

Trabajo Práctico 2: Git y Github

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

- ¿Qué es GitHub?

GitHub es un repositorio remoto, una plataforma online o comunidad donde podemos compartir nuestros códigos o trabajos de forma pública o privada.

- ¿Cómo crear un repositorio en GitHub?

Para crear un repositorio en GitHub, primero debemos tener el nuestro creado localmente con Git, luego ingresar a GitHub con una cuenta propia y crear el repositorio remoto, asociarlo con el local y luego hacer el *push* desde Git a GitHub. (Crear una copia de nuestro trabajo local en el servidor de GitHub).

- ¿Cómo crear una rama en Git?

Para crear un rama en Git debemos utilizar el comando `git branch` más el nombre que queramos ponerle a la rama. Ej: `git branch tp2`.

¿Cómo cambiar a una rama en Git?

Para cambiar de rama debemos utilizar el comando `git checkout` más el nombre de la rama a la cual nos queremos dirigir. Ej: `git checkout tp2`.

- ¿Cómo fusionar ramas en Git?

Las ramas en Git se pueden fusionar parandose sobre una rama y utilizando el comando `git merge` más el nombre de la otra rama a la cual queremos fusionar con la primera.

Ej:

`git branch tp2` (creamos la primer rama)

`git branch tp3` (creamos la segunda rama)

`git checkout tp2` (nos paramos y confirmamos que estamos parados sobre la primer rama)

`git merge tp3` (en esta instancia fusionamos la rama tp3 a la rama tp2)

- ¿Cómo crear un commit en Git?

Para crear un commit en Git utilizamos el comando: `git commit -m "un comentario"`

Este comentario entre comillas se utiliza normalmente para saber que cambios se hicieron.

Ej: `git commit -m "se finalizo el tp2"`

- ¿Cómo enviar un commit a GitHub?

Luego de hacer un commit sobre el git, como explico en la pregunta anterior, podemos enviar el commit a github haciendo un push sobre este. Con el comando: `git push origin (nombre de la rama)`

Ej: `git push origin tp2`

- ¿Qué es un repositorio remoto?

Un repositorio remoto es una copia de nuestro trabajo o proyecto guardada en una plataforma online.

- ¿Cómo empujar cambios a un repositorio remoto?

Para empujar o hacer un push en un repositorio remoto después de guardar tus cambios y hacer el commit debemos utilizar el comando: `git push origin` más la rama usada

Ej: `git push origin pt2`

- ¿Cómo tirar de cambios de un repositorio remoto?

Para tirar o hacer pull de un repositorio remoto debemos usar el comando: `git pull origin` más la rama usada.

Ej: `git pull origin pt2`

- ¿Qué es un fork de repositorio?

Un fork es una copia de un repositorio creada en una cuenta diferente permitiendo desarrollar cambios sin afectar el original.

- ¿Cómo crear un fork de un repositorio?

Para crear un fork de un proyecto vamos a su respectiva página en este caso en GitHub y seleccionamos el botón fork, donde vamos a poder cambiar su nombre o descripción.

Así crearemos una copia de repositorio en nuestro perfil de GitHub.

• Aclaración: Varias de estas preguntas no las encontré en la guía de estudio, utilicé material fuera de la carrera.

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

Para enviar una solicitud de extracción, una vez hecho los cambios de tu fork pasar por los comandos: `git add.` + `git commit -m "cambios"` + `git push origin` (+mas la rama)

Luego en github vamos al fork que hicimos anteriormente y presionamos la opción de "Compare & pull request".

- ¿Qué es un etiqueta en Git?

La etiqueta en git es una marca que se utiliza para marcar un commit en especial para identificarlo como destacado por la diferencia de sus cambios.

- ¿Cómo crear una etiqueta en Git?

Para crear una etiqueta en git se utiliza el comando: `git tag` mas nombre de la etiqueta.

Ej: `git tag versiontp2`

- ¿Cómo enviar una etiqueta a GitHub?

