

Resolución Práctico 2: Git y GitHub

Andrés Emanuel Meshler

Actividades

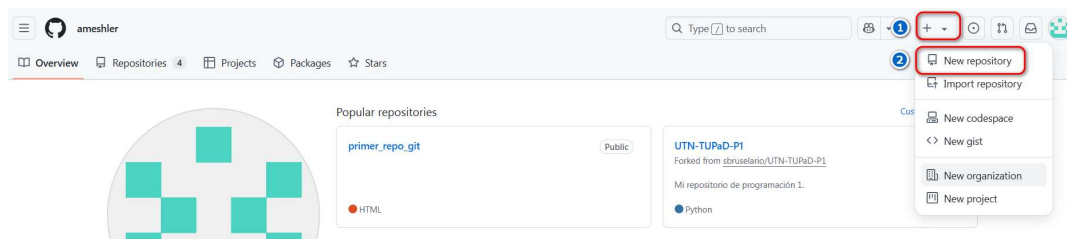
- 1) Contestar las siguientes preguntas utilizando las guías y documentación proporcionada (Desarrollar las respuestas):

- ¿Qué es GitHub?

GitHub es una plataforma web que se utiliza principalmente para alojar proyectos de software y facilitar la colaboración entre desarrolladores. Se basa en Git, un sistema de control de versiones que permite a los desarrolladores gestionar el código fuente de sus proyectos, llevar un registro de los cambios realizados. Facilitando, de este modo, el trabajo colaborativo en equipos de desarrollo.

- ¿Cómo crear un repositorio en GitHub?

Una vez registrados en GitHub debemos acceder con nuestras credenciales hacemos clic en el botón “+” y luego en “New Repository”



Con esto accedemos a la pagina donde podemos configurar el repositorio a crear:


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 *

 ameshler ▾

/

Colocar el nombre del repositorio aquí

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

Description (optional)

De manera opcional colocar una descripción aquí

- ☒  **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.

Seleccionar si el
acceso al repositorio
será público o
privado

Initialize this repository with:

☐ Add a README file

Opcionalmente tildar para que se genere un archivo
README.md

This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: None ▾

Este archivo indica qué archivos o carpetas
deben ser ignorados por Git al hacer commits.

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

Choose a license

License: None ▾

Podés seleccionar una licencia para tu proyecto
si es un proyecto público.

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

 You are creating a public repository in your personal account.

Click para finalizar y crear el repositorio

Create repository

- ¿Cómo crear una rama en Git?

Para crear una rama en Git debemos abrir en consola la ubicación de la rama principal de la cual queremos hacer el nuevo branch y utilizar el comando:

git branch nombre_del_branch

- ¿Cómo cambiar a una rama en Git?

Mediante el comando:

git checkout nombre_del_branch_de_destino

- ¿Cómo fusionar ramas en Git?

Parados en la rama principal utilizamos el comando:

git merge nombre_del_branch_a_fusionar

- ¿Cómo crear un commit en Git?

Para hacer un commit de los cambios realizados en nuestro código debemos tenerlos guardados en nuestro repositorio local y parados en la carpeta donde se encuentra ejecutar los siguientes comandos en orden:

git init (una única vez par inicializar el trackeo de Git)

git add . (para agregar al escenario de Git todos los archivos del proyecto)

git commit -m "descripción del contenido del commit realizado"

- ¿Cómo enviar un commit a GitHub?

Una vez realizado el commit para subirlo a Github se utiliza el comando:

git push origin <nombre_del_branch>

- ¿Qué es un repositorio remoto?

Un repositorio remoto es una versión de tu repositorio Git que se encuentra alojada en un servidor externo, generalmente en la nube, como GitHub.

- ¿Cómo agregar un repositorio remoto a Git?

Mediante el comando:

git remote add origin <url_del_repositorio_github>

- ¿Cómo empujar cambios a un repositorio remoto?

Mediante el comando:

git push origin <nombre_del_branch>

- ¿Cómo tirar de cambios de un repositorio remoto?

Mediante el comando:

git pull origin <nombre_del_branch>

- ¿Qué es un fork de repositorio?

Es una copia de un repositorio remoto de una cuenta diferente de Git hacia la nuestra, para poder aplicarle cambios sin afectar al repositorio original.

- ¿Cómo crear un fork de un repositorio?

Debemos estar logueados en GitHub, luego acceder a la url del repositorio que queremos copiar y luego hacer clic en el botón fork. Esto copiará el repositorio en nuestra cuenta de GitHub.

