# GIT

## Comandos

### 🔹 Zonas

➡️Tabla resumen: Las 3 zonas de Git

| **🔢 Zona** | **📄 Contenido** | **📦 Propósito** | **🛠️ Comandos involucrados** |
| --- | --- | --- | --- |
| **1. Working Directory** | Archivos reales que estás editando | Donde haces tus cambios locales | git status  git diff  git restore |
| **2. Staging Area** | Cambios seleccionados para el siguiente commit | Prepara los archivos para ser confirmados | git add  git restore –staged  git reset |
| **3. Repository** | Historial completo de commits confirmados | Registro permanente del proyecto | git commit  git log  git restore |

🔄 Flujo entre zonas

| **Acción** | **De → Hacia** | **Comando usado** |
| --- | --- | --- |
| Preparar cambios | Working Directory → Staging | git add archivo |
| Confirmar cambios | Staging → Repository | git commit -m "mensaje" |
| Deshacer cambios no guardados | Working Directory ← Repository | git restore archivo |
| Quitar del staging | Staging ← Working Directory | git restore --staged archivo |

✅ Aclaración:

* Las 3 zonas son **locales**.
* git push **no afecta tu Working Directory ni Staging Area**, solo **sube commits del repositorio local al remoto** (como GitHub, GitLab...).

### 🔹 git restore --staged archivo

📌 ¿Qué hace?

➡️ **Quita el archivo del Staging Area**, pero **mantiene los cambios en tu archivo**.

🧪 Ejemplo:

echo "Línea nueva" >> zonas.txt

git add zonas.txt # Lo preparas para commit (Staging)

git restore --staged zonas.txt # ❌ Lo sacas del Staging

✅ El archivo sigue con la línea nueva, pero ya **no se incluirá en el commit**  
✅ Ideal si te equivocaste al hacer git add

### 🔹 git restore archivo

📌 ¿Qué hace?

➡️ **Descarta los cambios hechos en tu archivo**, y lo deja como estaba en el último commit.

🧪 Ejemplo:

echo "Otra línea" >> zonas.txt # Modificas el archivo

git restore zonas.txt # 🔁 Revierte los cambios

❌ Se borra la línea "Otra línea"  
✅ El archivo vuelve exactamente como estaba en el último commit

## Preparación de las ramas principales

# Configura tu nombre de usuario global para los commits

*git config --global user.email jorge@gmail.com*

*git config --global user.name "Jorge"*

# Define que la rama principal por defecto se llame "main" en lugar de "master"

*git config --global user.name "Jorge"git config --global init.defaultBranch main*

# Inicializa un nuevo repositorio Git en la carpeta actual

*git init*

**Hacer modificaciones en los ficheros.**

# Añade todos los archivos al área de preparación (staging)

*git add .*

# Crea el primer commit con mensaje

*git commit -m "Primer commit con README"*

# Asocia tu repo local con el repositorio remoto en GitHub

*git remote add origin* [*https://github.com/jvicmar95/JORGE2025.git*](https://github.com/jvicmar95/JORGE2025.git)

# Sube la rama "main" y la configura para seguimiento remoto

*git push -u origin main*

**Creamos rama develop aparte de la main que ya existe**

*git checkout main*

*git pull origin main*

*git checkout -b develop*

*git push -u origin develop*

*git branch -a*

## Entender las zonas

🔹 Paso 1: Crear la rama test-zonas desde main

git checkout main

git pull origin main

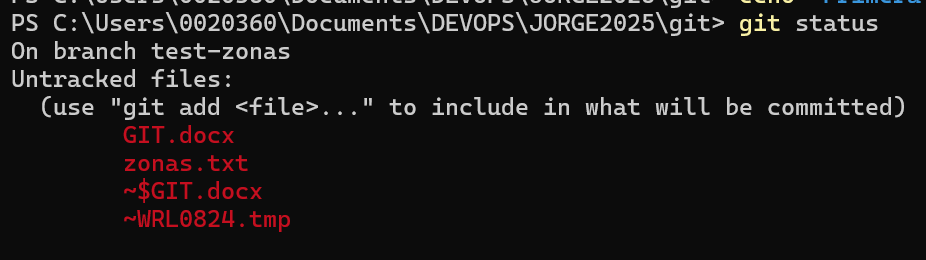
git checkout -b test-zonas

🔹 Paso 2: Crear un nuevo archivo (Working Directory)

echo "Primera línea del archivo" > zonas.txt

git status # 🔍 Git detecta todo como nuevo (untracked)