Para enviar una etiqueta a GitHub utilizamos el comando: `git push origin mas el nombre de la etiqueta que estemos usando`. Ej: `git push versiontp2`

O para subir todas las etiquetas que tengamos podemos usar: `git push origin --tags`

- ¿Qué es un historial de Git?

El historial en Git es una lista de todos los cambios hechos en un trabajo o proyecto guardados con commit. Podes ver o volver a cualquiera de estos cambios o puntos.

- ¿Cómo ver el historial de Git?

El historial de Git se puede visualizar con el comando: `git log`

- ¿Cómo buscar en el historial de Git?

Para buscar en el historial de git podemos usar el comando `git log`.

`git log --grep="palabra` (para buscar por palabra dentro de los comentarios de commit)

`git log -p archivo` (para buscar commits que modificaron una linea o archivo específico)

`git log --since="fecha"` Ej: "2025-03-17" (para buscar desde una fecha en específico)

`git log --until="fecha"` Ej: "2025-07-30" (para buscar hasta una fecha en específico)

- ¿Cómo borrar el historial de Git?

Para borrar el historial de Git podemos usar el comando: `git -rf .git`

Y luego iniciar desde cero con:

`git init`

`git add .`

`git commit -m`

- ¿Qué es un repositorio privado en GitHub?

Un repositorio privado en GitHub es proyector/trabajo en el que solo vos y las personas autorizadas tienen permiso de ver o modificar.

- ¿Cómo crear un repositorio privado en GitHub?

Crear un repositorio privado en GitHub es bastante sencillo, en tu perfil de Github vas a al botón "new" para crear un nuevo repositorio, luego de ponerle un nombre/descripcion tenés la opcion de crearlo en público o en privado.

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

Para invitar a alguien a tu repositorio privado podés ir a la opcion de "settings" de tu repositorio, luego a la opcion "collaboratos" nos pedirá que demosntremos que somos nosotros con nuestra contraseña y desde alli poedés añadir personas con el botón "add people" escribir el nombre del usuario a invitar y añadirlo.

- ¿Qué es un repositorio público en GitHub?

Cuando un repositorio es público, todos los usuarios con acceso a este mismo pueden usarlo, hacer un fork.

- ¿Cómo crear un repositorio público en GitHub?

Crear un repositorio público en GitHub vamos a nuestro perfil de Github, vamos al botón “new” para crear un nuevo repositorio, luego de ponerle un nombre/descripción tenés la opción de crearlo en público o en privado.

- ¿Cómo compartir un repositorio público en GitHub?

Para compartir un repositorio público en GitHub Solo debemos copiar el url del navegador de la página de repositorio y compartirlo por redes sociales o foros.

2) Realizar la siguiente actividad:

- Crear un repositorio.

