Contestar las siguientes preguntas utilizando las guías y documentación proporcionada

1. ¿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 una herramienta gratuita y de código abierto que permite el trabajo en equipo en proyectos, el intercambio de código y el trabajo conjunto de forma eficiente.

2. ¿Cómo crear un repositorio en GitHub?
Primero, hay que tener una cuenta de GitHub. Una vez creada la cuenta debemos crear un repositorio utilizando el boton “New” que se encuentra en la pagina principal de GitHub. Alli se le coloca: nombre, descripcion y se elige si queremos que sea publico o privado. Luego damos click en “Create Repository” y nos proporcionara el codigo necesario a ejecutar en nuestra terminal para subir los archivos.

3. ¿Cómo crear una rama en Git?
Para crear una nueva rama se usa el comando: git branch nombreDeLaRama.

4. ¿Cómo cambiar a una rama en Git?
Para cambiar de rama se usa el comando: git checkout nombreDeLaRama.

5. ¿Cómo fusionar ramas en Git?
Primero, debes estar en la rama a la que quieres fusionar los cambios. Por lo general es la rama “master” o “main”. Supongamos que queremos fusionar en la rama “master” los cambios de la rama “feature”.
Se ejecuta el comando: git checkout master (para moverse a la rama master)
Luego se ejecuta el comando: git merge feature (para fusionar los cambios dentro de la rama master)

6. ¿Cómo crear un commit en Git?
Para crear un commit en Git debemos agregar los cambios con el comando: git add .
Luego se ejecuta el comando: git commit -m “mensaje descriptivo de los cambios”
Y esto guarda el estado actual del codigo en el historial.

7. ¿Cómo enviar un commit a GitHub?
Primero debemos hacer el commit de forma local como se explico en el anterior punto y luego ejecutar el comando: git push origin nombreDeLaRama

8. ¿Qué es un repositorio remoto?
Los repositorios remotos son repositorios que estan alojados en internet. Estos nos permiten trabajar con desarrolladores de distintas partes del mundo. Un ejemplo de repositorio remoto serian los repositorios que podemos encontrar en GitHub, mientras que los repositorios que creamos nosotros en nuestra computadora son repositorios locales.

9. ¿Cómo agregar un repositorio remoto a Git?
Para vincular un repositorio remoto con uno local se utiliza el comando: git remote add [nombre] [url] y luego usando el comando git pull origin nombreDeLaRama nos traemos el codigo que se encuentra en el repositorio remoto.

10. ¿Cómo empujar cambios a un repositorio remoto?
Para enviar los cambios tenemos que hacer un commit y luego ejecutar el comando: git push origin nombreDeLaRama

11. ¿Cómo tirar de cambios de un repositorio remoto?
Para traernos el codigo del repositorio remoto utilizamos el comando: git pull origin nombreDeLaRama

12. ¿Qué es un fork de repositorio?
El Fork de un repositorio es una copia exacta de un repositorio remoto en nuestra cuenta de GitHub.

13. ¿Cómo crear un fork de un repositorio?
Buscamos un repositorio en GitHub y encotraremos un boton que dice “Fork”. Eso nos crea una copia del repositorio en nuestra cuenta.

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 porque ese cambio que hemos realizado nosotros, sería considerado como algo que a el repositorio original le vendrían 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 un etiqueta en Git?
Es una marca que se aplica a una confirmación para identificarla. Se utiliza para marcar un punto específico en el historial de un repositorio.

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. Por ejemplo: git tag v1.0
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

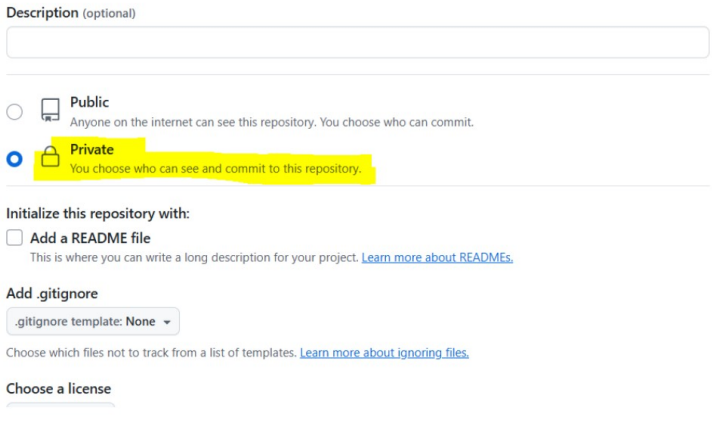
20. ¿Cómo ver el historial de Git?
Esto lo conseguimos con el comando de Git: git log

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. Esto es útil para proyectos que contienen información sensible o que aún están en desarrollo y no deseas que estén disponibles públicamente.

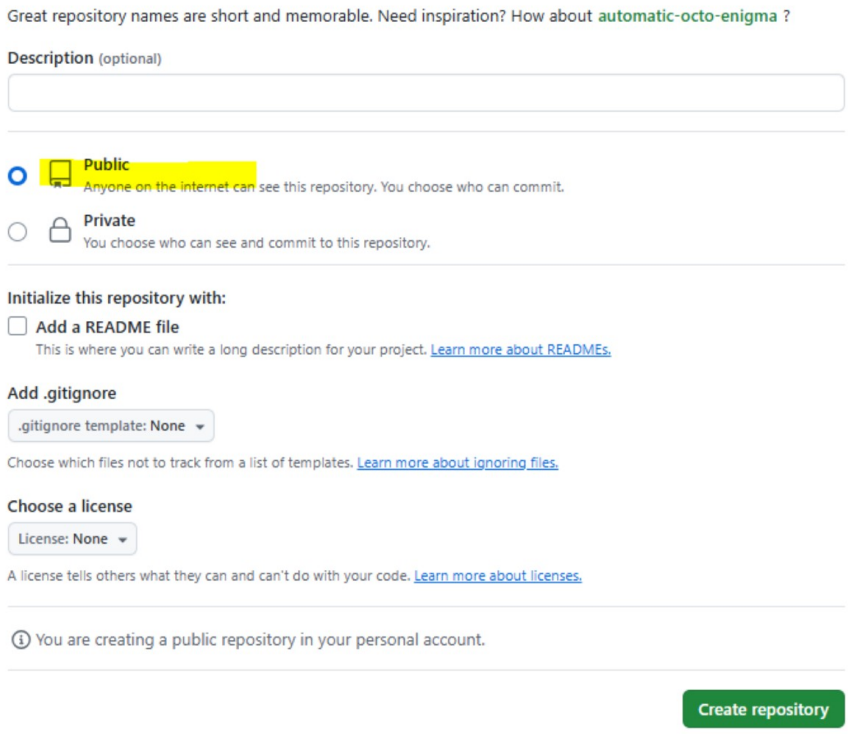
24. ¿Cómo crear un repositorio privado en GitHub?
Crear un nuevo repositorio, elegir nombre y agregar una descripcion. Abajo del apartado de descripcion se encuentra el apartado para elegir si hacer el repositorio publico o privado. Seleccionar repositorio privado y luego crear el repositorio.



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

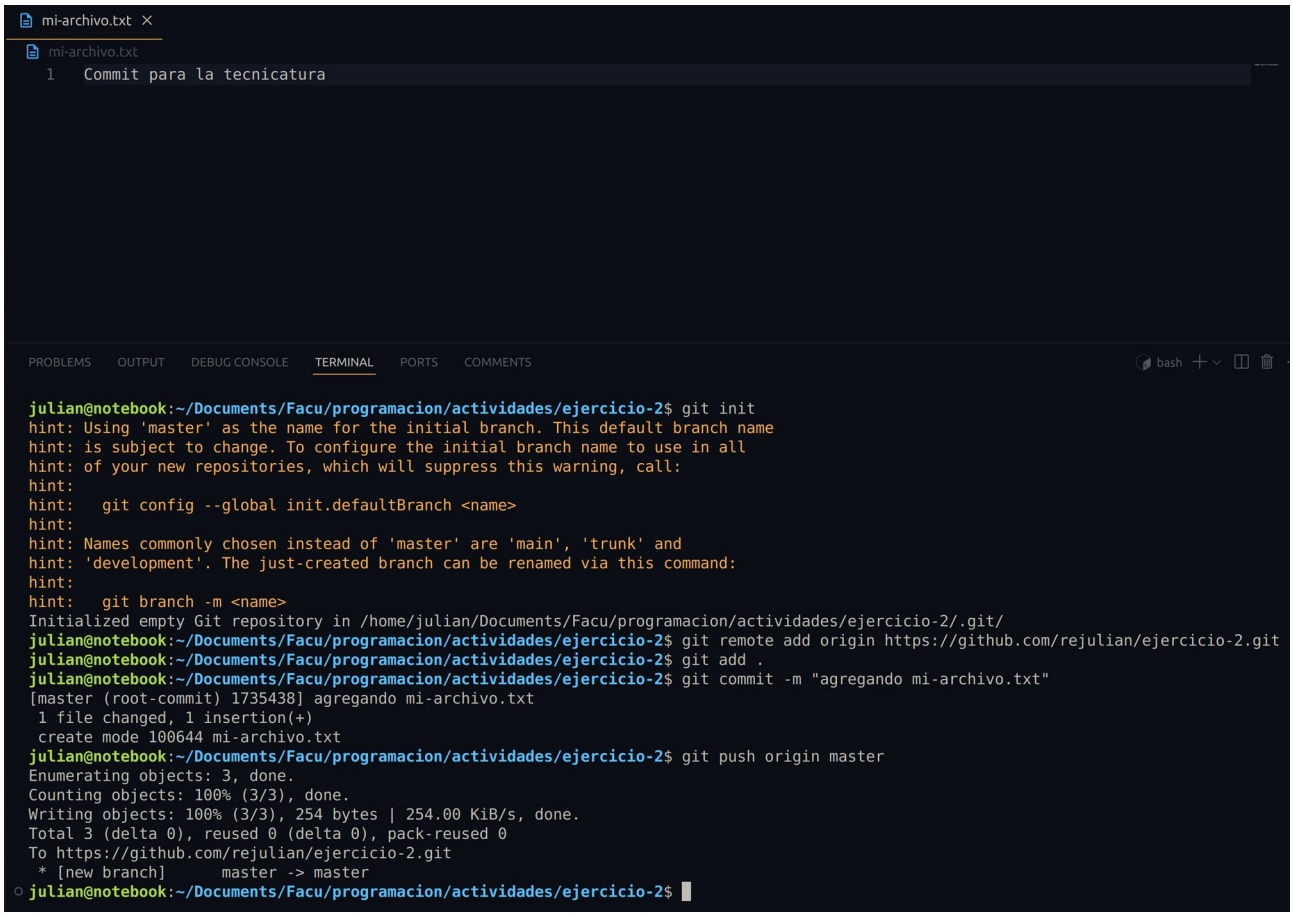
26. ¿Qué es un repositorio público en GitHub?
Un repositorio público en GitHub es un repositorio cuyo contenido es accesible a cualquier persona en Internet. Un repositorio público permite que cualquier persona pueda ver, clonar y, si tienen los permisos adecuados, contribuir al proyecto.

