

**Alumna: Estefanía Avalos**

## **Comisión: 24**

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

- ¿Qué es GitHub?

*Github es una plataforma en la cual se puede colaborar en proyectos a través de repositorios utilizando el sistema de control de versiones: Git*

- ¿Cómo crear un repositorio en GitHub?

- 1) *En el costado izquierdo de la pantalla principal como así también clickeando en la foto de nuestra cuenta que abre un menú -> mis repositorios -> nuevo*
- 2) *Se agrega el nombre, una descripción opcionalmente, definir si es público o privado, definir si queremos o no agregar un readme/gitignore.*
- 3) *Apretar el botón "crear repositorio"*

- ¿Cómo crear una rama en Git?

*Con el comando `git branch <nombre-de-la-branch>`*

- ¿Cómo cambiar a una rama en Git?

*Con el comando `git checkout <nombre-de-la-branch>`*

- ¿Cómo fusionar ramas en Git?

*Ubicados en la rama en la cual queremos traer cambios de otra se corre el comando `git merge <nombre-de-la-rama>`*

- ¿Cómo crear un commit en Git?

*`git commit -m <nombre-del-commit>`*

- ¿Cómo enviar un commit a GitHub?

*Con el comando `git push`*

- ¿Qué es un repositorio remoto?

*Es una versión del repositorio que no es local, que está alojado en la nube o servidor*

- ¿Cómo agregar un repositorio remoto a Git?

*Usando el comando: `git remote add origin https://github.com/usuario/repo.git`*

- ¿Cómo empujar cambios a un repositorio remoto?

*Con el comando `git push`*

- ¿Cómo tirar de cambios de un repositorio remoto?

*Con el comando `git pull`*

- ¿Qué es un fork de repositorio?

*Es un copia de un repositorio que se crea en una cuenta propia y en la cual se puede trabajar sin generar cambios en el original*

- ¿Cómo crear un fork de un repositorio?

*En la página principal del repositorio ir a fork -> create new fork*

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

- 1) *Se ingresa a la branch, puede aparecer un compare and pull request o bien ingresando a la pestaña pull request new.*
- 2) *La base será la rama principal (main, master) y la que se va a agregar es aquella rama que subimos y en la que estuvimos trabajando.*
- 3) *Después de chequear si hay algún conflicto y clickea el botón "create pull request"*

- ¿Cómo aceptar una solicitud de extracción?

*Se ingresa al pull request, se verifican los documentos que van a cambiarse en files changed y en conversation tenemos la opción de merge una vez que verificamos que los cambios están como deberían*

- ¿Qué es una etiqueta en Git?

*Una marcador de hitos en mi historial de versiones*

- ¿Cómo crear una etiqueta en Git?

*Con el comando git tag <nombre>*

- ¿Cómo enviar una etiqueta a GitHub?

*Con el comando git push origin <nombre-de-la-etiqueta>*

- ¿Qué es un historial de Git?

*Es el registro de todos los cambios hechos en un proyecto*

- ¿Cómo ver el historial de Git?

*Con el comando git log*

- ¿Cómo buscar en el historial de Git?

*git log --variable="valor"*

*git reflog para mostrar el historial de todos los cambios*

- ¿Cómo borrar el historial de Git?

*git filter-branch o git filter-repo se usan para reescribir commits/cambios*

- ¿Qué es un repositorio privado en GitHub?

*Es un repositorio que no está disponible para que lo vea todo el mundo, sino que solo aquellos a quienes se lo permite el dueño del repositorio*

- ¿Cómo crear un repositorio privado en GitHub?

*Al crear un repositorio te da la opción de que sea público o privado. Seleccionar la opción "privado" antes de crear el repositorio*

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

*En el repositorio -> settings -> collaborators -> indicar el usuario*

- ¿Qué es un repositorio público en GitHub?

*Un repositorio que está a la mano de cualquier persona que entre a mi github o que tenga su URL*

- ¿Cómo crear un repositorio público en GitHub?

*Al crear un repositorio te da la opción de que sea público o privado. Seleccionar la opción “público” antes de crear el repositorio*

- ¿Cómo compartir un repositorio público en GitHub?

*Se comparte la URL del repositorio*

## 2) Realizar la siguiente actividad:

- Crear un repositorio.
  - Dale un nombre al repositorio.
  - Elige 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 \*

 tefiavalos

/

Repository name \*


tp-2-programacion

✔ tp-2-programacion is available.


Great repository names are short and memorable. Need inspiration? How about **crispy-fortnight** ?

