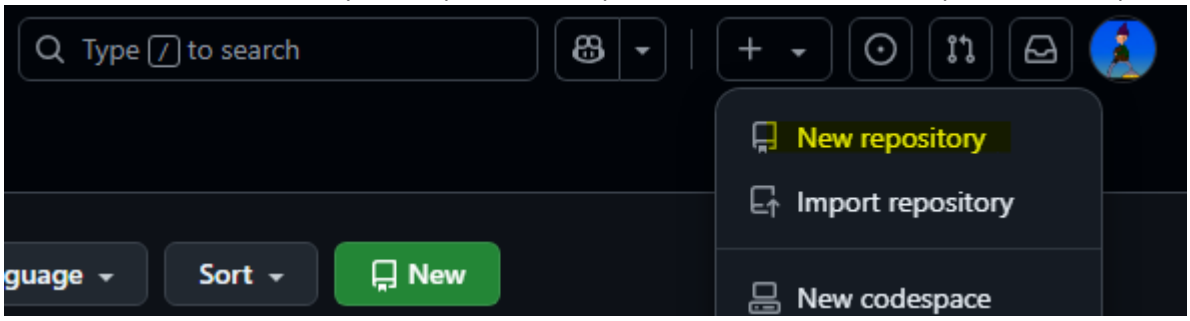
Actividades

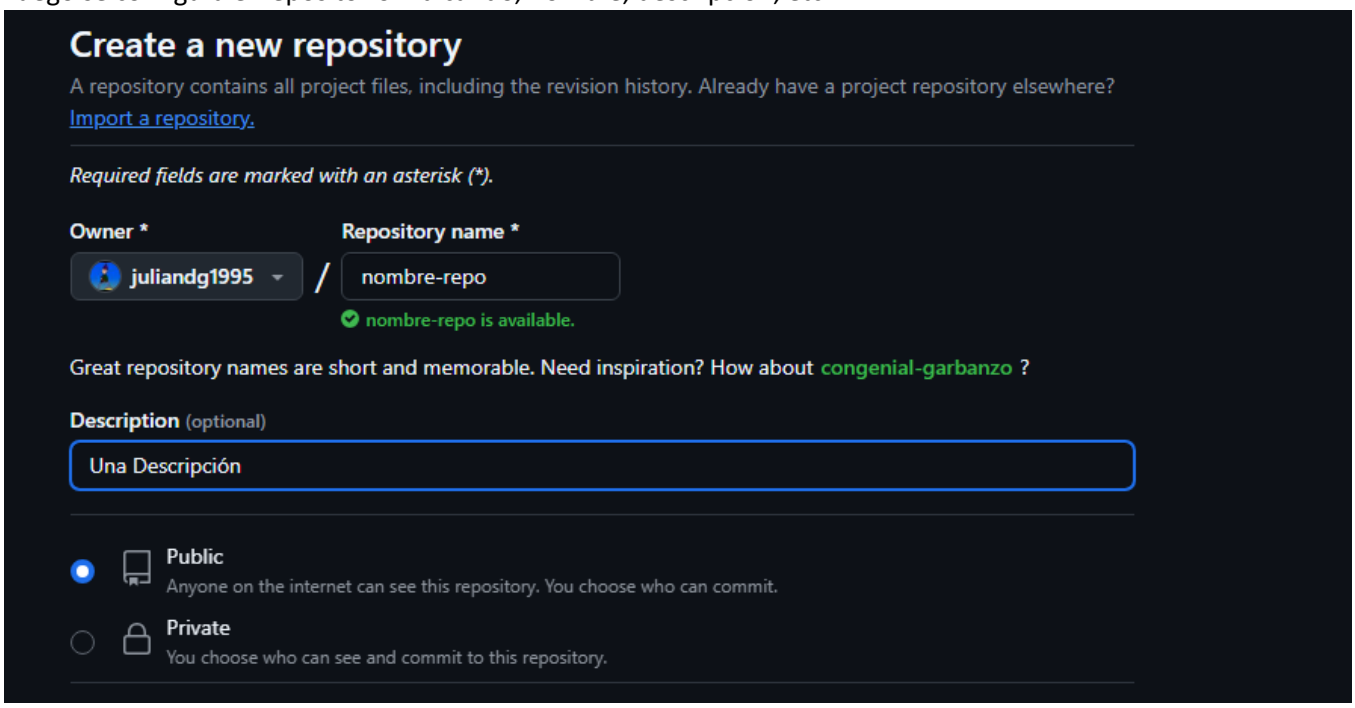
1) Contestar las siguientes preguntas utilizando las guías y documentación proporcionada (Desarrollar las respuestas) : • ¿Qué es GitHub?

