

Alumno: Damián Jorge Portillo, Comisión 19.

Repositorios:

<https://github.com/damianportillo/UTN-TUPaD-P1.git>

<https://github.com/damianportillo/UTN-TUPaD-P1-Tp2-Ac2.git>

<https://github.com/damianportillo/conflict-exercise.git>

Práctico 2: Git y GitHub

Actividades Resueltas

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

- **¿Qué es GitHub?**

GitHub es una plataforma de desarrollo colaborativo basada en la nube que permite a los desarrolladores alojar, gestionar y compartir código utilizando Git, un sistema de control de versiones.

- **¿Cómo crear un repositorio en GitHub?**

1- Entra a GitHub y haz clic en "New repository" (+).
2- Asigna un nombre, elige público o privado y haz clic en "Create repository".
3- Desde la terminal, enlaza tu proyecto con GitHub:
git init
git add .
git commit -m "Primer commit"
git branch -M main
git remote add origin <https://github.com/tu-usuario/mi-proyecto.git>
git push -u origin main

- **¿Cómo crear una rama en Git?**

git branch nombre-de-la-rama

- **¿Cómo cambiar a una rama en Git?**

git checkout nombre-de-la-rama

- **¿Cómo fusionar ramas en Git?**

Posicionado sobre la rama principal a la cual desea unir los cambios ejecuta:
git merge nombre-de-la-rama

- **¿Cómo crear un commit en Git?**

git commit -m "Mensaje del commit"

- **¿Cómo enviar un commit a GitHub?**
git push origin nombre-de-la-rama
- **¿Qué es un repositorio remoto?**
Un repositorio remoto en Git es una versión del repositorio que está alojada en un servidor externo, como GitHub, GitLab, o cualquier otro servicio de alojamiento de código. A diferencia de un repositorio local, que reside en la pc que trabajamos, el repositorio remoto permite que varias personas trabajen en el mismo proyecto, compartiendo y sincronizando cambios.
- **¿Cómo agregar un repositorio remoto a Git?**
git remote add origin <https://github.com/tu-usuario/nombre-del-repositorio.git>
- **¿Cómo empujar cambios a un repositorio remoto?**
git push origin nombre-de-la-rama
- **¿Cómo tirar de cambios de un repositorio remoto?**
git pull origin nombre-de-la-rama
- **¿Qué es un fork de repositorio?**
Un fork de repositorio es una copia de un repositorio que haces en tu propia cuenta de GitHub. Al hacer un fork, puedes realizar cambios y experimentos en el proyecto sin afectar el repositorio original.
- **¿Cómo crear un fork de un repositorio?**
 - 1- Voy repositorio en GitHub que deseo forkear.
 - 2- En la esquina superior derecha de la página, haz clic en el botón "Fork".
 - 3- GitHub abrirá una vista donde podrás modificar la descripción del repositorio a forkear. Y por último al confirmar creará una copia del repositorio en tu cuenta.
 - 4- Después de hacer el fork, puedes clonar el repositorio en tu computadora y empezar a trabajar en él.
- **¿Cómo enviar una solicitud de extracción (pull request) a un repositorio?**
 - 1- Haz los cambios en tu fork y súbelos a GitHub.
 - 2- Ve al repositorio original en GitHub.
 - 3- Haz clic en la pestaña "Pull requests" y luego en "New pull request".
 - 4- Selecciona tu rama (la que contiene los cambios) y la rama principal del repositorio original.
 - 5- Revisa las diferencias entre ambas ramas.
 - 6- Escribe un título y una descripción para tu solicitud de extracción.
 - 7- Haz clic en "Create pull request".
- **¿Cómo aceptar una solicitud de extracción?**
 - 1- Me posiciono en al repositorio donde se envió la solicitud de extracción en GitHub.

- 2- Hago clic en la pestaña "Pull requests".
- 3- Selecciono la solicitud de extracción que deseo aceptar.
- 4- Reviso los cambios propuestos y resuelvo cualquier conflicto si es necesario.
- 5- Hago clic en "Merge pull request".
- 6- Confirmo haciendo clic en "Confirm merge" para fusionar los cambios.
- 7- (Opcional) Selecciono "Delete branch" si es que ya no necesito la rama de la solicitud.

- **¿Qué es un etiqueta en Git?**

Una etiqueta (tag) en Git es una referencia a un commit específico en el historial de tu repositorio. Se usa para marcar puntos importantes, como versiones estables o lanzamientos de software. A diferencia de las ramas, las etiquetas no cambian, lo que las hace ideales para identificar hitos específicos en el proyecto, como una versión final. Son fijas y no se mueven, lo que facilita volver a esa versión en cualquier momento.

- **¿Cómo crear una etiqueta en Git?**

```
git tag v1.0.0  
git tag -a v1.0.0 -m "Lanzamiento de la versión 1.0.0"  
git tag v1.0.0 <commit-hash>
```

- **¿Cómo enviar una etiqueta a GitHub?**

```
git push origin v1.0.0  
git push --tags
```

- **¿Qué es un historial de Git?**

El historial de Git es un registro de todos los cambios (commits) realizados en un repositorio a lo largo del tiempo. Cada vez que haces un commit, Git guarda información sobre qué cambios se hicieron, cuándo, por quién y en qué archivo(s).

- **¿Cómo ver el historial de Git?**

```
git log
```

- **¿Cómo buscar en el historial de Git?**

```
git log --grep="término de búsqueda"  
git log --since="2023-01-01" --until="2023-12-31"  
git log -S"término a buscar" -- <archivo>  
git log --author="nombre del autor"  
git log | grep "término"
```

- **¿Cómo borrar el historial de Git?**

Borrar el historial de Git no es algo que se haga comúnmente, pero si realmente necesitas hacerlo, puedes usar algunos comandos para reescribir o eliminar el historial de un repositorio. Aquí te dejo algunas formas de hacerlo:

1- **Borrar todo el historial y mantener solo el último commit:**

- **Crear una rama sin historial**

```
git checkout --orphan nueva_rama
git add -A
git commit -m "Nuevo inicio sin historial"
```

- **Elimina la rama anterior y renombra la nueva:**

```
git branch -D main # O la rama original
git branch -m main
```

- **Empuja los cambios al remoto con --force:**

```
git push origin main --force
```

2- **Eliminar commits específicos:**

- **Usa rebase interactivo para eliminar o editar commits:**

```
git rebase -i --root
```

- **Luego, empuja con --force:**

```
git push origin main --force
```

IMPORTANTE: Borrar el historial puede causar pérdida de datos, así que hazlo con cuidado y asegúrate de que no afecte a otros colaboradores.

- **¿Qué es un repositorio privado en GitHub?**
Un repositorio privado en GitHub es un repositorio cuyo acceso está restringido solo a personas que tienen permisos específicos. Solo los usuarios autorizados (como los colaboradores) pueden ver, modificar o contribuir al código.
- **¿Cómo crear un repositorio privado en GitHub?**
 - 1- Inicio sesión en GitHub.
 - 2- Hago clic en el botón "+" en la esquina superior derecha y selecciona "New repository".
 - 3- Pongo un nombre y descripción (opcional) para el repositorio.
 - 4- Seleccione la opción "Private" bajo "Visibility".
 - 5- Hago clic en "Create repository".
- **¿Cómo invitar a alguien a un repositorio privado en GitHub?**
 - 1- Entro al repositorio privado en GitHub.
 - 2- Hago clic en la pestaña "Settings" (Configuración).
 - 3- En el menú de la izquierda, selecciono "Manage access".
 - 4- Hago clic en el botón "Invite a collaborator".
 - 5- Escribo el **nombre de usuario** de la persona que deseo invitar.
 - 6- Clic en "Add" para enviar la invitación.
- **¿Qué es un repositorio público en GitHub?**
Un repositorio público en GitHub es un repositorio cuyo código y contenido son accesibles a cualquier persona en Internet.
- **¿Cómo crear un repositorio público en GitHub?**
 - 1- Inicio sesión en GitHub.
 - 2- Hago clic en el botón "+" en la esquina superior derecha y selecciona "New repository".
 - 3- Pongo un nombre y descripción (opcional) para el repositorio.

4- Selecciono la opción "**Public**" bajo "**Visibility**".

5- Hago clic en "**Create repository**".

- **¿Cómo compartir un repositorio público en GitHub?**

Para compartir un repositorio público en GitHub, copio la URL del repositorio y la comparto con quien desee, ya que es accesible para todos.

2) Realizar la siguiente actividad:

- Crear un repositorio.
 - Dale un nombre al repositorio.
 - Elije el repositorio sea público.

- Inicializa el repositorio con un archivo.

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 * damianportillo / **Repository name *** UTN-TUPaD-P1-Tp2-Ac2

✔ UTN-TUPaD-P1-Tp2-Ac2 is available.