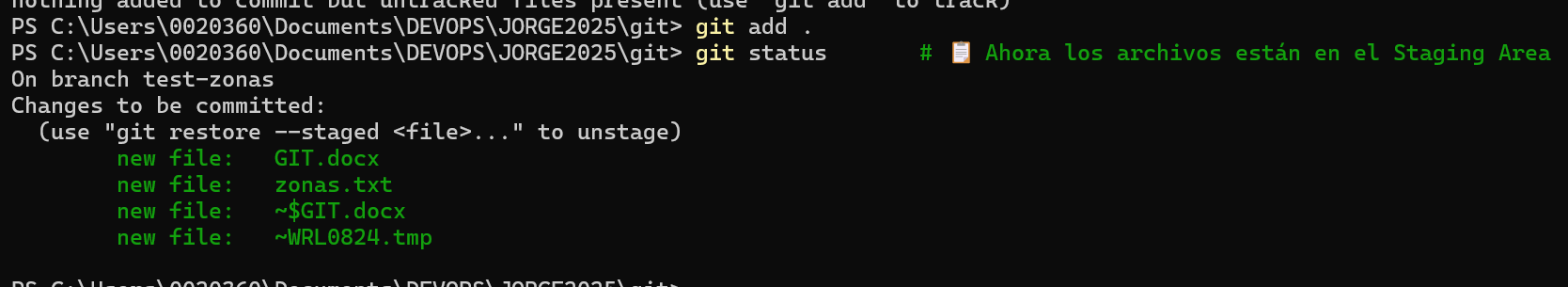
✅ **ZONA**: Los archivos están en el **Working Directory** (sin seguimiento)



🔹 Paso 3: Mover a Staging Area

git add .

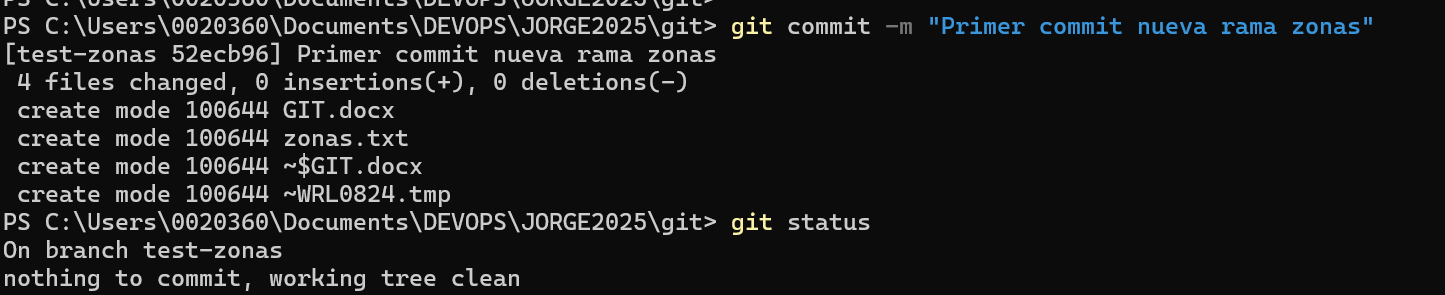
git status # 📋 Ahora los archivos están en el Staging Area