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

Debemos ir a la página de nuestro repositorio en GitHub (una vez subidos tus cambios al fork que creamos) y hacer clic en el botón **pull request** en la parte superior.

Esto nos llevará a una página donde podremos comparar los cambios entre nuestra rama y la rama principal del repositorio original.

Escribimos un título y descripción del pull request. Seleccionamos la rama base: En el campo "base", seleccionamos la rama del repositorio original a la que queremos que se fusionen los cambios. Luego, seleccionamos la rama de comparación: En el campo "compare", seleccionamos la rama de nuestro fork que contiene los cambios que queremos proponer.

Por último, debemos hacer clic en el botón verde que dice Create pull request.

- ¿Cómo aceptar una solicitud de extracción?

En la pestaña "Pull Requests" del repositorio de GitHub hacer clic en la pull request que se desea aceptar. Luego para aceptarla hacer clic en "Approve"

Primero debemos ir al repositorio original donde recibimos la solicitud de extracción (Pull Request). En la parte superior de la página del repositorio, hacer clic en la pestaña "Pull requests". Se desplegará una lista de Pull Request abiertas. Hacemos clic en la Pull Request que queremos revisar y aceptar.

Luego debemos revisar la descripción de la Pull Request proporcionada por quien la hizo y los diffs entre la rama del creador de la pull request y nuestra rama base.

De ser necesario podemos descargarnos y probar el código en local.

Si todo está bien podemos hacer clic en "Merge pull request", agregar un mensaje de manera opcional y finalmente hacemos clic en "Confirm merge"

- ¿Qué es una etiqueta en Git?

Es una referencia que se utiliza para marcar puntos específicos en el historial del repositorio, generalmente para señalar versiones importantes o hitos.

- ¿Cómo crear una etiqueta en Git?

Mediante el comando:

git tag <nombre-de-la-etiqueta>

- ¿Cómo enviar una etiqueta a GitHub?

Mediante el comando:

git push origin <nombre-de-la-etiqueta>

- ¿Qué es un historial de Git?

El historial de Git es un registro completo de todos los cambios (commits) que se han realizado en un repositorio a lo largo del tiempo. Es una línea de tiempo que muestra cómo ha evolucionado el código fuente, qué cambios se hicieron, quién los hizo y cuándo.

- ¿Cómo ver el historial de Git?

Mediante el comando:

git log

- ¿Cómo buscar en el historial de Git?

Mediante el comando:

git log

Con el agregado de:

--grep="palabra_clave" -> para buscar una palabra o frase en la descripción del commit.

--author="nombre_autor" -> para buscar por el nombre del autor del commit.

<nombre_del_archivo> -> para buscar cambios en un archivo específico.

--since="2025-01-01" --until="2025-03-27" -> para buscar en un rango de fechas.

- ¿Cómo borrar el historial de Git?

Mediante el comando:

- **git reset** -> Quita del stage todos los archivos y carpetas del proyecto.

- **git reset nombreArchivo** -> Quita del stage el archivo indicado.

- **git reset nombreCarpeta/** -> Quita del stage todos los archivos de esa carpeta.

- **git reset nombreCarpeta/nombreArchivo** -> Quita ese archivo del stage (que a la vez está dentro de una carpeta).

- **git reset nombreCarpeta/*.extensión** -> Quita todos los archivos que cumplan con la condición indicada previamente dentro de esa carpeta del stage.

- ¿Qué es un repositorio privado en GitHub?

Un repositorio privado en GitHub es un tipo de repositorio que solo es accesible para usuarios autorizados, a diferencia de los repositorios públicos, que son accesibles para cualquier persona. Los repositorios privados permiten mantener el código, los archivos y los datos de un proyecto de forma confidencial y controlada, de modo que solo ciertos colaboradores o personas específicas (autorizadas por el creador del repositorio privado) puedan ver, modificar o contribuir al proyecto.

- ¿Cómo crear un repositorio privado en GitHub?

Quando creamos un nuevo repositorio en GitHub tenemos la posibilidad de seleccionar la opción para que sea privado.

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 *

 ameshler ▾

Repository name *

/

Great repository names are short and memorable. Need inspiration? How about [curly-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.

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

En la página del repositorio, hacemos clic en la pestaña "Settings" que está en la parte superior derecha. En el menú de la izquierda, buscamos la sección "Access" y hacemos clic en "Collaborators". Verenis un campo de búsqueda ("Search for a collaborator") en el que escribimos el nombre de usuario de GitHub de la persona a la que queremos invitar.

