

Trabajo Practico N°2- Git y GitHub

Actividad 1:

Contestar las siguientes preguntas utilizando las guías y documentación proporcionada.

¿Qué es GitHub?

GitHub es una plataforma en la nube, que permite almacenar repositorios de código y otros, colaborar en proyectos con otros desarrolladores mediante funciones como pull requests, gestionar el control de versiones; sirviendo esencialmente como un espacio de trabajo remoto para equipos de desarrollo y proyectos de software.

¿Cómo crear un repositorio en GitHub?

- Entra a GitHub (con tu cuenta, sino crear una) y haz clic en + → New repository.
- Poner nombre al repositorio, elegir visibilidad (público/privado) y marca Add a README.
- Clic en Create repository.

¿Cómo crear una rama en Git?

Para crear una rama se escribe en la terminal: `git branch nombre` ; donde “nombre” es el nombre de la rama que se esta creando.

¿Cómo cambiar a una rama en Git?

Para cambiar de ra se escribe en la terminal: `git checkout nombre` ; donde “nombre” es el nombre de la rama a la cual queremos dirigirnos.

¿Cómo fusionar ramas en Git?

Primero no colocamos en una rama a la cual queremos fusionar otra rama, una vez en la rama se escribe en la terminal: `git merge nombre` ; donde “nombre” es el nombre de la rama que queremos fusionar.

¿Cómo crear un commit en Git?

Para crear un commit se escribe en la terminal: `git commit -m “descripcion”` ; donde descripción es la descripción que podemos hacer del commit.

¿Cómo enviar un commit a GitHub?

Para enviar un commit a GitHub se escribe en la terminal: `git push -u origin master` **¿Qué es un repositorio remoto?**

Un repositorio remoto es una versión almacenada en un servidor (como GitHub, GitLab o Bitbucket) de un proyecto con control de versiones Git, esto permite colaborar con otros desarrolladores, sincronizar cambios entre equipos y servir como respaldo del código.

¿Cómo agregar un repositorio remoto a Git?

Para hacerlo se escribe en la terminal: `git remote add origin URLdelRepositorio` **¿Cómo**

empujar cambios a un repositorio remoto?

Para actualizar cambios en el repositorio remoto se escribe en la terminal: `git push -u origin master`

¿Cómo tirar de cambios de un repositorio remoto?

Para obtener los últimos cambios del repositorio remoto en el repositorio local se escribe en la terminal: `git pull origin nom-de-rama` **¿Qué es un fork de repositorio?**

Un fork es una copia independiente de un repositorio de GitHub (o similar) que se aloja en mi cuenta personal, permitiéndome modificar el código sin afectar el proyecto original.

¿Cómo crear un fork de un repositorio?

Para esto se va primero al repositorio remoto, luego se busca el botón que dice fork, se le da click, se elige la cuenta de destino (otro repositorio remoto), con esto ya se tiene una copia del repositorio.

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

Una vez realizado los cambios o mejoras en mi fork, en la pestaña Pull Requests se le da click en New Pull Requests, se selecciona la rama del repositorio original a cambiary la rama de mi repositorio que propongo con cambios; se describen los cambios propuestos El dueño del proyecto revisará la request y decidirá si aceptarlo, sugerir cambios o rechazarlo.

¿Cómo aceptar una solicitud de extracción?

Primero se va al repositorio, a la pestaña Pull Requests, se abre el PR para revisarlo, si todo esta correcto se le da click en merge pull rquests, se confirma el merge, la plataforma cerrara automáticamente el PR, los cambios se integrarán y el historial mostrará el PR como parte del proyecto.

¿Qué es una etiqueta en Git?

En Git, una etiqueta (tag) es una referencia estática que marca un punto específico en el historial de commits (como una versión estable, lanzamiento o hito importante). A diferencia de las ramas, las etiquetas no cambian después de crearse.

¿Cómo crear una etiqueta en Git?

Para esto se escribe en la terminal: `git tag nombre-etiqueta` **¿Cómo enviar**

una etiqueta a GitHub?

Para esto se escribe en la terminal: `git push origin nombre-etiqueta`

¿Qué es un historial de Git?

El historial de Git es el registro completo de todos los commits (cambios guardados) realizados en un repositorio, mostrando la evolución del proyecto a lo largo del tiempo.

¿Cómo ver el historial de Git?

Para esto se escribe en la terminal: `git log` **¿Cómo buscar**

en el historial de Git?