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

Description (optional)
UTN Programación 1 - TP2 - Actividad 2

☒  **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.](#)

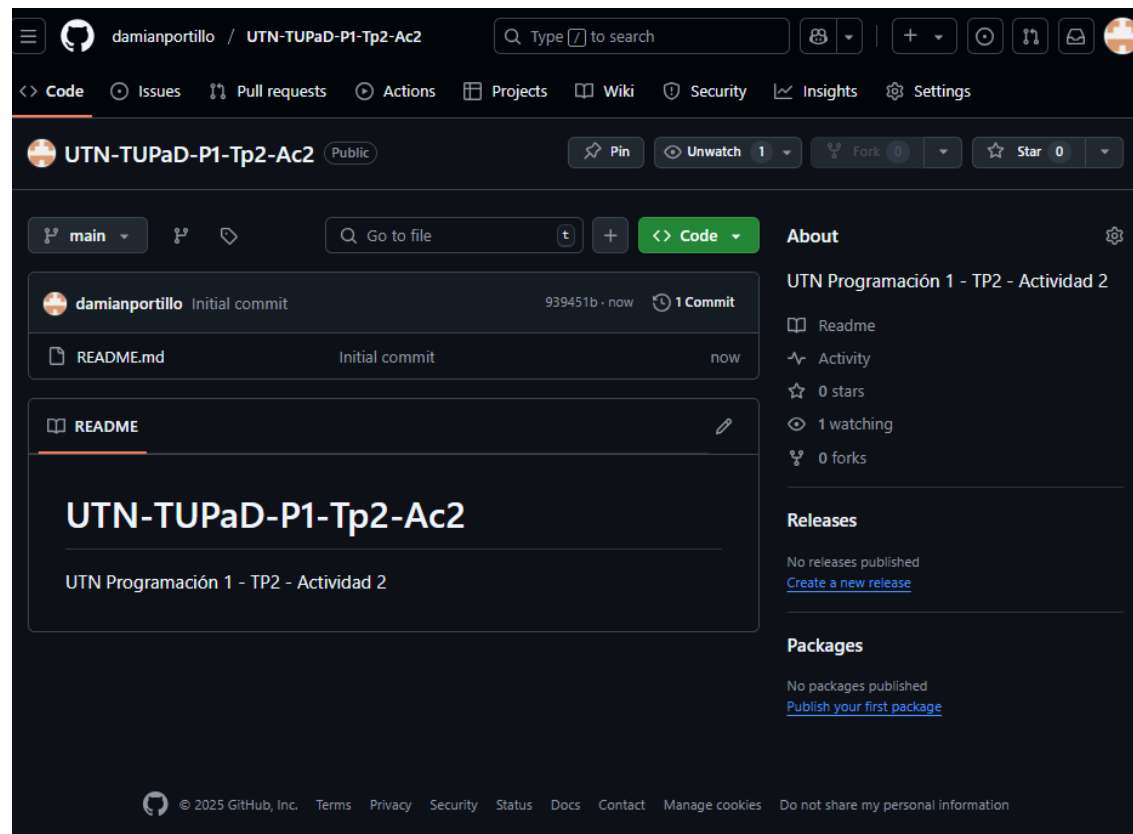
Choose a license
License: **None**

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

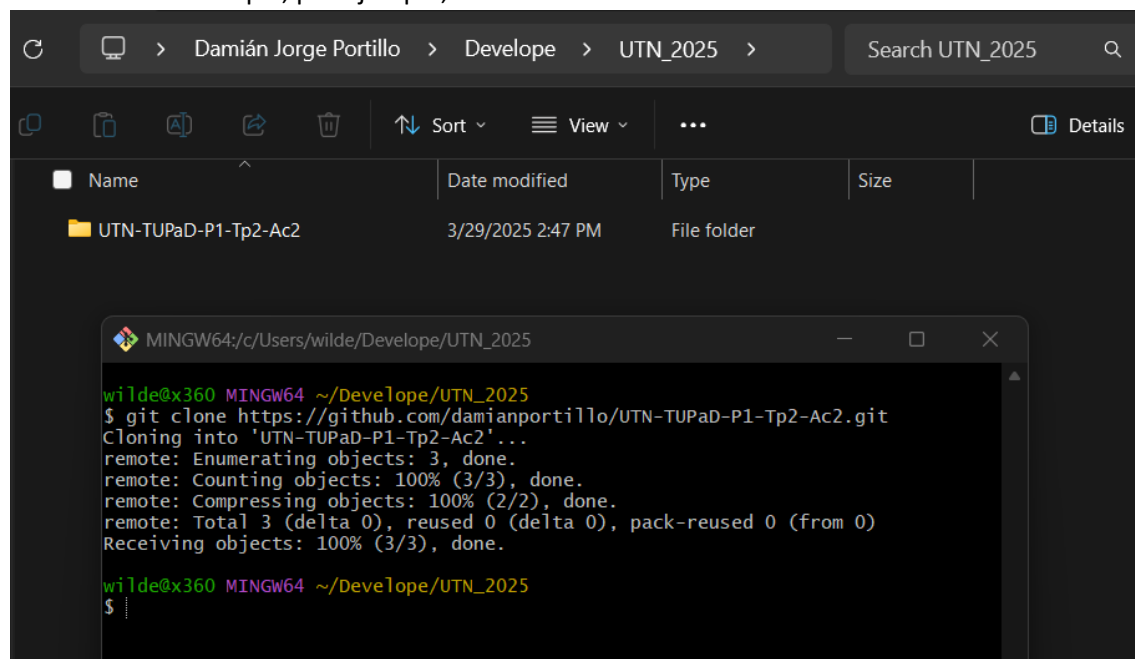
This will set  **main** as the default branch. Change the default name in your [settings](#).

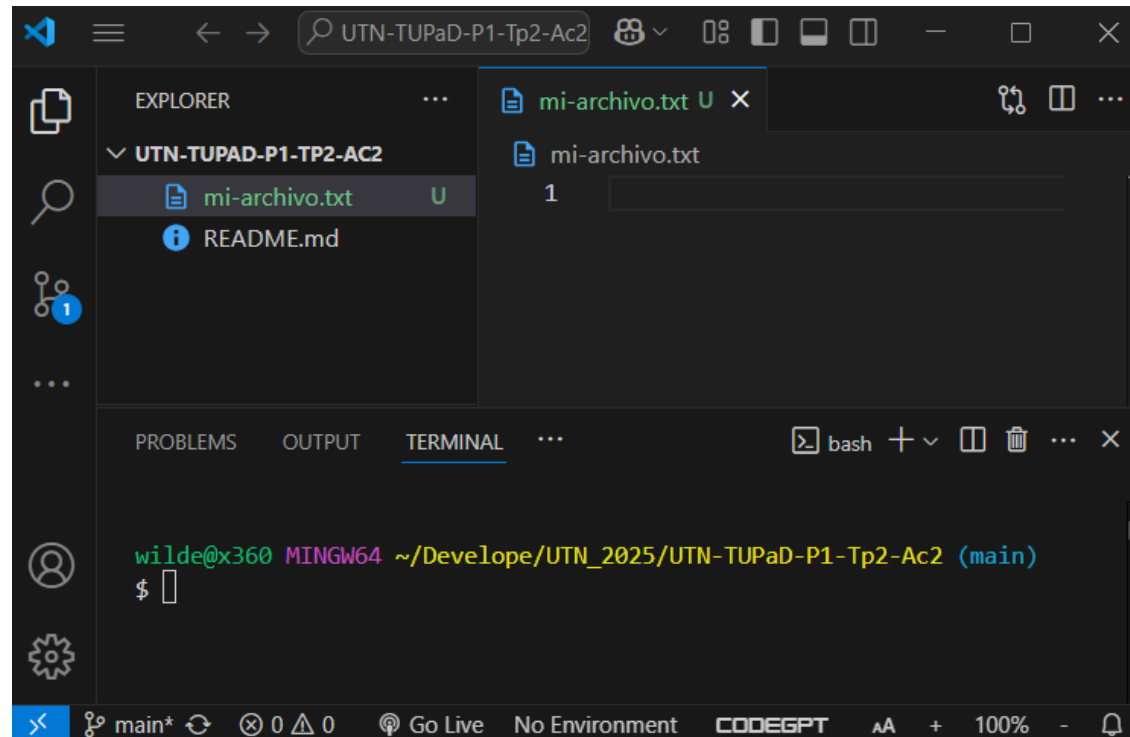
 You are creating a public repository in your personal account.

[Create repository](#)