🔹 Paso 4: Confirmar (commit) a Repository

git commit -m "Primer commit nueva rama zonas"

git status

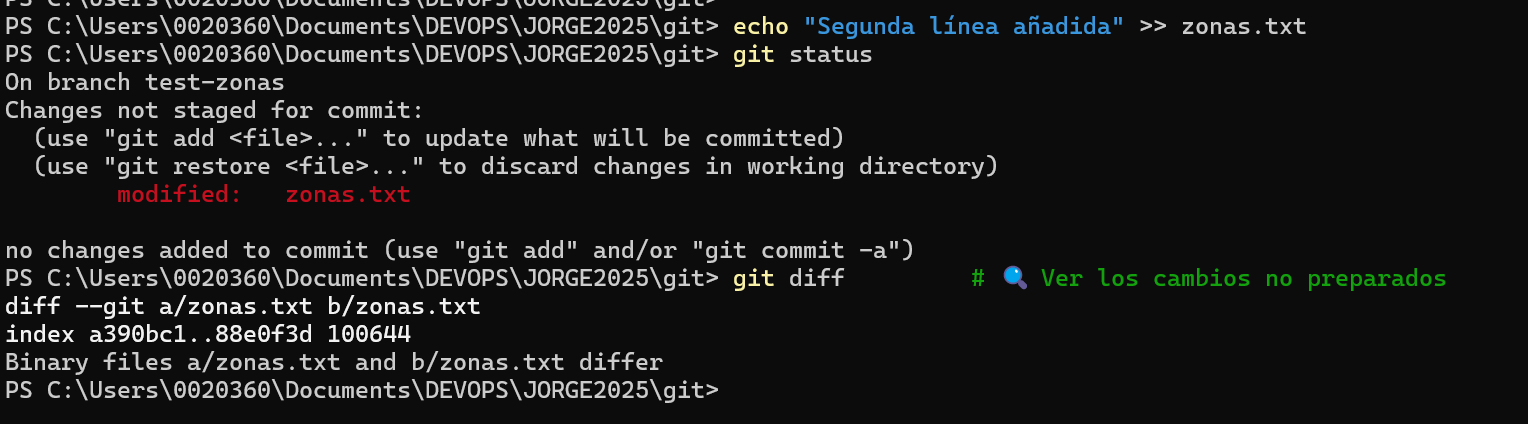


🔹 Paso 5: Modificar el archivo (Working Directory otra vez)

echo "Segunda línea añadida" >> zonas.txt

git status

git diff # 🔍 Ver los cambios no preparados

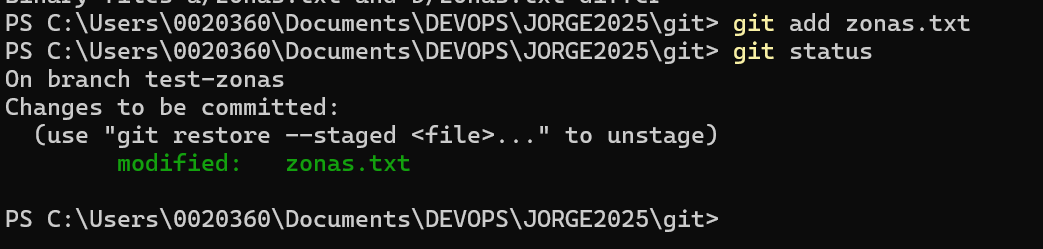


✅ Has modificado el archivo en el **Working Directory**

🔹 Paso 6: Pasar al Staging Area

git add zonas.txt

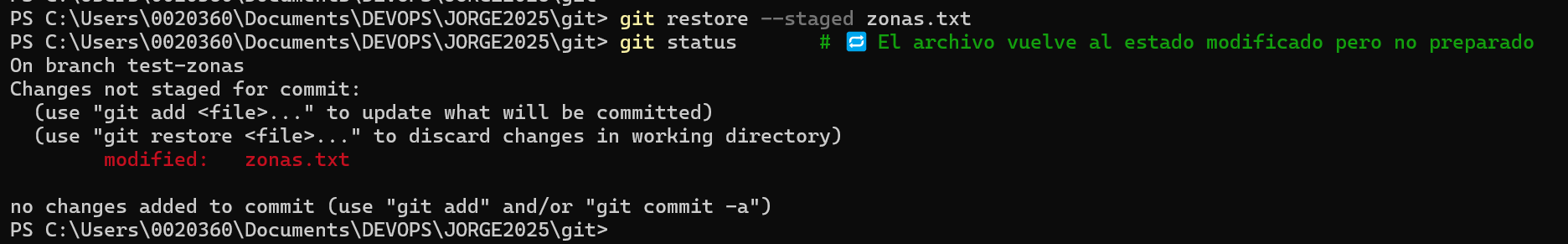
git status



🔹 Paso 7: Sacar el archivo del Staging (volver al Working Directory)

