

PROGRAMACIÓN I

Trabajo Práctico N.º 2: Git y GitHub

Nombre y Apellido: Pamela Zampieri

Comisión: 23

1) Contestar las siguientes preguntas utilizando las guías y documentación proporcionada (Desarrollar las respuestas):

1) ¿Qué es GitHub?

GitHub es una plataforma que ofrece alojamiento de repositorios de control de versiones, que permite a los desarrolladores almacenar y gestionar sus proyectos de software.

Es gratuita y de código abierto, sirve para almacenar nuestros proyectos online, permite el trabajo en colaboración con otros desarrolladores, el intercambio de código y el trabajo conjunto de forma eficiente.

2) ¿Cómo crear un repositorio en GitHub?

Hay que iniciar sesión en <https://github.com> (si no tenemos cuenta hay que crear una).

Seleccionar "New repository".

Completamos el formulario:

- Repository name: Elegimos un nombre (ej: mi-primer-repositorio)
- Description (esto es opcional): Una breve explicación de nuestro proyecto.
- Visibility: Elegir la visibilidad que queremos que tenga nuestro proyecto: puede ser público o privado.
- Se puede agregar también un README (recomendado si queremos que tenga una descripción inicial). Un .gitignore (para excluir ciertos archivos, dependiendo del lenguaje). O elegir una licencia (opcional, si es open source).
- Hacer clic en "Create repository".

3) ¿Cómo crear una rama en Git?

Para crear una nueva rama, usamos el comando: git branch.

\$ git branch nombreRama

4) ¿Cómo cambiar a una rama en Git?

Para cambiar a una rama en Git, usamos el comando: git checkout.

\$ git checkout nombreRama

5) ¿Cómo fusionar ramas en Git?

Una vez en la rama de destino, se usa el comando git merge para fusionar la rama de origen en la rama actual:

\$ git merge nombreRama

6) ¿Cómo crear un commit en Git?

Primero debemos realizar los cambios en los archivos de nuestro proyecto. Luego agregamos los cambios al área de preparación. Esto se hace con el comando git add.

Podemos agregar archivos específicos o todos los archivos modificados:

\$ git add archivo1 (se agrega el archivo "archivo1")

o

\$git add . (se agregan todos los archivos)

Una vez que los cambios están en el área de preparación, se puede hacer el commit con el comando git commit.

Se puede proporcionar un mensaje de commit que describa los cambios realizados:

\$ git commit -m "Inicio proyecto"

Esto guarda el estado actual del código en el historial del repositorio con el mensaje correspondiente.

7) ¿Cómo enviar un commit a GitHub?

Para enviar un commit a GitHub, debemos tener conectado nuestro repositorio local con el repositorio remoto de GitHub. Una vez que hicimos cambios y los confirmamos con un commit, usamos el siguiente comando para subirlos:

\$ git push -u origin main (o nombre de rama a la cual queramos subir los cambios)

8) ¿Qué es un repositorio remoto?

Un repositorio remoto es una versión del repositorio que está alojada en Internet o en una red, como por ejemplo en GitHub, GitLab o Bitbucket, entre otros. Sirve para compartir el código con otros desarrolladores y trabajar en equipo desde distintos lugares.

Está vinculado al repositorio local y permite sincronizar los cambios entre ambos mediante comandos como:

- **git push** : para subir tus cambios locales al remoto.
- **git pull o git fetch** : para descargar los cambios del remoto a la pc.

9) ¿Cómo agregar un repositorio remoto a Git?

Para agregar un repositorio remoto a Git, tenemos que utilizar el siguiente comando:
\$git remote add origin <URL-del-repositorio>

10) ¿Cómo empujar cambios a un repositorio remoto?

Para empujar los cambios a un repositorio remoto se utiliza el siguiente comando:
\$git push origin nombreRama

11) ¿Cómo tirar de cambios de un repositorio remoto?

Para tirar los cambios del repositorio remoto se utiliza el comando git pull para descargar y fusionar los cambios del repositorio remoto con la rama local:
\$git pull origin nombreRama

12) ¿Qué es un fork de repositorio?

Un *fork* es una copia de un repositorio que se crea en nuestra cuenta de GitHub a partir de otro repositorio público. Sirve principalmente para:

- Probar cambios sin afectar el proyecto original.
- Contribuir a proyectos de otros (se hace un *fork*, trabajamos en nuestra copia y luego podemos enviar un *pull request*).
- Usar un proyecto como base para crear nuestra propia versión personalizada.

Es muy útil en proyectos colaborativos y de código abierto.

13) ¿Cómo crear un fork de un repositorio?

Para crear un fork de un repositorio tenemos que:

Ingresar al repositorio público en GitHub que queremos *forkear*.

Hacemos clic en el botón **"Fork"**.

Elegimos nuestra cuenta personal o una organización donde queremos crear la copia.

Al tener una copia del repositorio original en nuestra cuenta podemos modificarlo sin afectar el proyecto original.

Esta copia la podemos clonar en nuestra pc local con **git clone**, trabajar en ella, y luego, si queremos podemos aportar al proyecto original, enviando un *pull request*.

14) ¿Cómo enviar una solicitud de extracción (pull request) a un repositorio?

Para hacer un *pull request*, debemos dirigirnos a la pestaña **"Pull requests"** del repositorio remoto en GitHub. Allí hacemos clic en **"New pull request"**. Luego, seleccionamos la rama con nuestros cambios y presionamos **"Create pull request"**.

Se abrirá una ventana a modo de resumen en donde se reflejarán los cambios que hemos realizado. Podemos agregar un título y una descripción más extensa del detalle de los cambios. Además de agregar reviewers (si estamos trabajando en un proyecto grupal o colaborativo en nuestra empresa por ejemplo) que nos aprueben ese nuevo pr. que estamos creando.

15) ¿Cómo aceptar una solicitud de extracción?

El autor (o maintainer) del repositorio verá la solicitud de *pull request* en la pestaña correspondiente. Podrá revisar los cambios propuestos y, si lo considera adecuado y no hay conflictos con la rama principal (por ejemplo, **main**), podrá hacer clic en **"Merge pull request"** para aceptar y fusionar los cambios al repositorio.