Una vez encontrado el usuario, lo seleccionamos y procedemos a elegir el nivel de permisos que le vamos a otorgar. Esto es: **Read** (solo podrá ver el repositorio pero no editarlo), **Write** (podrá ver y realizar cambios en el repositorio) y **Admin** (podrá gestionar el repositorio, agregar colaboradores, cambiar configuraciones, etc.).

Finalmente, después de elegir el nivel de permisos, hacemos clic en el botón "Add collaborator". GitHub enviará una invitación al usuario seleccionado.

- ¿Qué es un repositorio público en GitHub?

Un repositorio público en GitHub es un repositorio cuyo contenido es accesible y visible para cualquier persona en Internet. Cualquier persona puede ver, clonar, bifurcar (fork) o contribuir al proyecto, dependiendo de los permisos que se otorguen. Este tipo de repositorio es ideal para proyectos de código abierto, ya que permite que cualquiera pueda acceder al código, aprender de él, mejorar el proyecto o contribuir.

- ¿Cómo crear un repositorio público en GitHub?

En el menú de creación de nuevos repositorios se nos permite seleccionar si queremos que el repositorio sea público o privado.

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 *

/

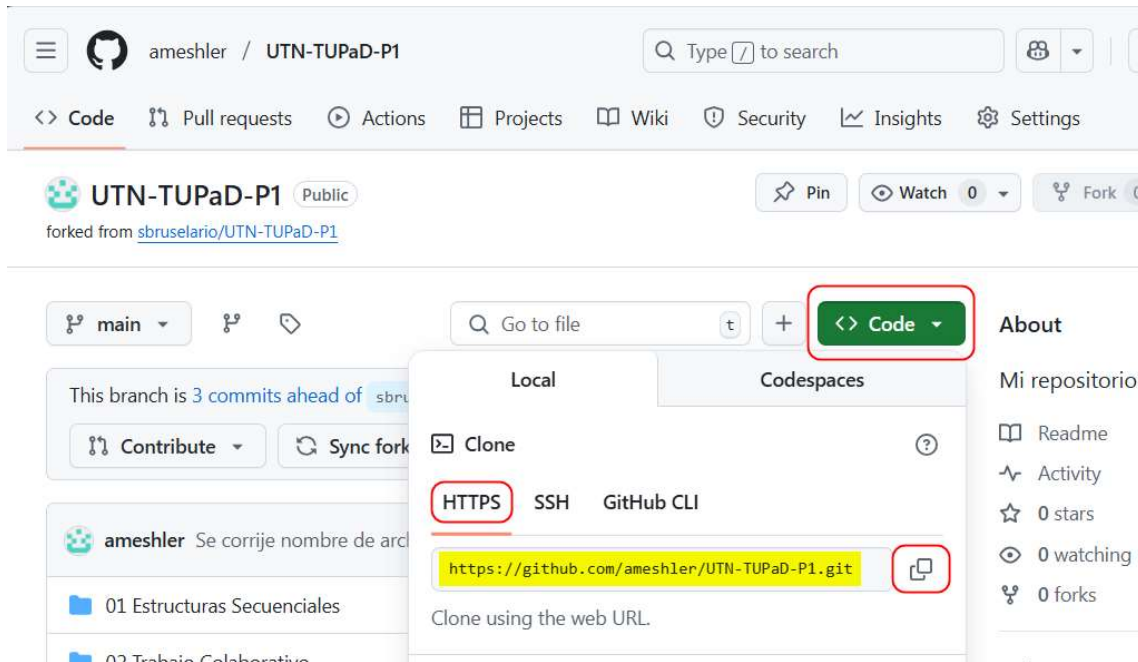
Great repository names are short and memorable. Need inspiration? How about [curly-fortnight](#) ?

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.

- ¿Cómo compartir un repositorio público en GitHub?

Se puede compartir mediante la url de la página principal del repositorio. La misma se puede copiar desde la barra de direcciones del navegador o haciendo clic en el botón "<> Code" y luego copiando el enlace de la sección HTTPS:



2) Realizar la siguiente actividad:

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

Actividad subida al repositorio: https://github.com/ameshler/repo_git_ei2

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".

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
```

- 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"`

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"`

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

- 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
```

- 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.

Actividad subida al repositorio: <https://github.com/ameshler/conflict-exercise>