git restore --staged zonas.txt

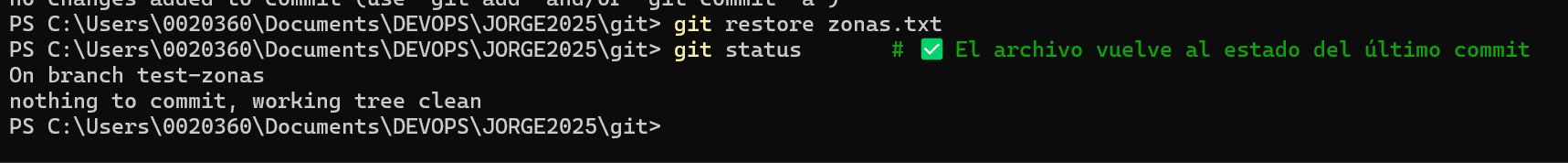
git status # 🔁 El archivo vuelve al estado modificado pero no preparado



🔹 Paso 8: Descartar los cambios del Working Directory

git restore zonas.txt

git status # ✅ El archivo vuelve al estado del último commit



🔹 Paso 9: Hacer un cambio definitivo y confirmarlo

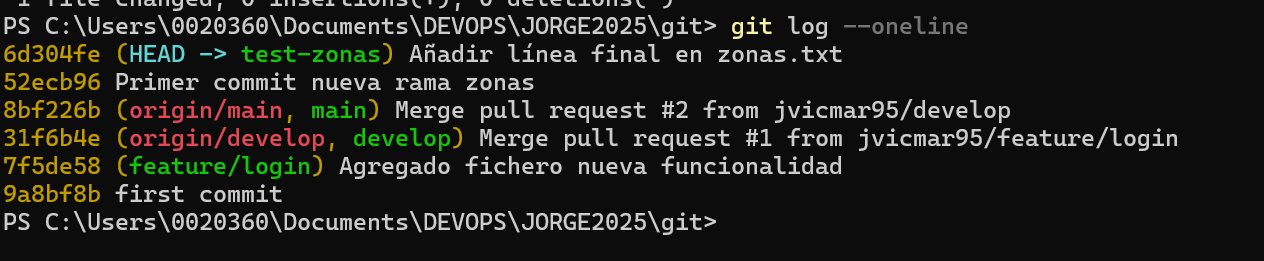
echo "Línea final confirmada" >> zonas.txt

git add zonas.txt

git commit -m "Añadir línea final en zonas.txt"

🔹 Paso 10: Ver historial de commits

git log –oneline



## Añadir nueva funcionalidad. Creación rama nueva funcionalidad con merge primero a develop y una vez probado merge a main