También puede dejar comentarios o solicitar modificaciones antes de aceptarlo, facilitando así una revisión colaborativa.

16) ¿Qué es un etiqueta en Git?

Una etiqueta en Git (*tag*) es una referencia fija a un punto específico en el historial de commits, usualmente utilizada para marcar versiones importantes de un proyecto, como lanzamientos (ej. v1.0, v2.1.3, etc.).

17) ¿Cómo crear una etiqueta en Git?

Git utiliza dos tipos principales de etiquetas: ligeras y anotadas.

Una etiqueta ligera es muy parecida a una rama que no cambia. Las etiquetas anotadas se guardan en la base de datos de Git como objetos enteros. Tienen un checksum; contienen el nombre del etiquetador, correo electrónico y fecha; tienen un mensaje asociado; y pueden ser firmadas y verificadas con GNU Privacy Guard (GPG).

Normalmente se recomienda que creemos etiquetas anotadas, de manera que tengamos toda esta información, pero si queremos una etiqueta temporal o por alguna razón no estamos interesados en esa información, entonces podemos usar las etiquetas ligeras.

\$git tag v1.0 (crear una etiqueta ligera)

\$git tag -a v1.0 -m "Versión 1.0 estable" (crear una etiqueta anotada)

18) ¿Cómo enviar una etiqueta a GitHub?

Para enviar una etiqueta (tag) a GitHub desde nuestro repositorio local, podemos hacerlo con estos comandos:

\$git push origin nombre_de_la_etiqueta

\$git push origin --tags (esto sube todas las etiquetas que tengamos creadas localmente al repositorio remoto en GitHub.)

19) ¿Qué es un historial de Git?

El historial de Git es el registro completo de todos los cambios que se han hecho en un repositorio a lo largo del tiempo. Incluye:

- Commits: Cada uno representa una "foto" del proyecto en un momento específico.
- Autor: Quién hizo el cambio.
- Mensaje del commit: Qué se cambió y por qué.
- Fecha y hora del cambio.
- Hash (ID único) de cada commit.

20) ¿Cómo ver el historial de Git?

Podemos ver el historial de Git usando:

\$git log (Esto muestra la lista de commits con toda su información.)

\$git log --oneline (Esto nos muestra solo los ID y mensajes de los commits, ideal para una vista rápida.)

21) ¿Cómo buscar en el historial de Git?

Para buscar en el historial de commits de Git, podemos utilizar varios comandos y opciones que nos permiten filtrar y localizar commits específicos.

- Para buscar commits que contengan una palabra o frase específica en el mensaje de commit, usamos git log con la opción **-grep**:

\$git log --grep="palabra clave"

- Para buscar commits que han modificado un archivo específico, usamos git log seguido del nombre del archivo:

\$git log -- nombre_del_archivo

- Para buscar commits en un rango de fechas específico, usamos las opciones --since y --until:

\$git log --since="2024-01-01" --until="2024-01-31"

Para encontrar commits hechos por un autor específico, usamos --author:

\$git log --author="Nombre del Autor"

22) ¿Cómo borrar el historial de Git?

Borrar el historial de Git es una acción que debe hacerse con cuidado, porque afecta todo el repositorio y puede romper cosas si estamos trabajando en equipo. Pero si realmente queremos empezar de cero y borrar todo el historial:

\$git checkout --orphan nuevaRama

\$git add .

\$git commit -m "Primer commit nuevaRama"

\$git branch -D main (eliminamos la rama anterior)

\$git branch -m main (renombramos la nueva rama con el nombre por ejemplo main)

\$git push -f origin main (forzamos el push al repositorio remoto)

Otra alternativa más segura es el comando **git reset**. Este se utiliza sobre todo para deshacer las cosas. Se mueve alrededor del puntero HEAD y opcionalmente cambia el index o área de ensayo y también puede cambiar opcionalmente el directorio de trabajo si se utiliza - hard. Esta última opción hace posible que este comando pueda perder tu trabajo si se usa incorrectamente, por lo que tenemos que asegurarnos de entenderlo antes de usarlo.

Existen distintas formas de utilizarlo:

\$git reset (Quita del stage todos los archivos y carpetas del proyecto.)

\$git reset nombreArchivo (Quita del stage el archivo indicado.)

\$git reset nombreCarpeta/ (Quita del stage todos los archivos de esa carpeta.)

\$ git reset nombreCarpeta/nombreArchivo (Quita ese archivo del stage que a la vez está dentro de una carpeta).

\$ git reset nombreCarpeta/*.*extensión (Quita todos los archivos que cumplan con la condición indicada previamente dentro de esa carpeta del stage.)

23) ¿Qué es un repositorio privado en GitHub?

Un repositorio privado en GitHub es un repositorio cuyo contenido no está disponible públicamente. Solo pueden acceder a él:

- El propietario del repositorio.
- Los colaboradores a los que se les haya dado permiso explícito.

24) ¿Cómo crear un repositorio privado en GitHub?

Hay que iniciar sesión en <https://github.com> (si no tenemos cuenta hay que crear una).

Seleccionar "New repository".

Completamos el formulario:

- Repository name: Elegimos un nombre (ej: mi-primer-repositorio)
- Description (esto es opcional): Una breve explicación de nuestro proyecto.
- Visibility: Elegir visibilidad **PRIVATE (privado)**.
- Se puede agregar también un README (recomendado si queremos que tenga una descripción inicial). Un .gitignore (para excluir ciertos archivos, dependiendo del lenguaje). O elegir una licencia (opcional, si es open source).
- Hacer clic en "Create repository".

25) ¿Cómo invitar a alguien a un repositorio privado en GitHub?

Para invitar a alguien a un repositorio privado debemos:

- Entrar al repositorio en GitHub.
- Hacer clic en la pestaña "Settings".
- En el menú lateral, seleccionar "Collaborators" (o "Manage access").
- Hacer clic en el botón "Add people".
- Ahí escribimos el nombre de usuario o email de GitHub de la persona que queremos invitar.
- Elegimos el nivel de acceso (por ejemplo: *read*, *write*, *admin*).
- Hacemos clic en "Add".

