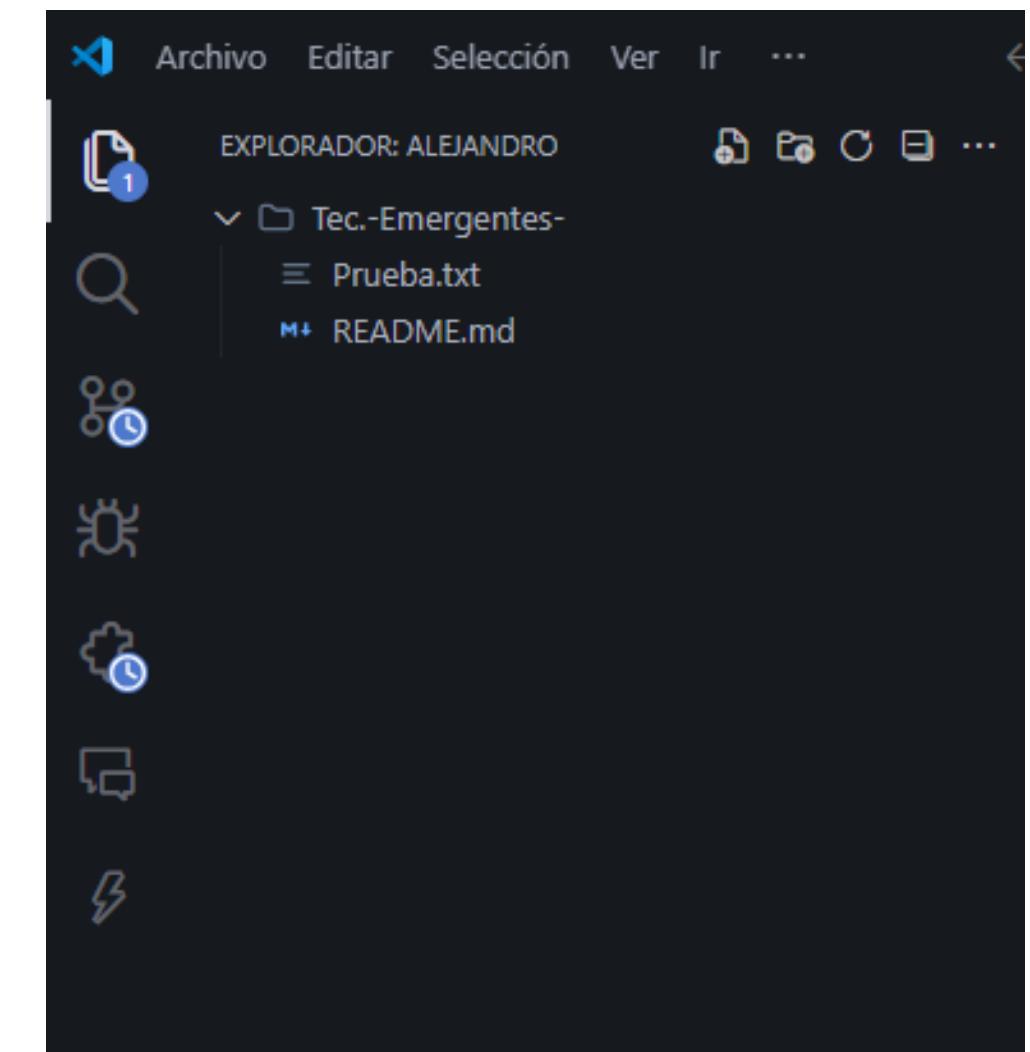
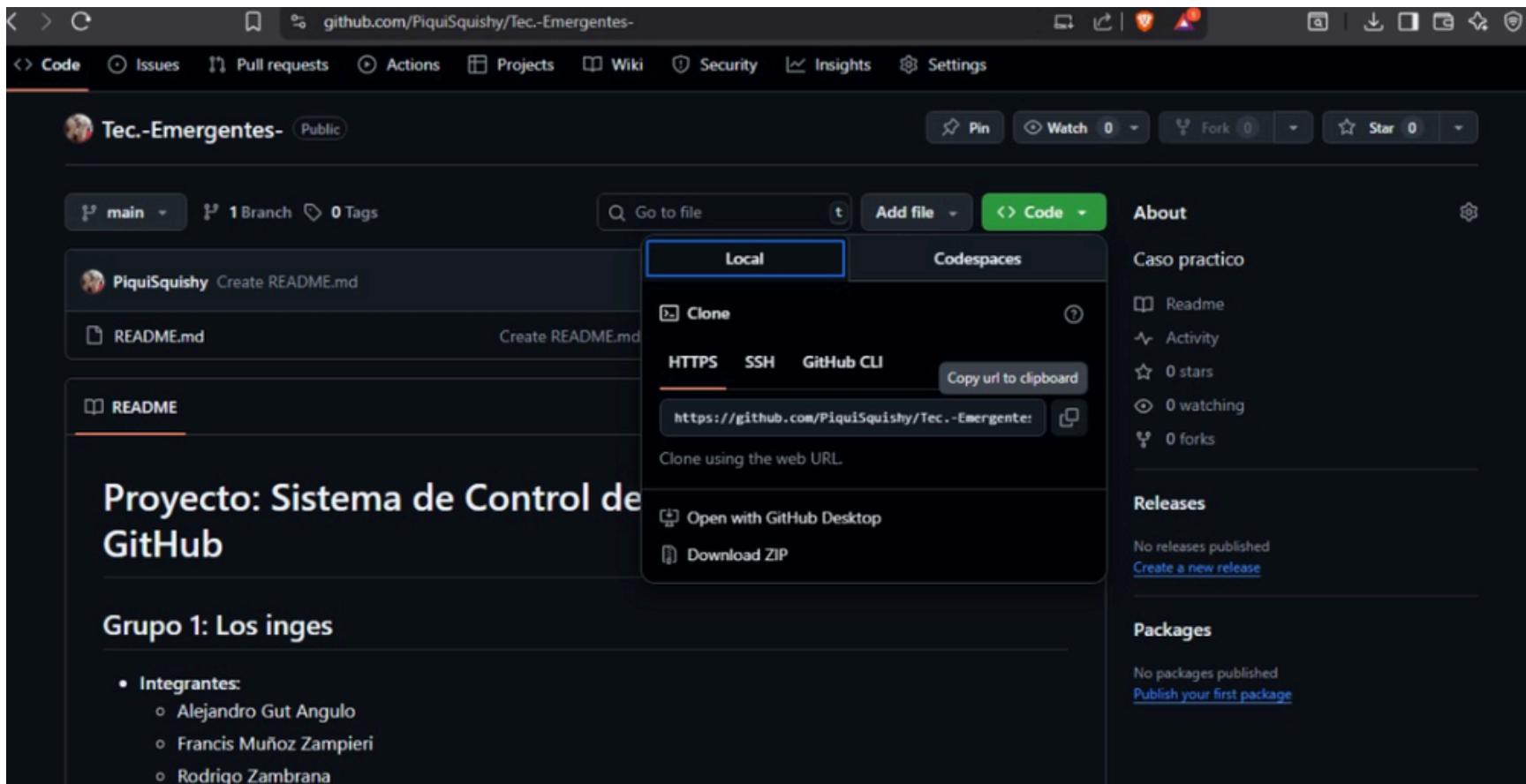
Cada miembro realiza cambios en su rama individual

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 repositorio 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.

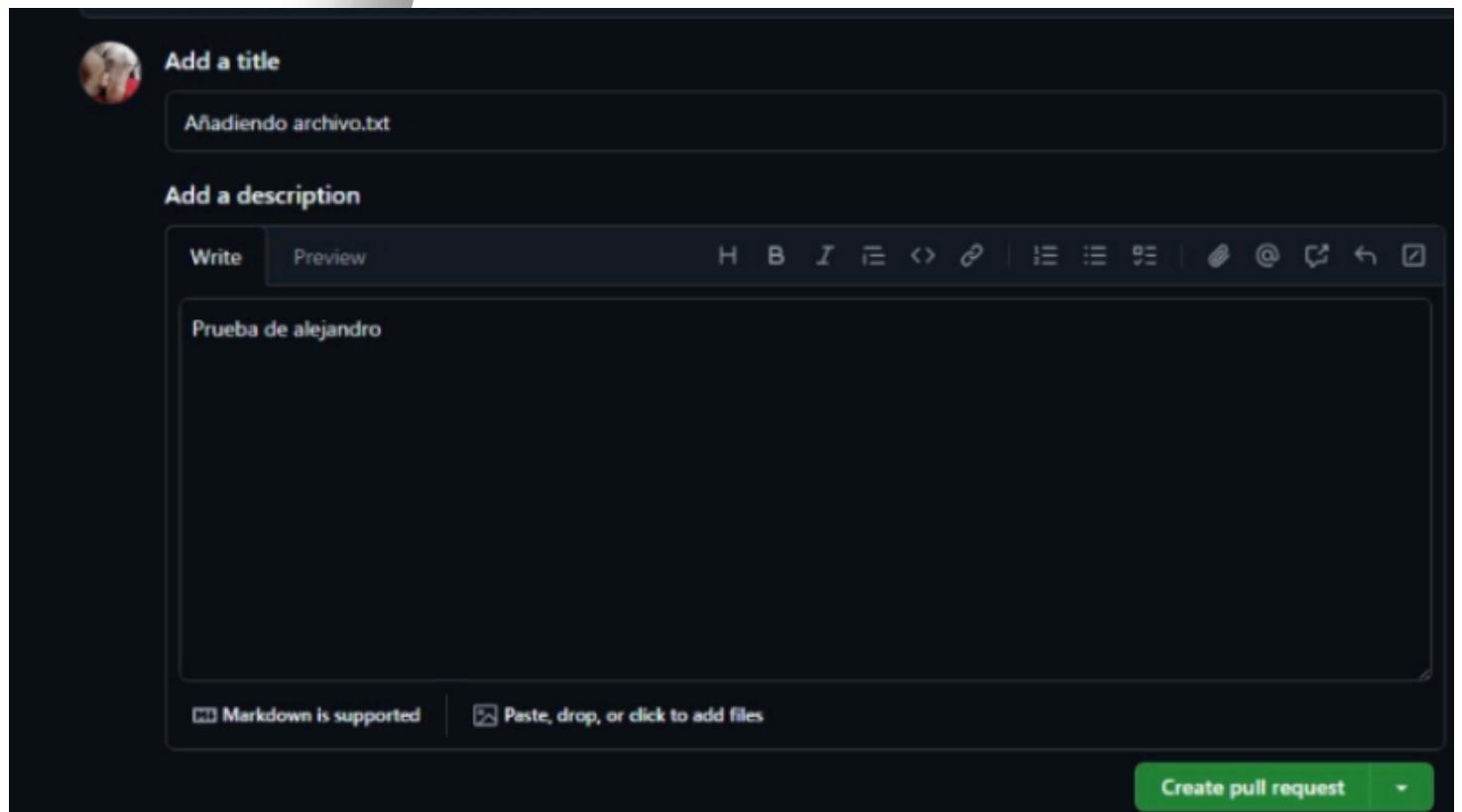
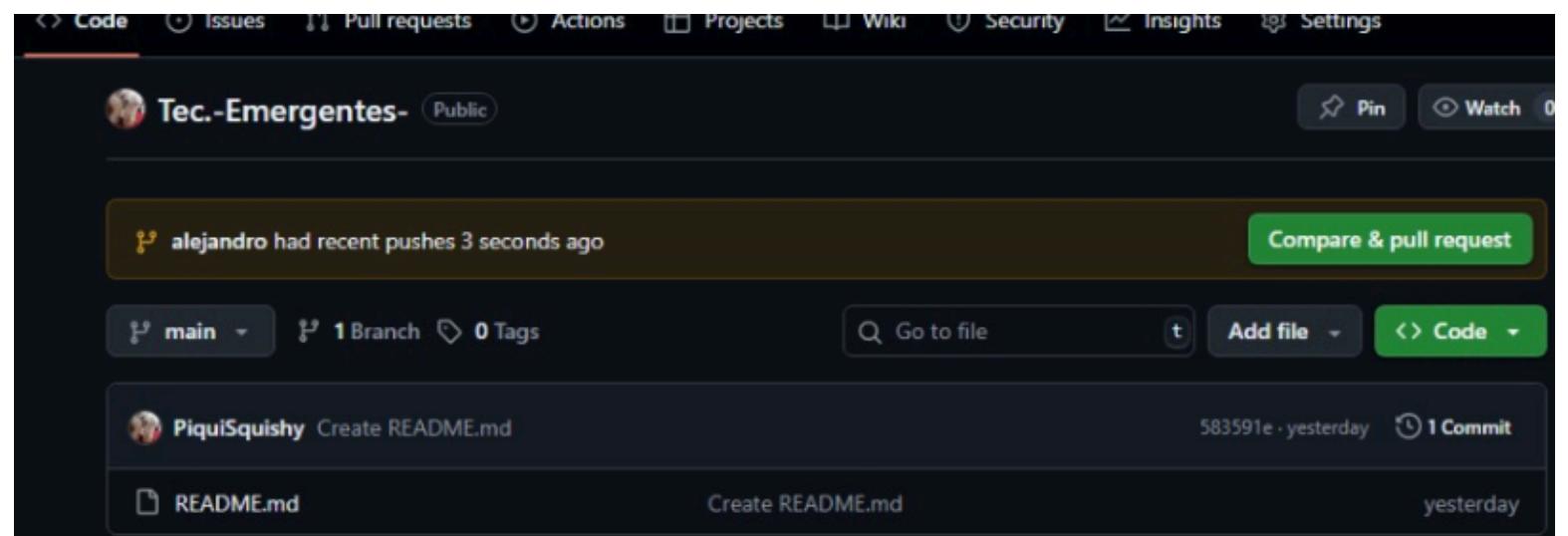
Paso 3: Editar archivos y hacer commits

```
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)

The screenshot shows a GitHub pull request interface. At the top, it says "Anadiendo archivo.txt" and "PiquiSquishy wants to merge 1 commit into main from alejandro". Below this, there are tabs for "ingles" and "español". A green box highlights the message "No conflicts with base branch" with the note "Merging can be performed automatically." A "Merge pull request" button is visible. On the right side, there are settings for the pull request, including "Assignees" (No one—assign yourself), "Labels" (None yet), "Projects" (None yet), "Milestone" (No milestone), and "Development" (None yet). A note states "Successfully merging this pull request may close these issues." Below the pull request details is a comment section with a "Write" button, a rich text editor toolbar, and a text input field for "Add your comment here...". A note at the bottom says "Markdown is supported" and "Paste, drop, or click to add files". Buttons for "Close pull request" and "Comment" are at the bottom.

The screenshot shows a GitHub commits page for the "main" branch. It lists two commits: "Merge pull request #1 from PiquiSquishy/alejandro" (authored by PiquiSquishy 6 minutes ago) and "Añadiendo archivo.txt" (authored by PiquiSquishy 13 minutes ago). Below these, another commit is listed: "Create README.md" (authored by PiquiSquishy yesterday). The commits are timestamped as "Commits on Aug 31, 2025" and "Commits on Aug 30, 2025".

Paso 7: Fusionar los cambios en main

The screenshot shows a GitHub commit history for the "main" branch. It includes the merge commit "Merge pull request #1 from PiquiSquishy/alejandro" (7b258da · now) and the commit "Añadiendo archivo.txt" (authored by PiquiSquishy 8 minutes ago). Below these, the commit "Create README.md" (authored by PiquiSquishy yesterday) is also shown. The commit history is timestamped as "7b258da · now" and "8 minutes ago" and "yesterday".

Noticias sobre GitHub



Reuters

Microsoft to add Anthropic's AI coding agent to its GitHub service

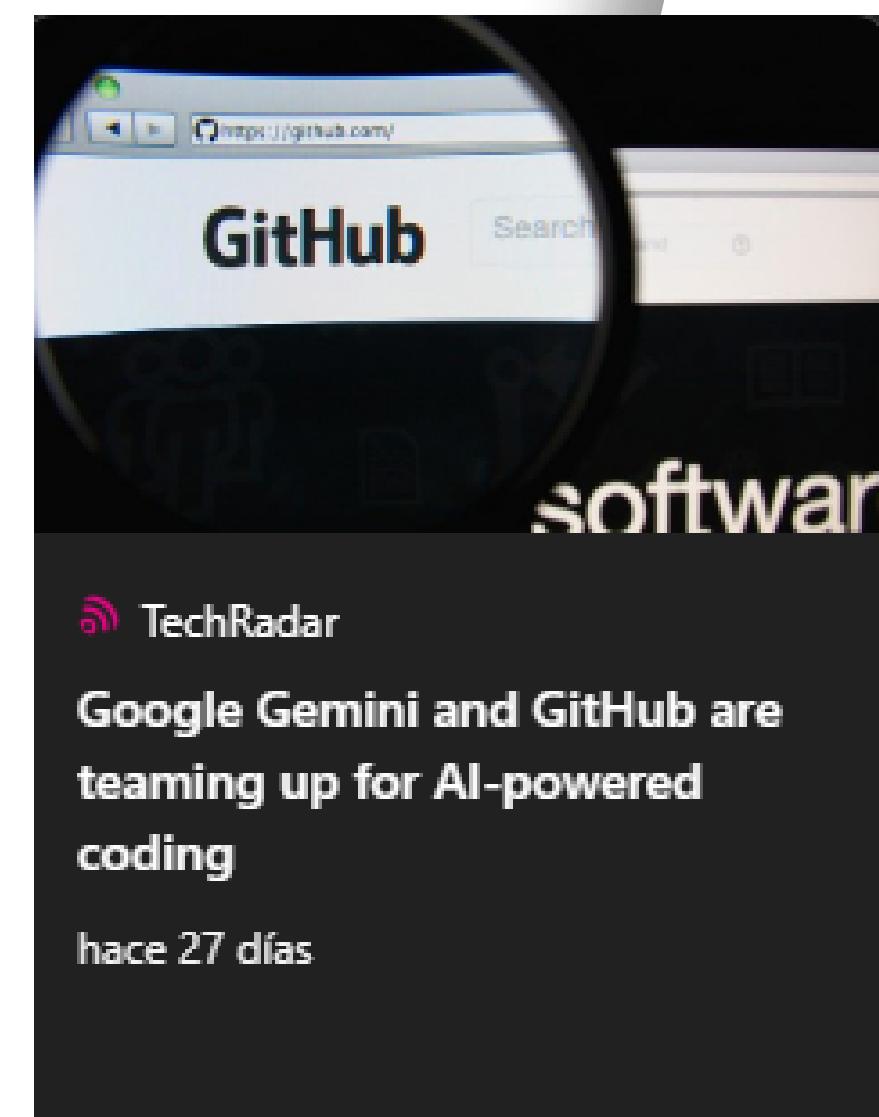
20 may 2025



IT Pro

GitHub just launched a new 'mission control center' for developers to delegate tasks to AI coding agents

hace 14 días



Noticias sobre GitHub

1. GitHub y Anthropic se unen: multiplica a los agentes de IA disponibles

En mayo de 2025, Microsoft anunció que integrará el agente de IA de la startup Anthropic en GitHub, junto con los de OpenAI y su propio agente. Esto marca una estrategia de GitHub hacia una plataforma de IA más abierta y neutral, ofreciendo a los desarrolladores mayor flexibilidad en la elección de herramientas inteligentes.

2. GitHub lanza agente de IA para corregir bugs automáticamente

También en mayo de 2025, GitHub introdujo un nuevo agente de IA integrado en GitHub Copilot, diseñado para tareas como corregir errores (bugs) y añadir funcionalidades automáticamente. Esto transforma a Copilot de una herramienta de sugerencia de código a un asistente de codificación más autónomo y proactivo.

3. GitHub implementa “Mission Control”: gestiona tus agentes de IA desde cualquier parte

En agosto de 2025, GitHub lanzó en vista previa pública la nueva interfaz “Mission Control” (Agents Panel), accesible desde cualquier página en GitHub. Permite a los desarrolladores delegar tareas en agentes de Copilot, realizar pruebas y generar pull requests sin salir del flujo de trabajo, mejorando la continuidad y eficiencia del desarrollo.

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.



Gracias

Alberto Navarro