Se puede buscar de varias formas ej:

Por mensaje: `git log --grep="texto"` , donde texto es lo que se quiere buscar dentro de los commits

Por nombre: `git log --author="nombre"` , donde nombre es el nombre del que hizo el commits

¿Cómo borrar el historial de Git?

Se debe eliminar el archivo `.git` que se genera en el repositorio.

¿Qué es un repositorio privado en GitHub?

Es un proyecto alojado en la plataforma cuya visibilidad y acceso están restringidos únicamente a usuarios específicos autorizados por el propietario.

¿Cómo crear un repositorio privado en GitHub?

Cuando se crea el repositorio se debe elegir el icono de privado en vez de público.

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

Primero ir a seting, ir a collaborators, hacer click en add people, poner las credenciales de la persona o personas invitadas, elegir el nivel de permiso (read, write, o admin) y por ultimo hacer click en Add [usuario] to this repository.

¿Qué es un repositorio público en GitHub?

Es un proyecto abierto, accesible para cualquier usuario de internet, donde el código fuente, el historial de cambios y las contribuciones son visibles sin restricciones.

¿Cómo crear un repositorio público en GitHub?

Cuando se crea el repositorio se debe elegir el icono de público en vez de privado.

¿Cómo compartir un repositorio público en GitHub?

La forma más sencilla es pasar una copiar de la URL del repositorio en el navegador.

Actividad 2:

<https://github.com/Torres7417/tp2-actividad2>

Actividad 3:

<https://github.com/Torres7417/conflict-exercise>

Paso 1:

The screenshot shows the GitHub 'New repository' page. The repository is set to be 'Public'. Under 'Initialize this repository with:', the 'Add a README file' checkbox is checked. The '.gitignore' template is set to 'None'. The license is set to 'None'. The default branch is 'main'. A green 'Create repository' button is at the bottom right.

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.](#)

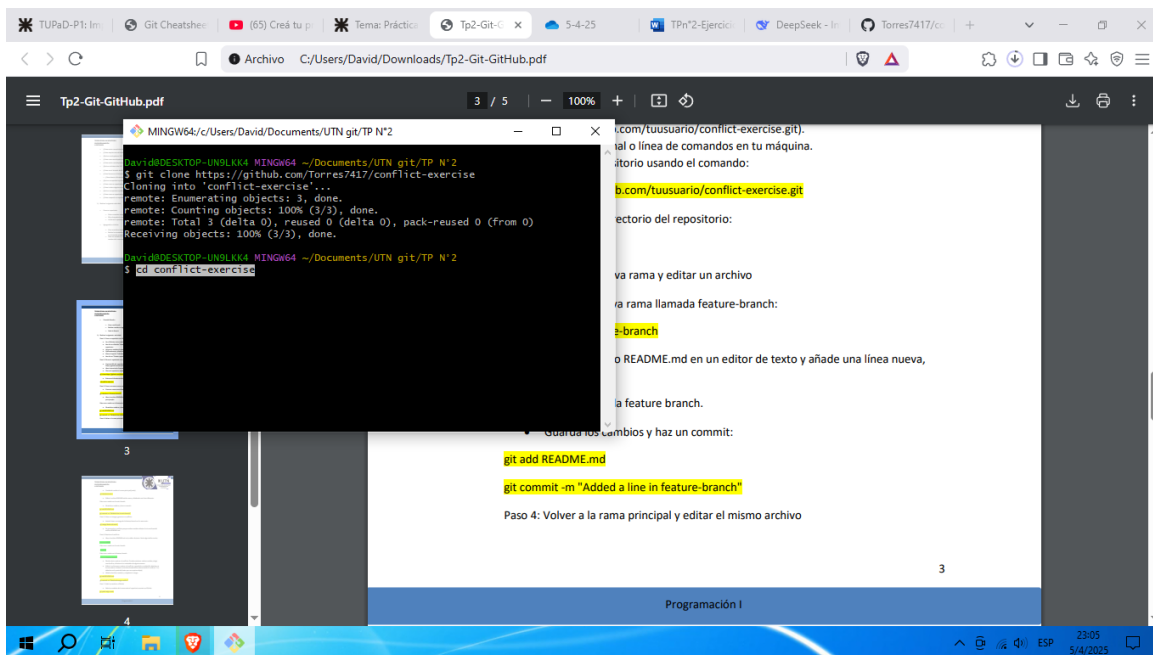
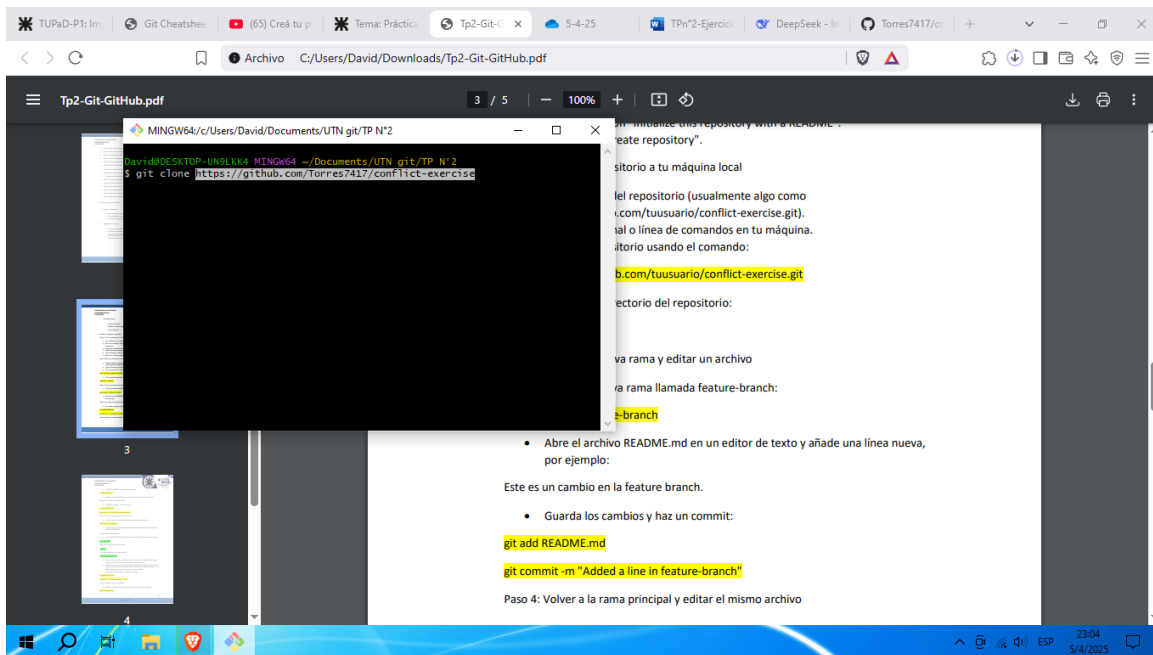
Choose a license
License: None
A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

