TP 2: GIT Y GITHUB

NOMBRE: GINO CANEVARO

1) Contestar las siguientes preguntas utilizando las guías y documentación proporcionada

(Desarrollar las respuestas) :

A. ¿Qué es GitHub?

Es una plataforma que ofrece alojamiento de repositorios de control de versiones, que permite a los desarrolladores almacenar y gestionar sus proyectos de software. Es gratuita y de código abierto, permitiendo trabajar en equipo con proyectos

B. ¿Cómo crear un repositorio en GitHub?

Tenemos que iniciar sesión en GitHub, dirigirnos a nuestro perfil, ir a “New repository”, y configuramos nuestro repositorio (nombre, descripción, elegir entre que sea público o privado), también es recomendable añadirle un archivo README para documentar el proyecto, luego de todo eso apretamos en “Create repository” y listo, nuevo repositorio creado.

C. ¿Cómo crear una rama en Git?

Tenemos que utilizar el comando git branch, es decir

$ git branch nuevaRama

Cabe aclarar que al crear una nueva rama no significa que estamos en esa misma rama, sino que estamos en la rama master.

D. ¿Cómo cambiar a una rama en Git?

Utilizamos el comando comando git checkout:

$ git checkout nuevaRama

Y si queremos crear una nueva rama saltando a ella directamente

$ git checkout -b nuevaRama

E. ¿Cómo fusionar ramas en Git?

Primero, hay que estar en la rama que queremos fusionar los cambios. (master o main) u otra rama en la que estemos haciendo los cambios.

Intentemos fusionar la rama ‘ramaNueva’ a la rama master, entonces vamos a pararnos en la principal:

$ git checkout master

Ahora el comando git merge para fusionar la rama de origen en la rama actual:

$ git merge nuevaRama

Este comando incorpora los cambios de nuevaRama en master.

F. ¿Cómo crear un commit en Git?

Hacer los cambios en los archivos del proyecto que deseas guardar en el commit y luego agregar los cambios al área de preparación. Esto se hace con el comando git add.

$ git add archivo1 o $git add .

Luego git commit + mensaje.

$ git commit –m “Mensaje de los cambios”

Esto guarda el estado actual del código en el historial del repositorio con el mensaje.

G. ¿Cómo enviar un commit a GitHub?

Clonamos el repositorio de GitHub a nuestra máquina local:

git clone https://github.com/tu_usuario/tu_repositorio.git

cd tu_repositorio

Hacer cambios en los archivos del repo

git add nombre_del_archivo o git add .

Realizamos un commit de los cambios:

git commit -m "Descripción de los cambios"

Y para enviar los commits al repositorio en GitHub:

git push origin nombre_de_la_rama

H. ¿Qué es un repositorio remoto?

Un repositorio remoto es una versión de tu proyecto almacenada en un servidor externo (como GitHub, GitLab, Bitbucket o un servidor privado), que permite a múltiples desarrolladores colaborar, sincronizar cambios y mantener un historial centralizado del código.

I. ¿Cómo agregar un repositorio remoto a Git?

$ git remote add [nombre] [url]

$ git remote –v

[nombre] [url]

Para traer toda la información del repositorio remoto que aún no tienes en tu repositorio local, usa el

$ git fetch [nombre]

J. ¿Cómo empujar cambios a un repositorio remoto?

git pull origin nombre_de_la_rama

Empuja tus cambios al repositorio remoto:

git push origin nombre_de_la_rama

K ¿Cómo tirar de cambios de un repositorio remoto?

pull para descargar y fusionar los cambios del repositorio remoto con tu rama local:

git pull origin nombre_de_la_rama

Si estás trabajando en la rama main:

git pull origin main

L. Qué es un fork de repositorio?

Un fork es una copia personal de un repositorio remoto que te permite experimentar, modificar y contribuir al proyecto sin afectar el repositorio original. Es esencial para contribuir a proyectos de código abierto o crear derivados de un software existente.

M. ¿Cómo crear un fork de un repositorio?

Ve al repositorio: Encuentra el repositorio que deseas copiar.

Haz clic en "Fork": Está en la esquina superior derecha.

Elige tu cuenta: Selecciona dónde quieres guardar la copia.

Espera: GitHub creará la copia en tu cuenta.

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

Nos dirigiremos a la solapa de Pull requests , le damos click en new pull request, encontraremos 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). Click en Create pull request , más abajo mencionamos el por qué ese cambio que hicimos.

Ñ. ¿Cómo aceptar una solicitud de extracción?