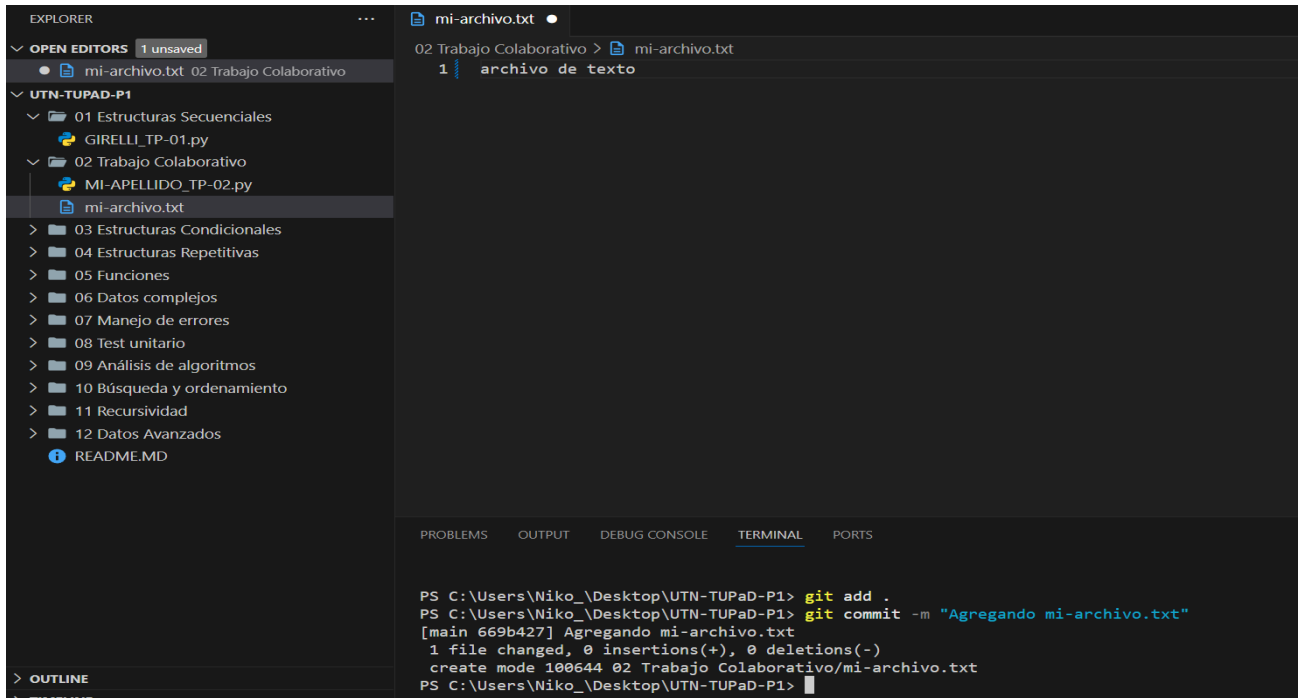
o Dale un nombre al repositorio.

o Elije el repositorio sea público.

o Inicializa el repositorio con un archivo.

Agregando un Archivo

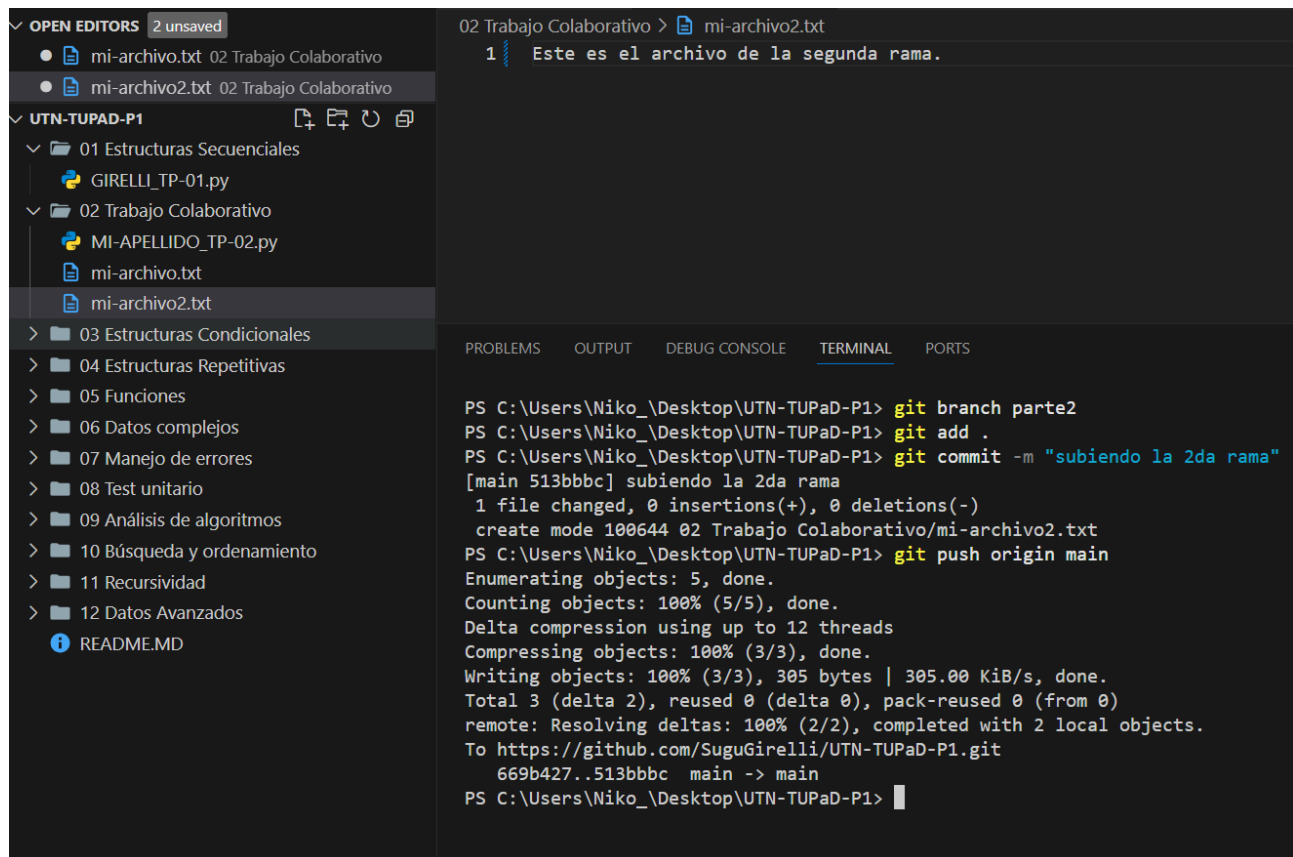
- o Crea un archivo simple, por ejemplo, "mi-archivo.txt".
- o Realiza los comandos `git add .` y `git commit -m "Agregando mi-archivo.txt"` en la línea de comandos.
- o Sube los cambios al repositorio en GitHub con `git push origin main` (o el nombre de la rama correspondiente).



```
PS C:\Users\Niko_\Desktop\UTN-TUPaD-P1> git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 342 bytes | 342.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/SuguGirelli/UTN-TUPaD-P1.git
    bc6531e..669b427  main -> main
PS C:\Users\Niko_\Desktop\UTN-TUPaD-P1>
```

