

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.

Repositorio: https://github.com/gabimancini/UTN-TUPaD-P1_Mancini/tree/main/

Actividades

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

- **¿Qué es GitHub?**

GitHub es una plataforma en línea para el almacenamiento, gestión y colaboración en proyectos de software. Se basa en Git, un sistema de control de versiones que permite a los desarrolladores trabajar en equipo de manera eficiente.

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

Ir a [GitHub](#) e **iniciar sesión** con tu cuenta.

Hacer clic en el botón "+" en la esquina superior derecha.

Seleccionar **"New repository" (Nuevo repositorio)**

Escribir un nombre único para tu repositorio.

Description (Descripción): Opcional, puedes agregar información sobre el proyecto.

Visibility (Visibilidad):

- **Public:** Cualquier persona puede verlo.

- **Private:** Solo tú y los colaboradores pueden verlo.

Initialize this repository with a README: Si lo marcas, se creará un archivo **README.md** automáticamente.

- **¿Cómo crear una rama en Git?**

git branch nombre-de-rama

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

git checkout nombre-de-rama

- **¿Cómo fusionar ramas en Git?**

git merge nombre-de-rama

- **¿Cómo crear un commit en Git?**

git commit -m "Mensaje a enviar"

- **¿Cómo enviar un commit a GitHub?**

git push

- **¿Qué es un repositorio remoto?**

Un repositorio remoto es una versión de tu repositorio que está almacenada en un servidor en línea (como GitHub, GitLab o Bitbucket).

- **¿Cómo agregar un repositorio remoto a Git?**

git remote add origin url-repo

- **¿Cómo empujar cambios a un repositorio remoto?**

git push

- **¿Cómo tirar de cambios de un repositorio remoto?**

git pull

- **¿Qué es un fork de repositorio?**

Un fork es una copia de un repositorio en tu cuenta de GitHub. Se usa para modificar un proyecto sin afectar el original y se puede proponer cambios al autor original con un pull request.

- **¿Cómo crear un fork de un repositorio?**

Posicionarse en el repositorio en GitHub que se quiere hacer un fork. En la parte superior derecha de la página del repositorio, hacer clic en el botón **"Fork"**. GitHub copia el repositorio a tu cuenta..

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

- Posicionarse en el repositorio original en GitHub.
- Hacer clic en la pestaña "Pull Requests".
- Presionar el botón "New Pull Request".
- Seleccionar tu fork y la rama con los cambios.
- Escribir un título y una descripción clara de los cambios que hiciste.
- Hacer clic en "Create Pull Request".

- **¿Cómo aceptar una solicitud de extracción?**

Ir al repositorio en GitHub.
Hacer clic en la pestaña "Pull Requests".
Abrir la solicitud de extracción que se desea revisar.
Revisar los cambios y, si es necesario, deja comentarios.
Si todo está bien, presionar "Merge Pull Request".
Elegir un método: Merge commit, Squash and merge, Rebase and merge
Hacer clic en "Confirm merge".

- **¿Qué es un etiqueta en Git?**

Una **etiqueta (tag)** en Git es un marcador que se usa para señalar versiones específicas en el historial del repositorio.

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

`git tag -a nombre-etiqueta -m "Mensaje de la etiqueta"`

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

`git push origin nombre-etiqueta` o `git push origin --tags`

- **¿Qué es un historial de Git?**

El historial de Git muestra todos los cambios hechos en el repositorio. Se usa para ver commits, ramas y etiquetas.

- **¿Cómo ver el historial de Git?**

`git log`

`git log --oneline --graph --decorate --all`

`git blame archivo.txt`

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

`git reflog`

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

`git reset --hard HEAD~3`

- **¿Qué es un repositorio privado en GitHub?**

Un repositorio privado en GitHub es un proyecto que solo tú y las personas que invites pueden ver y modificar. Es útil para trabajo privado o proyectos en desarrollo.

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

Selecciona "Private" para que sea privado al crear un repositorio.

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

- Clic en "Settings" (Configuración).
- Seleccionar "Manage access".
- Clic en "Invite a collaborator".
- Escribir el nombre de usuario o correo de la persona y envía la invitación.
- La persona debe aceptar la invitación para acceder al repositorio.

- **¿Qué es un repositorio público en GitHub?**

Un repositorio público es un proyecto visible para cualquier persona en GitHub. Es ideal para proyectos de código abierto o cuando quieres compartir código con el mundo.