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

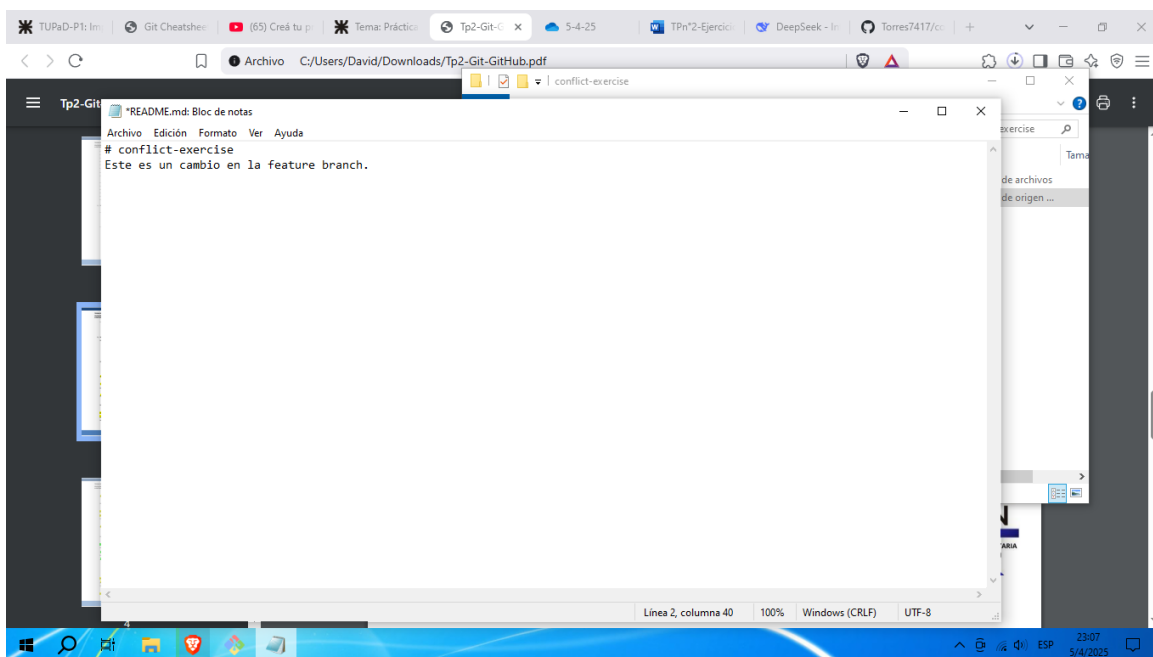
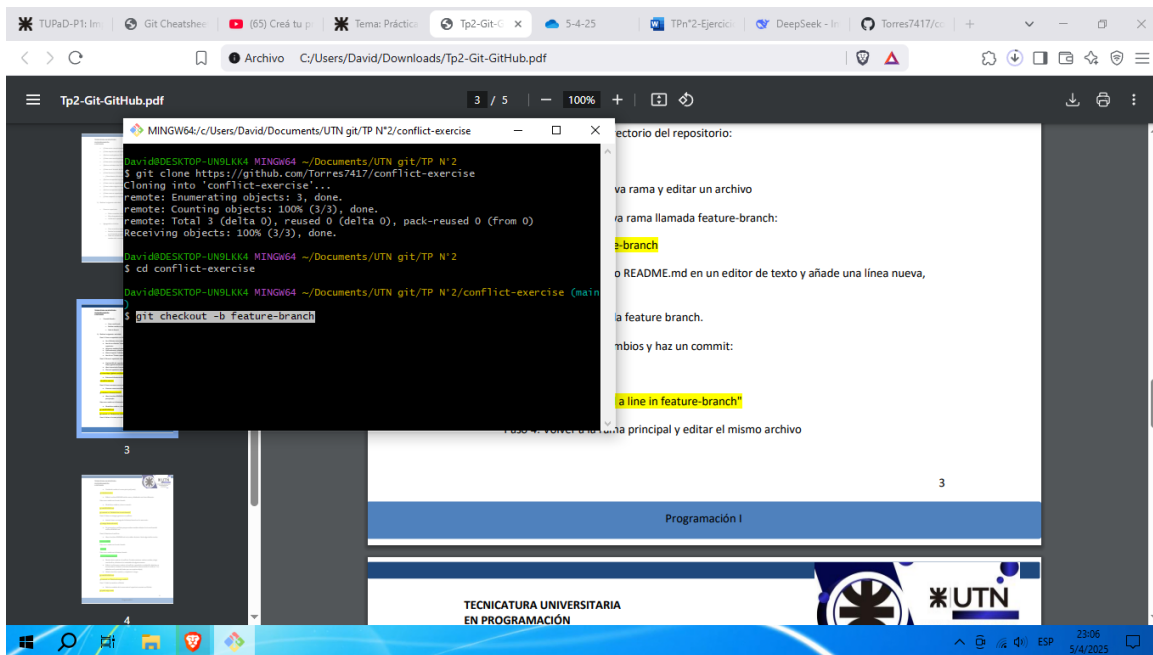
☐ You are creating a public repository in your personal account.