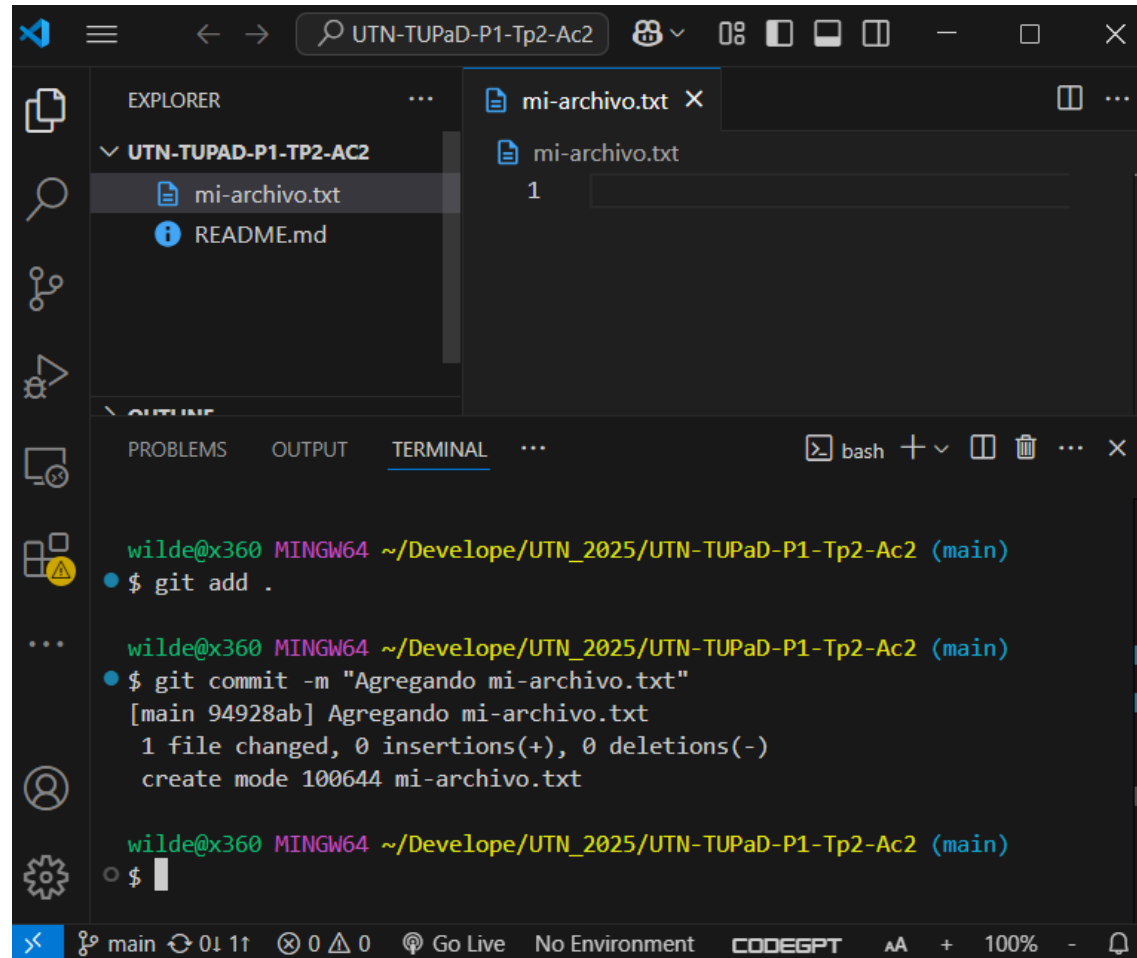


- Agregando un Archivo
 - Crea un archivo simple, por ejemplo, "mi-archivo.txt".





- Realiza los comandos `git add .` y `git commit -m "Agregando mi-archivo.txt"` en la línea de comandos.



The screenshot shows the Visual Studio Code interface. The Explorer panel on the left shows a project named 'UTN-TUPAD-P1-TP2-AC2' with two files: 'mi-archivo.txt' and 'README.md'. The Editor panel on the right shows the content of 'mi-archivo.txt', which is a single line: '1'. The Terminal panel at the bottom shows a bash shell session with the following commands and output:

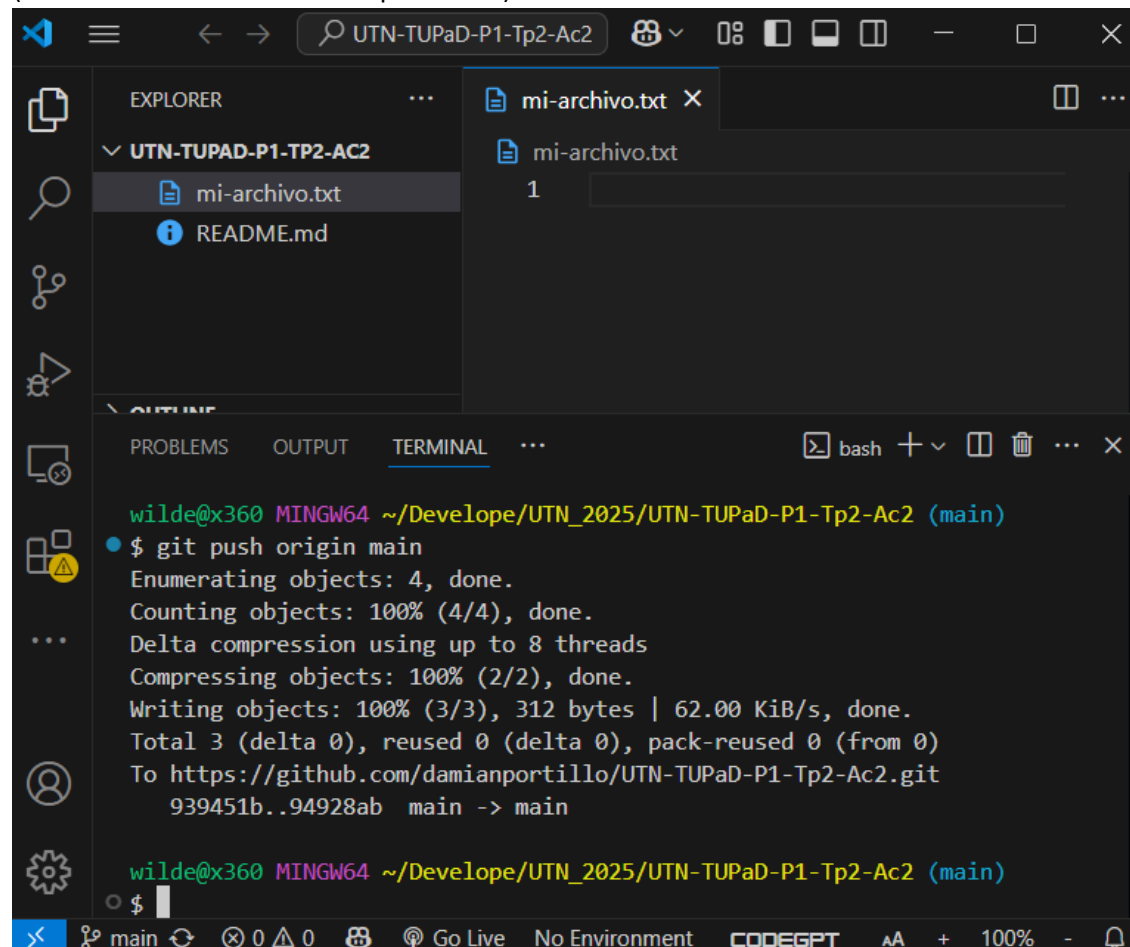
```
wilde@x360 MINGW64 ~/Develo/UTN_2025/UTN-TUPaD-P1-Tp2-Ac2 (main)
$ git add .

wilde@x360 MINGW64 ~/Develo/UTN_2025/UTN-TUPaD-P1-Tp2-Ac2 (main)
$ git commit -m "Agregando mi-archivo.txt"
[main 94928ab] Agregando mi-archivo.txt
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 mi-archivo.txt

wilde@x360 MINGW64 ~/Develo/UTN_2025/UTN-TUPaD-P1-Tp2-Ac2 (main)
$
```

The status bar at the bottom indicates the current branch is 'main', with 0 changes and 0 deletions. It also shows 'Go Live', 'No Environment', 'CODEGPT', and a zoom level of 100%.

- Sube los cambios al repositorio en GitHub con `git push origin main` (o el nombre de la rama correspondiente).



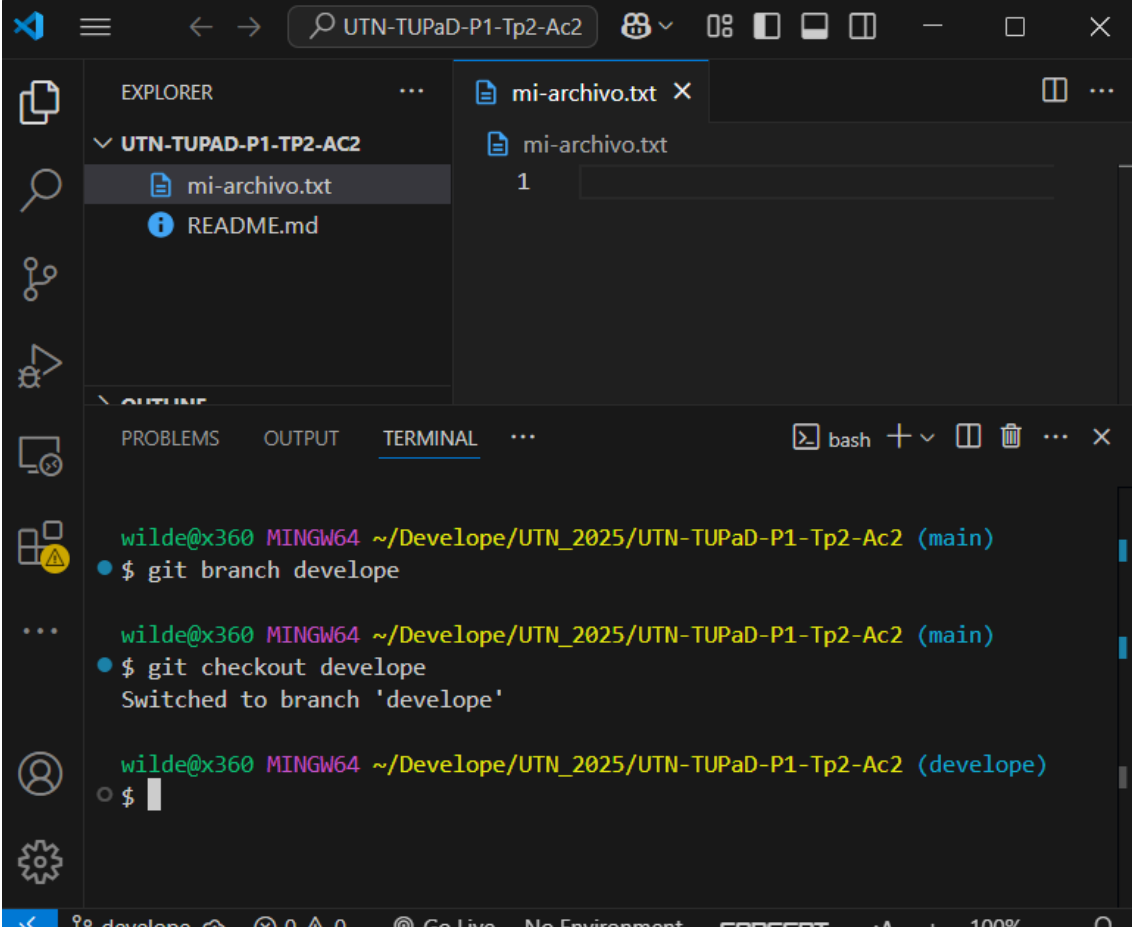
The screenshot shows the Visual Studio Code interface. The Explorer panel on the left shows a project named 'UTN-TUPAD-P1-TP2-AC2' with files 'mi-archivo.txt' and 'README.md'. The main editor shows 'mi-archivo.txt' with a single line of text. The Terminal panel at the bottom shows the following output:

```
wilde@x360 MINGW64 ~/Develo/UTN_2025/UTN-TUPaD-P1-Tp2-Ac2 (main)
$ git push origin main
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), 312 bytes | 62.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/damianportillo/UTN-TUPaD-P1-Tp2-Ac2.git
939451b..94928ab main -> main

wilde@x360 MINGW64 ~/Develo/UTN_2025/UTN-TUPaD-P1-Tp2-Ac2 (main)
$
```