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

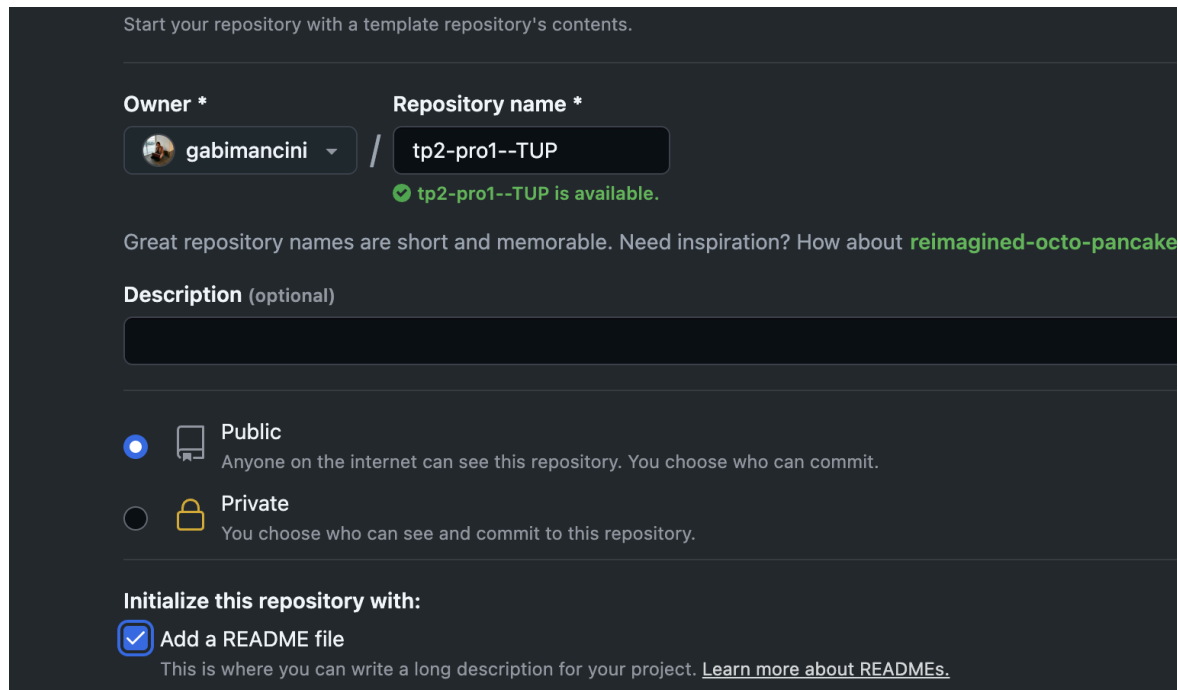
Selecciona "Public" para que sea público al crear un repositorio.

- ¿Cómo compartir un repositorio público en GitHub?

A través de la URL

2) Realizar la siguiente actividad:

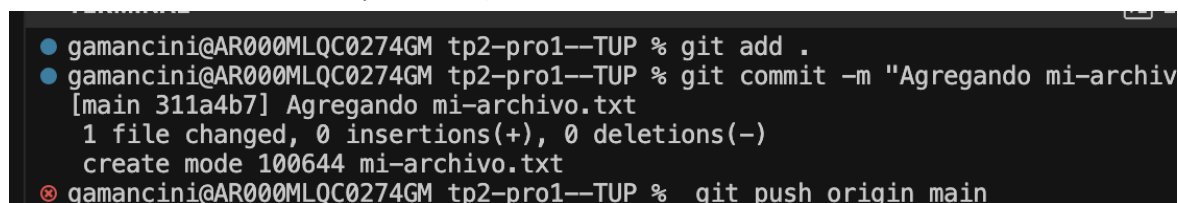
- Crear un repositorio.
 - Dale un nombre al repositorio.
 - Elije el repositorio sea público.
 - Inicializa el repositorio con un archivo.



The screenshot shows the GitHub 'Create new repository' page. At the top, it says 'Start your repository with a template repository's contents.' Below this, there are two main fields: 'Owner *' and 'Repository name *'. The 'Owner' field has a dropdown menu showing 'gabimancini' with a profile picture. The 'Repository name' field has a text input with 'tp2-pro1--TUP'. Below the repository name field, there is a green checkmark and the text 'tp2-pro1--TUP is available.' Below these fields, there is a text input for 'Description (optional)'. Further down, there are two radio button options for visibility: 'Public' (selected) and 'Private'. The 'Public' option has a description: 'Anyone on the internet can see this repository. You choose who can commit.' The 'Private' option has a description: 'You choose who can see and commit to this repository.' At the bottom, there is a section 'Initialize this repository with:' with a checked checkbox for 'Add a README file'. Below this checkbox, there is a note: 'This is where you can write a long description for your project. [Learn more about READMEs.](#)'

