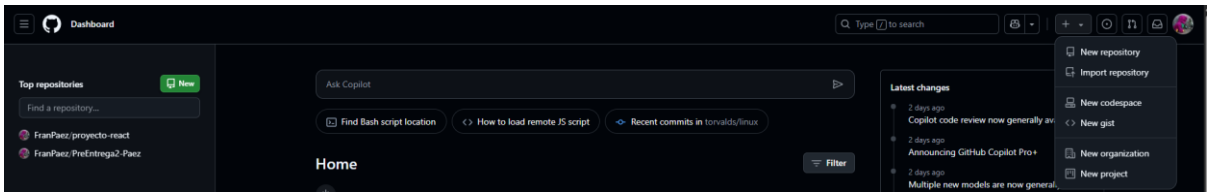
TRABAJO PRACTICO 2 – TRABAJO COLABORATIVO

Primera parte del trabajo:

1. ¿Qué es GitHub?
   Respuesta: GitHub es una plataforma donde los desarrolladores pueden guardar y gestionar sus proyectos de programación. Permite realizar cambios, seguir el historial del código y colaborar con otras personas usando Git.

2. ¿Cómo crear un repositorio en GitHub?
   Respuesta: Primero tenes que crearte una cuenta en GitHub, seguido de eso haces clic en el símbolo “+” de la derecha y haces clic en "nuevo repositorio". Luego colocarle el nombre que quieras y estaría listo el repositorio.



3. ¿Cómo crear una rama en Git?
   Respuesta: Git te crea por defecto la rama **"master"** o **"main"**. Luego para crear una adicional se utiliza el comando **"git branch nombre-rama"**.

4. ¿Cómo cambiar a una rama en Git?
   Respuesta: Para moverte entre diferentes ramas se utiliza el comando **"git checkout nombre-rama"**.

5. ¿Cómo fusionar ramas en Git?
   Respuesta: Para fusionar una rama en la rama actual se utiliza el comando **"git merge nombre-rama"**.

6. ¿Cómo crear un commit en Git?
   Respuesta: Para crear un commit se debe ejecutar el comando **"git commit"**. Podes agregarle el parámetro **-m "..."** para escribir un mensaje donde están los puntos suspensivos. Se lo considera buena práctica para llevar un seguimiento más específico.

7. ¿Cómo enviar un commit a GitHub?
   Respuesta: Para enviar el commit, una vez agregados al stage con **"git add"** y creado el commit con **"git commit"** se utiliza **"git push"** para enviarlo hacia el repositorio remoto.

8. ¿Qué es un repositorio remoto?
   Respuesta: Un repositorio remoto de GitHub es una copia local de tu proyecto que está almacenada en los servidores de GitHub.

9. ¿Cómo agregar un repositorio remoto a Git?
   Respuesta: Una vez iniciado git en el proyecto, vinculada la cuenta y agregado los archivos al stage, hay que ir a la página de GitHub, nuevo repositorio, le colocamos el nombre y podes crearlo de forma pública o privada. Una vez creado te muestra los comandos a seguir para enviar los archivos al repositorio remoto.

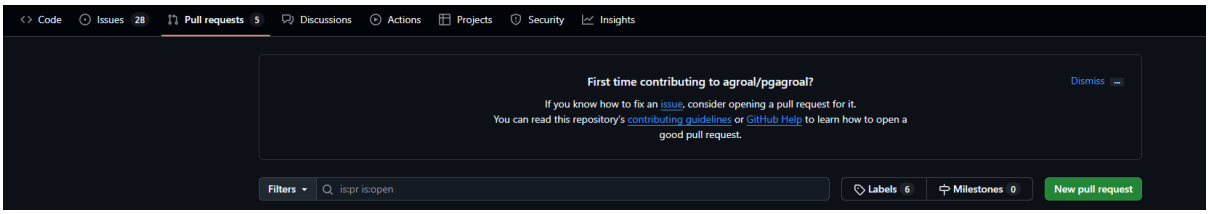
10. ¿Cómo empujar cambios a un repositorio remoto?
    Respuesta: Esto se realiza con el comando **"git push"**.

11. ¿Cómo tirar de cambios de un repositorio remoto?
    Respuesta: Esto se realiza con el comando **"git pull"**.