El usuario invitado recibirá una notificación por mail o en GitHub. Deberá aceptar la invitación. Cuando la acepte, podrá ver y trabajar en el repositorio según los permisos dados.

26) ¿Qué es un repositorio público en GitHub?

Un repositorio público en GitHub es un repositorio cuyo contenido es accesible a cualquier persona. A diferencia de un repositorio privado, que está restringido a un

grupo específico de colaboradores, un repositorio público permite que cualquier persona pueda ver, clonar y, si tienen los permisos adecuados, contribuir al proyecto.

27) ¿Cómo crear un repositorio público en GitHub?

Hay que iniciar sesión en <https://github.com> (si no tenemos cuenta hay que crear una).

Seleccionar "New repository".

Completamos el formulario:

- Repository name: Elegimos un nombre (ej: mi-primer-repositorio)
- Description (esto es opcional): Una breve explicación de nuestro proyecto.
- Visibility: Elegir visibilidad **PUBLIC (público)**.
- Se puede agregar también un README (recomendado si queremos que tenga una descripción inicial). Un .gitignore (para excluir ciertos archivos, dependiendo del lenguaje). O elegir una licencia (opcional, si es open source).
- Hacer clic en "Create repository".

28) ¿Cómo compartir un repositorio público en GitHub?

Para compartir un repositorio público debemos:

- Iniciar sesión en <https://github.com>
- Ingresar a nuestro repositorio en GitHub
- Copiamos la URL del repositorio
- Hacemos clic en `<> Code` y nos va a mostrar una URL como esta:
<https://github.com/usuario/nombre-del-repo.git>
- La tenemos que compartir como cualquier link: en un mail, mensaje, documento, chat, etc. Cualquier persona con ese enlace podrá entrar, ver el código, descargarlo o hacer un fork.

2) Realizar la siguiente actividad:

A. Crear un repositorio:


- Dale un nombre al repositorio.
- Elige que el repositorio sea público.

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 *

 PamesZampieri

Repository name *

TP2-Ej2-Zampieri-Pame

✓ TP2-Ej2-Zampieri-Pamela is available.

Great repository names are short and memorable. Need inspiration? How about [cautious-fishstick](#) ?

Description (optional)

Este repositorio responde a la consigna 2 del TP2 de Programación I: Git y GitHub.



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.

Initialize this repository with:



Add a README file

This is where you can write a long description for your project. [Learn more about READMEs](#).

Add .gitignore

.gitignore template: None

Choose which files not to track from a list of templates. [Learn more about ignoring files](#).

Choose a license

License: None

A license tells others what they can and can't do with your code. [Learn more about licenses](#).

This will set `main` as the default branch. Change the default name in your [settings](#).

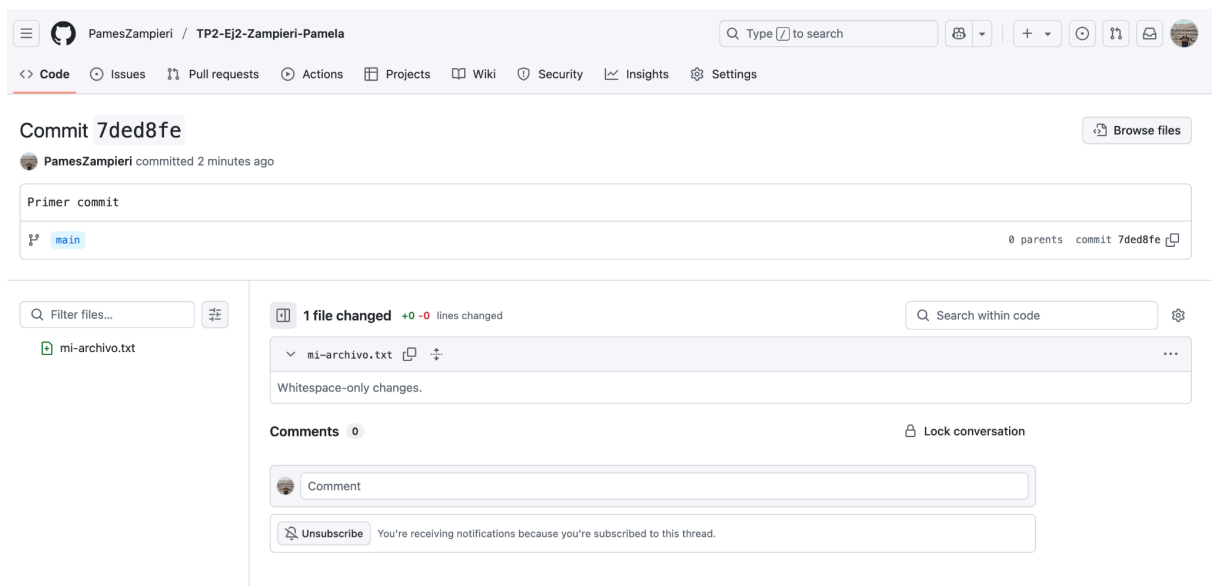
① You are creating a public repository in your personal account.

- Inicializa el repositorio con un archivo.

B. 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).

```
mi-archivo.txt x
mi-archivo.txt
1
PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS
pamezampieri@MBP-de-Pamela TP2-ProgramaciónI-ZampieriPamela % git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint: git config --global init.defaultBranch <name>
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint: git branch -m <name>
Initialized empty Git repository in /Users/pamezampieri/Desktop/TP2-ProgramaciónI-ZampieriPamela/.git/
pamezampieri@MBP-de-Pamela TP2-ProgramaciónI-ZampieriPamela % git branch -M main
pamezampieri@MBP-de-Pamela TP2-ProgramaciónI-ZampieriPamela % git add .
pamezampieri@MBP-de-Pamela TP2-ProgramaciónI-ZampieriPamela % git commit -m "Primer commit"
[main (root-commit) 7ded8fe] Primer commit
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 mi-archivo.txt
pamezampieri@MBP-de-Pamela TP2-ProgramaciónI-ZampieriPamela % git remote add origin https://github.com/PamesZampieri/TP2-Ej2-Zampieri-Pamela.git
pamezampieri@MBP-de-Pamela TP2-ProgramaciónI-ZampieriPamela % git push -u origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 240 bytes | 240.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/PamesZampieri/TP2-Ej2-Zampieri-Pamela.git
 * [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
pamezampieri@MBP-de-Pamela TP2-ProgramaciónI-ZampieriPamela %
```



C. Creando Branchs

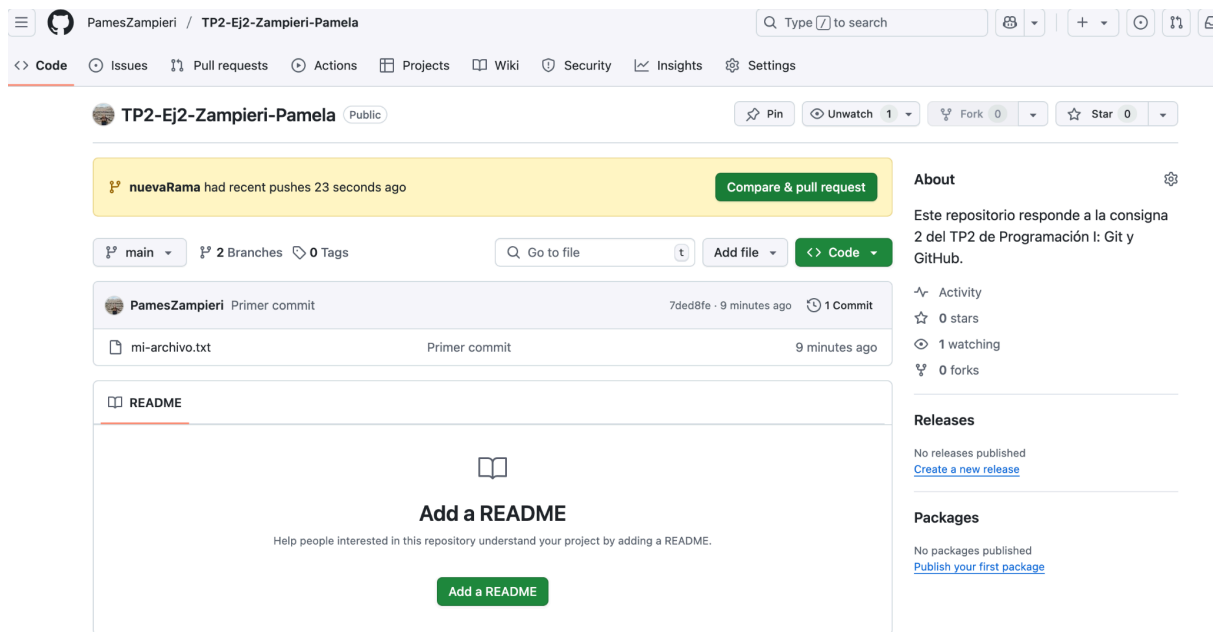
- Crear una Branch
- Realizar cambios o agregar un archivo
- Subir la Branch

```
pamezampieri@MBP-de-Pamela TP2-ProgramaciónI-ZampieriPamela % git branch
* main
pamezampieri@MBP-de-Pamela TP2-ProgramaciónI-ZampieriPamela % git checkout -b nuevaRama
Switched to a new branch 'nuevaRama'
pamezampieri@MBP-de-Pamela TP2-ProgramaciónI-ZampieriPamela % git branch
main
* nuevaRama
pamezampieri@MBP-de-Pamela TP2-ProgramaciónI-ZampieriPamela % git status
On branch nuevaRama
nothing to commit, working tree clean
pamezampieri@MBP-de-Pamela TP2-ProgramaciónI-ZampieriPamela % git status
On branch nuevaRama
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   mi-archivo.txt

no changes added to commit (use "git add" and/or "git commit -a")
pamezampieri@MBP-de-Pamela TP2-ProgramaciónI-ZampieriPamela % git add .
pamezampieri@MBP-de-Pamela TP2-ProgramaciónI-ZampieriPamela % git commit -m "Primer commit en nuevaRama"
[nuevaRama cececd6] Primer commit en nuevaRama
1 file changed, 1 insertion(+)
pamezampieri@MBP-de-Pamela TP2-ProgramaciónI-ZampieriPamela % git push
fatal: The current branch nuevaRama has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream origin nuevaRama

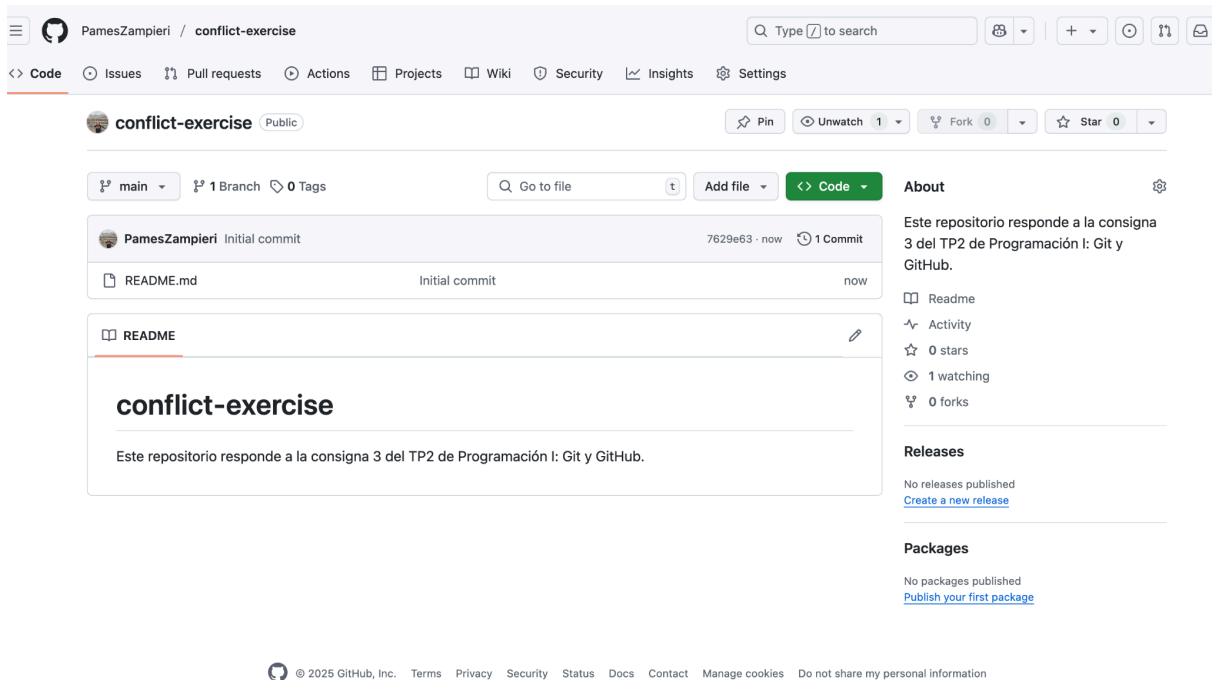
pamezampieri@MBP-de-Pamela TP2-ProgramaciónI-ZampieriPamela % git push origin nuevaRama
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Writing objects: 100% (3/3), 293 bytes | 293.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'nuevaRama' on GitHub by visiting:
remote:   https://github.com/PamesZampieri/TP2-Ej2-Zampieri-Pamela/pull/new/nuevaRama
remote:
To https://github.com/PamesZampieri/TP2-Ej2-Zampieri-Pamela.git
 * [new branch]   nuevaRama -> nuevaRama
pamezampieri@MBP-de-Pamela TP2-ProgramaciónI-ZampieriPamela %
```



3) Realizar la siguiente actividad:

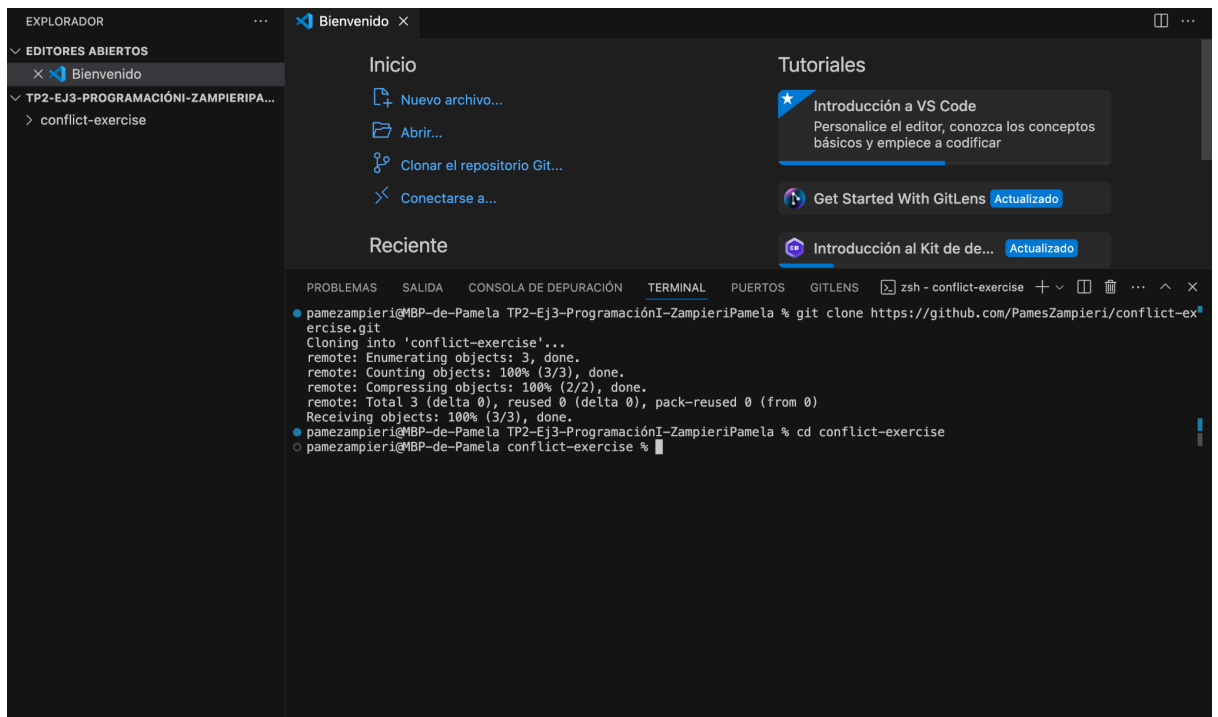
A. Paso 1: Crear un repositorio en GitHub

- 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".



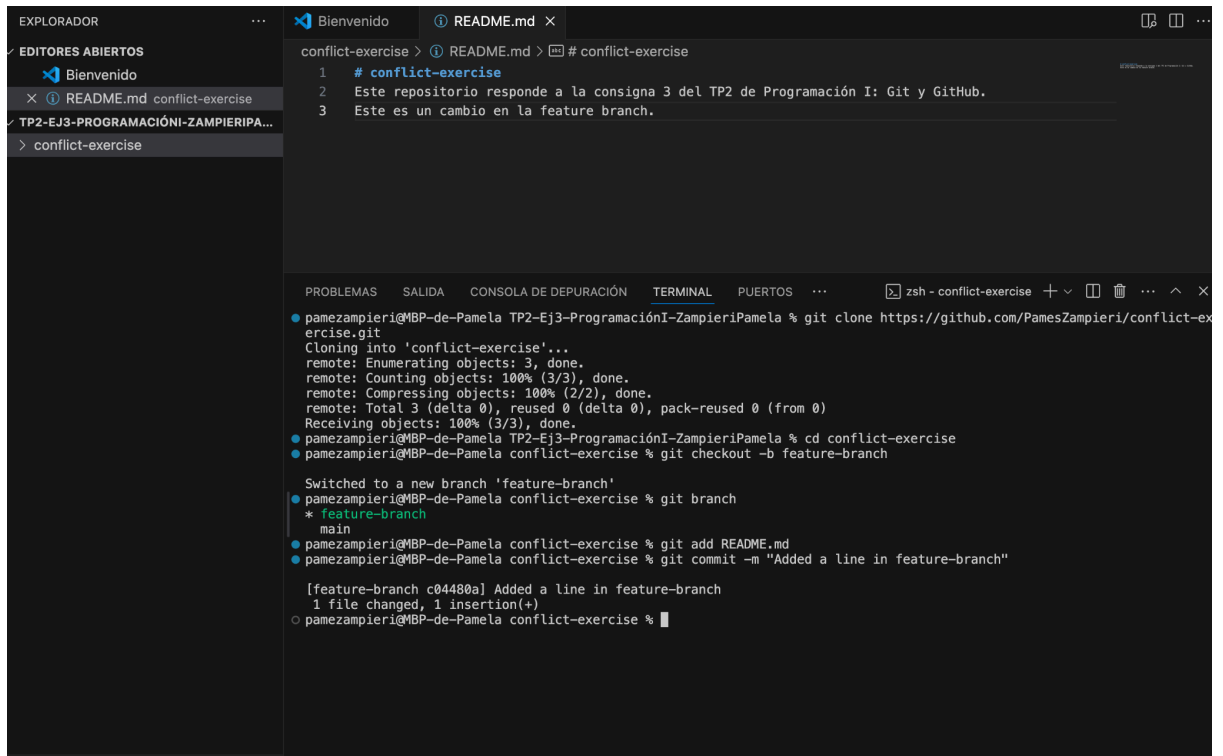
B. Paso 2: Clonar el repositorio a tu máquina local