- Agregando un Archivo

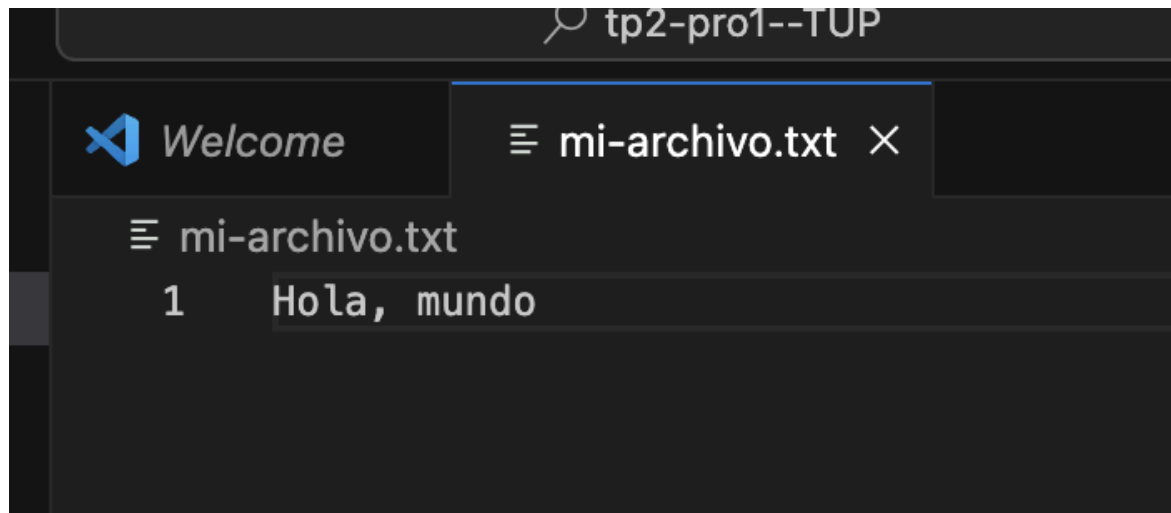
- Crea un archivo simple, por ejemplo, "mi-archivo.txt".
- Realiza los comandos git add . y git commit -m "Agregando mi-archivo.txt" en la línea de comandos.
- Sube los cambios al repositorio en GitHub con git push origin main (o el nombre de la rama correspondiente).



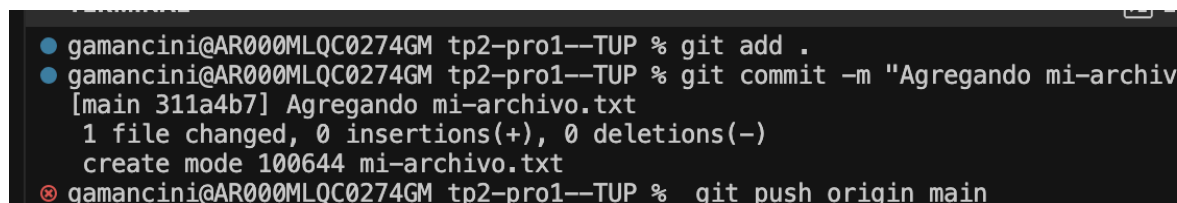
```
gamancini@AR000MLQC0274GM tp2-pro1--TUP % git add .
gamancini@AR000MLQC0274GM tp2-pro1--TUP % git commit -m "Agregando mi-archivo.txt"
[main 311a4b7] Agregando mi-archivo.txt
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 mi-archivo.txt
gamancini@AR000MLQC0274GM tp2-pro1--TUP % git push origin main
```

**TECNICATURA UNIVERSITARIA
EN PROGRAMACIÓN
A DISTANCIA**

- 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".

Start your repository with a template repository's contents.

Owner *

 gabimancini ▾

Repository name *

conflict-exercise

✔ conflict-exercise is available.

Great repository names are short and memorable. Need inspiration? How about **verbose-succot**

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.](#)