**Paso 1: Asegúrate de estar en develop y actualizado**

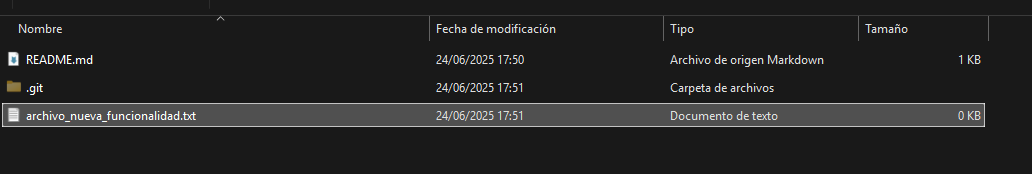
*git checkout develop*

*git pull origin develop*

**Paso 2: Crea la rama feature desde develop y cámbiate a ella**

*git checkout -b feature/login*

**Paso 3: Crea el archivo nuevo para la funcionalidad**



**Paso 4: Añade el archivo y haz commit**

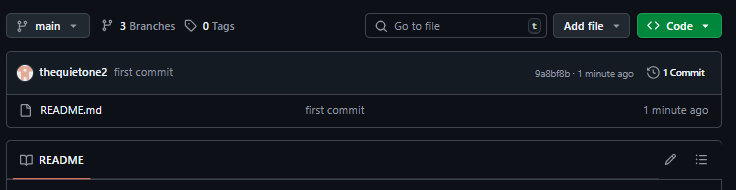
*git add .*

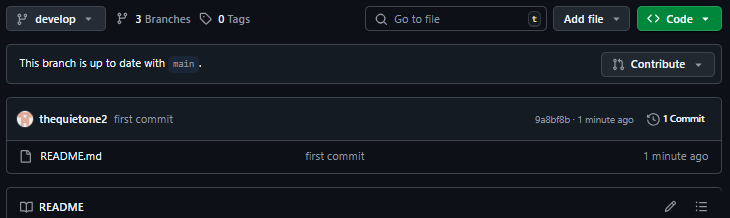
*git commit -m "Agregado fichero nueva funcionalidad"*

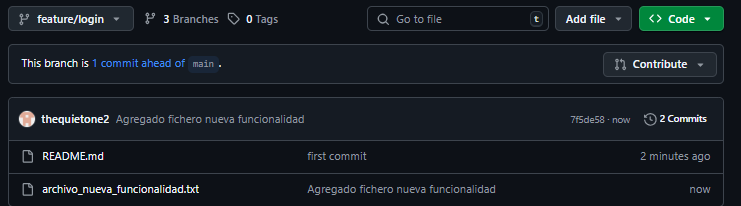
**Paso 5: Sube la rama feature/login a remoto**

