

Trabajo Práctico 2: Git y GitHub - TUP - Axel Gomez

Actividad 1.

1. ¿Qué es GitHub?

GitHub es una plataforma de desarrollo colaborativo basada en la nube que permite a los desarrolladores alojar, gestionar y compartir código utilizando Git. Proporciona herramientas para el control de versiones, revisión de código, gestión de proyectos y colaboración en equipo. GitHub permite a los desarrolladores trabajar en proyectos de código abierto o privados, facilitando la integración con herramientas CI/CD, automatización de flujos de trabajo y la documentación del código.

2. ¿Cómo crear un repositorio en GitHub?

Para crear un repositorio en GitHub se siguen los siguientes pasos

1. Se ingresa en tu cuenta de [GitHub](https://github.com).
2. Se hace click en el botón "+" en la esquina superior derecha y selecciona **"New repository"**.
3. Ingresas un nombre para el repositorio en **Repository name**.
4. (Opcional) Agrega una descripción en **Description**.
5. Eliges la visibilidad del repositorio:
 - **Public**: Cualquiera puede verlo.
 - **Private**: Solo tú y las personas con acceso pueden verlo.
6. (Opcional) Selecciona la opción **Add a README file** para incluir un archivo de descripción.
7. (Opcional) Agrega un archivo **.gitignore** para excluir archivos no deseados del control de versiones.
8. (Opcional) Selecciona una licencia para el repositorio.
9. Clickeas en **Create repository**.

3. ¿Cómo crear una rama en Git?

Una rama en Git permite trabajar en nuevas funcionalidades o correcciones sin afectar la rama principal (**main** o **master**). Para crear una nueva rama, se usa el siguiente comando en la terminal:

```
git branch nombre-de-la-rama
```

Por ejemplo, si deseas crear una rama llamada `feature-login`:

```
git branch feature-login
```

4. ¿Cómo cambiar a una rama en Git?

Para cambiar a una rama, se usa el siguiente comando:

```
git checkout nombre-de-la-rama
```

Por ejemplo, para cambiar a la rama `feature-login`:

```
git checkout feature-login
```

A partir de Git 2.23, se recomienda usar `git switch`:

```
git switch feature-login
```

5. ¿Cómo fusionar ramas en Git?

Para fusionar los cambios de una rama a otra (por ejemplo, de `feature-login` a `main`), se siguen los siguientes pasos

Asegurate de estar en la rama donde quieres fusionar los cambios:

```
git checkout main
```

Ejecuta el comando de fusión:

```
git merge feature-login
```

Si hay conflictos en la fusión, Git los mostrará y deberás resolverlos manualmente antes de completar el merge.

6. ¿Cómo crear un commit en Git?

Un commit en Git guarda los cambios realizados en el código en el historial del repositorio.

Para hacer un commit, sigue estos pasos:

Agrega los archivos al área de preparación (staging area):

```
git add .
```

O agrega archivos específicos:

```
git add archivo.txt
```

Creas un commit con un mensaje descriptivo:

```
git commit -m "Mensaje descriptivo del cambio"
```

7. ¿Cómo enviar un commit a GitHub?

Para que se suba el commit a GitHub, se siguen estos pasos:

Asegurarse de estar en la rama correcta:

```
git checkout nombre-de-la-rama
```

Envía los cambios al repositorio remoto en GitHub con este comando:

```
git push origin nombre-de-la-rama
```

Si es la primera vez que subes una rama nueva, usa:

```
git push -u origin nombre-de-la-rama
```

8. ¿Qué es un repositorio remoto?

Un **repositorio remoto** es una versión de un repositorio que está alojada en una plataforma en línea como **GitHub, GitLab o Bitbucket**. A diferencia del repositorio local (almacenado en tu computadora), el repositorio remoto permite la colaboración entre múltiples desarrolladores al proporcionar un lugar común donde pueden compartir, fusionar y administrar el código.