12. ¿Qué es un fork de repositorio?
    Respuesta: Un fork realiza una copia de un repositorio de otro usuario de GitHub en tu cuenta de forma totalmente independiente al proyecto de esa tercera persona.

13. ¿Cómo crear un fork de un repositorio?
    Respuesta: En caso de trabajarlo localmente podes realizar una clonación del repositorio con el comando **"git clone url-repositorio"**. Sino desde GitHub yendo a donde este el repositorio y haciendo clic en fork.

14. ¿Cómo enviar una solicitud de extracción (pull request) a un repositorio?
    Respuesta: Dentro del repositorio en GitHub, existe un apartado llamado "Pull requests" donde podés enviar tus cambios que crees que ese proyecto debería aplicar y los mismos van a ser revisados por alguien para aprobarlos o no.



15. ¿Cómo aceptar una solicitud de extracción?
    Respuesta: El autor del repositorio es quien va a ver en sus pull requests los mensajes enviados con los cambios solicitados. En caso de querer aprobarla solo deberá hacer clic en **“Merge pull requests”** y esto hará que los cambios se sumen a la rama principal.

16. ¿Qué es una etiqueta en Git?
    Respuesta: Git cuenta con la posibilidad de marcar algunos cambios del historial como importantes, estas marcas importantes se las considera como etiquetas. Una buena práctica es usarla para las versiones del desarrollo.

17. ¿Cómo crear una etiqueta en Git?
    Respuesta: No existe un solo tipo de etiqueta en Git, por lo tanto, hay más de una forma de crearla:
    Etiqueta ligera: es simplemente un nombre que apunta a un commit específico, sin información adicional. Se coloca **“git tag v1.0.0”** y esto crea una etiqueta llamada `v1.0.0` que apunta al commit actual.
    Etiqueta anotada: Las etiquetas anotadas son más completas y recomendadas, ya que almacenan información adicional como el nombre del autor, la fecha, y un mensaje de descripción. Se coloca **“git tag -a v1.0.0 -m "Primera versión estable"”** y esto crea una etiqueta anotada llamada `v1.0.0` con el mensaje “Primera versión estable”

18. ¿Cómo enviar una etiqueta a GitHub?
    Respuesta: Por defecto estas etiquetas no se suben, en caso de querer hacerlo se debe colocar **“git push origin v1.0.0”** en caso de querer enviar solamente esa etiqueta. En caso de querer enviar todas basta con poner **“git push --tags”** y en caso de necesitar eliminar etiquetas se utiliza **“git tag -d v1.0.0”** para eliminar una etiqueta local y **“git push origin --delete tag v1.0.0”** para eliminar una etiqueta del repositorio remoto.

19. ¿Qué es un historial de Git?
    Respuesta: El historial de Git son todos los cambios en secuencia que se realizaron en el repositorio. Cada cambio se guarda como commit y cada commit contiene información sobre cada uno de estos cambios.

20. ¿Cómo ver el historial de Git?
    Respuesta: Se puede realizar con el comando **“git log”**. Este comando mostrara todos los commits donde los primeros en aparecer son los últimos commits realizados. En caso de querer ver una cierta cantidad en específico ya que nuestro proyecto puede contar con muchos, se utiliza el signo **“-”** con el número de commits que se quieren visualizar quedando el comando así: **“git log –5"**

21. ¿Cómo buscar en el historial de Git?
    Respuesta: Para buscar algún commit en específico en el historial existen algunos parámetros para agregarle al “git log” para realizar la búsqueda:
    Por palabra clave: Al utilizar **–grep=”palabra-clave”** buscaremos todos los commits cuyo mensaje contengan esa palabra quedando **“git log –grep=”palabra-clave”**.
    Por rango de fecha: Para buscar en un rango de fechas se utiliza **“--since”** y **“--until”**, quedando asi **“git log –since=”2025-01-01" --until=”2025-03-31””** (Tener en cuenta que se utiliza el formato de fecha **YYYY-MM-DD**).
    Por autor: Para buscar algún commit por nombre de quien lo realizo se usa **“--author”** quedando: **“git log –author=”nombre-autor””**.

22. ¿Cómo borrar el historial de Git?

    Respuesta: Para deshacer cosas se utiliza el comando **"git reset"**, este tiene varias formas de ser usado:

    "git reset": Quita del stage todos los archivos y carpetas del proyecto.

    "git reset nombre-archivo": Quita del stage el archivo indicado.

    "git reset nombre-carpeta/": Quita del stage la carpeta indicada.

    "git reset nombre-carpeta/nombre-archivo": Quita del stage el archivo indicado que este dentro de la carpeta indicada.

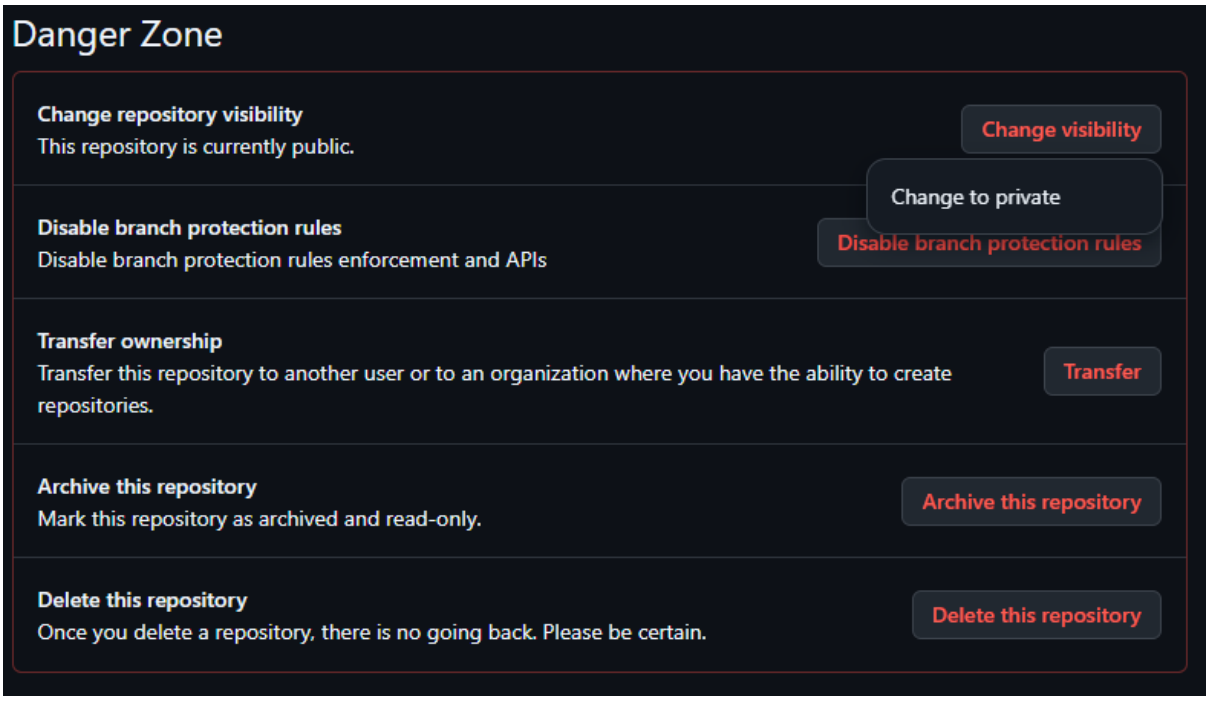
    "git reset nombre-carpeta/.extension":* Quita del stage todos los archivos que cumplan con ese formato de archivo dentro de la carpeta indicada.

23. ¿Qué es un repositorio privado en GitHub?

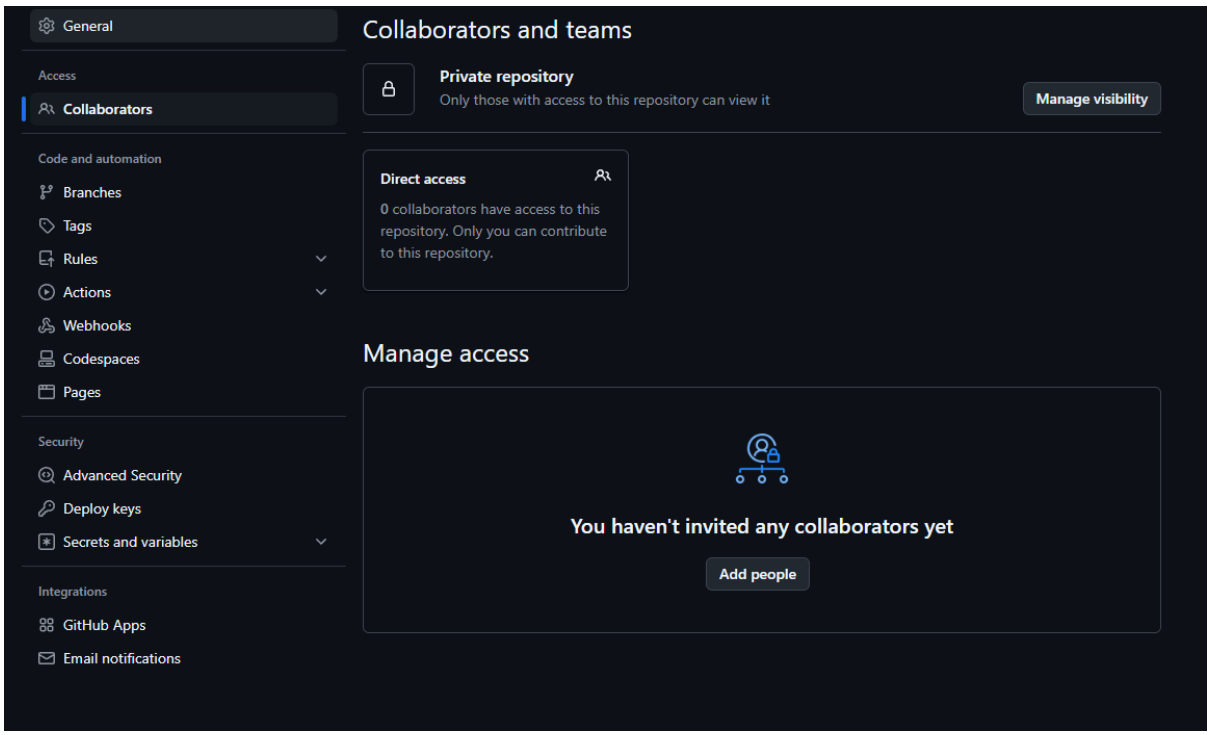
    Respuesta: Un repositorio privado es aquel que, como su nombre indica, el acceso está limitado a ciertos usuarios determinados por el creador del repositorio.

24. ¿Cómo crear un repositorio privado en GitHub?

    Respuesta: A la hora de crear un repositorio como vimos en el punto número dos, GitHub nos da la opción de crear el repositorio en privado o en público, en este caso marcaremos en privado. En todo caso, dentro de la configuración del repositorio, nos vamos abajo de todo a una zona en rojo marcada como **"danger zone"** y tenemos la opción de cambiarlo de público a privado si asi deseamos, esta zona hay que tener cuidado con que cambiamos ya que podríamos eliminar el repositorio completo.



25. ¿Cómo invitar a alguien a un repositorio privado en GitHub?
    Respuesta: Dentro de las configuraciones de nuestro repositorio tenemos la opción de “collaborators” como primera, ahí dentro podremos agregar a gente para que pueda obtener acceso a este repositorio.



26. ¿Qué es un repositorio público en GitHub?
    Respuesta: Un repositorio público es aquel el cual esta visible para cualquier persona que solamente tenga el link o, mismo navegando por GitHub lo encuentre. Al ser publico cualquier persona puede aportar al proyecto creando “pull requests”, puede clonarlo en su PC de forma independiente al original entre otras cosas. De todas maneras, el manejo total del repositorio siempre está en manos del autor de este.

27. ¿Cómo crear un repositorio público en GitHub?
    Respuesta: Al igual que en el punto 2, cuando creamos un repositorio GitHub nos ofrece crearlo en “publico” o en “privado”, simplemente seleccionamos en “publico” y estaría. Esto puede cambiarse en cualquier momento desde la configuración del repositorio, en la “danger zone”.

28. ¿Cómo compartir un repositorio público en GitHub?

Respuesta: En este caso es más sencillo, para compartirlo solo basta con enviar el link al repositorio y la otra persona ya tendrá acceso para aportar al proyecto.

Segunda parte del trabajo: Realiza la siguiente actividad
Estas consignas fueron realizadas en el siguiente repositorio:
https://github.com/FranPaez/prueba-git

- 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

Tercera parte del trabajo:

Este ejercicio se realizó en el siguiente repositorio:
https://github.com/FranPaez/conflict-exercise

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