- Creando Branchs

- Crear una Branch



The screenshot shows the Visual Studio Code interface with a project named 'UTN-TUPaD-P1-Tp2-Ac2'. The Explorer panel on the left shows files 'mi-archivo.txt' and 'README.md'. The main editor shows 'mi-archivo.txt' with line 1. The TERMINAL panel at the bottom shows the following commands and output:

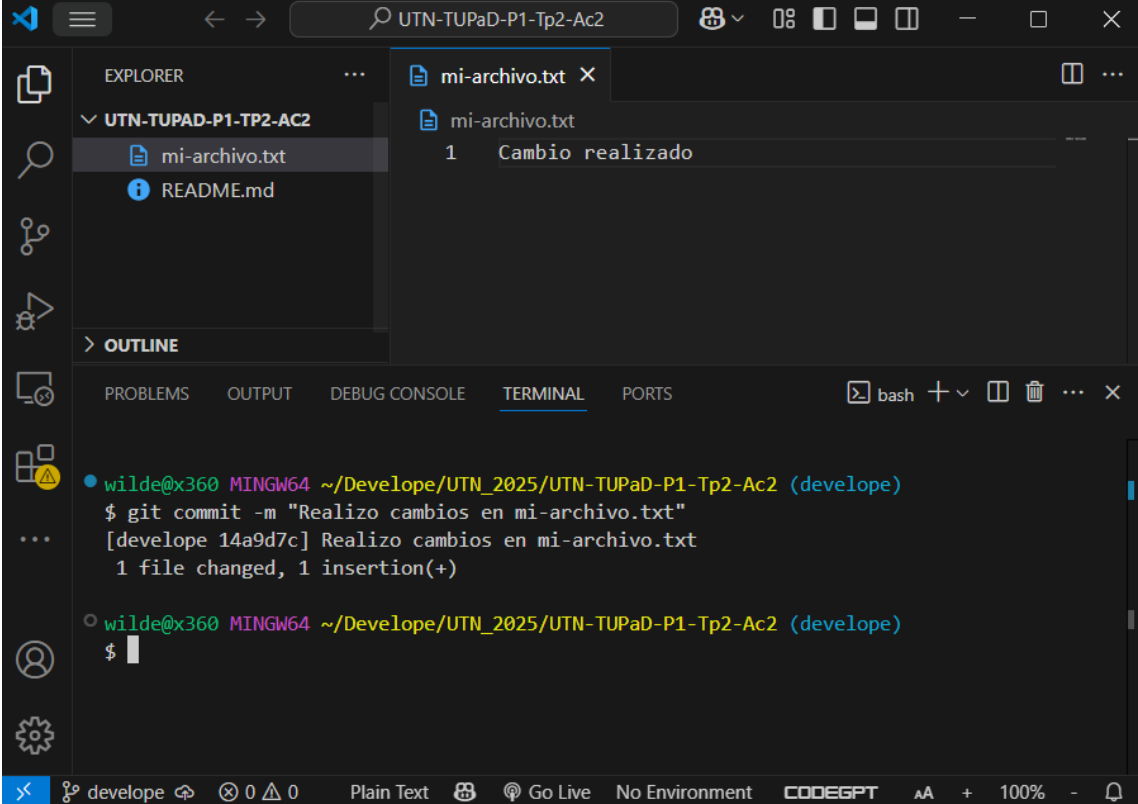
```
wilde@x360 MINGW64 ~/Develope/UTN_2025/UTN-TUPaD-P1-Tp2-Ac2 (main)
$ git branch developpe

wilde@x360 MINGW64 ~/Develope/UTN_2025/UTN-TUPaD-P1-Tp2-Ac2 (main)
$ git checkout developpe
Switched to branch 'developpe'

wilde@x360 MINGW64 ~/Develope/UTN_2025/UTN-TUPaD-P1-Tp2-Ac2 (developpe)
$
```

The status bar at the bottom indicates the current branch is 'developpe'.

- Realizar cambios o agregar un archivo

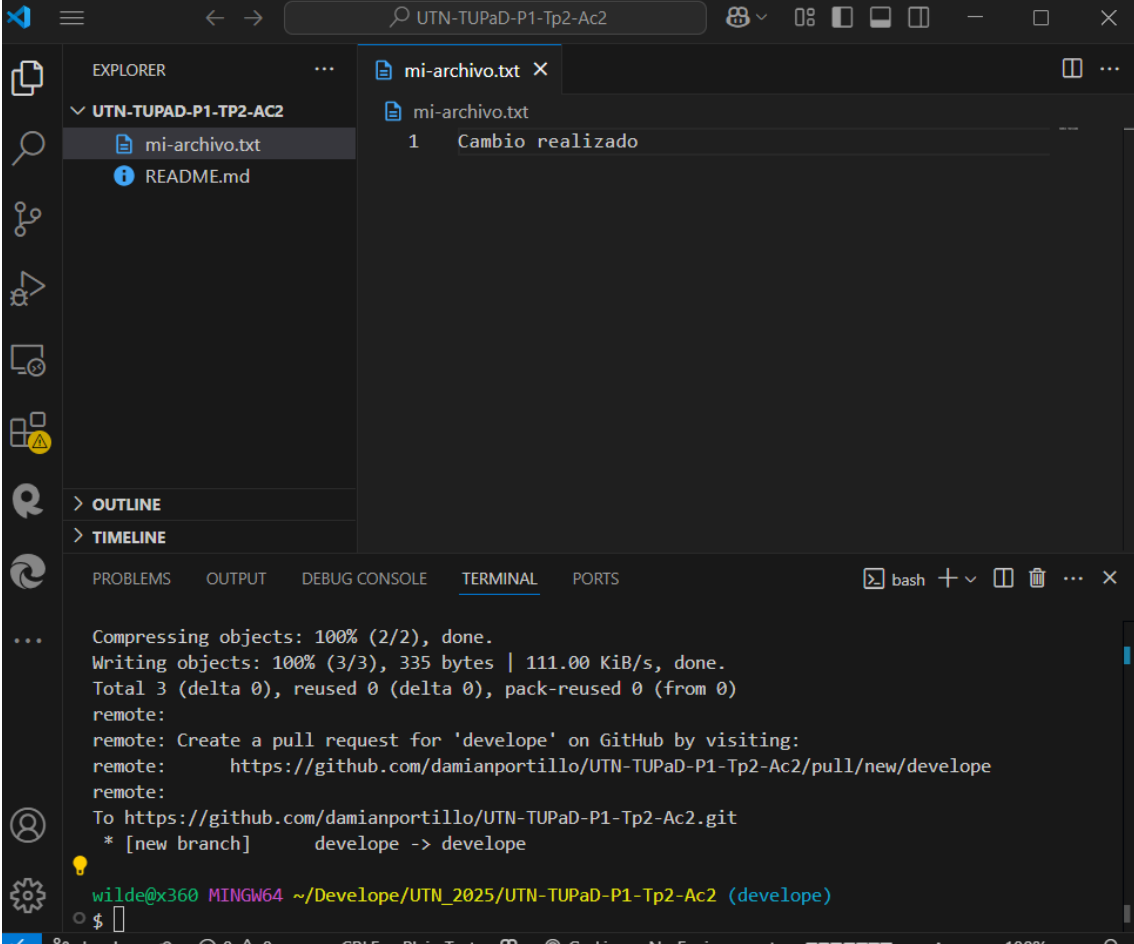


The screenshot shows the Visual Studio Code (VS Code) interface. The Explorer panel on the left shows a project named 'UTN-TUPAD-P1-TP2-AC2' with two files: 'mi-archivo.txt' and 'README.md'. The file 'mi-archivo.txt' is open in the editor, showing the text '1 Cambio realizado'. The Terminal panel at the bottom shows the following commands and output:

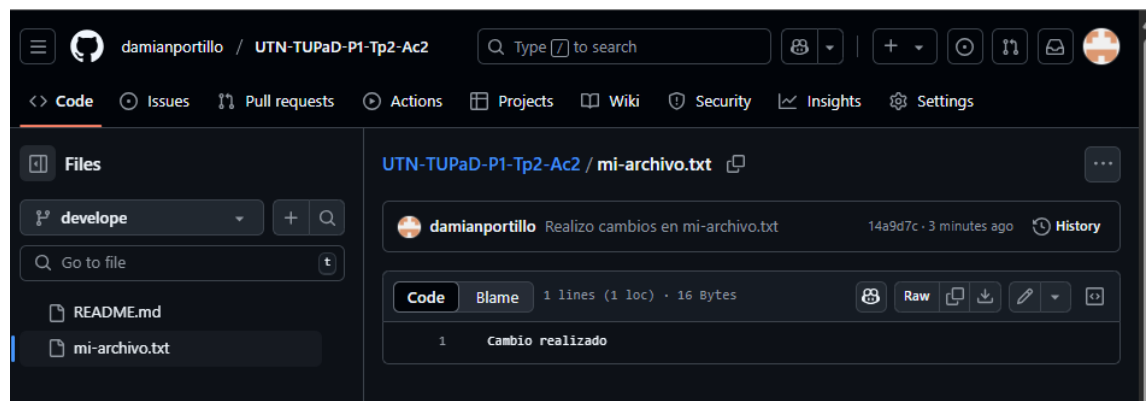
```
wilde@x360 MINGW64 ~/Develope/UTN_2025/UTN-TUPAD-P1-Tp2-Ac2 (develope)
$ git commit -m "Realizo cambios en mi-archivo.txt"
[develope 14a9d7c] Realizo cambios en mi-archivo.txt
1 file changed, 1 insertion(+)

wilde@x360 MINGW64 ~/Develope/UTN_2025/UTN-TUPAD-P1-Tp2-Ac2 (develope)
$
```

- Subir la Branch



The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left shows a project named 'UTN-TUPAD-P1-TP2-AC2' with files 'mi-archivo.txt' and 'README.md'. The main editor area shows 'mi-archivo.txt' with the content '1 Cambio realizado'. The TERMINAL panel at the bottom shows the output of a git commit command, including the message 'Cambio realizado' and the commit hash '14a9d7c'. The terminal prompt is 'wilde@x360 MINGW64 ~/Develo/UTN_2025/UTN-TUPAD-P1-Tp2-Ac2 (develop)'.

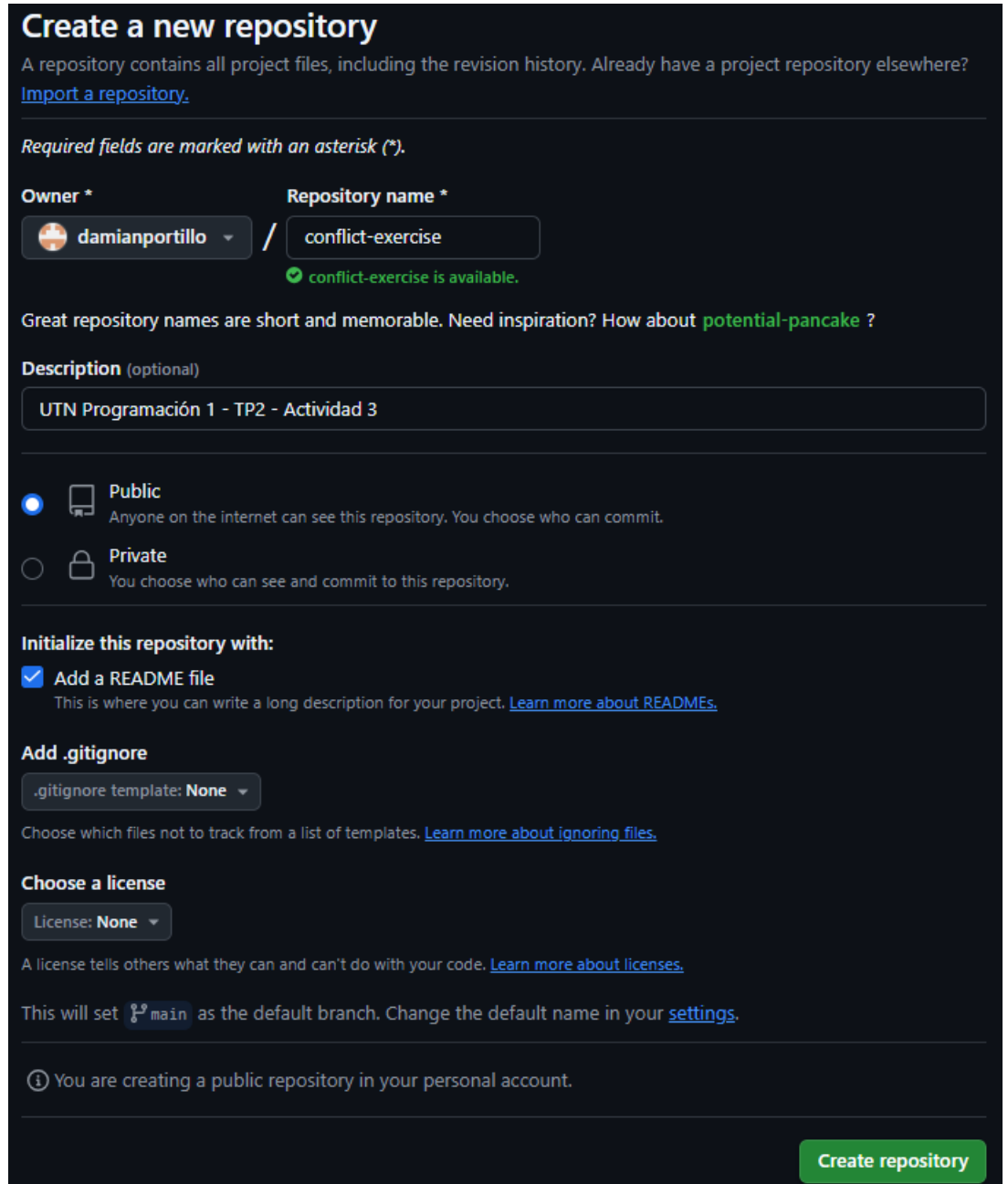


3) Realizar la siguiente actividad:

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".




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 * **Repository name ***


 damianportillo / conflict-exercise


✔ conflict-exercise is available.

Great repository names are short and memorable. Need inspiration? How about **potential-pancake** ?

Description (optional)

UTN Programación 1 - TP2 - Actividad 3

☒  **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](#).

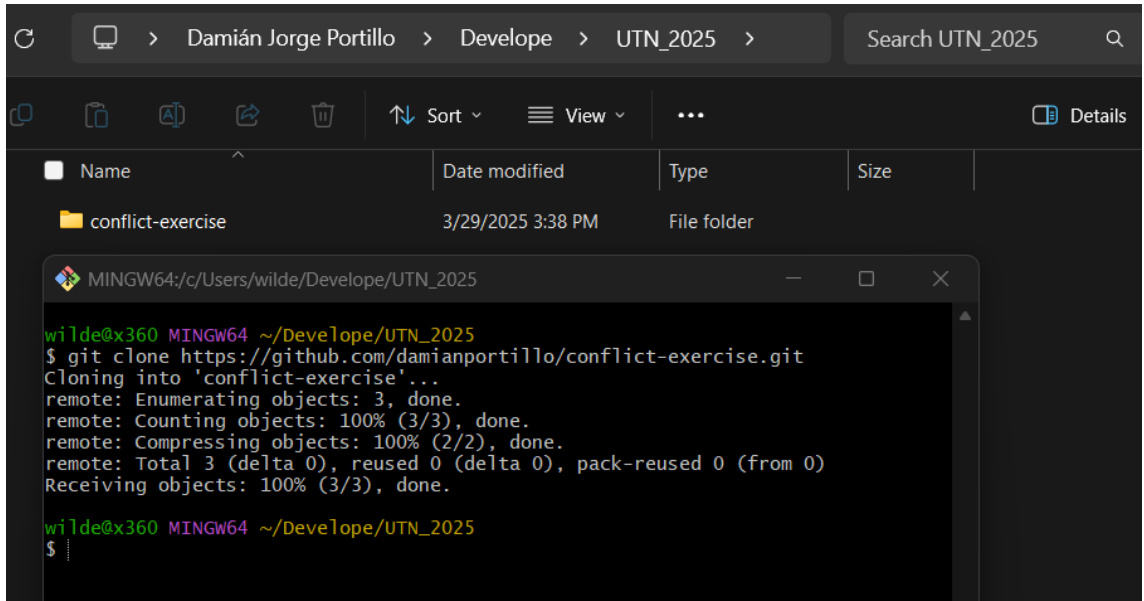
 You are creating a public repository in your personal account.