- 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`



C. 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"`



```
conflict-exercise > ① README.md > ② # conflict-exercise
1  # conflict-exercise
2  Este repositorio responde a la consigna 3 del TP2 de Programación I: Git y GitHub.
3  Este es un cambio en la feature branch.

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS ... zsh - conflict-exercise + - □ ... ^ ×

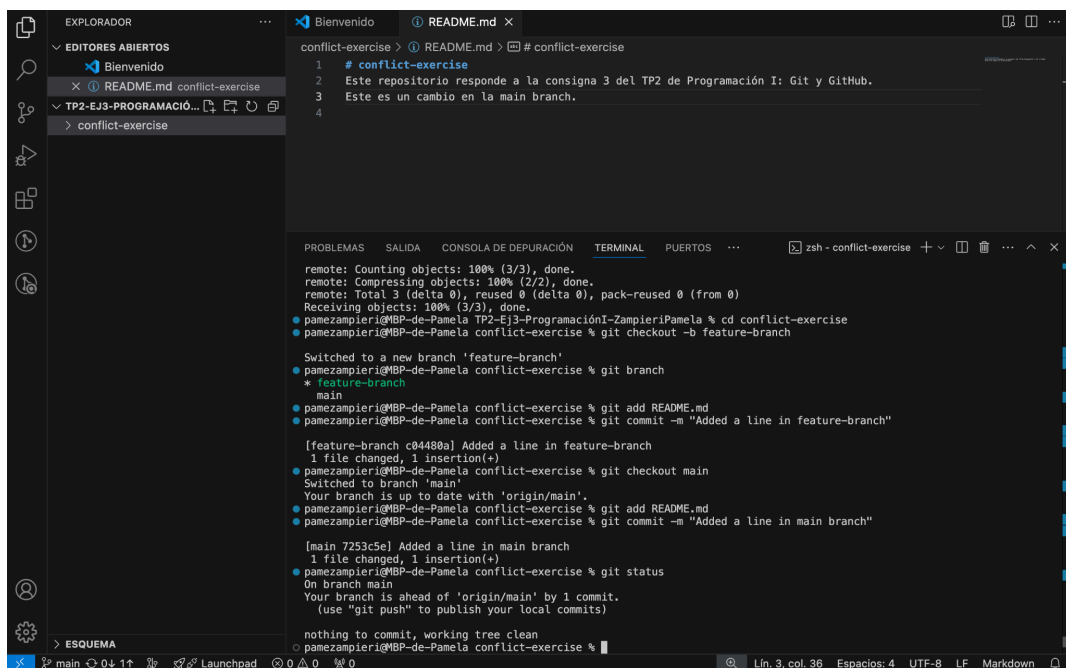
● pamezampieri@MBP-de-Pamela TP2-Ej3-ProgramaciónI-ZampieriPamela % git clone https://github.com/PamesZampieri/conflict-exercise.git
Cloning into 'conflict-exercise'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.
● pamezampieri@MBP-de-Pamela TP2-Ej3-ProgramaciónI-ZampieriPamela % cd conflict-exercise
● pamezampieri@MBP-de-Pamela conflict-exercise % git checkout -b feature-branch

Switched to a new branch 'feature-branch'
● pamezampieri@MBP-de-Pamela conflict-exercise % git branch
* feature-branch
  main
● pamezampieri@MBP-de-Pamela conflict-exercise % git add README.md
● pamezampieri@MBP-de-Pamela conflict-exercise % git commit -m "Added a line in feature-branch"

[feature-branch c04480a] Added a line in feature-branch
1 file changed, 1 insertion(+)
○ pamezampieri@MBP-de-Pamela conflict-exercise %
```

D. 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"



```
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.
● pamezampieri@MBP-de-Pamela TP2-Ej3-ProgramaciónI-ZampieriPamela % cd conflict-exercise
● pamezampieri@MBP-de-Pamela conflict-exercise % git checkout main

Switched to a new branch 'feature-branch'
● pamezampieri@MBP-de-Pamela conflict-exercise % git branch
* feature-branch
  main
● pamezampieri@MBP-de-Pamela conflict-exercise % git add README.md
● pamezampieri@MBP-de-Pamela conflict-exercise % git commit -m "Added a line in feature-branch"

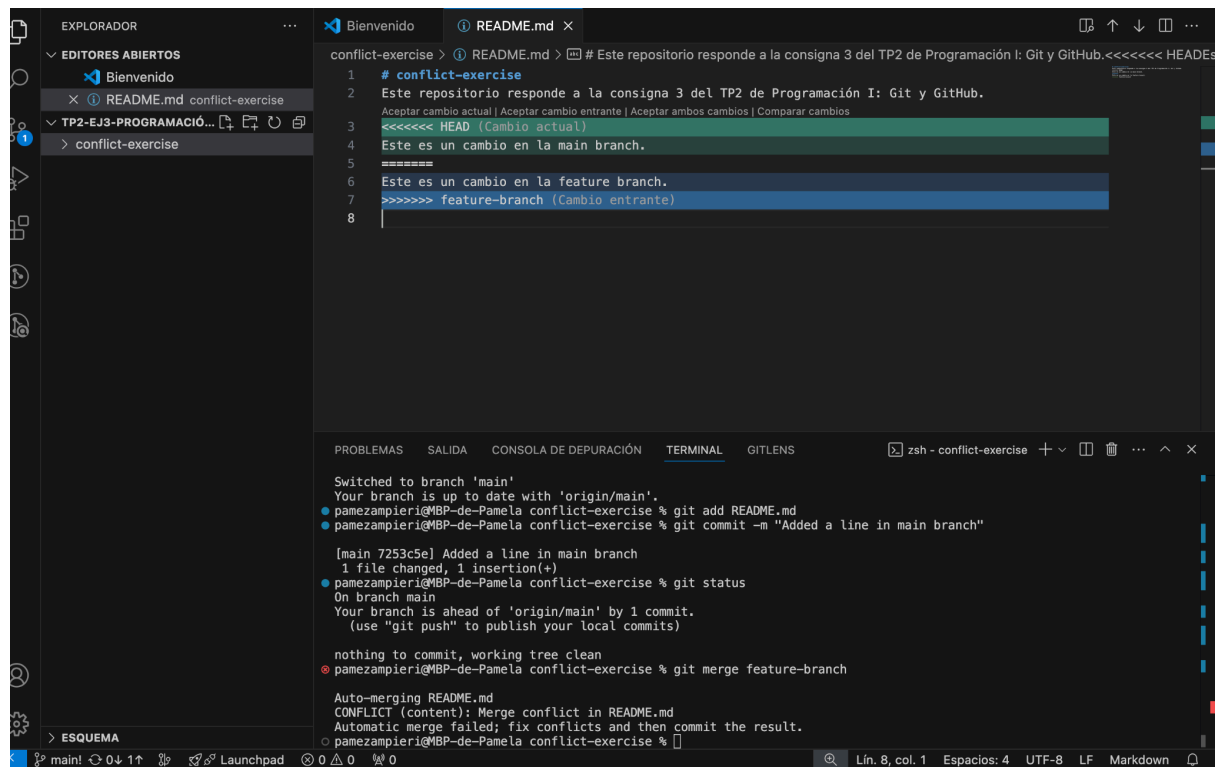
[feature-branch c04480a] Added a line in feature-branch
1 file changed, 1 insertion(+)
● pamezampieri@MBP-de-Pamela conflict-exercise % git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
● pamezampieri@MBP-de-Pamela conflict-exercise % git add README.md
● pamezampieri@MBP-de-Pamela conflict-exercise % git commit -m "Added a line in main branch"

[main 7253c5e] Added a line in main branch
1 file changed, 1 insertion(+)
● pamezampieri@MBP-de-Pamela conflict-exercise % git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
(use "git push" to publish your local commits)

nothing to commit, working tree clean
○ pamezampieri@MBP-de-Pamela conflict-exercise %
```

E. 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.



```
conflict-exercise > 1 README.md x
conflict-exercise > 1 README.md > # Este repositorio responde a la consigna 3 del TP2 de Programación I: Git y GitHub.<<<<<< HEAD
1 # conflict-exercise
2 Este repositorio responde a la consigna 3 del TP2 de Programación I: Git y GitHub.
3 <<<<<< HEAD (Cambio actual)
4 Este es un cambio en la main branch.
5 =====
6 Este es un cambio en la feature branch.
7 >>>>>> feature-branch (Cambio entrante)
8

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL GITLENS
zsh - conflict-exercise + - - - - -
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
pamezampieri@MBP-de-Pamela conflict-exercise % git add README.md
pamezampieri@MBP-de-Pamela conflict-exercise % git commit -m "Added a line in main branch"

[main 7253c5e] Added a line in main branch
1 file changed, 1 insertion(+)
pamezampieri@MBP-de-Pamela conflict-exercise % git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
(use "git push" to publish your local commits)

nothing to commit, working tree clean
pamezampieri@MBP-de-Pamela conflict-exercise % git merge feature-branch

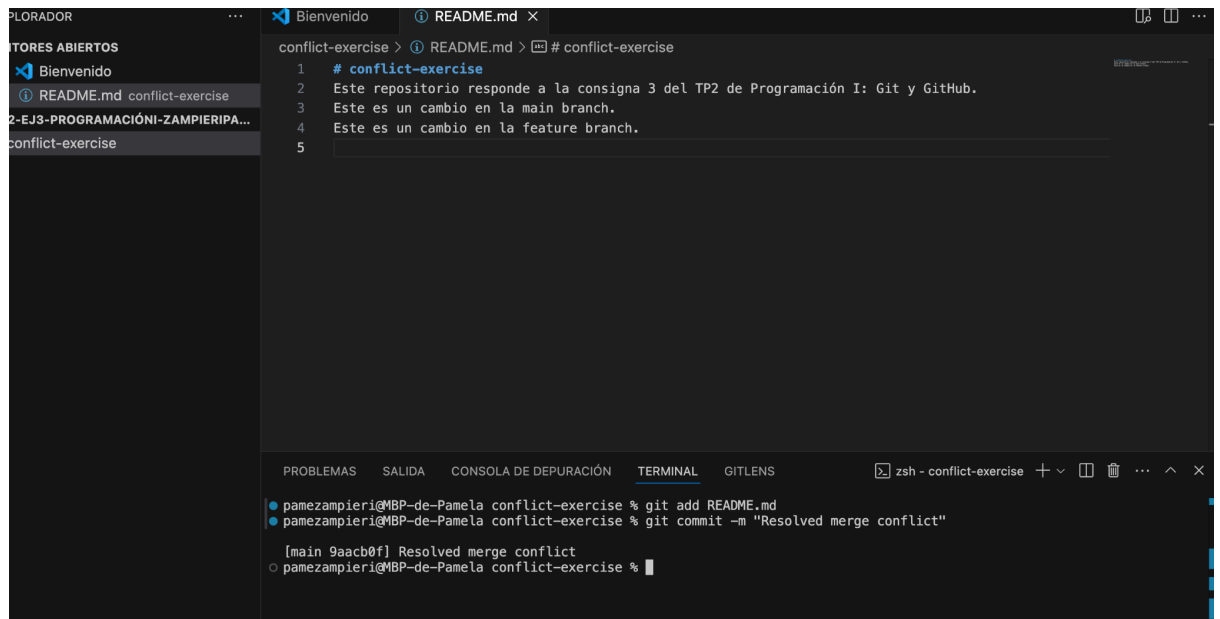
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.
pamezampieri@MBP-de-Pamela conflict-exercise %
```

F. 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
```

- 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 estés 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"`



The screenshot shows a code editor with a sidebar on the left containing a file explorer with 'BIENVENIDO', 'README.md', and 'conflict-exercise'. The main editor area displays the content of 'README.md' with the following text:

```
1 # conflict-exercise
2 Este repositorio responde a la consigna 3 del TP2 de Programación I: Git y GitHub.
3 Este es un cambio en la main branch.
4 Este es un cambio en la feature branch.
5
```

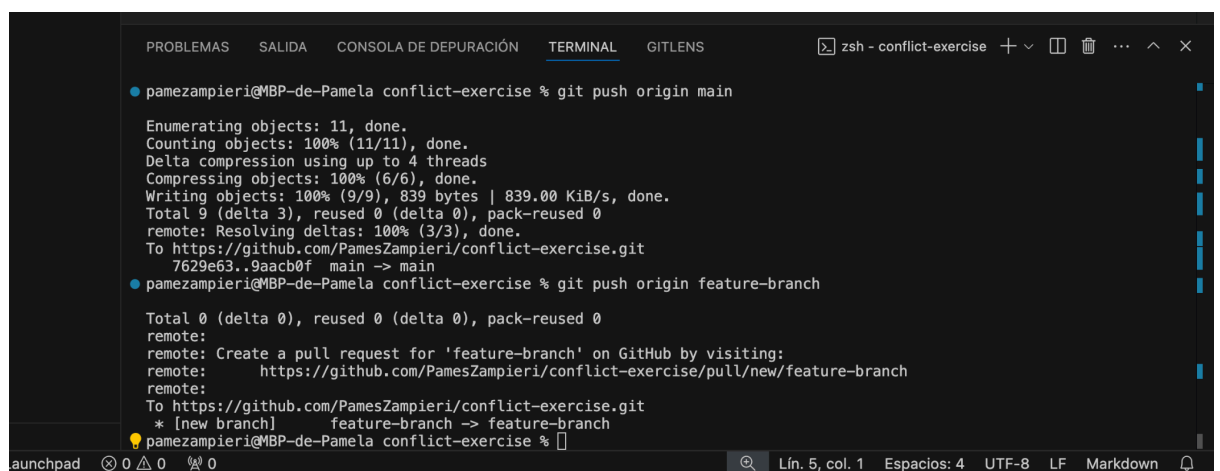
At the bottom, a terminal window shows the following commands and output:

```
pamezampieri@MBP-de-Pamela conflict-exercise % git add README.md
pamezampieri@MBP-de-Pamela conflict-exercise % git commit -m "Resolved merge conflict"

[main 9aacb0f] Resolved merge conflict
pamezampieri@MBP-de-Pamela conflict-exercise %
```

G. 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`



The screenshot shows a terminal window with the following output:

```
pamezampieri@MBP-de-Pamela conflict-exercise % git push origin main
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 4 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (9/9), 839 bytes | 839.00 KiB/s, done.
Total 9 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), done.
To https://github.com/PamesZampieri/conflict-exercise.git
 7629e63..9aacb0f  main -> main
pamezampieri@MBP-de-Pamela conflict-exercise % git push origin feature-branch
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'feature-branch' on GitHub by visiting:
remote:   https://github.com/PamesZampieri/conflict-exercise/pull/new/feature-branch
remote:
To https://github.com/PamesZampieri/conflict-exercise.git
 * [new branch]      feature-branch -> feature-branch
pamezampieri@MBP-de-Pamela conflict-exercise %
```

H. 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.

PamesZampieri

conflict-exercise

Type to search

Code

Issues

Pull requests

Actions

Projects

Wiki

Security

Insights

Settings

conflict-exercise

Public

Pin

Unwatch 1

Fork 0

Star 0

main

2 Branches

0 Tags

Go to file

t

Add file

Code

PamesZampieri

Resolved merge conflict

9aacb0f · 2 minutes ago

4 Commits

README.md

Resolved merge conflict

2 minutes ago

README

conflict-exercise

Este repositorio responde a la consigna 3 del TP2 de Programación I: Git y GitHub. Este es un cambio en la main branch. Este es un cambio en la feature branch.

About

Este repositorio responde a la consigna 3 del TP2 de Programación I: Git y GitHub.

Readme

Activity

0 stars

1 watching

0 forks

Releases

No releases published

Create a new release

Packages

No packages published

Publish your first package

© 2025 GitHub, Inc.

Terms

Privacy

Security

Status

Docs

Contact

Manage cookies

Do not share my personal information

github.com/PamesZampieri/conflict-exercise/commit/9aacb0fe0e7d3876ba58bbc4bf4dc915f3a76c40

Commit 9aacb0f

Browse files

PamesZampieri

committed 3 minutes ago

Resolved merge conflict

main

2 parents 7253c5e + c04488a commit 9aacb0f

Filter files...

1 file changed

+1 -0 lines changed

Search within code

README.md

@@ -1,3 +1,4 @@

1 1 # conflict-exercise

2 2 Este repositorio responde a la consigna 3 del TP2 de Programación I: Git y GitHub.

3 3 Este es un cambio en la main branch.

4 + Este es un cambio en la feature branch.

Comments 0

Lock conversation

Comment

Unsubscribe You're receiving notifications because you're subscribed to this thread.