

Práctico 2: Git y GitHub

Objetivo:

El estudiante desarrollará competencias para trabajar con Git y GitHub, aplicando conceptos fundamentales de control de versiones, colaboración en proyectos y resolución de conflictos, en un entorno simulado y guiado.

Resultados de aprendizaje:

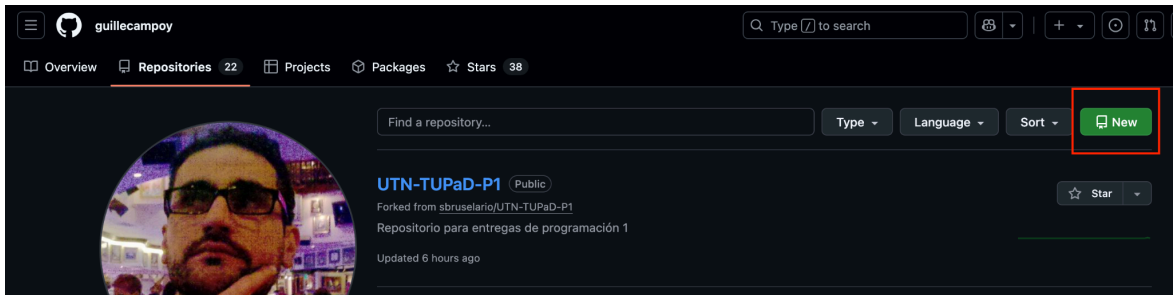
1. Comprender los conceptos básicos de Git y GitHub: Identificar y explicar los principales términos y procesos asociados con Git y GitHub, como repositorios, ramas, commits, forks, etiquetas y repositorios remotos.
2. Manejar comandos esenciales de Git: Ejecutar comandos básicos para crear, modificar, fusionar y gestionar ramas, commits y repositorios, tanto en local como en remoto.
3. Aplicar técnicas de colaboración en GitHub: Configurar y utilizar repositorios remotos, realizar forks, y gestionar pull requests para facilitar el trabajo colaborativo.
4. Resolver conflictos en un entorno de control de versiones: Identificar, analizar y solucionar conflictos de merge generados en un flujo de trabajo con múltiples ramas.

Actividades

- 1) Contestar las siguientes preguntas utilizando las guías y documentación proporcionada (Desarrollar las respuestas) :
 - ¿Qué es GitHub?
Es un repositorio de código, donde uno puede guardar sus proyectos, usa git, como sistema de control de versiones. Permite compartir el código con otras personas.

- ¿Cómo crear un repositorio en GitHub?

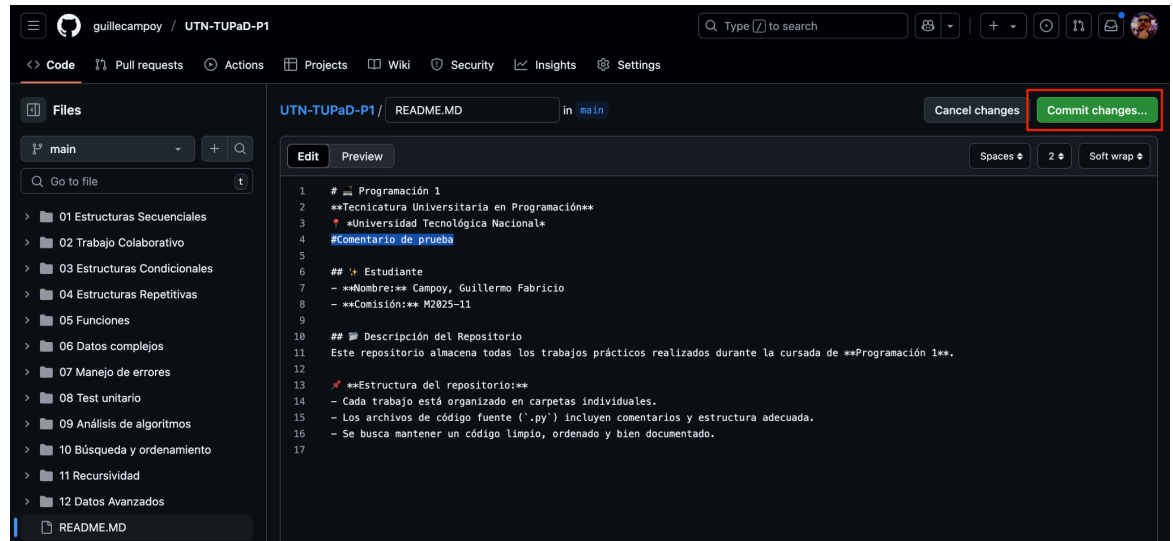
Primeramente hay que tener una cuenta en el sitio, luego desde nuestro perfil, seleccionamos la opción de repositorios y luego nuevo (o New), proporcionamos los datos de nombre, descripción, si será un repositorio público o privado, indicar si crea un archivo readme, seleccionamos el tipo de lenguaje para que configure un archivo .gitignore por defecto y finalmente el tipo de licencia.