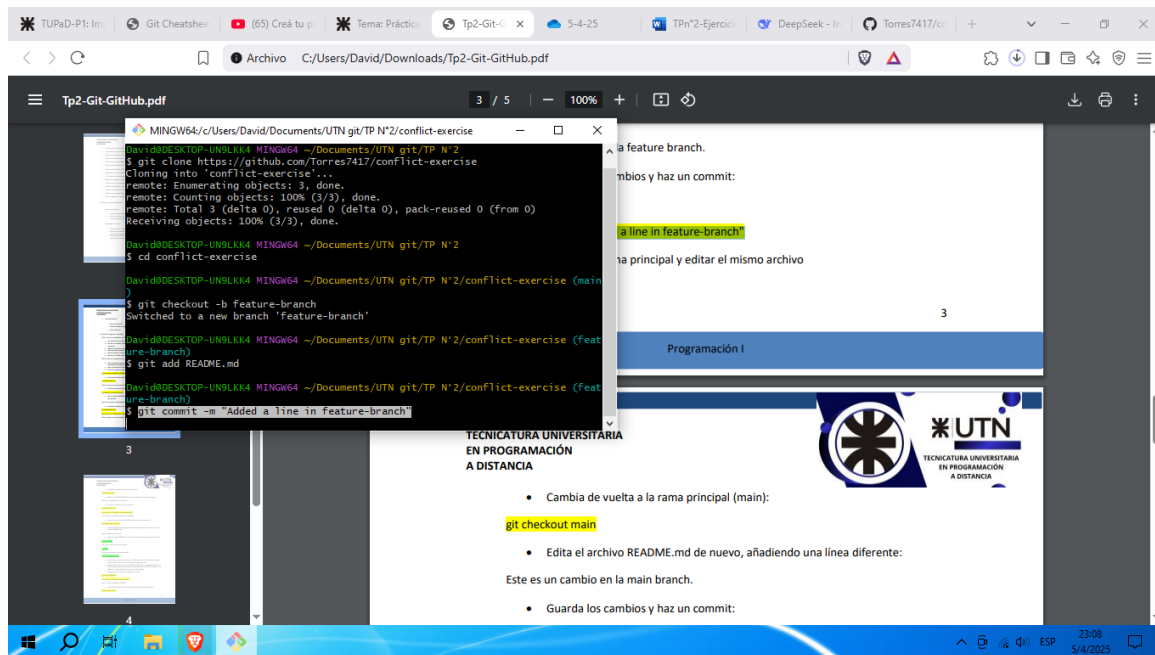
Create repository

Paso 2:

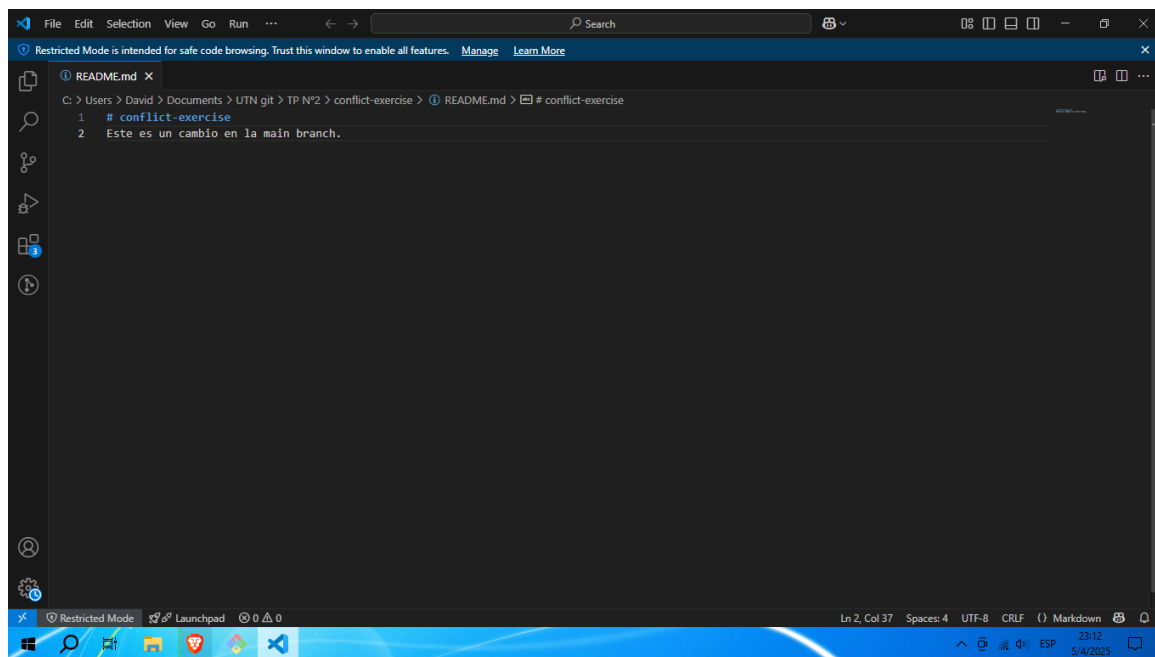


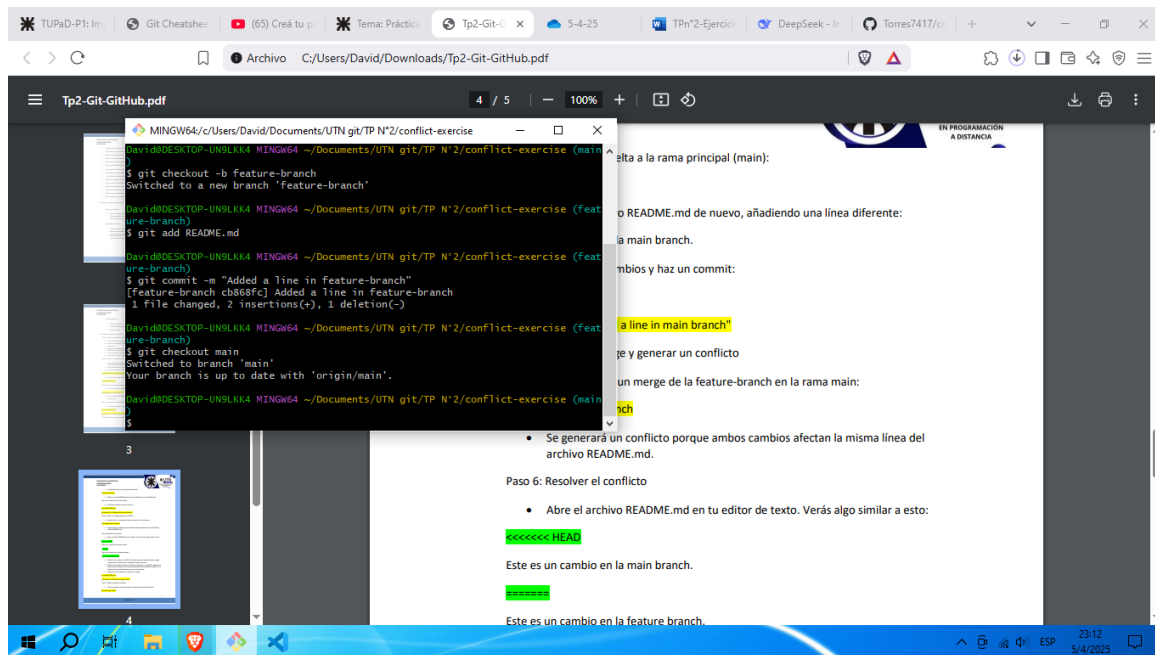