Los repositorios remotos facilitan la integración de cambios, la gestión de ramas y la implementación de flujos de trabajo colaborativos, como **pull requests** y revisiones de código.

9. ¿Cómo agregar un repositorio remoto a Git?

Para agregar un repositorio remoto en Git, usa el siguiente comando:

```
git remote add origin URL-del-repositorio
```

Ejemplo: `git remote add origin https://github.com/usuario/repo.git`

Para verificar que se agregó correctamente el repositorio remoto, se usa:

```
git remote -v
```

10. ¿Cómo empujar cambios a un repositorio remoto?

Para subir cambios desde el repositorio local al remoto, se usa comando **git push**:

```
git push origin nombre-de-la-rama
```

Ejemplo, si estás en la rama main:

```
git push origin main
```

Si es la primera vez que subes una nueva rama, se usa:

```
git push -u origin nombre-de-la-rama
```

11. ¿Cómo tirar de cambios de un repositorio remoto?

Para obtener los últimos cambios de un repositorio remoto y fusionarlos en el código local, se usa el comando **git pull**:

```
git pull origin nombre-de-la-rama
```

Ejemplo, si deseas obtener los cambios de la rama **main**:

```
git pull origin main
```

12. ¿Qué es un fork de repositorio?

Un **fork** es una copia de un repositorio que se crea en tu cuenta de GitHub u otra plataforma, permitiéndote modificar el código sin afectar el repositorio original.

Los forks son útiles para:

- Contribuir a proyectos de código abierto.
- Hacer pruebas sin alterar el repositorio principal.
- Personalizar un proyecto existente.

Cuando haces un fork, puedes realizar cambios en tu copia y luego proponer que estos cambios se integren al repositorio original mediante un **pull request**.

13. ¿Cómo crear un fork de un repositorio?

Para hacer un fork de un repositorio en GitHub:

1. Inicias sesión en [GitHub](#).
2. Vas al repositorio que deseas forkear.
3. Haz clic en el botón **Fork** en la parte superior derecha de la página.
4. GitHub creará una copia del repositorio en tu cuenta.

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

Un **pull request (PR)** es una solicitud para fusionar cambios desde una rama de tu repositorio (o de un fork) hacia la rama principal de otro repositorio en GitHub.

Pasos para enviar un pull request en GitHub

1. **Asegúrate de que tu código está actualizado** en tu rama local y que los cambios están subidos a GitHub.
2. **Ve al repositorio en GitHub** donde deseas hacer la solicitud.
3. **Navegas a la pestaña "Pull requests"** y haz clic en **"New pull request"**.
4. **Selecciona las ramas** de origen y destino:
 - **Base branch:** La rama donde se fusionarán los cambios (por lo general, **main**).
 - **Compare branch:** Tu rama con los cambios.
5. **Revisa los cambios** y agrega un título y descripción clara del PR.
6. **Haz clic en "Create pull request"**.
7. Esperas la revisión y aprobación del propietario del repositorio.

15. ¿Cómo aceptar una solicitud de extracción?

Para aceptar un **pull request (PR)** en GitHub:

1. **Ve a la pestaña "Pull requests"** en el repositorio.
2. **Selecciona el pull request** que deseas aceptar.
3. **Revisa los cambios** y verificas que no hay conflictos.

4. Si todo está bien, seleccionas en **"Merge pull request"**.
5. Confirmas la fusión haciendo clic en **"Confirm merge"**.
6. (Opcional) **Elimina la rama** después de la fusión si ya no es necesaria.

Si hay conflictos, primero debes resolverlos antes de aceptar el PR.

16. ¿Qué es una etiqueta en Git?

Una **etiqueta (tag)** en Git es un marcador usado para señalar versiones específicas del código, comúnmente para identificar lanzamientos o versiones estables (como **v1.0**, **v2.3**).

Las etiquetas pueden ser:

- **Anotadas (annotated)**: Contienen información adicional (autor, fecha, mensaje).
- **Ligeras (lightweight)**: Solo marcan un commit sin metadatos adicionales.