- ¿Cómo crear una rama en Git?
Parados en la carpeta donde se encuentra el proyecto, escribimos en línea de comandos
git branch <nombre_branch>
- ¿Cómo cambiar a una rama en Git?
Parados en la carpeta donde está el proyecto, escribimos en línea de comandos
git checkout <nombre_branch>
- ¿Cómo fusionar ramas en Git?
Parados en la carpeta donde está el proyecto, escribimos en línea de comandos
git merge <nombre_rama>, lo que ocasiona que realiza una fusión o merge con la rama actual en la que estamos trabajando con la rama <nombre_rama>
- ¿Cómo crear un commit en Git?
Para crear un commit, primero se debe utilizar el comando add, de algún archivo que contenga cambios, luego ejecutamos el comando git commit, que confirma los cambios (snapshot o instantánea del trabajo en ese momento), se requiere indicar un comentario, se puede utilizar el comando **git commit -m "mensaje"**, se puede combinar el de la siguiente forma **git commit -am "mensaje"**, este último comando agrega todos los archivos modificados, y confirma con el mensaje indicado.

- ¿Cómo enviar un commit a GitHub?

Podemos utilizar directamente el editor de texto online de github, y nos permite revisar y enviar los cambios correspondientes



- ¿Qué es un repositorio remoto?

Un repositorio remoto, es una copia de nuestro proyecto local, con sus propias ramas, podemos tener ramas locales como remotas (pueden o no tener el mismo nombre) y nos permite compartir nuestro trabajo y colaborar en un mismo proyecto con diferentes personas

- ¿Cómo agregar un repositorio remoto a Git?

Se ejecuta el comando **git clone <repositorio_remoto>** hay varias formas en las cuales se puede utilizar, que dependen de las capacidades o funcionalidades del sitio donde esté alojado el repositorio, en el caso de github tenemos tres formas, utilizando **HTTPS git clone <url_proyecto>** requiere usar usuario y contraseña, con **SSH** para lo que se necesita vincular la cuenta con las Keys locales o bien utilizar el **cliente oficial de github** (previamente instalado y configurado)

- ¿Cómo empujar cambios a un repositorio remoto?

En este caso, asumiendo que ya tenemos vinculado nuestro repositorio local con github, se ejecuta el comando **git push origin <nombre_de_rama>**, previamente agregando los cambios con el comando **add** y ejecutando luego el comando **commit**, combinando ambas instrucciones en una sola línea **git commit -am "mensaje"**

- ¿Cómo tirar de cambios de un repositorio remoto?