Description (optional)

---

☒  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

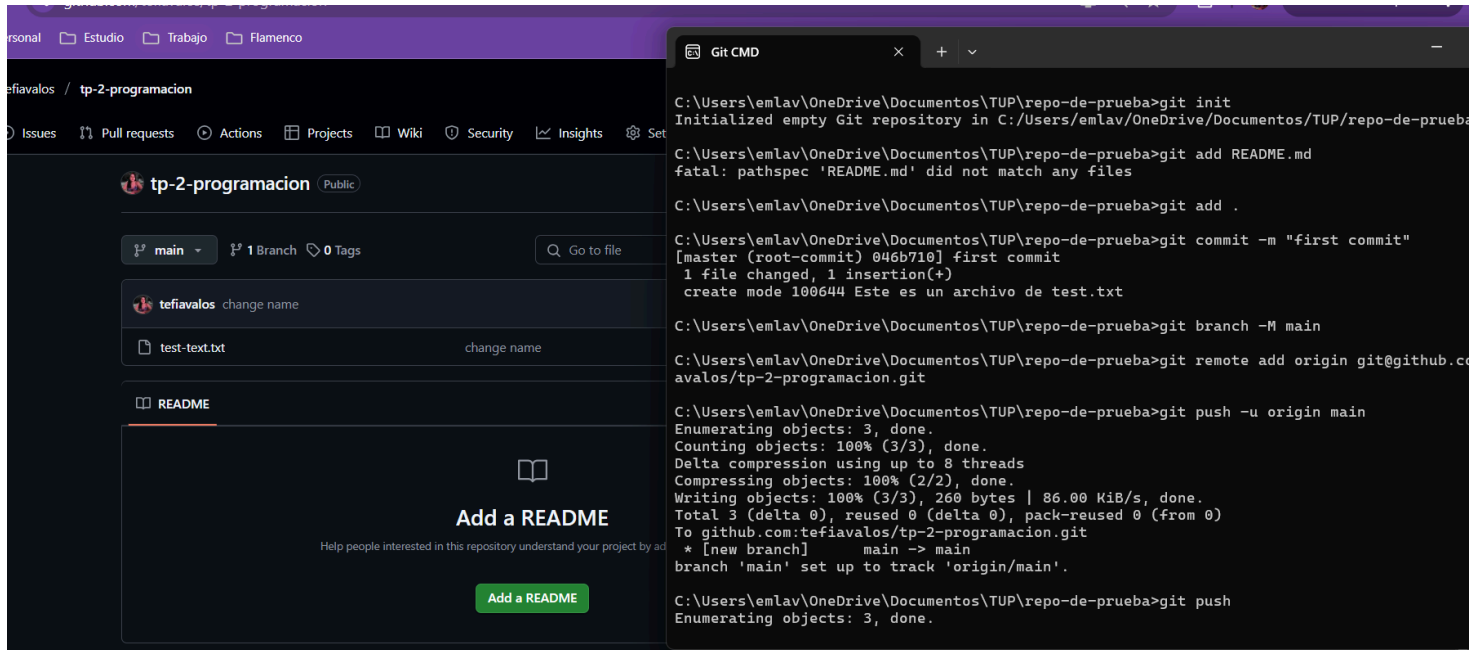
.gitignore template: None

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

Choose a license

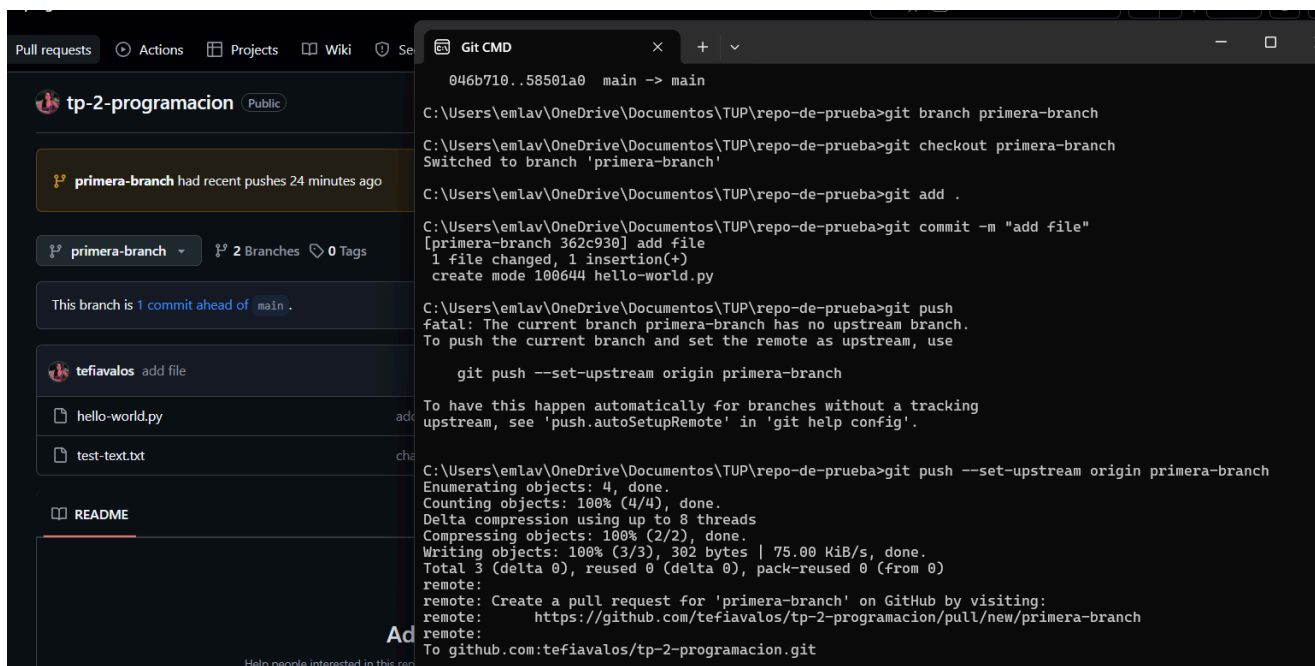
- Agregando un Archivo

- 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



### 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".

Owner \* / Repository name \*

tefiavalos / conflict-exercise

✓ conflict-exercise is available.

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

Description (optional)

This is the exercise 3 of TP2 - Programación 1 -TUP

☒ 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](#).

*i* 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
```

- Entra en el directorio del repositorio:

## cd conflict-exercise

```
C:\Users\emlav\OneDrive\Documentos\TUP>git clone https://github.com/tefiavalos/conflict-exercise
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.

C:\Users\emlav\OneDrive\Documentos\TUP>cd conflict-exercise

C:\Users\emlav\OneDrive\Documentos\TUP\conflict-exercise>git checkout -b "feature-branch"
Switched to a new branch 'feature-branch'
```

### 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"
```

```
C:\Users\emlav\OneDrive\Documentos\TUP>cd conflict-exercise

C:\Users\emlav\OneDrive\Documentos\TUP\conflict-exercise>git checkout -b "feature-branch"