17. ¿Cómo crear una etiqueta en Git?

Para crear una etiqueta, puedes usar uno de los siguientes comandos:

Etiqueta ligera: `git tag nombre-de-la-etiqueta`

Ejemplo: `git tag v1.0`

Etiqueta anotada (recomendada para versiones oficiales):

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

Ejemplo: `git tag -a v1.0 -m "Primera versión estable"`

Si necesitas etiquetar un commit anterior, usa:

`git tag -a v1.0 commit_id`

Para listar las etiquetas existentes: `git tag`

18. ¿Cómo enviar una etiqueta a GitHub?

Después de crear una etiqueta en local, debes enviarla al repositorio remoto en GitHub:

```
git push origin nombre-de-la-etiqueta
```

Ejemplo: `git push origin v1.0`

Si deseas subir todas las etiquetas creadas: `git push origin --tags`

Para eliminar una etiqueta en remoto: `git push --delete origin nombre-de-la-etiqueta`

Para eliminar una etiqueta en local: `git tag -d nombre-de-la-etiqueta`

19. ¿Qué es un historial de Git?

El **historial de Git** es el registro de todos los cambios realizados en un repositorio, incluyendo commits, fusiones (merges), ramas y etiquetas.

El historial permite ver qué cambios se han hecho, quién los realizó y cuándo, facilitando la colaboración y el control de versiones.

20. ¿Cómo ver el historial de Git?

Para ver el historial de commits, usa:

```
git log
```


Ver historial de commits en una sola línea:

```
git log --oneline
```

Ver historial con detalles de cambios por autor:

```
git log --author="nombre"
```

Ver cambios de un archivo específico:

```
git log -- nombre-del-archivo
```

Ver historial de ramas con un gráfico visual:

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

Si solo deseas ver los últimos commits: `git log -n 5`

21 ¿Cómo buscar en el historial de Git?

Para buscar en el historial de Git, puedes usar el comando `git log` con diferentes opciones de filtrado.

Buscar por mensaje de commit: `git log --grep="texto a buscar"`

Ejemplo: `git log --grep="fix login bug"`

Buscar por autor: `git log --author="nombre"`

Ejemplo: `git log --author="Juan Pérez"`

Buscar cambios en un archivo específico: `git log -- nombre-del-archivo`

Ejemplo: `git log -- index.js`

Buscar commits con una palabra clave en los cambios realizados: `git log -S "palabra clave"`


Ejemplo: `git log -S "login"`

22. ¿Cómo borrar el historial de Git?

Git no permite borrar el historial de commits de manera directa, ya que está diseñado para ser un sistema seguro de control de versiones. Sin embargo, hay formas de eliminar el historial en casos extremos.

Opción 1: Reescribir el historial con un nuevo commit inicial (destructivo)

```
git checkout --orphan nueva-rama  
  
git add -A  
  
git commit -m "Nuevo inicio del historial"  
  
git branch -D main  
  
git branch -m main  
  
git push -f origin main
```

 : Esto sobrescribirá el historial en el repositorio remoto y puede causar problemas a otros colaboradores.

Opción 2: Eliminar el repositorio y crear uno nuevo

Si necesitas borrar el historial completamente sin afectar archivos:

1. Elimina el repositorio en GitHub.
2. Crea uno nuevo y vuelve a subir el código.

23. ¿Qué es un repositorio privado en GitHub?

Un **repositorio privado** en GitHub es un repositorio que solo pueden ver y acceder los usuarios con permisos específicos. Es útil para proyectos internos, comerciales o cualquier código que no deba ser público.

24. ¿Cómo crear un repositorio privado en GitHub?

1. Inicias sesión en **GitHub** y haz clic en el botón "+" → **"New repository"**.

2. Escribe un **nombre** para el repositorio.
3. (Opcional) Agrega una **descripción**.
4. Selecciona la opción "**Private**" para que el repositorio sea privado.
5. (Opcional) Agrega un **README**, un archivo **.gitignore** o una licencia.
6. Haz clic en "**Create repository**".

Solo las personas a quienes invites podrán acceder al código.

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

1. Abre el repositorio en GitHub.
2. Ve a la pestaña "**Settings**".
3. En la barra lateral, selecciona "**Collaborators**".
4. Haz clic en "**Add people**".
5. Escribe el **nombre de usuario o correo electrónico** de la persona a invitar.
6. Haz clic en "**Add collaborator**".
7. La persona recibirá una invitación y deberá aceptarla para acceder al repositorio.

26. ¿Qué es un repositorio público en GitHub?

Un **repositorio público** en GitHub es un repositorio accesible para cualquier persona en Internet. Cualquiera puede ver el código, clonarlo y contribuir (si el propietario lo permite). Es ideal para proyectos de código abierto o documentación compartida.

27. ¿Cómo crear un repositorio público en GitHub?

1. Inicias sesión en **GitHub** y haz clic en "+" → **"New repository"**.
2. Ingresa un **nombre** para el repositorio.
3. (Opcional) Agrega una **descripción**.
4. Selecciona la opción **"Public"**.
5. (Opcional) Agrega un **README**, un **.gitignore** o una licencia.
6. Haz clic en **"Create repository"**.

28. ¿Cómo compartir un repositorio público en GitHub?

Para compartir un repositorio público en GitHub, puedes copiar y enviar el enlace directo del repositorio:

<https://github.com/usuario/nombre-del-repositorio>

Actividad 2)

Enlace del repositorio: <https://github.com/Axel3890/Ejercicio-2---Pr-ctico-2-Git-y-GitHub.git>

```
● $ git log -n 5
commit 483122e16283b32ab6a5e28a1169c0c7a711d2da (HEAD -> Rama-practica, origin/Rama-practica)
Author: Axel Gomez <111620032+Axel3890@users.noreply.github.com>
Date:   Wed Mar 26 22:09:27 2025 -0300

    Archivo Index creado con otra rama

commit f48a42e67a2870d802dee0aa001af9849a99ddc8 (origin/main, origin/HEAD, main)
Author: Axel Gomez <111620032+Axel3890@users.noreply.github.com>
Date:   Wed Mar 26 22:05:53 2025 -0300

    Agregando archivo de ejemplo

commit 907b82d19f7c5c4a87f67c17d6634a6bda97c552
Author: Axel Gomez <111620032+Axel3890@users.noreply.github.com>
Date:   Wed Mar 26 22:04:05 2025 -0300

    Initial commit
```

Actividad 3)

Enlace del repositorio: <https://github.com/Axel3890/Ejercicio-Conflicto.git>

Ejercicio-Conflicto

Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes

<<<<<< HEAD (Current Change)

Este es un cambio en la main branch.

=====

Este es un cambio en la feature branch.

>>>>>> feature-branch (Incoming Change)

```
● $ git log -n 5
commit a7a7d236cd3d6ce4f8e8d4f13d306f26f4cec8d9 (HEAD -> main, origin/main, origin/HEAD)
Merge: 4bfd256 467a331
Author: Axel Gomez <111620032+Axel3890@users.noreply.github.com>
Date: Thu Mar 27 10:36:46 2025 -0300

    Resolved merge conflict

commit 4bfd256d0961a886d85106da1425c94e7cb04ff6
Author: Axel Gomez <111620032+Axel3890@users.noreply.github.com>
Date: Wed Mar 26 23:17:21 2025 -0300

    Added a line in main branch

commit 467a3310f9037fcb9107110df3d1256ed4269f71 (origin/feature-branch, feature-branch)
Author: Axel Gomez <111620032+Axel3890@users.noreply.github.com>
Date: Wed Mar 26 23:15:44 2025 -0300

    Added a line in feature-branch

commit 639c9130f8ee99d88deb72e72571f123a157a940
Author: Axel Gomez <111620032+Axel3890@users.noreply.github.com>
Date: Wed Mar 26 22:10:36 2025 -0300

    Initial commit
```