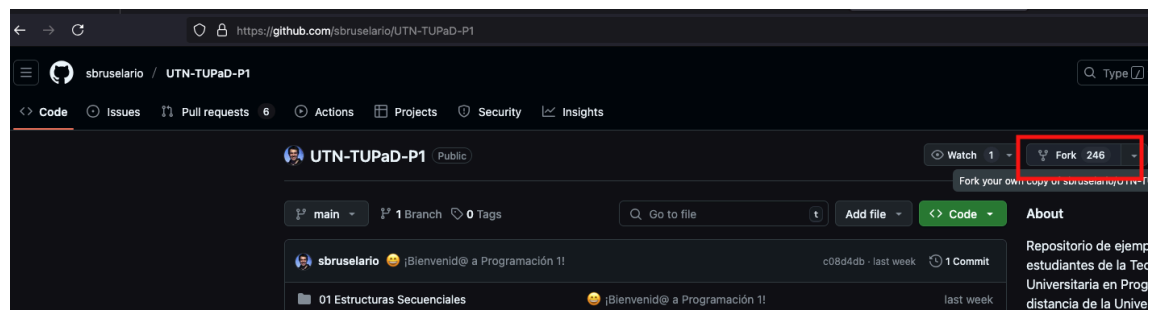
Para ello se utiliza el comando **pull**, parado en el repositorio local, puedo ejecutar por ejemplo **git pull origin main**, donde origin es el nombre asignado al repositorio remoto y main el nombre de la rama remota.

- ¿Qué es un fork de repositorio?

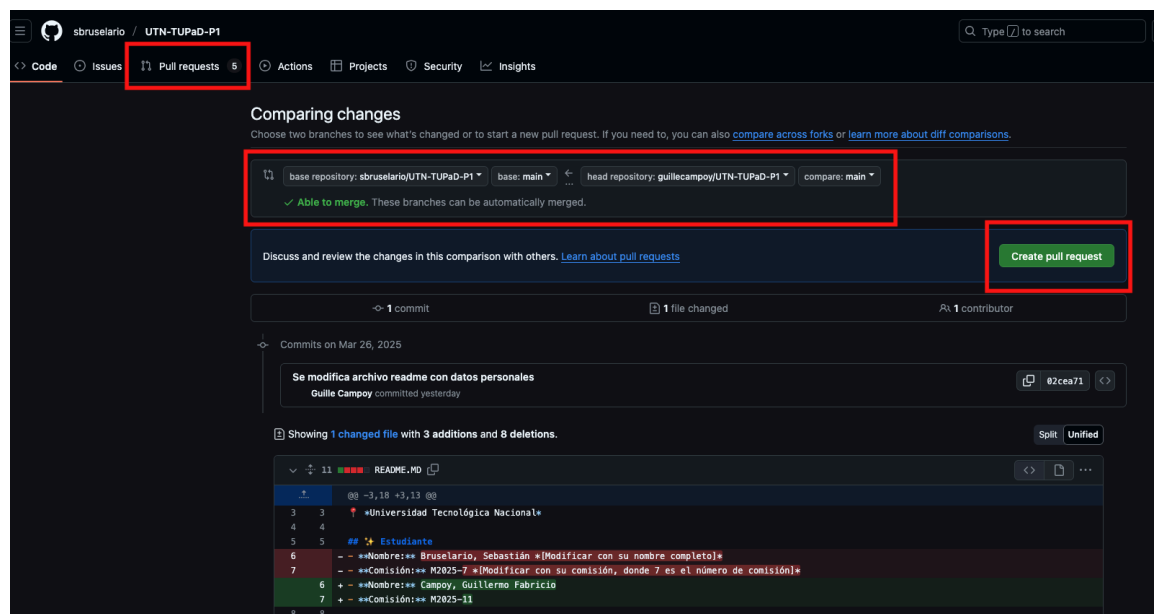
Un fork es una copia de un repositorio remoto, alojado en nuestro repositorio personal, por ejemplo el fork que hemos realizado en nuestro espacio personal de github sobre el proyecto base con la estructura propuesta por los docentes. Nos permite incluso tener sincronizado nuestro fork con el proyecto original, al momento de realizar un fork podemos incluso solo traer la rama principal o todas las ramas que tenga.

- ¿Cómo crear un fork de un repositorio?

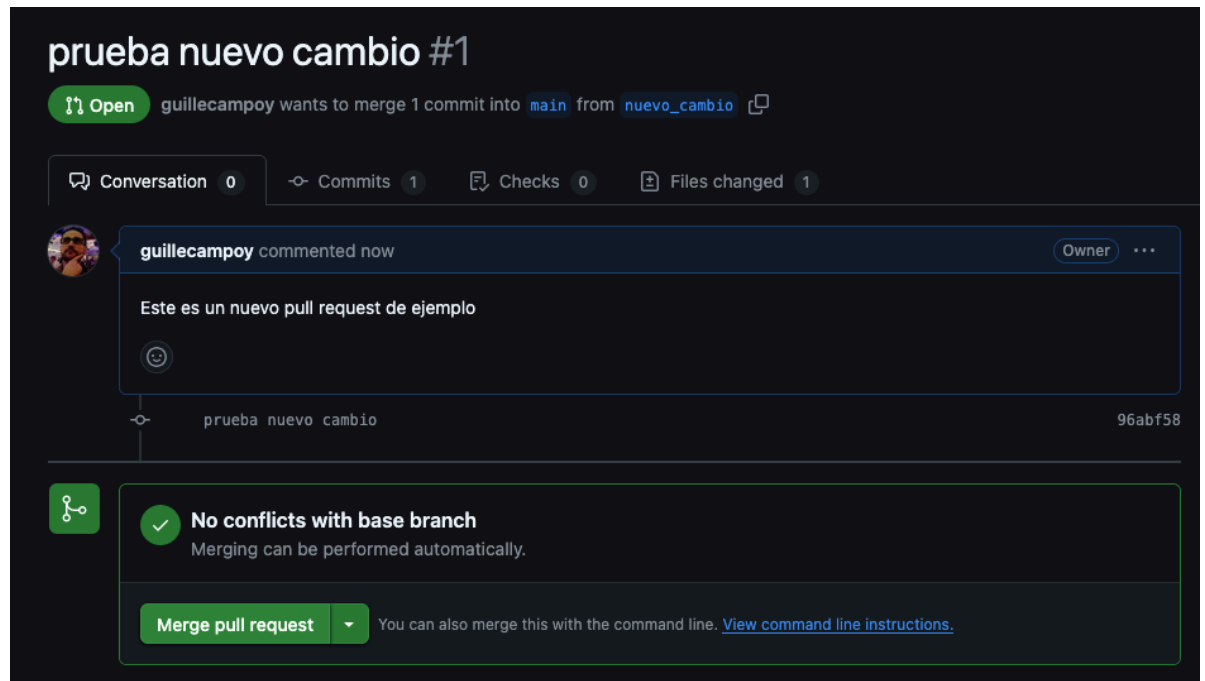
Voy a explicarlo asumiendo que estamos en github, nos posicionamos en el repositorio original o sobre el que queremos realizar un fork y seleccionamos el botón a la derecha “fork”, esto nos redirige a una pantalla donde nos indica en qué espacio queremos guardar el fork, en este caso es nuestra cuenta personal, pero puede ser otro tipo de espacio, ejemplo una cuenta de la empresa.



- ¿Cómo enviar una solicitud de extracción (pull request) a un repositorio?
Desde la interfaz visual de github por ejemplo podemos hacer un pull request en este caso, al repositorio original desde el propio sitio, por ejemplo si nuestro fork contiene cambios, en este caso mi rama personal 'main' tiene cambios respecto a la rama main del proyecto base. se pueden revisar los mismos y enviar la petición al repositorio original.
También puedo localmente trabajar en una rama por ejemplo **origin/nueva_funcionalidad** hacer un push de la rama con los cambios a github y luego puedo realizar un pull request de **origin/nueva_funcionalidad** a **origin/main** para integrar la nueva funcionalidad a la rama principal.
Adjunto captura de caso 1 con la ui de github

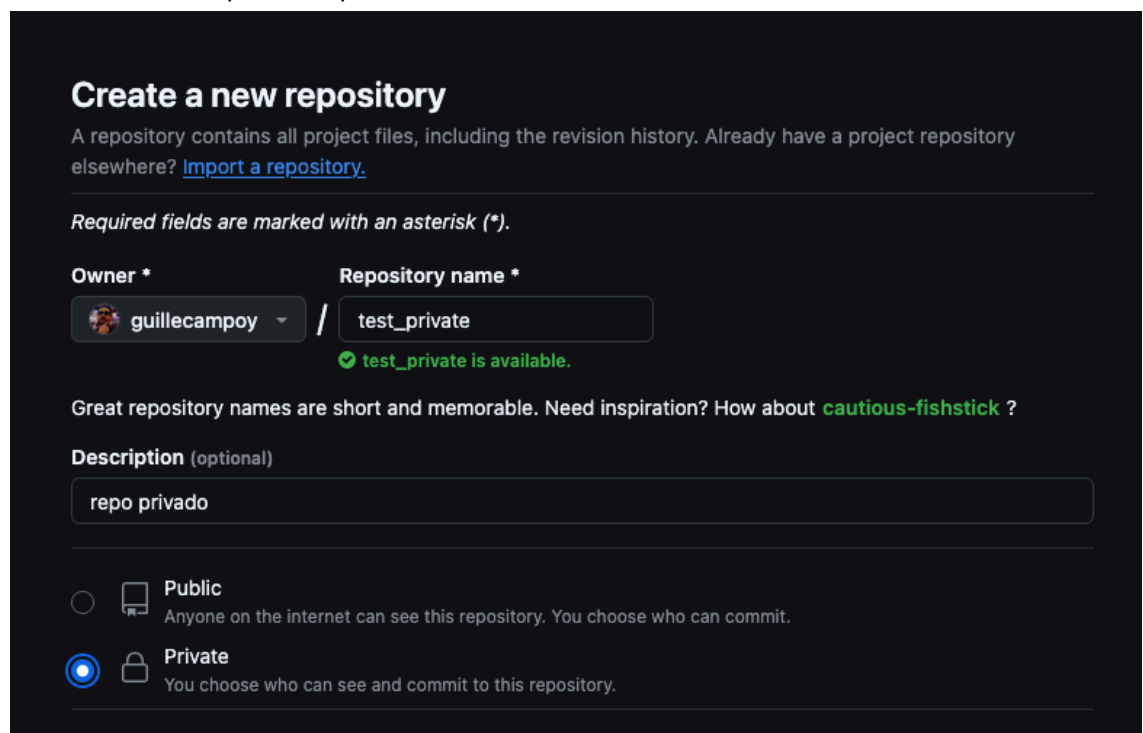


- ¿Cómo aceptar una solicitud de extracción?
Para demostrar el ejemplo cree una nueva rama en mi repositorio local llamada `nuevo_cambio`, agregue alguna modificación y realice un `push` al repositorio remoto con la nueva rama, luego desde la UI de github abra un nuevo pull request con los cambios para revisar y aceptar los mismos en la rama `main`. Confirmamos en el botón `Merge pull request`.



- ¿Qué es una etiqueta en Git?
Son referencias a un punto específico en la historia de un repositorio, se pueden usar por ejemplo para marcar releases o cambios mayores en nuestro código y así tenerlo ordenado.
- ¿Cómo crear una etiqueta en Git?
hay dos tipos de etiquetas, notada o ligeras, la diferencia entre la segunda y la primera es que la primera es más completa y contiene información, como descripción y datos del autor
 - Ligera : `git tag nombre-etiqueta`
 - Anotada: `git tag -a nombre-etiqueta -m "Mensaje"`
- ¿Cómo enviar una etiqueta a GitHub?
Para hacerlo se utiliza esta línea de comandos: **`git push origin nombre-etiqueta`**
- ¿Qué es un historial de Git?
Es la secuencia de cambios registrados en un repositorio
- ¿Cómo ver el historial de Git?
el comando es **`git log`**

- ¿Cómo buscar en el historial de Git?
Si lo que buscamos es algún commit específico podemos utilizar el comando **git log --grep="mensaje buscado"**
- ¿Cómo borrar el historial de Git?
Si no hemos subido los cambios al repositorio remoto podemos hacer con el siguiente comando **git reset --hard HEAD~n** siendo n al número de commits locales hacia atrás que quiero eliminar
- ¿Qué es un repositorio privado en GitHub?
Indica que solo las personas a las que se le garantice el acceso lo podrán ver y trabajar con el
- ¿Cómo crear un repositorio privado en GitHub?