• ¿Cómo crear un repositorio en GitHub?

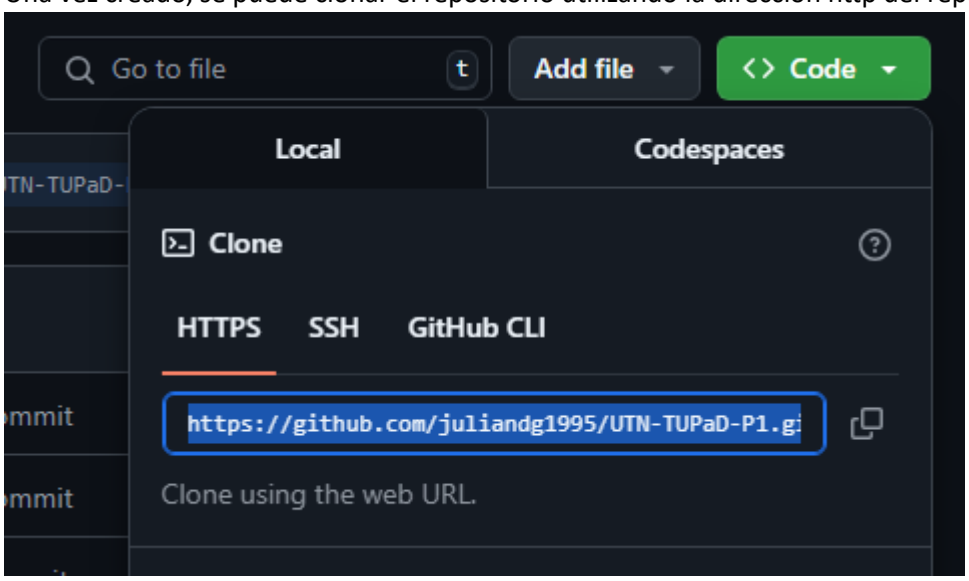
- Dentro de la cuenta GitHub, y de la opción “Mis repositorios”, se clickea en la opción “New Repository”



- Luego se configura el repositorio indicando, nombre, descripción, etc



Una vez creado, se puede clonar el repositorio utilizando la dirección http del repositorio:



Ya creado, se puede allí clonar el repositorio local utilizando el siguiente comando en consola:

- ***git remote add origin*** https://github.com/tu-usuario/nombre-del-repositorio.git

• ¿Cómo crear una rama en Git?

- Hay dos opciones:
  - ***git checkout -b*** [nombre-de-la-rama]
  - ***git branch*** [nombre-de-la-rama]

• ¿Cómo cambiar a una rama en Git?

- Ya posicionado en una rama padre:
  - ***Git checkout*** *[nombre-rama-destino]*

• ¿Cómo fusionar ramas en Git?

- Ya posicionado en la rama a fusionar:
  - ***Git merge*** *[nombre-de-rama-a-fusionar]*

• ¿Cómo crear un commit en Git?

- ***Git commit -m*** *[mensaje descriptivo del commit]*

• ¿Cómo enviar un commit a GitHub?

- ***Git add .***
- ***Git commit -m*** *[mensaje descriptivo de los cambios]*
- ***Git push origin*** *[nombre-rama-en-github]*

• ¿Qué es un repositorio remoto?

- Es un repositorio virtual, donde podemos trabajar de manera sincrónica con nuestro repositorio local, y también de manera colaborativa con otros programadores, haciendo control de cambios y versiones, entre otras ventajas.

• ¿Cómo agregar un repositorio remoto a Git?

- ***git remote add origin*** [https://github.com/tu-usuario/nombre-del-repositorio.git]

• ¿Cómo empujar cambios a un repositorio remoto?

- ***Git push origjn [nombre-repo-remoto]***

• ¿Cómo tirar de cambios de un repositorio remoto?

- ***Git pull [nombre-repo_remoto]***

• ¿Qué es un fork de repositorio?

- Es una copia de un repositorio remoto, que permite trabajar con él sin afectar el repositorio original

• ¿Cómo crear un fork de un repositorio?

- En github, ya parados sobre el repositorio, existe una opción llamada fork, que copia el repositorio indicado en nuestra colección de repositorios personal.