Switched to a new branch 'feature-branch'

C:\Users\emlav\OneDrive\Documentos\TUP\conflict-exercise>git add .

C:\Users\emlav\OneDrive\Documentos\TUP\conflict-exercise>git commit -m "Added a line in feature-branch"
[feature-branch 9400162] Added a line in feature-branch
1 file changed, 3 insertions(+)

C:\Users\emlav\OneDrive\Documentos\TUP\conflict-exercise>git checkout main
Switched to branch 'main'
```

### 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"
```

```
C:\Users\emlav\OneDrive\Documentos\TUP\conflict-exercise>git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

C:\Users\emlav\OneDrive\Documentos\TUP\conflict-exercise>git add .

C:\Users\emlav\OneDrive\Documentos\TUP\conflict-exercise>git commit -m "Added a line in main"
[main 91ef1f5] Added a line in main
1 file changed, 3 insertions(+)
```

## 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.

## Paso 6: Resolver el conflicto

```
C:\Users\emlav\OneDrive\Documentos\TUP\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.

C:\Users\emlav\OneDrive\Documentos\TUP\conflict-exercise>git add .

C:\Users\emlav\OneDrive\Documentos\TUP\conflict-exercise>git commit -m "Conflict resolved"
[main ef5c8d8] Conflict resolved
```

- 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"
```

## Paso 7: Subir los cambios a GitHub

- Sube los cambios de la rama main al repositorio remoto en GitHub:

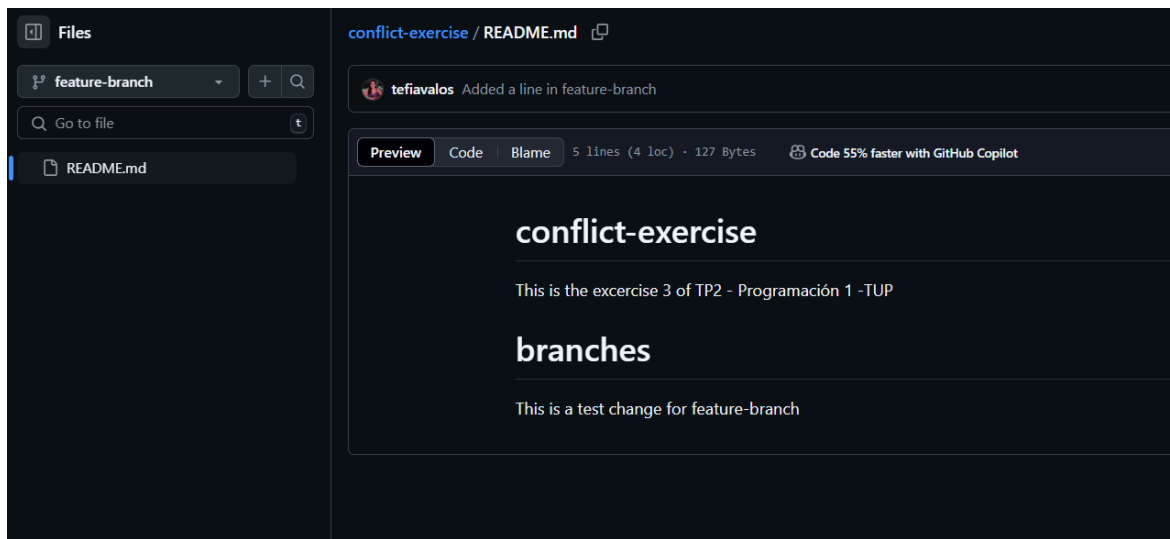
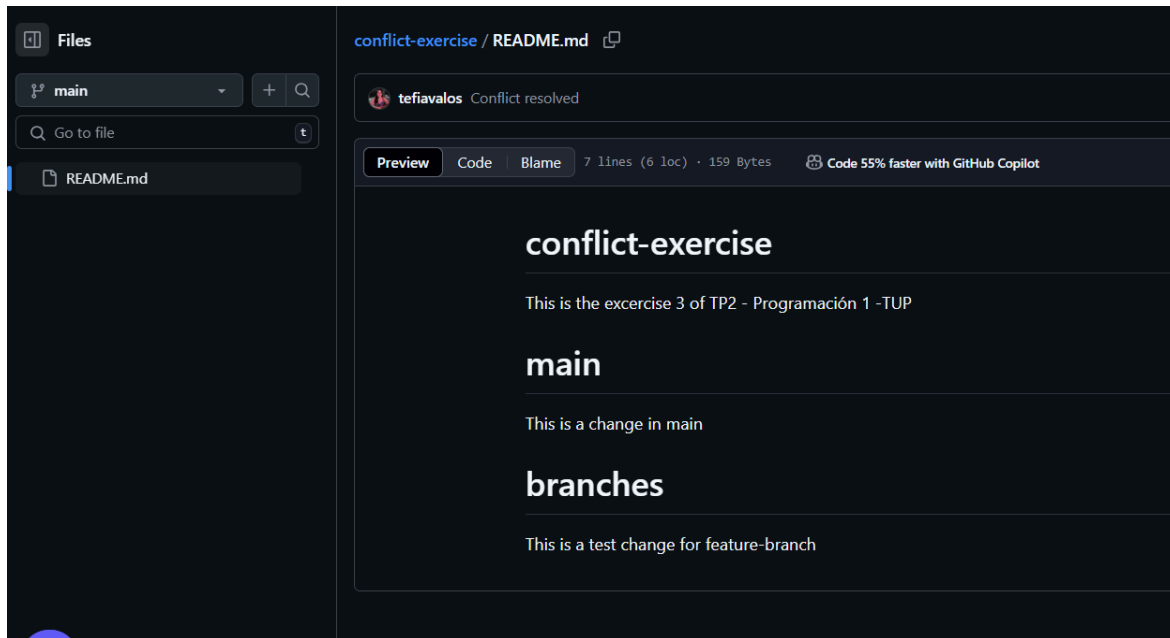
```
git push origin main
```

- También sube la feature-branch si deseas:

```
git push origin feature-branch
```

## 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.



```
C:\Users\emlav\OneDrive\Documentos\TUP>cd conflict-exercise

C:\Users\emlav\OneDrive\Documentos\TUP\conflict-exercise>git checkout -b "feature-branch"
Switched to a new branch 'feature-branch'

C:\Users\emlav\OneDrive\Documentos\TUP\conflict-exercise>git add .

C:\Users\emlav\OneDrive\Documentos\TUP\conflict-exercise>git commit -m "Added a line in feature-branch"
[feature-branch 9400162] Added a line in feature-branch
 1 file changed, 3 insertions(+)

C:\Users\emlav\OneDrive\Documentos\TUP\conflict-exercise>git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

C:\Users\emlav\OneDrive\Documentos\TUP\conflict-exercise>git add .

C:\Users\emlav\OneDrive\Documentos\TUP\conflict-exercise>git commit -m "Added a line in main"
[main 91ef1f5] Added a line in main
 1 file changed, 3 insertions(+)

C:\Users\emlav\OneDrive\Documentos\TUP\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.

C:\Users\emlav\OneDrive\Documentos\TUP\conflict-exercise>git add .

C:\Users\emlav\OneDrive\Documentos\TUP\conflict-exercise>git commit -m "Conflict resolved"
[main ef5c8d8] Conflict resolved

C:\Users\emlav\OneDrive\Documentos\TUP\conflict-exercise>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.
```