Create repository

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



The screenshot shows a file explorer window with the path 'Damián Jorge Portillo > Developpe > UTN_2025'. A folder named 'conflict-exercise' is listed, modified on 3/29/2025 at 3:38 PM. Below the file explorer, a terminal window is open with the following output:

```
MINGW64:~/Develo.../UTN_2025
wilde@x360 MINGW64 ~/Develo.../UTN_2025
$ git clone https://github.com/damianportillo/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.

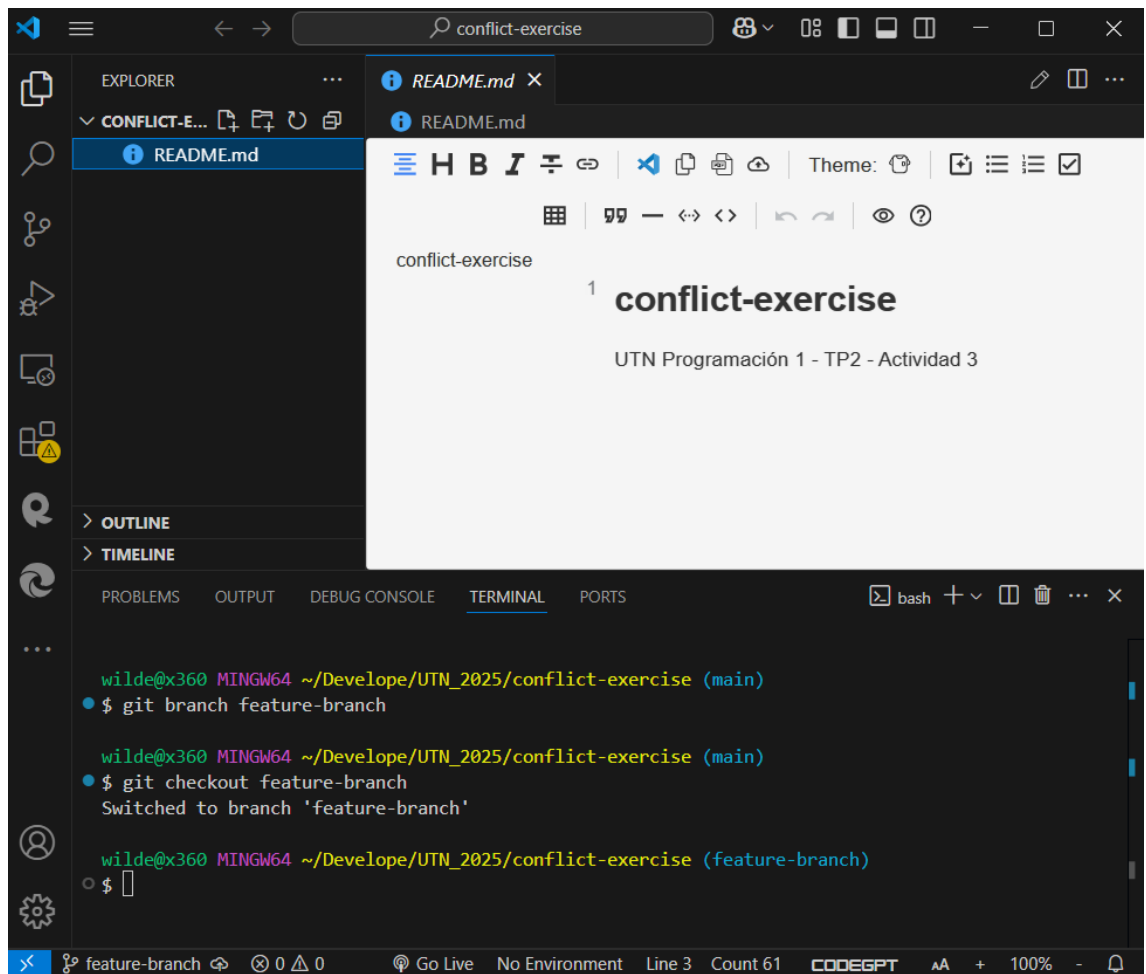
wilde@x360 MINGW64 ~/Develo.../UTN_2025
$
```

- 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`



The screenshot shows the Visual Studio Code interface. The Explorer panel on the left shows a file named 'README.md' selected. The main editor area displays the content of 'README.md', which includes the title 'conflict-exercise' and the subtitle 'UTN Programación 1 - TP2 - Actividad 3'. The Terminal panel at the bottom shows a bash shell with the following commands and output:

```
wilde@x360 MINGW64 ~/Develope/UTN_2025/conflict-exercise (main)
$ git branch feature-branch

wilde@x360 MINGW64 ~/Develope/UTN_2025/conflict-exercise (main)
$ git checkout feature-branch
Switched to branch 'feature-branch'

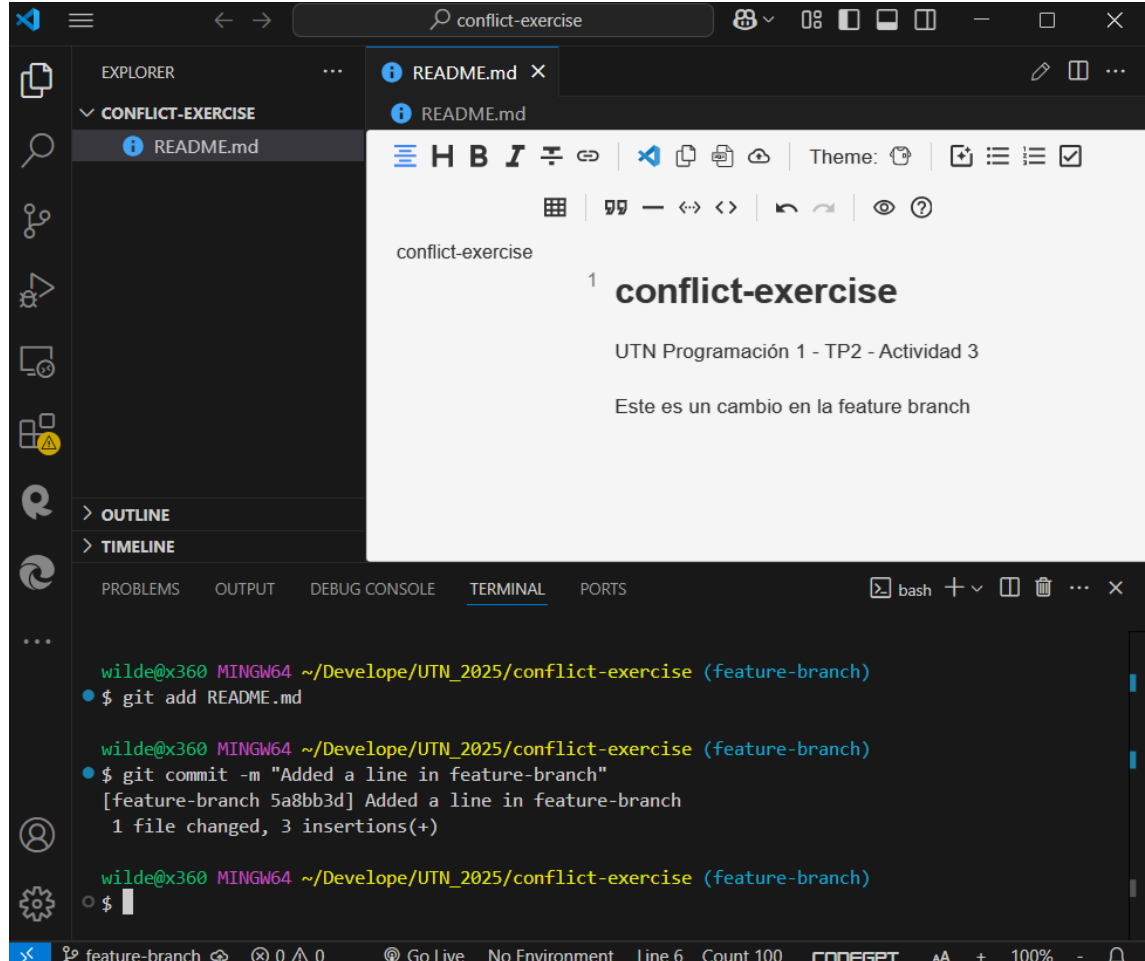
wilde@x360 MINGW64 ~/Develope/UTN_2025/conflict-exercise (feature-branch)
$
```

- 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"



The screenshot shows the Visual Studio Code interface. The Explorer pane on the left shows the file structure of the 'CONFLICT-EXERCISE' project, with 'README.md' selected. The main editor area displays the content of 'README.md', which includes the title 'conflict-exercise', the subtitle 'UTN Programación 1 - TP2 - Actividad 3', and the text 'Este es un cambio en la feature branch'. The bottom panel shows the terminal with the following commands and output:

```
wilde@x360 MINGW64 ~/Develope/UTN_2025/conflict-exercise (feature-branch)
$ git add README.md

wilde@x360 MINGW64 ~/Develope/UTN_2025/conflict-exercise (feature-branch)
$ git commit -m "Added a line in feature-branch"
[feature-branch 5a8bb3d] Added a line in feature-branch
1 file changed, 3 insertions(+)

wilde@x360 MINGW64 ~/Develope/UTN_2025/conflict-exercise (feature-branch)
$
```

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

- Cambia de vuelta a la rama principal (main):