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

- Luego de realizar un push de algún commit con cambios, ir a la pagina de github, y clickear en la opción "Compare & pull request" después de abrir la rama, o ir a “Pull Requests” y clickear en “New Pull Request”
- Seleccionar la rama base (posiblemente ‘main’ o ‘master’) y la rama modificada.
- Ponerle un título al PR y una descripción de los cambios, y clickear en “Create Pull Request”
- Una vez enviada la solicitud de extracción, los mantenedores del repositorio revisarán tus cambios. Pueden comentar, solicitar cambios o aprobar el PR.

• ¿Cómo aceptar una solicitud de extracción?

- Localizar la pestaña Solicitudes de extracción
- Seleccionar la solicitud de extracción que se quiere aprobar

- Revisar los cambios propuestos
- Escribir un comentario que resuma la opinión sobre los cambios
- Seleccionar Aprobar
- Hacer clic en Enviar revisión
-

• ¿Qué es un etiqueta en Git?

- En Git, una etiqueta es una referencia que señala un punto específico en el historial de un proyecto. Se puede usar para marcar confirmaciones importantes, como versiones beta o cambios significativos en el código

• ¿Cómo crear una etiqueta en Git?

- Para crear una etiqueta en Git, se utiliza el comando ***git tag***.

1. Escribir git tag seguido del nombre de la etiqueta. Por ejemplo, git tag v1.4.
2. Para crear una etiqueta anotada, se utiliza la opción -a. Por ejemplo, git tag -a v1.4 -m 'my version 1.4'.
3. Para crear una etiqueta ligera, se utiliza el comando ***git tag <tagname>-lw***.
4. Para ver la información de una etiqueta, se utiliza el comando git show. Por ejemplo, git show v1.4.

• ¿Cómo enviar una etiqueta a GitHub?

- Para enviar una etiqueta a GitHub, puedes usar el comando *git push* con la opción *--tags*.
- Ejecutar ***git push origin [etiqueta]*** para enviar la etiqueta al servidor github.
- También se pueden crear directamente desde GitHub
    - En el nombre del repo, clickear en “Incidencias” o “Solicitudes de incorporación de cambios”.
    - Por encima de la lista de incidencias, clickear en “Labels”
    - Clickear en “Nueva etiqueta”
-

• ¿Qué es un historial de Git?

- El historial de Git es el registro de todos los cambios realizados en un repositorio a lo largo del tiempo. Git rastrea cada modificación en los archivos mediante confirmaciones (commits), que contienen información sobre qué se cambió, quién lo cambió y cuándo.

• ¿Cómo ver el historial de Git?

- Se puede visualizar el historial con el comando ***git log*** , que muestra una lista de todos los commits en orden cronológico inverso. También es posible utilizar herramientas como ***git blame*** para ver quién modificó cada línea de un archivo, o ***git reflog*** para consultar acciones recientes, incluso aquellas que no aparecen en el historial normal.

• ¿Cómo buscar en el historial de Git?

- Para buscar en el historial de Git, se pueden utilizar diferentes comandos según lo que se necesite encontrar:
    - **Buscar por mensaje de *commit***
      Se puede emplear ***git log --grep="palabra clave"*** para filtrar los *commits* que contengan una palabra específica en el mensaje.
    - **Buscar por autor**
      Se puede usar ***git log --author="nombre"*** para visualizar los *commits* realizados por un usuario en particular.
    - **Buscar por cambios en archivos**
      El comando ***git log -- filename.ext*** permite ver todos los *commits* que afectaron un archivo específico.
    - **Buscar por contenido modificado**
      Se puede utilizar ***git log -S "texto"*** para localizar *commits* donde se haya agregado o eliminado una línea con cierto texto.

- **Búsqueda avanzada con git grep**
  Para encontrar una palabra o expresión en todas las versiones de los archivos, se puede emplear ***git grep "palabra".***
- **Ver cambios línea por línea**
  El comando ***git blame archivo.ext*** muestra cada línea de un archivo junto con el autor y el *commit* que la modificó.

• ¿Cómo borrar el historial de Git?

- En Git, no existe un comando directo para borrar completamente el historial, pero se pueden emplear diferentes enfoques según lo que se quiera lograr:

  - **Reiniciar el repositorio**
    Si se desea eliminar todo el historial y comenzar desde cero, se puede ejecutar:

    (Windows Powershell)
    ```
    Remove-Item -Recurse -Force .git
    git init
    git commit -m "Inicio de nuevo historial"
    ```

    Esto eliminará por completo el repositorio Git y creará uno nuevo, perdiendo toda la información previa.