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
```

```
● gamancini@AR000MLQC0274GM ~ % git clone https://github.com/gabimancini/conflict-exercis
Cloning into 'conflict-exercise'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.
● gamancini@AR000MLQC0274GM ~ % cd conflict-exercise
● gamancini@AR000MLQC0274GM conflict-exercise % git checkout -b feature-branch
Switched to a new branch 'feature-branch'
○ gamancini@AR000MLQC0274GM conflict-exercise % █
```

- 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.

❗ README.md ✕

❗ README.md > `abc` # conflict-exercise

1 # conflict-exercise

2 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

TECNICATURA UNIVERSITARIA
EN PROGRAMACIÓN
A DISTANCIA

Programación I



3

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

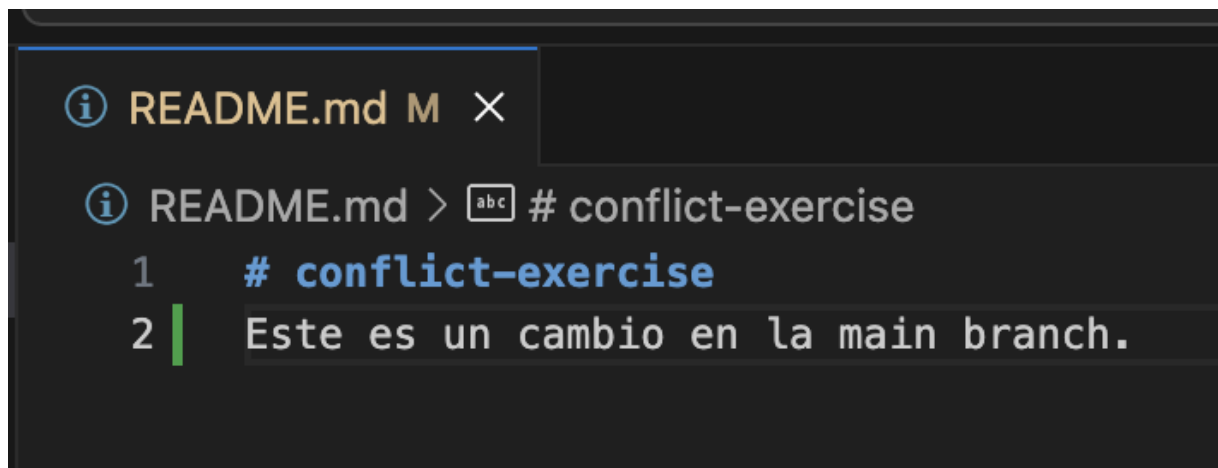
```
git checkout main
```

```
gamancini@AR000MLQC0274GM conflict-exercise % git add README.md
gamancini@AR000MLQC0274GM conflict-exercise % git commit -m "Added a line in feature-branch"

[feature-branch 7b942cb] Added a line in feature-branch
 1 file changed, 2 insertions(+), 1 deletion(-)
gamancini@AR000MLQC0274GM conflict-exercise % git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
gamancini@AR000MLQC0274GM conflict-exercise %
```

- 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
```

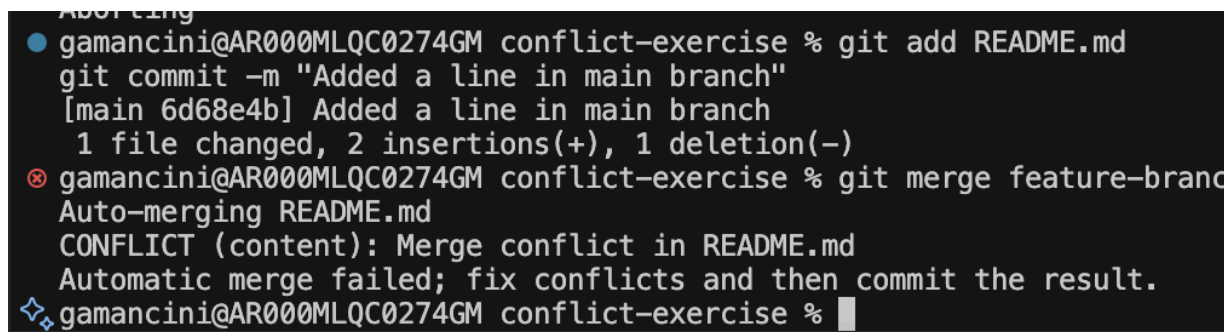
```
git commit -m "Added a line in main branch"
```

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.



```
gamancini@AR000MLQC0274GM conflict-exercise % git add README.md
git commit -m "Added a line in main branch"
[main 6d68e4b] Added a line in main branch
1 file changed, 2 insertions(+), 1 deletion(-)
gamancini@AR000MLQC0274GM 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.
gamancini@AR000MLQC0274GM conflict-exercise %
```

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"
```

Paso 7: Subir los cambios a GitHub

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

```
git push origin main
```

4

Programación I

TECNICATURA UNIVERSITARIA
EN PROGRAMACIÓN
A DISTANCIA



- También sube la feature-branch si deseas:

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

```
Automatic merge failed; fix conflicts and then commit the result.  
● gamancini@AR000MLQC0274GM conflict-exercise % git add README.md  
git commit -m "Resolved merge conflict"  
[main e9a34eb] Resolved merge conflict  
✖ gamancini@AR000MLQC0274GM conflict-exercise % git push origin 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.