*git push origin feature/login*

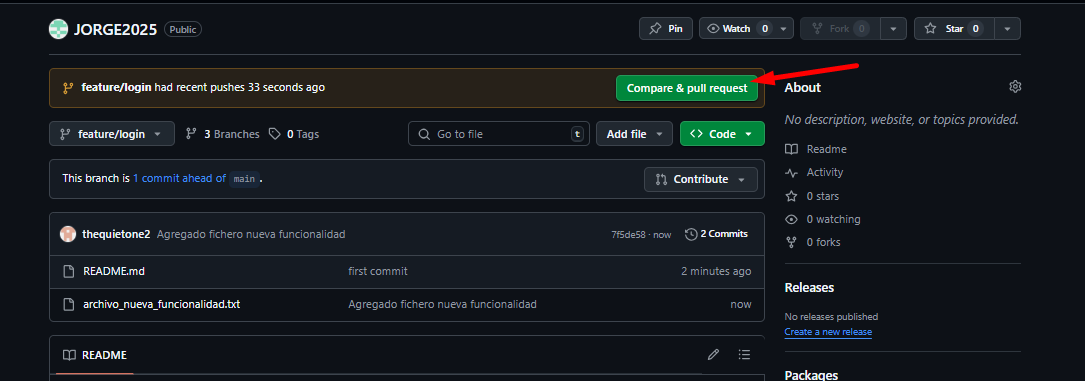
**Paso 6: Comprobación**

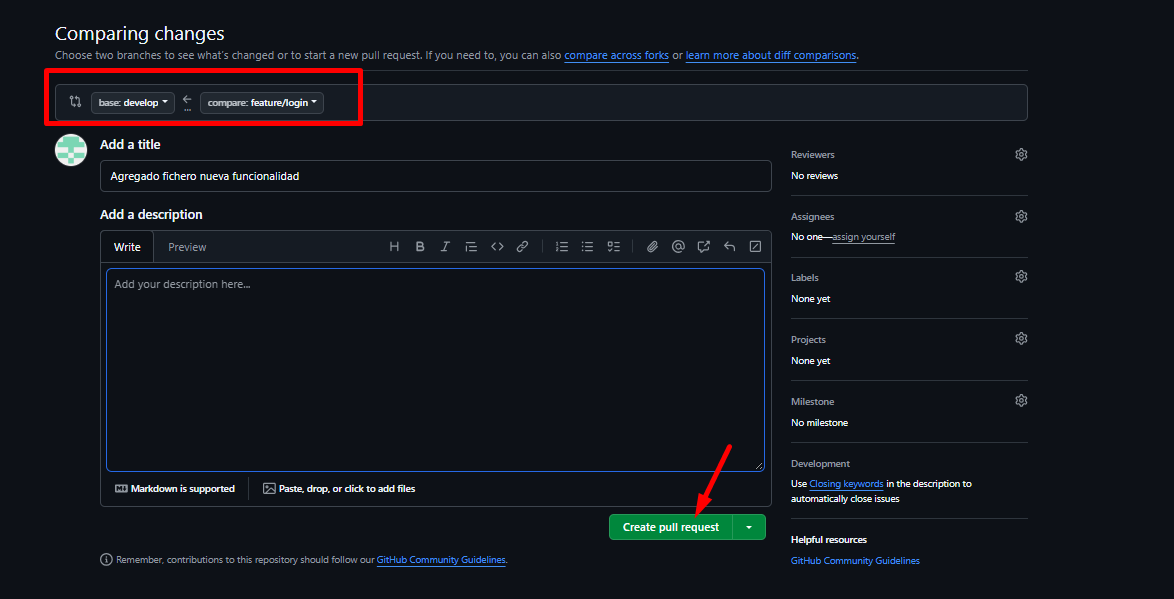




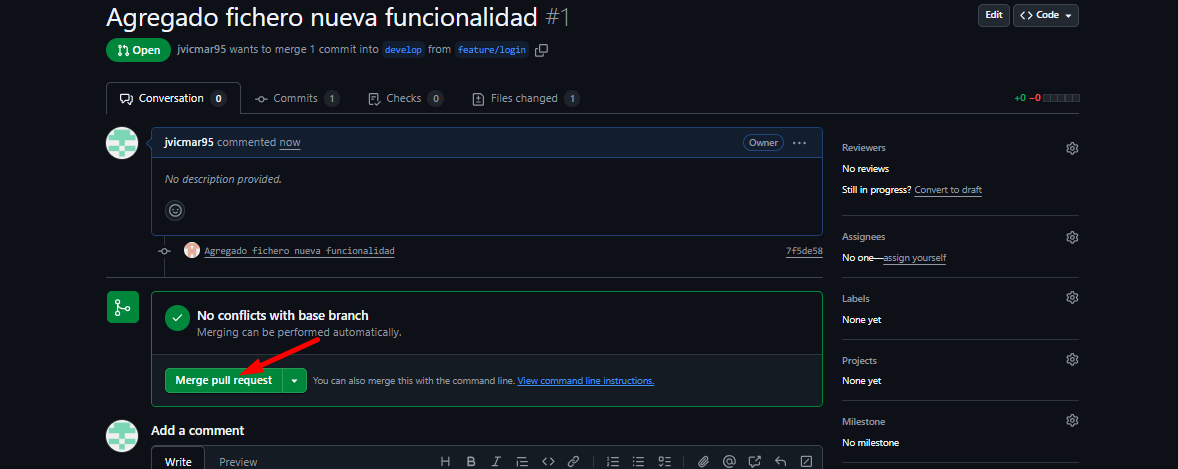


**Paso 7: Creación del pull request**

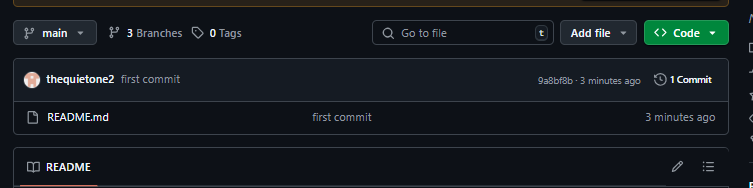


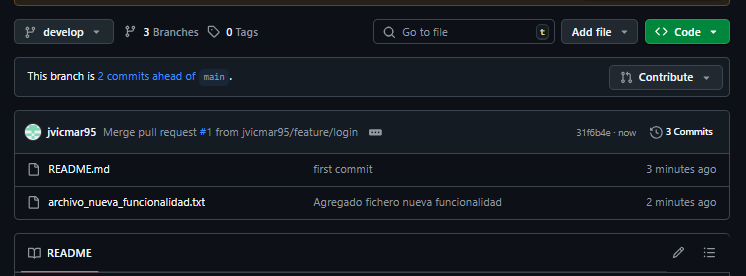


**Paso 8: Hacer el merge (en una empresa seria el responsable)**



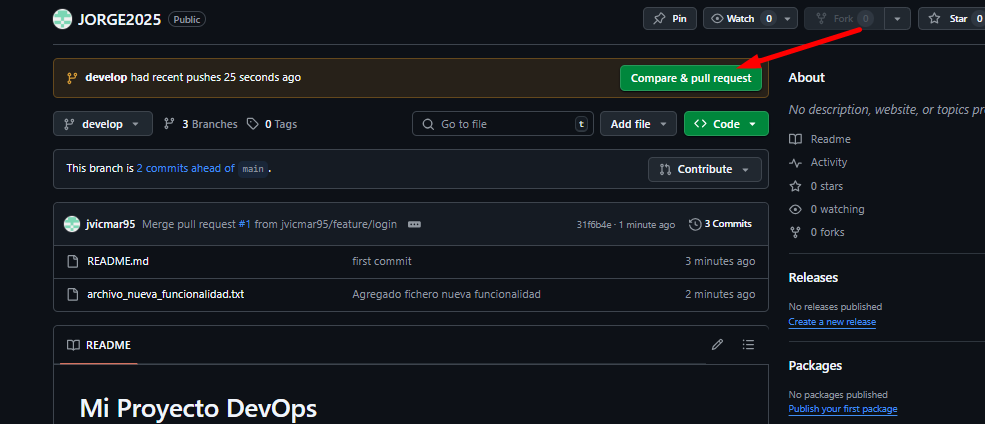