  - **Reescribir el historial**
    Para eliminar todos los *commits* y conservar los archivos, se puede usar:

    ```
    git checkout --orphan nueva-rama
    git add .
    git commit -m "Nuevo comienzo"
    git branch -D main
    git branch -m main
    ```

    Esto crea una nueva rama sin historial y elimina la anterior.

  - **Borrar *commits* específicos**
    Si solo se necesita eliminar ciertos *commits*, git rebase o git reset pueden ser opciones:

    ```
    git reset --hard <hash>
    git push –force
    ```

    Esta acción deshace cambios hasta el *commit* seleccionado.

  - Si se necesita navegar o revertir cambios dentro del historial, Git ofrece comandos como ***git revert*** y ***git reset***, para revertir o eliminar un comit y volver a una versión anterior.

• ¿Qué es un repositorio privado en GitHub?

- Un repositorio privado en GitHub es un espacio donde se almacenan archivos y versiones de código que solo pueden ser vistos y gestionados por personas con permisos específicos. A diferencia de los repositorios públicos, que están disponibles para cualquier usuario de GitHub, los privados ofrecen mayor control sobre la privacidad y seguridad del proyecto.
- Algunas características clave de los repositorios privados incluyen:
  - **Acceso restringido**: Solo los colaboradores autorizados pueden ver y modificar el contenido.
  - **Control de permisos**: Se pueden asignar roles específicos, como lectura o escritura, para cada usuario.
  - **Repositorios ilimitados**: GitHub permite a todos los usuarios crear repositorios privados sin límite, aunque ciertas funciones avanzadas requieren planes pagos.
  - **Mayor seguridad**: Al no ser visibles públicamente, es posible proteger información sensible, evitando accesos no autorizados.

- Este tipo de repositorio es ideal para proyectos en desarrollo, código propietario o trabajos en equipo que requieren mantener la confidencialidad del contenido.

• ¿Cómo crear un repositorio privado en GitHub?

- Dentro de GitHub, y habiendonos loggeado en nuestra cuenta, podemos sencillamente clickear en el botón “+” en la esquina superior derecha de la pantalla. Esto nos abrirá un menú desplegable, y sólo debemos elegir la opción “New Repository” o “Nuevo Repositorio”.
  Una vez allí, sólo debemos asignarle un nombre y apretar en el raddiobutton “Private” si queremos que sea privado o “Public” si queremos que sea público.

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

- Para invitar a alguien a colaborar en un repositorio, simplemente debemos loggearnos en la cuenta, ir al repositorio que deseamos compartir → “Settings” → “Collaborators” (en el menú de navegación a la izquierda de la pantalla) → “Add People” → Ingresar el mail o nombre de usuario de la persona que queramos invitar, y luego se le enviará la notificación con la invitación.

• ¿Qué es un repositorio público en GitHub?

- Un repositorio público en GitHub es un espacio donde se almacenan archivos y versiones de código que están disponibles para cualquier usuario de la plataforma. A diferencia de los repositorios privados, que requieren permisos para acceder, los públicos pueden ser vistos, clonados y utilizados libremente.
- Algunas características clave de los repositorios públicos incluyen:
  - **Acceso abierto**: Cualquier persona puede ver y descargar el contenido del repositorio.
  - **Colaboración global**: Se pueden recibir contribuciones de desarrolladores de todo el mundo mediante solicitudes de extracción (*pull requests*).
  - **Visibilidad y reconocimiento**: Son ideales para proyectos de código abierto, permitiendo que otros los encuentren y los usen.
  - **Licencias**: Es recomendable establecer una licencia de código abierto para definir cómo se puede utilizar y modificar el contenido.
- Este tipo de repositorio es útil para compartir proyectos, fomentar la colaboración y contribuir al desarrollo de software de código abierto. ¿Se está pensando en crear o contribuir a uno?

• ¿Cómo crear un repositorio público en GitHub?

- De la misma manera en la que se crea uno privado, sólo que en vez de seleccionar el raddiobutton “Private”, se utiliza la opción “Public”.

• ¿Cómo compartir un repositorio público en GitHub?

- Basta con compartir el link del repositorio, donde estará la dirección para realizar el pull de datos.