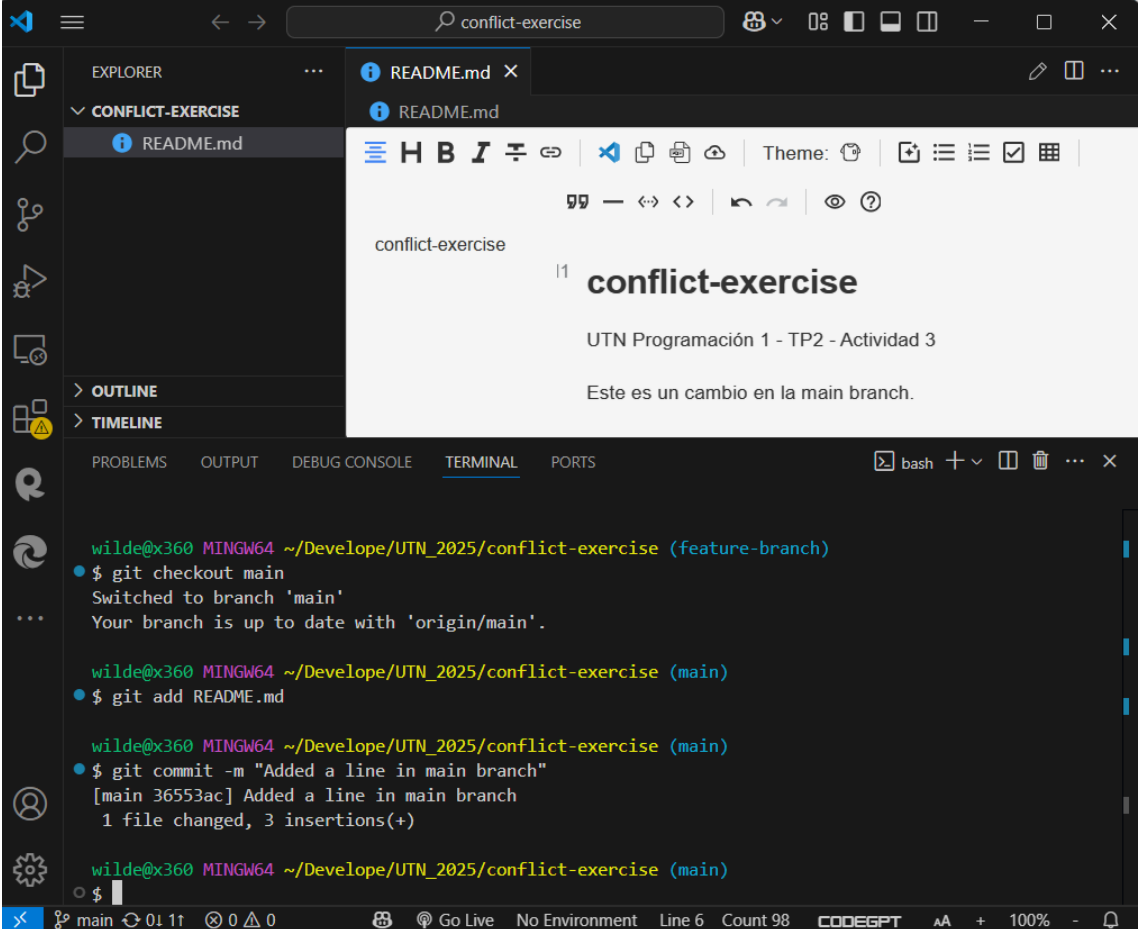
`git checkout main`

- 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"



```
wilde@x360 MINGW64 ~/Develope/UTN_2025/conflict-exercise (feature-branch)
• $ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

wilde@x360 MINGW64 ~/Develope/UTN_2025/conflict-exercise (main)
• $ git add README.md

wilde@x360 MINGW64 ~/Develope/UTN_2025/conflict-exercise (main)
• $ git commit -m "Added a line in main branch"
[main 36553ac] Added a line in main branch
1 file changed, 3 insertions(+)

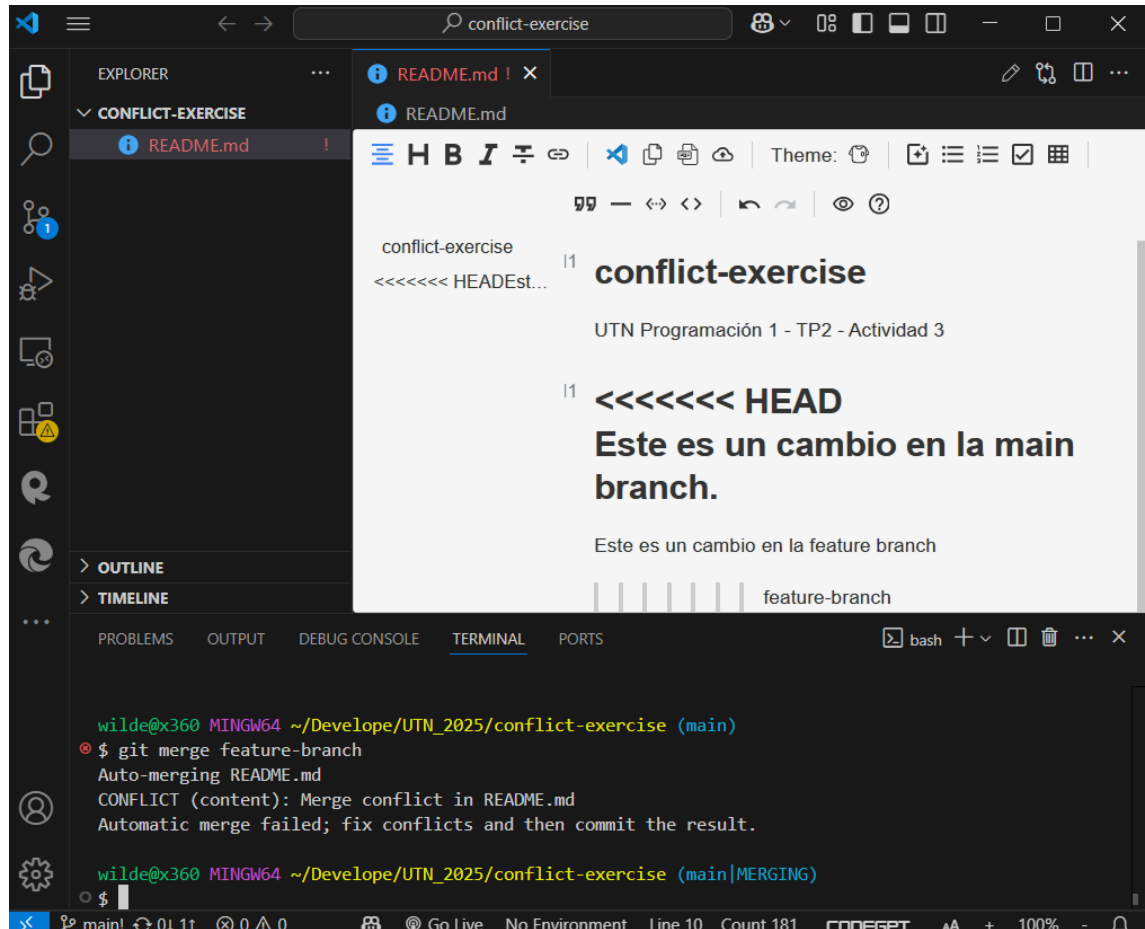
wilde@x360 MINGW64 ~/Develope/UTN_2025/conflict-exercise (main)
• $
```

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
<<<<<<< HEADEst...

conflict-exercise
UTN Programación 1 - TP2 - Actividad 3

<<<<<<< HEAD
Este es un cambio en la main
branch.

Este es un cambio en la feature branch
feature-branch

wilde@x360 MINGW64 ~/Develo.../UTN_2025/conflict-exercise (main)
$ 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.

wilde@x360 MINGW64 ~/Develo.../UTN_2025/conflict-exercise (main|MERGING)
$
```

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.

```
=====
```

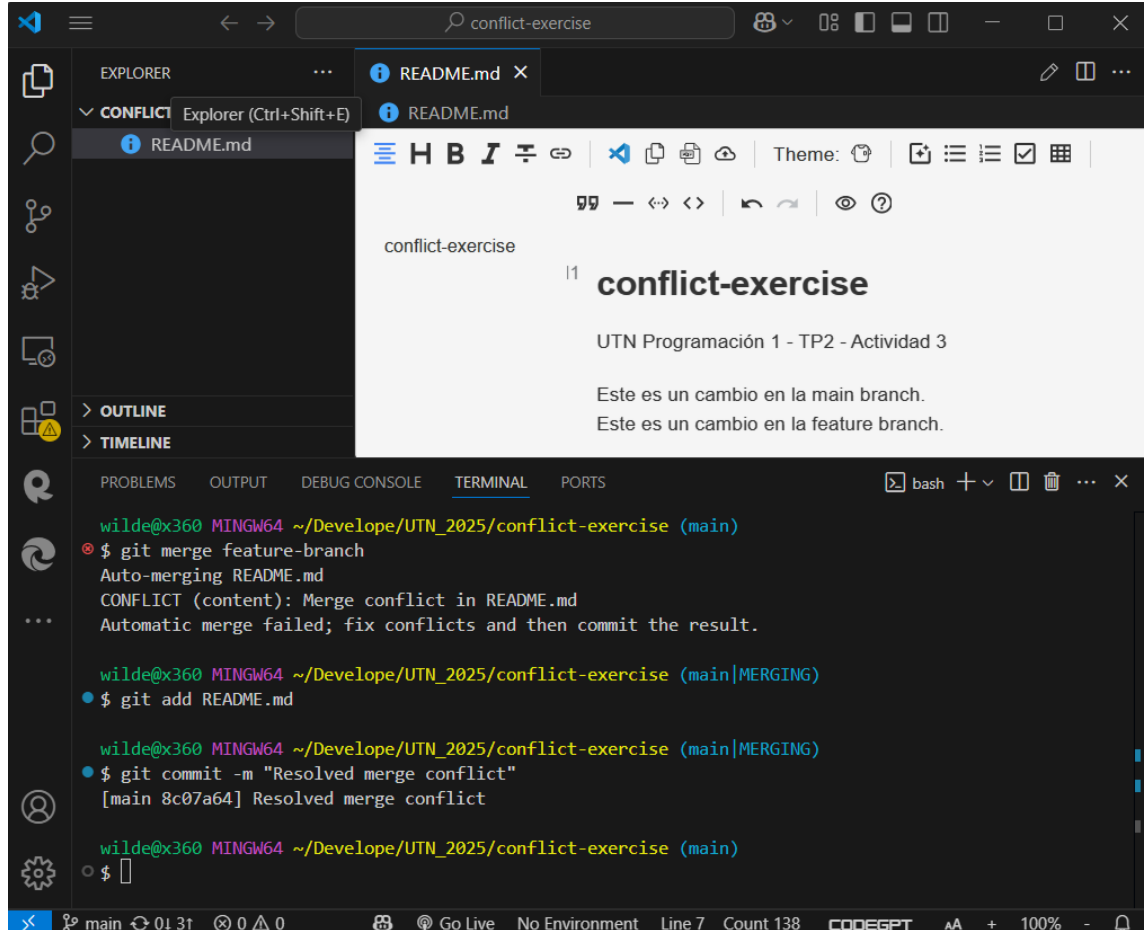
Este es un cambio en la feature branch.