- Creando Branchs

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



The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar shows a project named 'UTN-TUPAD-P1' with a folder structure including '01 Estructuras Secuenciales', '02 Trabajo Colaborativo' (selected), '03 Estructuras Condicionales', '04 Estructuras Repetitivas', '05 Funciones', '06 Datos complejos', '07 Manejo de errores', '08 Test unitario', '09 Análisis de algoritmos', '10 Búsqueda y ordenamiento', '11 Recursividad', '12 Datos Avanzados', and a 'README.MD' file. The Editor pane shows '02 Trabajo Colaborativo > mi-archivo2.txt' with the content '1 Este es el archivo de la segunda rama.' The bottom panel shows the 'TERMINAL' tab with the following output:

```
PS C:\Users\Niko_\Desktop\UTN-TUPaD-P1> git branch parte2
PS C:\Users\Niko_\Desktop\UTN-TUPaD-P1> git add .
PS C:\Users\Niko_\Desktop\UTN-TUPaD-P1> git commit -m "subiendo la 2da rama"
[main 513bbbc] subiendo la 2da rama
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 02 Trabajo Colaborativo/mi-archivo2.txt
PS C:\Users\Niko_\Desktop\UTN-TUPaD-P1> git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 305 bytes | 305.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/SuguGirelli/UTN-TUPaD-P1.git
 669b427..513bbbc main -> main
PS C:\Users\Niko_\Desktop\UTN-TUPaD-P1>
```

3) Para esta actividad voy a dejar los títulos de los pasos y las imágenes que explican lo que sucede en cada paso.

Paso 1: Crear un repositorio en GitHub.

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 * SuguGirelli / **Repository name *** conflict-exercise

✔ conflict-exercise is available.

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

Description (optional)

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](#).

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

[Create repository](#)

Paso 2: Clonar el repositorio a tu máquina local.