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk ().*

Owner * guillecampoy / **Repository name *** test_private

✔ test_private is available.

Great repository names are short and memorable. Need inspiration? How about **cautious-fishstick** ?

Description (optional)

repo privado

☐ **Public**
Anyone on the internet can see this repository. You choose who can commit.

☒ **Private**
You choose who can see and commit to this repository.

- ¿Cómo invitar a alguien a un repositorio privado en GitHub?
Desde el repositorio privado se pueden agregar colaboradores al proyecto mediante su usuario o su correo que registro en github.
- ¿Qué es un repositorio público en GitHub?
Quiere decir que el mismo puede ser accedido por cualquier persona, se puede clonar, realizar fork, ver historial de cambios

**TECNICATURA UNIVERSITARIA
EN PROGRAMACIÓN
A DISTANCIA**

- ¿Cómo crear un repositorio público en GitHub?
Desde nuestro perfil → repositorios → seleccionamos new (o nuevo) y podemos especificar algunos datos básicos como si es público o privado, licencias, etc.
- ¿Cómo compartir un repositorio público en GitHub?
Simplemente nos paramos en el sitio del repositorio → botón verde <> Code, se despliega y ofrece diferentes métodos.

2) Realizar la siguiente actividad:

- Crear un repositorio.
 - Dale un nombre al repositorio.
https://github.com/guillecampoy/test_basicos
 - Elige que el repositorio sea público.
 - Inicializa el repositorio con un archivo.
- Agregando un Archivo
 - Crea un archivo simple, por ejemplo, "mi-archivo.txt".
 - Realiza los comandos `git add .` y `git commit -m "Agregando mi-archivo.txt"` en la línea de comandos.
 - Sube los cambios al repositorio en GitHub con `git push origin main` (o el nombre de la rama correspondiente).
- Creando Branchs
 - Crear una Branch
 - Realizar cambios o agregar un archivo
 - Subir la Branch
 - https://github.com/guillecampoy/test_basicos/tree/feature-basics

3) Realizar la siguiente actividad:

Paso 1: Crear un repositorio en GitHub

- Ve a GitHub e inicia sesión en tu cuenta.
- Haz clic en el botón "New" o "Create repository" para crear un nuevo repositorio.
- Asigna un nombre al repositorio, por ejemplo, conflict-exercise. • Opcionalmente, añade una descripción.
- Marca la opción "Initialize this repository with a README".
- Haz clic en "Create repository".
- <https://github.com/guillecampoy/conflict-exercise>

Paso 2: Clonar el repositorio a tu máquina local

- Copia la URL del repositorio (usualmente algo como `https://github.com/tuusuario/conflict-exercise.git`).
- Abre la terminal o línea de comandos en tu máquina.
- Clona el repositorio usando el comando:
`git clone https://github.com/tuusuario/conflict-exercise.git`
- Entra en el directorio del repositorio:
- `cd conflict-exercise`

Paso 3: Crear una nueva rama y editar un archivo

- Crea una nueva rama llamada `feature-branch`:
`git checkout -b feature-branch`
- Abre el archivo `README.md` en un editor de texto y añade una línea nueva, por ejemplo:
Este es un cambio en la feature branch.
Guarda los cambios y haz un commit:
`git add README.md`
`git commit -m "Added a line in feature-branch"`

Paso 4: Volver a la rama principal y editar el mismo archivo

- Cambia de vuelta a la rama principal (`main`):
`git checkout main`
- Edita el archivo `README.md` de nuevo, añadiendo una línea diferente: Este es un cambio en la main branch.
- Guarda los cambios y haz un commit:
`git add README.md`
`git commit -m "Added a line in main branch"`

Paso 5: Hacer un merge y generar un conflicto

- Intenta hacer un merge de la `feature-branch` en la rama `main`: `git merge feature-branch`
- Se generará un conflicto porque ambos cambios afectan la misma línea del archivo `README.md`.

TECNICATURA UNIVERSITARIA EN PROGRAMACIÓN A DISTANCIA

Paso 6: Resolver el conflicto

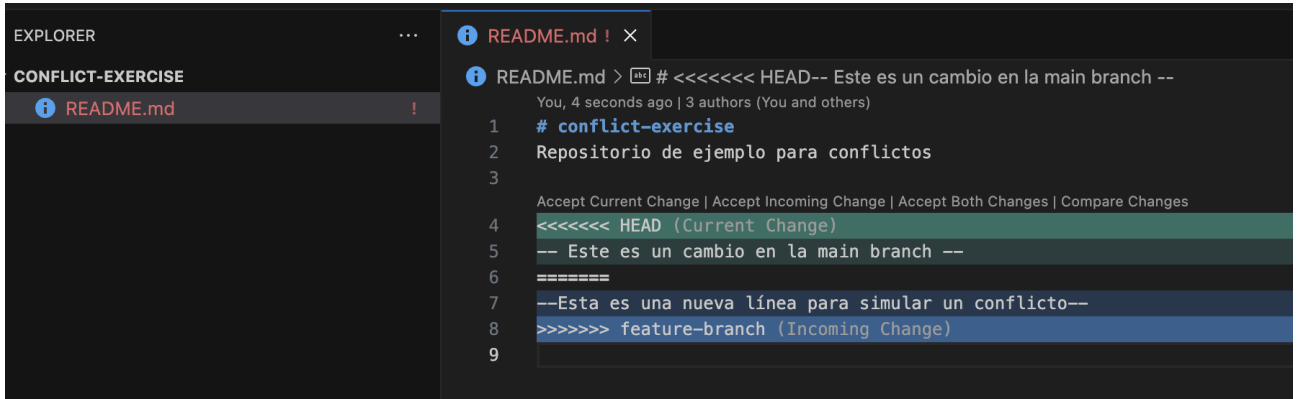
- Abre el archivo README.md en tu editor de texto. Verás algo similar a esto: <<<<<<< HEAD

Este es un cambio en la main branch.

=====

Este es un cambio en la feature branch.

>>>>>> feature-branch



```
EXPLORER
CONFLICT-EXERCISE
  README.md

README.md
1 # <<<<<<< HEAD-- Este es un cambio en la main branch --
2 You, 4 seconds ago | 3 authors (You and others)
3 # conflict-exercise
4 Repositorio de ejemplo para conflictos
5
6 Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
7 <<<<<<< HEAD (Current Change)
8 -- Este es un cambio en la main branch --
9 =====
10 --Esta es una nueva línea para simular un conflicto--
11 >>>>>> feature-branch (Incoming Change)
```

- Decide cómo resolver el conflicto. Puedes mantener ambos cambios, elegir uno de ellos, o fusionar los contenidos de alguna manera.
- Edita el archivo para resolver el conflicto y guarda los cambios(Se debe borrar lo marcado en verde en el archivo donde estes solucionando el conflicto. Y se debe borrar la parte del texto que no se quiera dejar).
- Añade el archivo resuelto y completa el merge:
git add README.md
git commit -m "Resolved merge conflict"

Paso 7: Subir los cambios a GitHub

- Sube los cambios de la rama main al repositorio remoto en GitHub: git push origin main
- También sube la feature-branch si deseas:

git push origin feature-branch

<https://github.com/guillecampoy/conflict-exercise>

Paso 8: Verificar en GitHub

- Ve a tu repositorio en GitHub y revisa el archivo README.md para confirmar que los cambios se han subido correctamente.
- Puedes revisar el historial de commits para ver el conflicto y su resolución.

<https://github.com/guillecampoy/conflict-exercise/commits/main/>