**Paso 9: Comprobación**

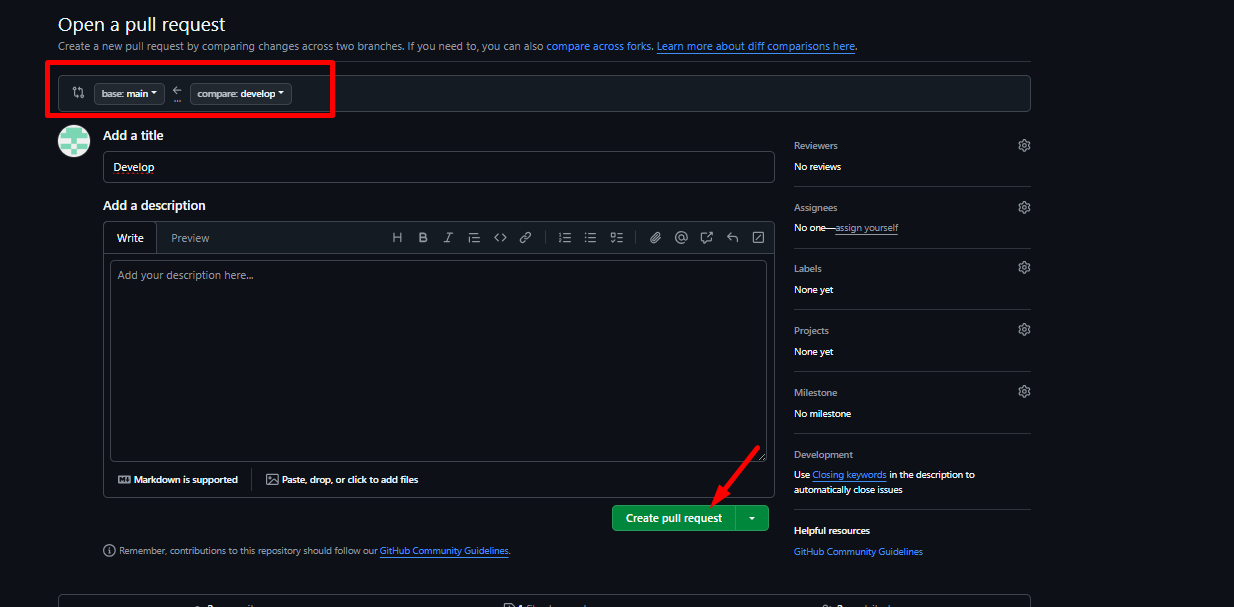


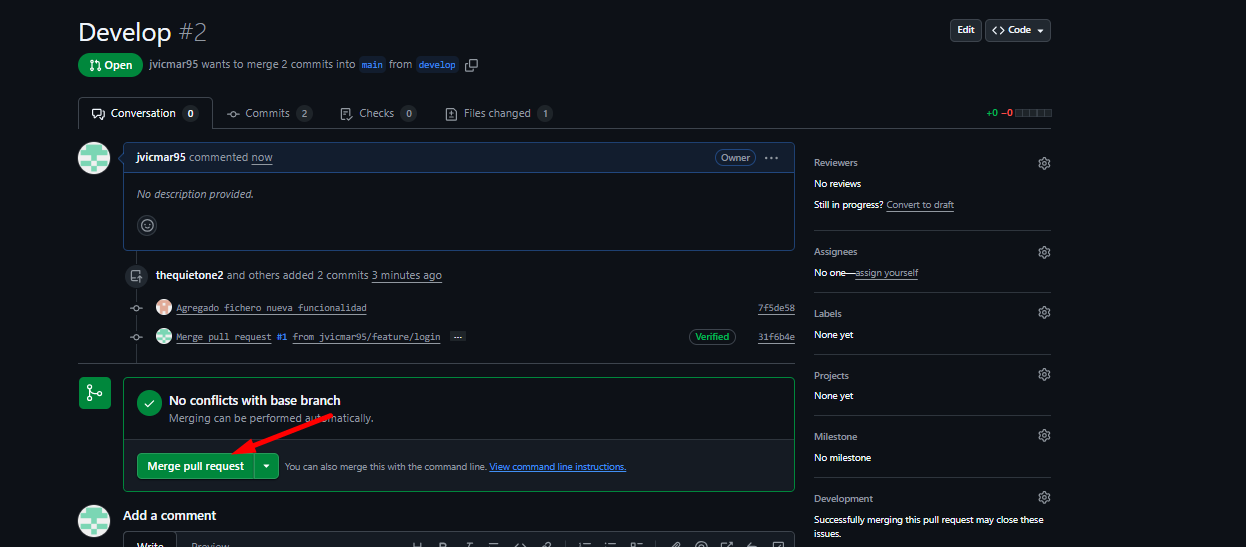


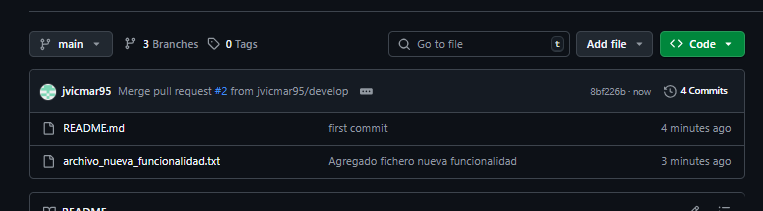
Ahora probaríamos que la nueva funcionalidad esta bien en develop y procedemos a pasarlo a la rama main

**Paso 10: Hacer pull request y merge desde develop a main**









**Paso 11: Borrar rama nueva funcionalidad**

*git checkout main*

*git branch -d feature/login*

*git push origin --delete feature/login*

**Paso 12: Actualizar localmente**

Tras haber hecho el merge en developer y el main nos traemos los cambios a nuestro local

*git checkout develop*

*git pull origin develop*

*git checkout main*

*git pull origin main*