Paso 3: Crear una nueva rama y editar un archivo.

```
Símbolo del sistema
Microsoft Windows [Versión 10.0.26100.3476]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\Niko_>git clone https://github.com/SuguGirelli/conflict-exercise
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.

C:\Users\Niko_>cd conflict-exercise

C:\Users\Niko_\conflict-exercise>git add README.md

C:\Users\Niko_\conflict-exercise>git commit -m "Added a line in feature-branch"
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

C:\Users\Niko_\conflict-exercise>
```

Aca me salte un paso y lo vuelvo a realizar.

```
Símbolo del sistema
Microsoft Windows [Versión 10.0.26100.3476]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\Niko_>git clone https://github.com/SuguGirelli/conflict-exercise.git
fatal: destination path 'conflict-exercise' already exists and is not an empty director
y.

C:\Users\Niko_>cd conflict-exercise

C:\Users\Niko_\conflict-exercise>git checkout -b feature-branch
Switched to a new branch 'feature-branch'

C:\Users\Niko_\conflict-exercise>notepad README.md

C:\Users\Niko_\conflict-exercise>
```

```
README.md
Archivo  Editar  Ver

# conflict-exercise
tp2 actividad 3

Este es un cambio en la feature Branch.
```



```
Símbolo del sistema
Microsoft Windows [Versión 10.0.26100.3476]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\Niko_>git clone https://github.com/SuguGirelli/conflict-exercise.git
fatal: destination path 'conflict-exercise' already exists and is not an empty director
y.

C:\Users\Niko_>cd conflict-exercise

C:\Users\Niko_\conflict-exercise>git checkout -b feature-branch
Switched to a new branch 'feature-branch'

C:\Users\Niko_\conflict-exercise>notepad README.md

C:\Users\Niko_\conflict-exercise>git add README.md

C:\Users\Niko_\conflict-exercise>git commit -m "Added a line in feature-branch"
[feature-branch 6cac7dd] Added a line in feature-branch
1 file changed, 2 insertions(+)

C:\Users\Niko_\conflict-exercise>
```

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

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

C:\Users\Niko_\conflict-exercise>notepad README.md

C:\Users\Niko_\conflict-exercise>
```

```
README.md
Archivo  Editar  Ver

# conflict-exercise
tp2 actividad 3
|
Este es un cambio en la main branch.
```

Paso 5: Hacer un merge y generar un conflicto

```
C:\Users\Niko_\conflict-exercise>git add README.md

C:\Users\Niko_\conflict-exercise>git commit -m "Added a line in a main branch"
[main 3e9a58e] Added a line in a main branch
1 file changed, 2 insertions(+)

C:\Users\Niko_\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\Niko_\conflict-exercise>
```

Paso 6: Resolver el conflicto

```
C: > Users > Niko_ > Desktop > README.md > # conflict-exercise
1  # conflict-exercise
2  tp2 actividad 3
3
4  Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
5  <<<<<< HEAD (Current Change)
6  Este es un cambio en la main branch.
7  =====
8  >>>>>> feature-branch (Incoming Change)
9
```

```
C: > Users > Niko_ > Desktop > README.md > # conflict-exercise
1  # conflict-exercise
2  tp2 actividad 3
3
4
5  Este es un cambio en la main branch.
6
7
```

```
PS C:\Users\Niko_\Desktop\UTN-TUPaD-P1> code README.md
PS C:\Users\Niko_\Desktop\UTN-TUPaD-P1> git add README.md
PS C:\Users\Niko_\Desktop\UTN-TUPaD-P1> git commit -m "Resolved merge conflict"
On branch main
Your branch is up to date with 'origin/main'.
```

Paso 7: Subir los cambios a GitHub

```
PS C:\Users\Niko_\Desktop\UTN-TUPaD-P1> git push origin main
Everything up-to-date
PS C:\Users\Niko_\Desktop\UTN-TUPaD-P1> 
```

Paso 8: Verificar en GitHub

Showing 1 changed file with 2 additions and 0 deletions.

Split Unified

2 README.md	
...	...
1	1 # conflict-exercise
2	2 tp2 actividad 3
3	+
4	+ Este es un cambio en la feature Branch.