Examinar los cambios en la pestaña "Files changed”, lee la descripción y los comentarios de la solicitud, si hay preguntas dejar comentarios. En caso de que esté todo correcto, la aprobamos, si hace falta modificar algo hay que detallarlo, luego de esto y de resolver conflictos, clickeamos en “Merge pull request”

O. ¿Qué es una etiqueta en Git?

Es una funcionalidad que permite etiquetar puntos importantes del historial, se suele usar para marcar versiones de lanzamiento

P. ¿Cómo enviar una etiqueta a GitHub?

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

git tag

Una vez que has creado la etiqueta en tu repositorio local, necesitas empujarla al repositorio remoto en GitHub

git push origin v1.0

Para empujar todas las etiquetas creadas, usar:

git push origin –tags

Q. ¿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 en que se encuentra el proyecto

R. ¿Cómo ver el historial de Git?

Es muy simple, con el siguiente comando:

git log

Cabe aclarar que el orden está invertido, es decir, los últimos realizados aparecen primeros

S. ¿Cómo buscar en el historial de Git?

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"

T. ¿Como borrar el historial de Git?

Las distintas formas según el caso:

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 está dentro de una carpeta)

U. ¿Qué es un repositorio privado en GitHub?

Es un tipo de repositorio en el que el contenido solo tiene acceso disponible el autor y las respectivas personas autorizadas, nadie más.

V. ¿Cómo crear un repositorio privado en GitHub?

Iniciamos sesión en GitHub, vamos a la sección “creación de repositorios”, hacemos click en el + y seleccionamos “New Repository” o “New” sea como aparezca, y ahí elegimos “Private”

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

Una vez en el repositorio, nos vamos a “settings”, luego seleccionamos “collaborators”, y de ahí en “Add people”, y ya podremos añadir gente al repositorio privado

X. ¿Qué es un repositorio público en GitHub?

Es un tipo de repositorio en el cual el acceso contenido está disponible a todo público, a cualquier persona en internet

Y. ¿Cómo crear un repositorio público en GitHub?

Hacemos lo mismo que con el anterior, solo que esta vez ponemos en “public”

Z. ¿Cómo compartir un repositorio público en GitHub?

Es con el link del mismo, una vez que lo tenemos lo compartimos en donde queramos y listo

2) Realizar la siguiente actividad:

- *Crear un repositorio.*

o Dale un nombre al repositorio.

o Elije el repositorio sea público.

o Inicializa el repositorio con un archivo.

- *Agregando un Archivo*

o Crea un archivo simple, por ejemplo, "mi-archivo.txt".

o Realiza los comandos git add . y git commit -m "Agregando mi-archivo.txt"

en la línea de comandos.

o Sube los cambios al repositorio en GitHub con git push origin main (o el

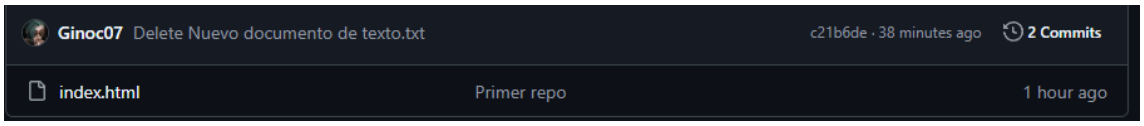
nombre de la rama correspondiente).

- *Creando Branchs*

o Crear una Branch

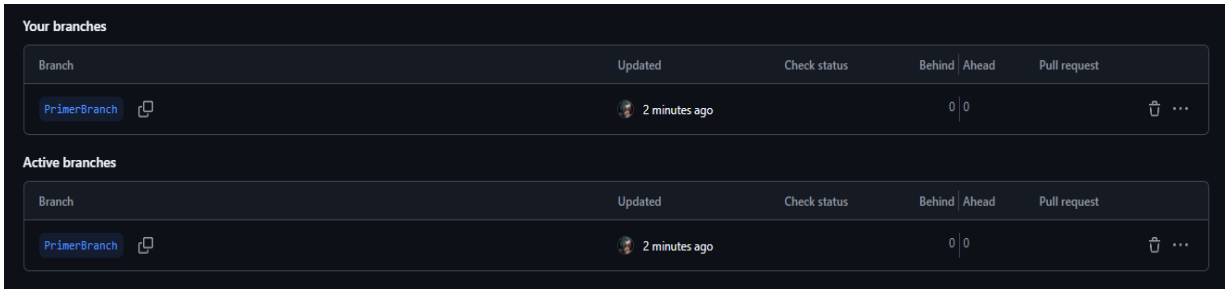
o Realizar cambios o agregar un archivo

o Subir la Branch




```
gino@DESKTOP-R8NPIR8 MINGW64 ~/Desktop/primer-repo-git
$ git init
Initialized empty Git repository in C:/Users/gino/Desktop/primer-repo-git/.git/

gino@DESKTOP-R8NPIR8 MINGW64 ~/Desktop/primer-repo-git (master)
$ git add .

gino@DESKTOP-R8NPIR8 MINGW64 ~/Desktop/primer-repo-git (master)
$ git commit -m "Primer repo"
```

```
gino@DESKTOP-R8NPIR8 MINGW64 ~/Desktop/primer-repo-git (master)
$ git commit -m "Primer repo"
[master (root-commit) e71d61c] Primer repo
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 Nuevo documento de texto.txt
 create mode 100644 index.html

gino@DESKTOP-R8NPIR8 MINGW64 ~/Desktop/primer-repo-git (master)
$ git remote add origin https://github.com/Ginoc07/PrimerRepo.git

gino@DESKTOP-R8NPIR8 MINGW64 ~/Desktop/primer-repo-git (master)
$ git push -u origin master
info: please complete authentication in your browser...
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 244 bytes | 244.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Ginoc07/PrimerRepo.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.
```



3) Realizar la siguiente actividad:

Paso 1: Crear un repositorio en GitHub

• Ve a GitHub e inicia sesión en tu cuenta.

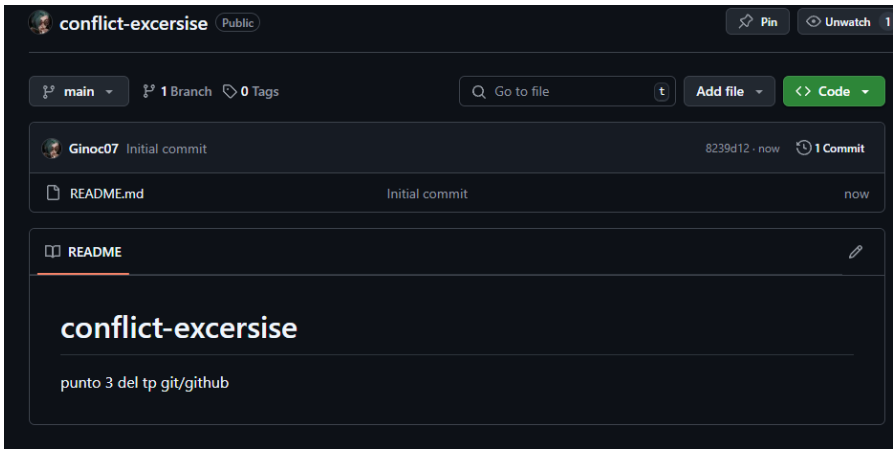
• Haz clic en el botón "New" o "Create repository" para crear un nuevo repositorio.

• 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).

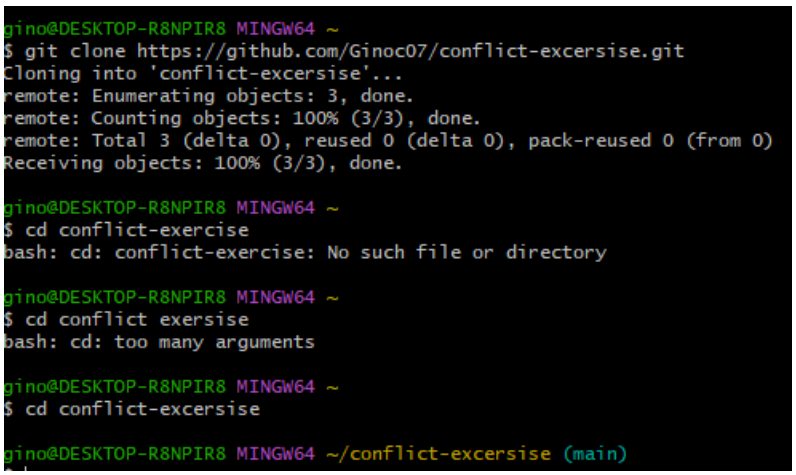
• Abre la terminal o línea de comandos en tu máquina.

• 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" (Ya lo había guardado desde el github)

```
gino@DESKTOP-R8NPIR8 MINGW64 ~/conflict-excersise (main)
$ git checkout -b feature-branch
Switched to a new branch 'feature-branch'

gino@DESKTOP-R8NPIR8 MINGW64 ~/conflict-excersise (feature-branch)
$ git add README.md

gino@DESKTOP-R8NPIR8 MINGW64 ~/conflict-excersise (feature-branch)
$ git commit -m "Added a line in feature-branch"
On branch feature-branch
nothing to commit, working tree clean

gino@DESKTOP-R8NPIR8 MINGW64 ~/conflict-excersise (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"

```
gino@DESKTOP-R8NPIR8 MINGW64 ~/conflict-excersise (feature-branch)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

gino@DESKTOP-R8NPIR8 MINGW64 ~/conflict-excersise (main)
$ git add README.md

gino@DESKTOP-R8NPIR8 MINGW64 ~/conflict-excersise (main)
$ git commit -m "Added a line in main branch"
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
```

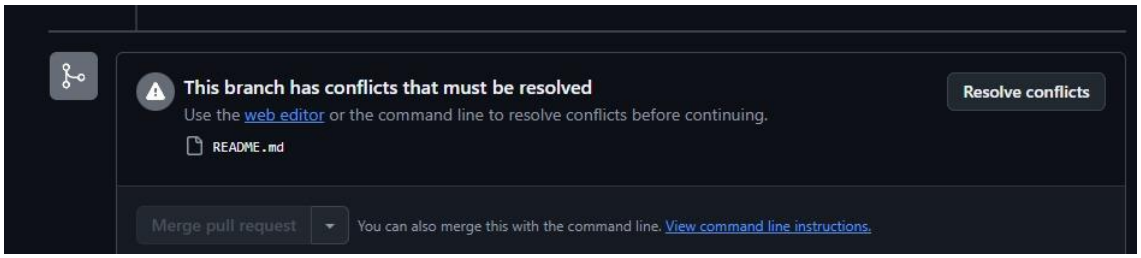
Paso 5: Hacer un merge y generar un conflicto

• 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

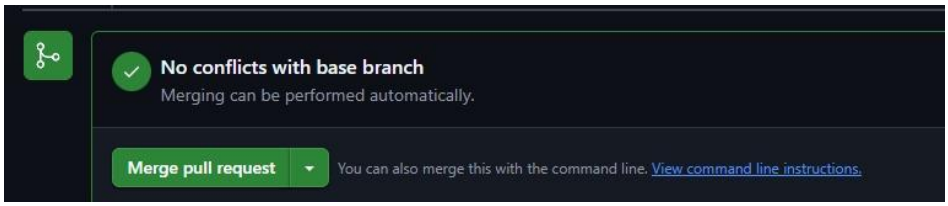
• Decide cómo resolver el conflicto. Puedes mantener ambos cambios, elegir uno de ellos, o fusionar los contenidos de alguna manera.

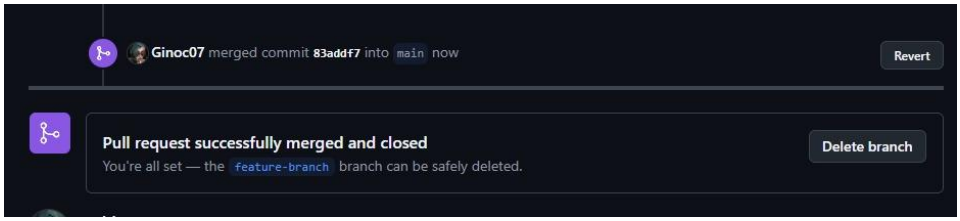
• Edita el archivo para resolver el conflicto y guarda los cambios(Se debe borrar lo marcado en verde en el archivo donde estes solucionando el conflicto. Y se debe borrar la parte del texto que no se quiera dejar).

• 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

```
gino@DESKTOP-R8NPIR8 MINGW64 ~/conflict-excersise (main)
$ git push origin main
To https://github.com/Ginoc07/conflict-excersise.git
 ! [rejected]        main -> main (fetch first)
error: failed to push some refs to 'https://github.com/Ginoc07/conflict-excersis
e.git'
hint: Updates were rejected because the remote contains work that you do not
hint: have locally. This is usually caused by another repository pushing to
hint: the same ref. If you want to integrate the remote changes, use
hint: 'git pull' before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.

gino@DESKTOP-R8NPIR8 MINGW64 ~/conflict-excersise (main)
$ git pull
remote: Enumerating objects: 17, done.
remote: Counting objects: 100% (17/17), done.
remote: Compressing objects: 100% (10/10), done.
remote: Total 15 (delta 4), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (15/15), 4.40 KiB | 102.00 KiB/s, done.
From https://github.com/Ginoc07/conflict-excersise
   8239d12..2582b7f  main       -> origin/main
Updating 8239d12..2582b7f
Fast-forward
 README.md | 4 +++-
```

```
 README.md | 4 +++-
 1 file changed, 3 insertions(+), 1 deletion(-)

gino@DESKTOP-R8NPIR8 MINGW64 ~/conflict-excersise (main)
$ git push origin main
Everything up-to-date

gino@DESKTOP-R8NPIR8 MINGW64 ~/conflict-excersise (main)
$ git push origin feature-branch
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'feature-branch' on GitHub by visiting:
remote:      https://github.com/Ginoc07/conflict-excersise/pull/new/feature-b
ch
remote:
To https://github.com/Ginoc07/conflict-excersise.git
 * [new branch]      feature-branch -> feature-branch

gino@DESKTOP-R8NPIR8 MINGW64 ~/conflict-excersise (main)
```

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.

Verifiqué y está todo en orden, también vi el historial.