```
>>>>>>> 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 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"
```



```
wilde@x360 MINGW64 ~/Develope/UTN_2025/conflict-exercise (main)
$ 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.

wilde@x360 MINGW64 ~/Develope/UTN_2025/conflict-exercise (main|MERGING)
$ git add README.md

wilde@x360 MINGW64 ~/Develope/UTN_2025/conflict-exercise (main|MERGING)
$ git commit -m "Resolved merge conflict"
[main 8c07a64] Resolved merge conflict

wilde@x360 MINGW64 ~/Develope/UTN_2025/conflict-exercise (main)
$
```

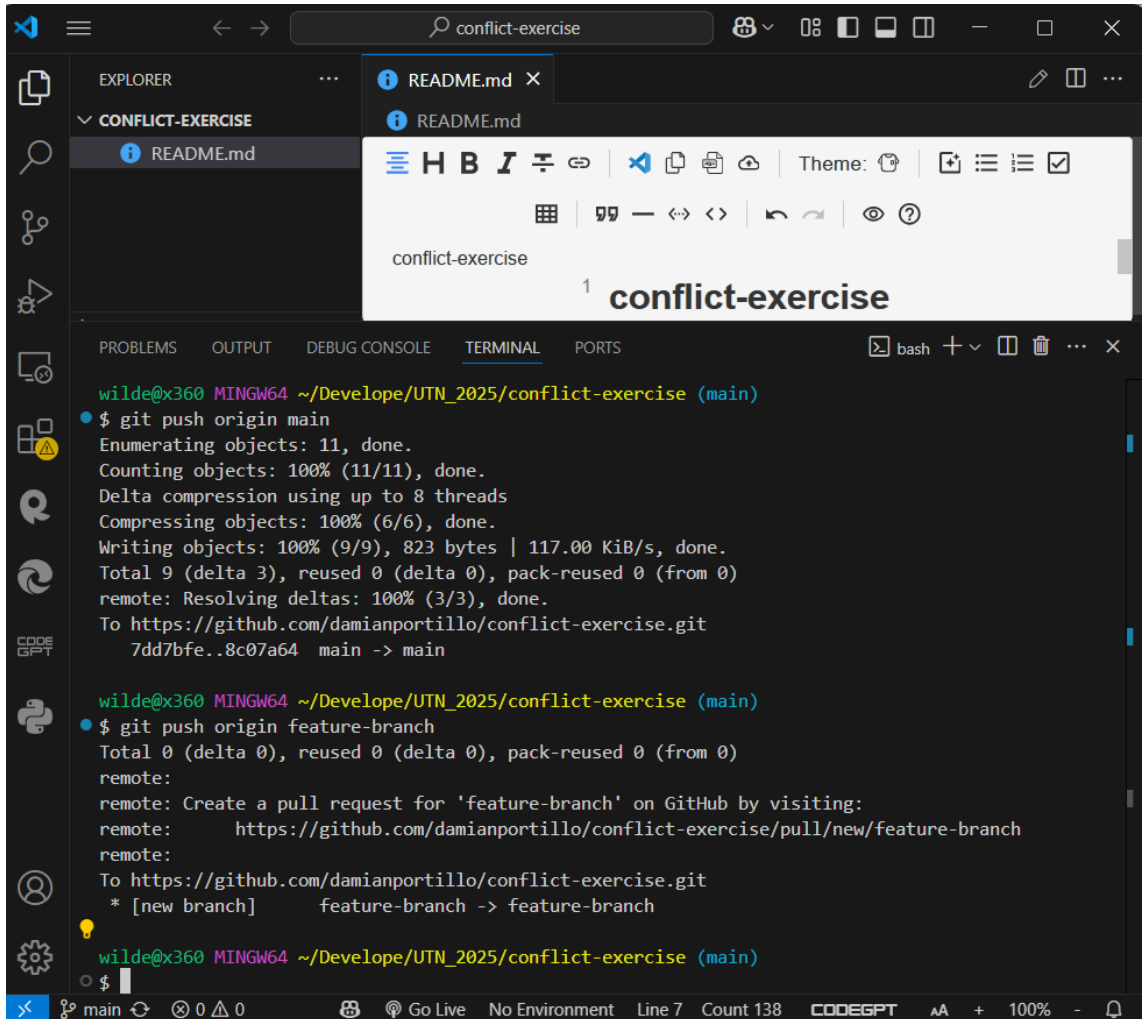
Paso 7: Subir los cambios a GitHub

- 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



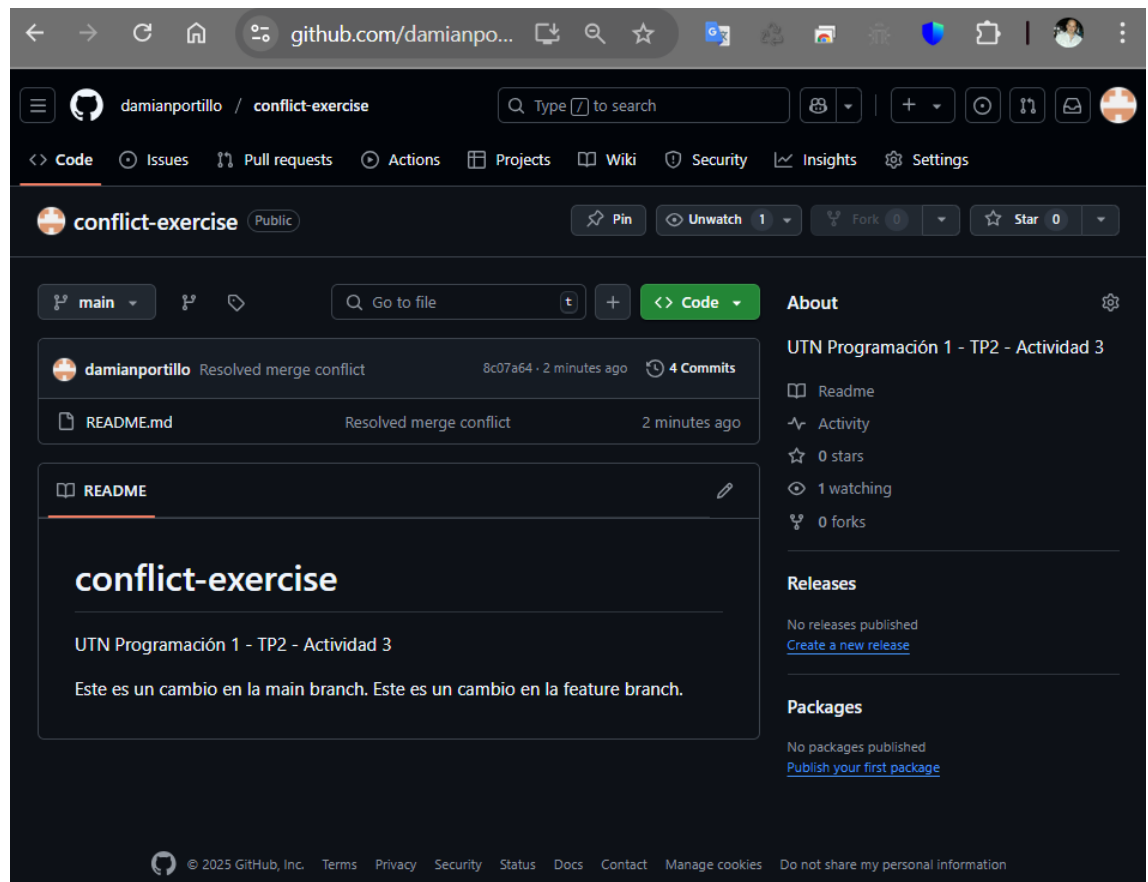
```
wilde@x360 MINGW64 ~/Develope/UTN_2025/conflict-exercise (main)
$ git push origin main
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 8 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (9/9), 823 bytes | 117.00 KiB/s, done.
Total 9 (delta 3), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (3/3), done.
To https://github.com/damianportillo/conflict-exercise.git
7dd7bfe..8c07a64  main -> main

wilde@x360 MINGW64 ~/Develope/UTN_2025/conflict-exercise (main)
$ git push origin feature-branch
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'feature-branch' on GitHub by visiting:
remote:   https://github.com/damianportillo/conflict-exercise/pull/new/feature-branch
remote:
To https://github.com/damianportillo/conflict-exercise.git
 * [new branch]      feature-branch -> feature-branch

wilde@x360 MINGW64 ~/Develope/UTN_2025/conflict-exercise (main)
$
```

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.