Paso 3:



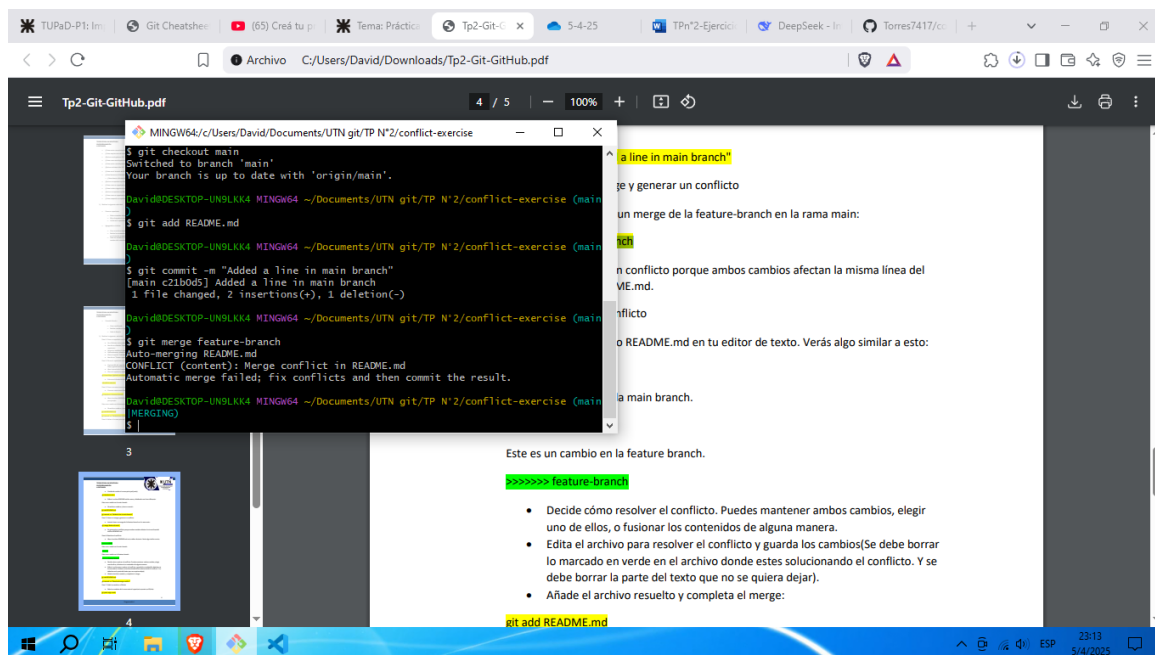


Paso 4:

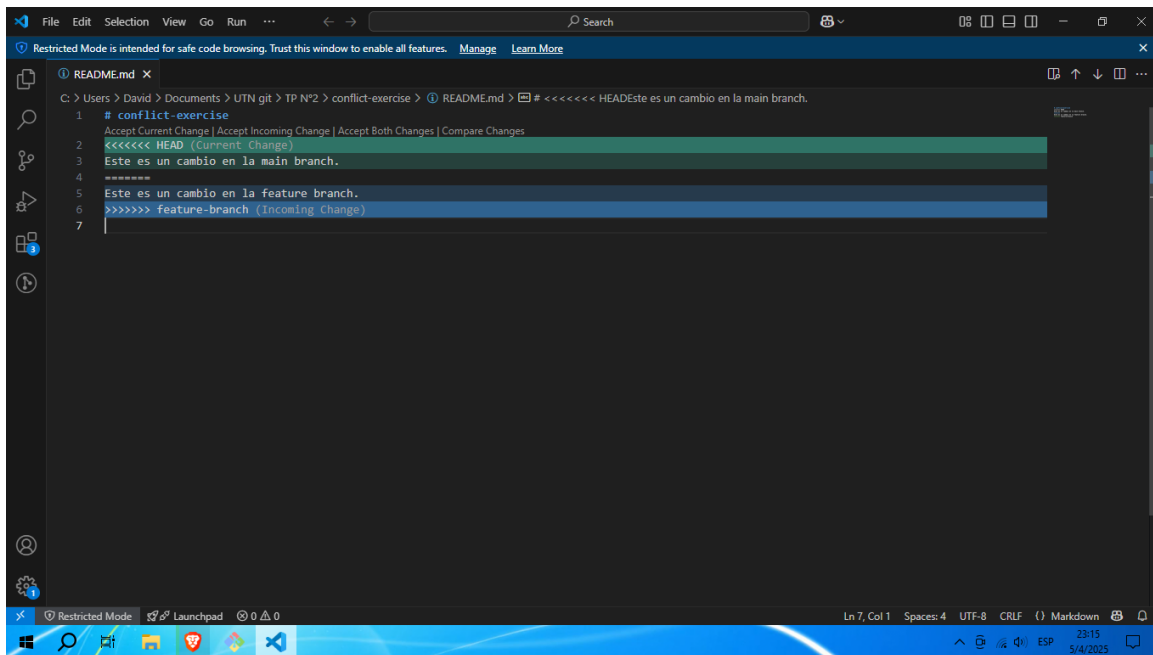




Paso 5:



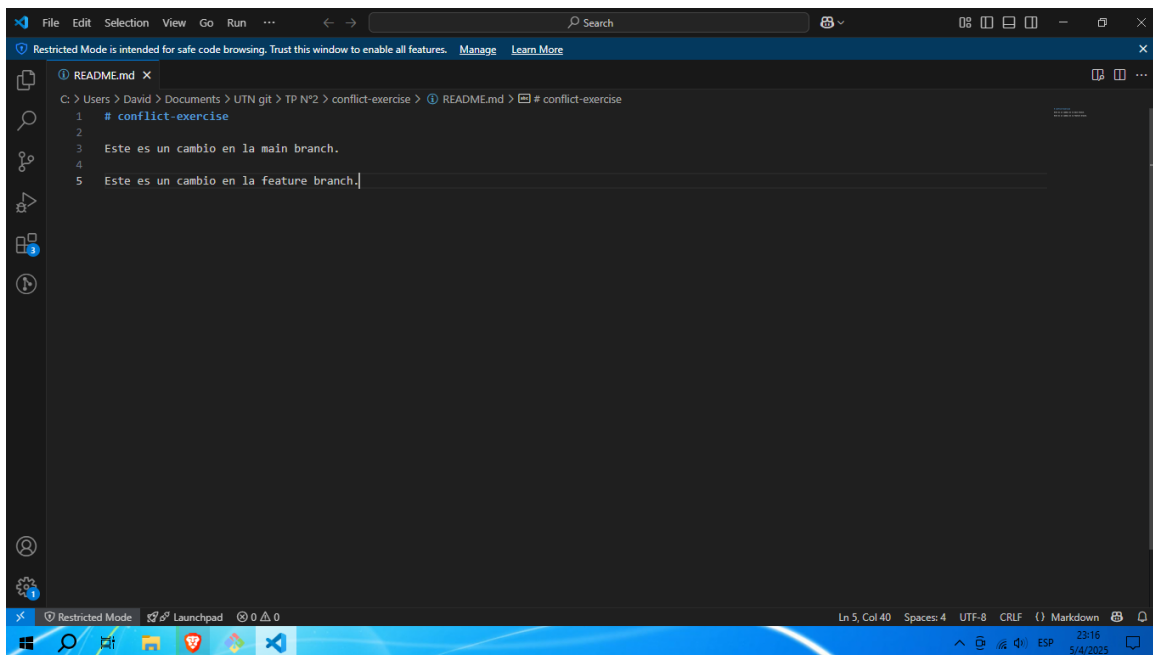
Paso 6:



This screenshot shows a Visual Studio Code editor window with a file named `README.md` open. The file content is as follows:

```
1 # conflict-exercise
2 Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
3 <<<<<< HEAD (Current Change)
4 Este es un cambio en la main branch.
5 =====
6 Este es un cambio en la feature branch.
7 >>>>>> feature-branch (Incoming Change)
```

The editor interface includes a menu bar (File, Edit, Selection, View, Go, Run, ...), a search bar, and a status bar at the bottom indicating 'Ln 7, Col 1', 'Spaces: 4', 'UTF-8', 'CRLF', and 'Markdown'. The Windows taskbar is visible at the bottom of the screen.



This screenshot shows the same Visual Studio Code editor window after the conflict has been resolved. The file content is now:

```
1 # conflict-exercise
2
3 Este es un cambio en la main branch.
4
5 Este es un cambio en la feature branch.
```

The status bar at the bottom now indicates 'Ln 5, Col 40', 'Spaces: 4', 'UTF-8', 'CRLF', and 'Markdown'. The Windows taskbar remains visible at the bottom.

```
MINGW64/c/Users/David/Documents/UTN git/TP N°2/conflict-exercise
David@DESKTOP-UN9LKK4 MINGW64 ~/Documents/UTN git/TP N°2/conflict-exercise (main)
$ git checkout -b feature-branch
Switched to a new branch 'feature-branch'

David@DESKTOP-UN9LKK4 MINGW64 ~/Documents/UTN git/TP N°2/conflict-exercise (feature-branch)
$ git add README.md

David@DESKTOP-UN9LKK4 MINGW64 ~/Documents/UTN git/TP N°2/conflict-exercise (feature-branch)
$ git commit -m "Added a line in feature-branch"
[feature-branch cb868fc] Added a line in feature-branch
1 file changed, 2 insertions(+), 1 deletion(-)

David@DESKTOP-UN9LKK4 MINGW64 ~/Documents/UTN git/TP N°2/conflict-exercise (feature-branch)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

David@DESKTOP-UN9LKK4 MINGW64 ~/Documents/UTN git/TP N°2/conflict-exercise (main)
$ git add README.md

David@DESKTOP-UN9LKK4 MINGW64 ~/Documents/UTN git/TP N°2/conflict-exercise (main)
$ git commit -m "Added a line in main branch"
[main c21b0d9] Added a line in main branch
1 file changed, 2 insertions(+), 1 deletion(-)

David@DESKTOP-UN9LKK4 MINGW64 ~/Documents/UTN git/TP N°2/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.

David@DESKTOP-UN9LKK4 MINGW64 ~/Documents/UTN git/TP N°2/conflict-exercise (main)
$ git add README.md

David@DESKTOP-UN9LKK4 MINGW64 ~/Documents/UTN git/TP N°2/conflict-exercise (main)
$ git commit -m "Resolved merge conflict"
[main 8aceee9] Resolved merge conflict

David@DESKTOP-UN9LKK4 MINGW64 ~/Documents/UTN git/TP N°2/conflict-exercise (main)
$
```

Paso 7:

The video frame shows a terminal window with the following output:

```
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.

David@DESKTOP-UN9LKK4 MINGW64 ~/Documents/UTN git/TP N°2/conflict-exercise (main)
$ git add README.md

David@DESKTOP-UN9LKK4 MINGW64 ~/Documents/UTN git/TP N°2/conflict-exercise (main)
$ git commit -m "Resolved merge conflict"
[main 8aceee9] Resolved merge conflict

David@DESKTOP-UN9LKK4 MINGW64 ~/Documents/UTN git/TP N°2/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), 840 bytes | 840.00 KiB/s, done.
Total 9 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), done.
To https://github.com/Torres7417/conflict-exercise
cc7305ff..8aceee9 main -> main
```

The document titled 'Tp2-Git-GitHub.pdf' shows the following text:

4

Programación I

UTN

TECNICATURA UNIVERSITARIA EN PROGRAMACIÓN A DISTANCIA

5

git push origin feature-branch

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.

Paso 8:

