1- ¿Qué es GitHub?

GitHub es una plataforma de desarrollo colaborativo para alojar proyectos utilizando el sistema de control de versiones Git. Principalmente, se utiliza para la creación y gestión de código fuente de programas informáticos, permitiendo a desarrolladores trabajar juntos en proyectos desde cualquier parte del mundo.

2- ¿Como Crear un repositorio en GitHub?

- Inicia sesión en tu cuenta de GitHub.
- En la esquina superior derecha, haz clic en el icono "+" y luego selecciona "New repository".
- Completa los detalles del repositorio:
    - **Repository name**: El nombre del repositorio.
    - **Description**: Una descripción opcional del repositorio.
    - **Public or Private**: Elige si quieres que el repositorio sea público o privado.
- Opcionalmente, puedes inicializar el repositorio con un README, un archivo .gitignore o una licencia.
- Haz clic en "Create repository".

3- ¿Cómo crear una rama en Git?

Para crear una rama en Git utilizaremos el comando **git branch nombre-rama**.

4- ¿Cómo cambiar a una rama en Git?

Para cambiar de rama en Git utilizamos el comando **git checkout rama.**

5- ¿Cómo fusionar ramas en Git?
Cambia a la rama destino: Primero, debes hacer checkout a la rama donde quieres integrar los cambios. Esta suele ser `main` o `master`, o una rama de desarrollo (`develop`).
git checkout <rama_destino>
Reemplaza `<rama_destino>` con el nombre de la rama a la que quieres fusionar los cambios (por ejemplo, `main`).
Ejecuta el comando de fusión: Una vez que estás en la rama destino, utiliza el comando `git merge` para integrar los cambios de la otra rama.
git merge <rama_a_fusionar>
Reemplaza `<rama_a_fusionar>` con el nombre de la rama que quieres fusionar en la rama actual (por ejemplo, `feature-login`).

6- ¿Cómo crear un commit en Git?
Para crear un commit en git utilizamos el comando
git init (iniciamos un repositorio)
git add . (Agregar archivos modificados)
git commit -m "Agregamos Mensaje" **Creamos el Commit**

7 - ¿Cómo enviar un commit a GitHub?
Inicializá Git (si todavía no lo hiciste):
```
git init
```
Agregá los archivos que querés subir:
```
git add .
```
Creá un commit con un mensaje descriptivo:
```
git commit -m "Primer commit: sube archivos iniciales"
```
Agregá el repositorio remoto de GitHub:
```
git remote add origin https://github.com/tu-usuario/tu-repo.git
```
Subí los cambios a GitHub:
```
git push -u origin main
```

8- ¿Qué es un repositorio remoto?

Los repositorios remotos son versiones de tu proyecto que están hospedadas en Internet o en cualquier otra red. Puedes tener varios de ellos, y en cada uno tendrás generalmente permisos de solo lectura o de lectura y escritura. Colaborar con otras personas implica gestionar estos repositorios remotos enviando y trayendo datos de ellos cada vez que necesites compartir tu trabajo. Gestionar repositorios remotos incluye saber cómo añadir un repositorio remoto, eliminar los remotos que ya no son válidos, gestionar varias ramas remotas, definir si deben rastrearse o no y más.

9- ¿Cómo agregar un repositorio remoto a git?

Utiliza el comando git remote add para vincular tu repositorio local con un repositorio remoto y asignar un nombre a ese remoto: $ git remote add [nombre] [url] Ponerle un nombre te ayuda a referenciar de manera más fácil en lugar de usar la URL completa. Puedes usar el nombre en la línea de comandos. Para asegurarte de que el remoto se ha agregado correctamente y verificar el nombre que le has dado, usa el comando: $ git remote –v Esto mostrará una lista de todos los remotos configurados con sus URLs: [nombre] [url] Para traer toda la información del repositorio remoto que aún no tienes en tu repositorio local, usa el comando git fetch seguido del nombre del remoto: $ git fetch [nombre] Este comando descarga todos los cambios del repositorio remoto asociado con el nombre, pero no fusiona esos cambios con tu rama actual. Es útil para ver qué cambios están disponibles en el remoto.

10- ¿Cómo empujar cambios a un repositorio remoto?

Antes de empujar tus cambios, es una buena práctica obtener los últimos cambios del repositorio remoto para evitar conflictos: git pull origin nombre_de_la_rama Empuja tus cambios al repositorio remoto: git push origin nombre_de_la_rama

11- ¿Cómo tirar de cambios de un repositorio remoto?
Como mencionamos en el punto anterior, para tirar los cambios del repositorio remoto Usa el comando git pull para descargar y fusionar los cambios del repositorio remoto con tu rama local: git pull origin nombre_de_la_rama Por ejemplo, si estás trabajando en la rama main: git pull origin main

12- ¿Qué es un fork de repositorio?
En muchas ocasiones podemos ver en github repositorios que son de nuestro agrado y donde nosotros queremos tener una copia y hacerle algunas modificaciones, para ello github nos ofrece la implementación de fork. Buscamos algún repositorio que nos parezca interesante. Este puede ser cualquier proyecto de código abierto que quieras modificar o utilizar como base para tu propio trabajo. En la página del repositorio, busca el botón "Fork" ubicado en la parte superior derecha de la página. Haz clic en el botón "Fork". Esto creará una copia del repositorio en tu propia cuenta de GitHub. Esta copia será independiente del repositorio original. Cualquier cambio que realices en tu fork no afectará al repositorio original, los cambios que hagamos no irán al repositorio original del autor de ese proyecto, si no a nuestra copia local. Entonces solo resta clonar ese repositorio que está en nuestro canal para poder trasladarlo a alguna carpeta deseada y seguir trabajando desde nuestro repositorio local.

13- ¿Cómo crear un fork de un repositorio?
Para entender esto podes por ejemplo crear una cuenta adicional en github para que puedas desde allí hacer un Fork de alguno de los repositorios de tu cuenta original, clonarlo en alguna carpeta y en algún archivo que nosotros notamos que podría mejorarse algo, procedemos a realizar algún cambio, lo agregamos al stage y lo commiteamos (todo esto desde nuestro repositorio local remoto) y luego mandamos los cambios a el forking que hicimos con un git push origin master. Ese cambio que hemos hecho queremos que sea visto por el creador del repositorio original (en este caso nosotros, desde nuestra cuenta original), debemos hacérselo notar puesto que en el repositorio original los cambios que hicimos previamente, no se verán reflejados.

14- ¿Cómo enviar una solicitud de extracción (pull request) a un repositorio?
Para hacer el pull request nos dirigiremos a la solapa de Pull requests allí daremos click en new pull request, veremos una ventana a modo de resumen en donde se reflejarán los cambios que hemos hecho nosotros en comparación al repositorio original (el código original, mejor dicho). Daremos click en Create pull request donde veremos el asunto (colocamos algún mensaje global) y más abajo tenemos suficiente lugar para poder

explayarnos en mencionar el porqué ese cambio que hemos realizado nosotros, sería considerado como algo que a el repositorio original le vendría bien agregarlo.

15- ¿Cómo aceptar una solicitud de extracción?
El autor del repositorio verá en sus pull requests el mensaje que le hemos enviado, para que lo pueda observar y si lo considera realizar el cambio pertinente (además de poder responderle al usuario que le ha propuesto ese cambio). Lo bueno de todo esto es que si el usuario original considera que esta modificación es buena y no genera conflictos con la rama maestra de su repositorio local remoto, puede clickear en Merge pull request y de esta manera sumará a su repositorio los cambios que hizo un usuario (en modo de ayuda)

16- ¿Qué es una etiqueta en Git?
Es una referencia a un punto específico en la historia de tu repositorio. Se utiliza comúnmente para marcar puntos de lanzamiento de versiones (por ejemplo, `v1.0`, `v2.3.5`).

17- ¿Cómo crear una etiqueta en Git?
Git utiliza dos tipos principales de etiquetas: ligeras y anotadas. Una etiqueta ligera es muy parecida a una rama que no cambia - simplemente es un puntero a un commit específico. Sin embargo, las etiquetas anotadas se guardan en la base de datos de Git como objetos enteros. Tienen un checksum; contienen el nombre del etiquetador, correo electrónico y fecha; tienen un mensaje asociado; y pueden ser firmadas y verificadas con GNU Privacy Guard (GPG). Normalmente se recomienda que crees etiquetas anotadas, de manera que tengas toda esta información; pero si quieres una etiqueta temporal o por alguna razón no estás interesado en esa información, entonces puedes usar las etiquetas ligeras.

18- ¿Cómo enviar una etiqueta a GitHub?
Primero, debes crear una etiqueta en tu repositorio local. Puedes crear una etiqueta anotada (que incluye información adicional como el autor y la fecha) o una etiqueta ligera (simplemente un puntero a un commit): Ej. etiqueta ligera: git tag v1.0 Podes verificar las etiquetas que has creado localmente con: git tag. Una vez que has creado la etiqueta en tu repositorio local, necesitas empujarla al repositorio remoto en GitHub. Puedes hacer esto con el siguiente comando: git push origin v1.0 (origin es el nombre del repositorio remoto (por defecto suele ser origin) y v1.0 es el nombre de la etiqueta.) Para empujar todas las etiquetas creadas, usar: git push origin --tags

19- ¿Qué es un historial de Git?
El historial de Git es una secuencia de todos los cambios realizados en un repositorio de Git. Cada cambio en el repositorio se guarda como un commit, y cada commit contiene información sobre el estado del proyecto en un momento específico, incluyendo: Identificador del commit Autor Fecha de realización Mensaje enviado

20- ¿Cómo ver el historial de Git?
Esto lo conseguimos con el comando de Git: git log Con tipear este comando en el bash de Git podremos apreciar el histórico de commits, estando situados en la carpeta de nuestro proyecto. El listado de commits estará invertido, es decir, los últimos realizados aparecen los primeros. El comando git log --oneline es una forma compacta de visualizar el historial de commits en un repositorio Git. Muestra un resumen conciso de los commits recientes, con cada commit representado en una sola línea. Si tu proyecto ya tiene muchos commits, quizás no quieras mostrarlos todos, ya que generalmente no querrás ver cosas tan antiguas como el origen del repositorio. Para ver un número de logs determinado introducimos ese número como opción, con el signo "-" delante (-1, -8, -12...). Por ejemplo, esto muestra los últimos tres commits: git log -3 Si queremos que el log también nos muestre los cambios en el código de cada commit podemos usar la opción -p. Esta opción genera una salida mucho más larga, por lo que seguramente nos tocará movernos en la salida con los cursores y usaremos CTRL + Z para salir. git log -2 -p

21- ¿Cómo buscar en el historial de Git?
Para buscar en el historial de commits de Git, puedes utilizar varios comandos y opciones que te permiten filtrar y localizar commits específicos. Para buscar commits que contengan una palabra o frase específica en el mensaje de commit, usa git log con la opción –grep: git log --grep="palabra clave" Para buscar commits que han modificado un archivo específico, usa git log seguido del nombre del archivo: git log -- nombre_del_archivo Para buscar commits en un rango de fechas específico, usa las opciones --since y --until: git log --since="2024-01-01" --until="2024-01-31" Para encontrar commits hechos por un autor específico, usa --author: git log --author="Nombre del Autor"

22- ¿Cómo borrar el historial de Git?
El comando git reset se utiliza sobre todo para deshacer las cosas, como posiblemente puedes deducir por el verbo. Se mueve alrededor del puntero HEAD y opcionalmente cambia el index o área de ensayo y también puede cambiar opcionalmente el directorio de trabajo si se utiliza - hard. Esta última opción hace posible que este comando pueda perder tu trabajo si se usa incorrectamente, por lo que asegúrese de entenderlo antes de usarlo. Existen distintas formas de utilizarlo: • 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.

23- ¿Qué es un repositorio privado en GitHub?
Un repositorio privado en GitHub es un tipo de repositorio en el que el contenido solo es accesible para usuarios específicos que han sido autorizados. A diferencia de los repositorios públicos, donde cualquier persona puede ver y clonar el contenido, un repositorio privado limita el acceso a los colaboradores que tú elijas. Esto es útil para proyectos que contienen información sensible o que aún están en desarrollo y no deseas que estén disponibles públicamente.

24- ¿Cómo crear un repositorio privado en GitHub?

Inicia sesión en GitHub:

- Abre tu navegador web y ve a https://github.com/.
- Ingresa tu nombre de usuario o correo electrónico y tu contraseña.
- Haz clic en el botón "Sign in" (Iniciar sesión).

Accede a la creación de un nuevo repositorio:

- Una vez que hayas iniciado sesión, mira la esquina superior derecha de la página. Verás un icono de **"+" (Sign in)**. Haz clic en este icono.
- En el menú desplegable que aparece, selecciona la opción **"New repository"** (Nuevo repositorio).

Configura tu nuevo repositorio:

- Serás redirigido a la página "Create a new repository" (Crear un nuevo repositorio). Aquí deberás completar algunos campos:
    - **Owner:** (Propietario) Por defecto, será tu nombre de usuario. Puedes cambiarlo si perteneces a alguna organización.
    - **Repository name:** (Nombre del repositorio) Elige un nombre descriptivo para tu proyecto.
    - **Description (optional):** (Descripción - opcional) Puedes agregar una breve descripción de tu repositorio.

Selecciona la privacidad del repositorio:

- Justo debajo de la descripción, verás una sección llamada **"Privacy"** (Privacidad). Aquí tienes dos opciones:
    - **Public:** (Público) Cualquiera en Internet puede ver este repositorio. Tú eliges quién puede hacer commits.

    - **Private:** (Privado) Solo tú y las personas que invites pueden ver y hacer commits en este repositorio.
- **Haz clic en el botón de radio (el círculo) junto a "Private"** para seleccionar esta opción. El botón debería llenarse o resaltarse indicando que está seleccionado.

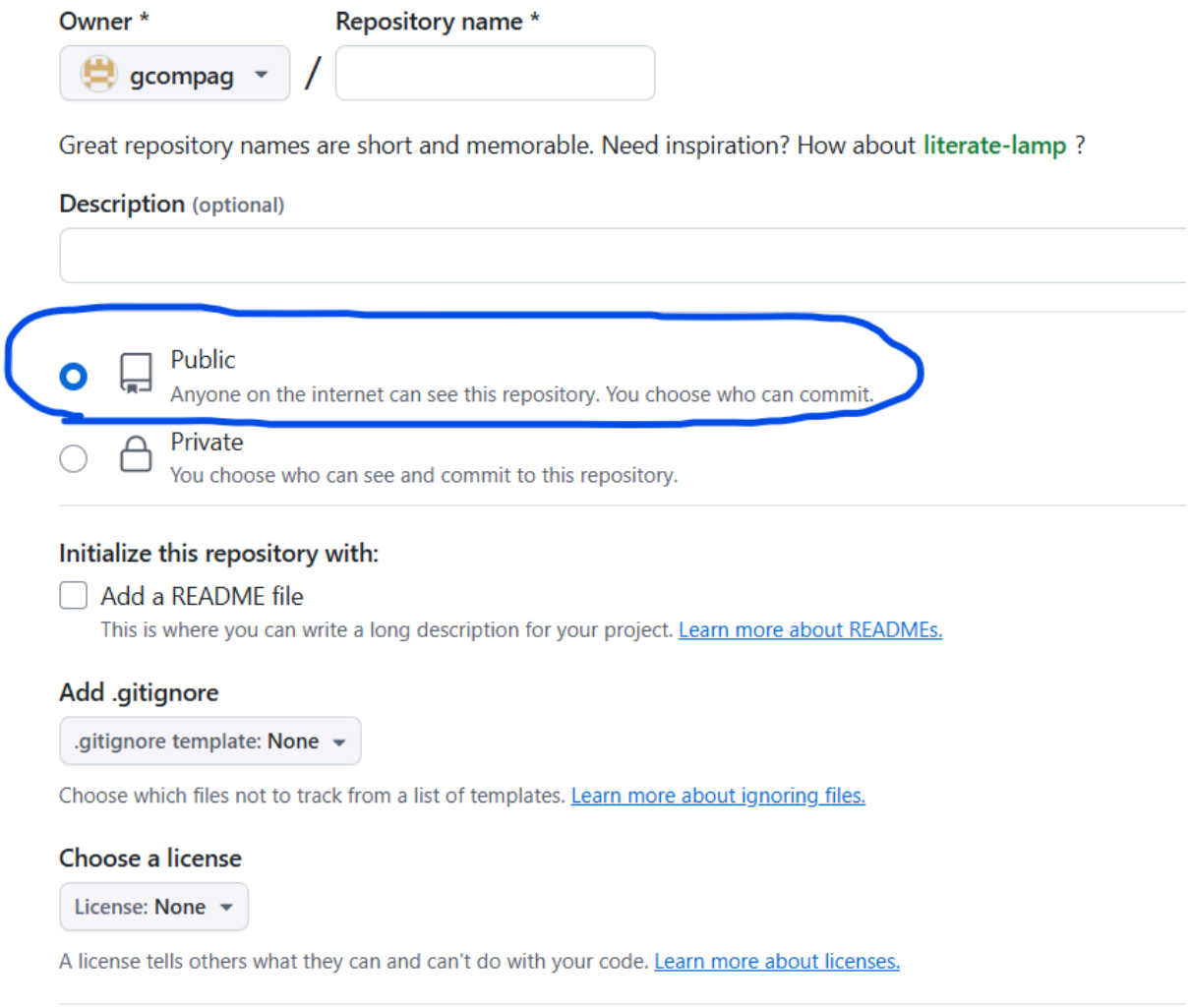
25- ¿Cómo invitar a alguien a un repositorio privado en GitHub?

Invitar a alguien a un repositorio privado en GitHub es un proceso sencillo, pero requiere permisos adecuados. Accede al repositorio, haz clic en la pestaña "Settings" del repositorio. Está en la parte superior del repositorio, junto a las pestañas como "Code" y "Issues".
Selecciona "Collaborators" en el menú de la izquierda. Esto te llevará a la página donde puedes administrar colaboradores.
En la sección "Collaborators", haz clic en el botón "Add people" e ingresa el nombre de usuario de GitHub de la persona que deseas invitar. Selecciona el nivel de acceso que deseas otorgar: Read, Triage, Write, Maintain, o Admin. Haz clic en el botón "Add" para enviar la invitación.

26- ¿Qué es un repositorio público en GitHub?
Un repositorio público en GitHub es un repositorio cuyo contenido es accesible a cualquier persona en Internet. A diferencia de un repositorio privado, que está restringido a un grupo específico de colaboradores, un repositorio público permite que cualquier persona pueda ver, clonar y, si tienen los permisos adecuados, contribuir al proyecto.

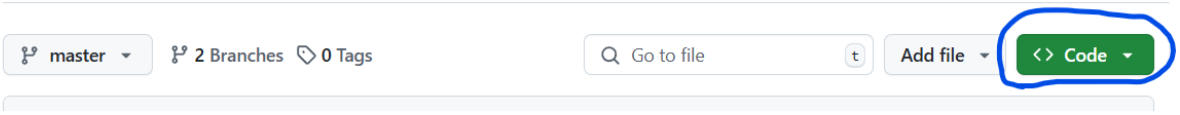
27- ¿Cómo crear un repositorio público en GitHub?
Inicia sesión en GitHub Ingresa a la página de creación de repositorios: En la esquina superior derecha de la página principal, debes hacer clic en el botón “+” y seleccionar “New Repository” o hacer clic en “New”:

Completar la información del repositorio (nombre del repositorio, descripción) Seleccionar la configuración de privacidad:



28- ¿Cómo compartir un repositorio público en GitHub?

La forma más sencilla de compartir tu repositorio es proporcionar el enlace directo al mismo. Accede a tu repositorio, copia la URL de tu repositorio que se encuentra en un cuadro de texto que dice "<> Code":



Repositorio Fork de ejemplo: https://github.com/gcompag/UTN-TUPaD-P1-GC.git
Repositorio Actividad 2 - TP 2: https://github.com/gcompag/repo-tp2.git
Repositorio Actividad 3 - TP 3: https://github.com/gcompag/conflict-exercise.git