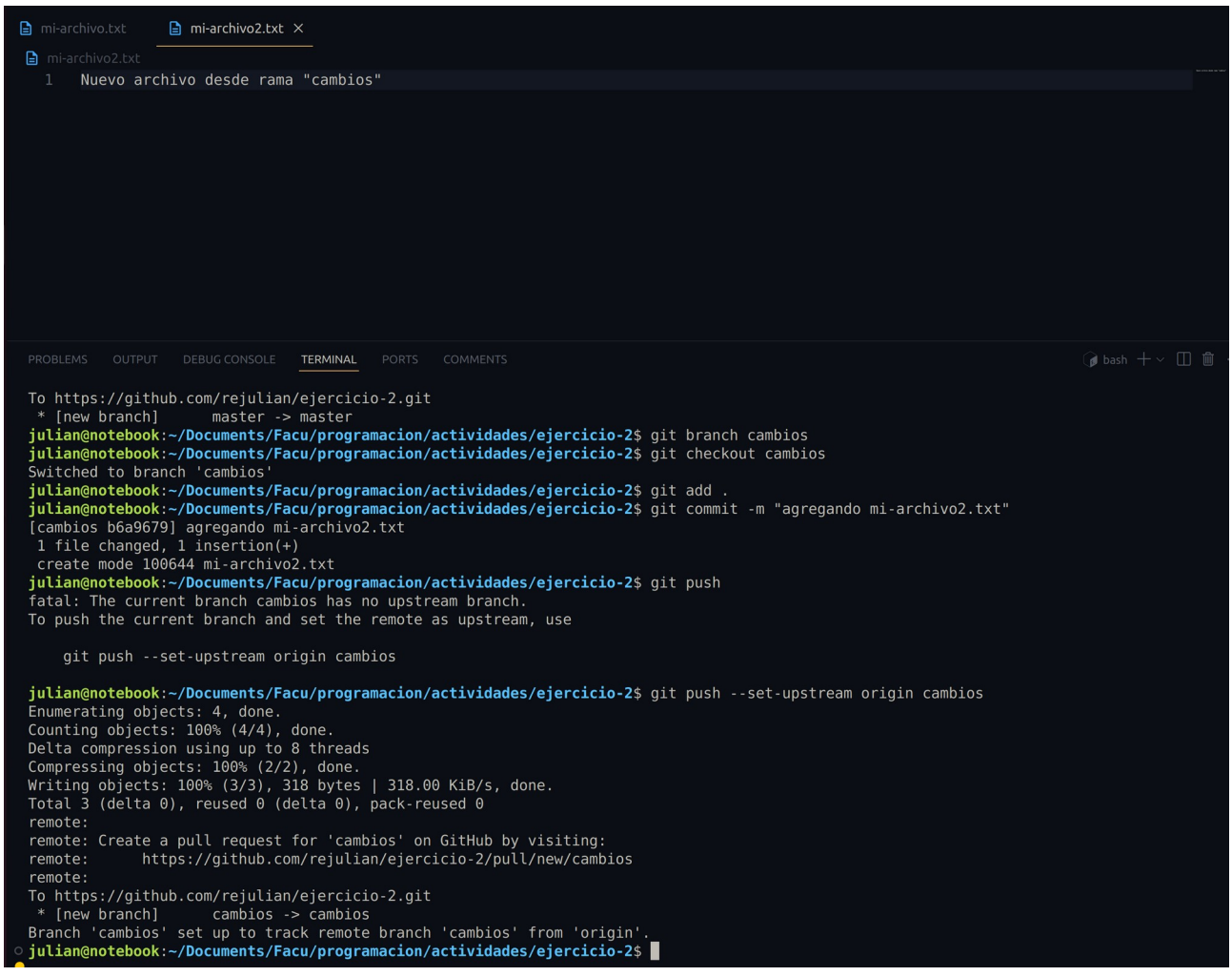
27. ¿Cómo crear un repositorio público en GitHub?
Crear un nuevo repositorio, elegir nombre y agregar una descripcion. Abajo del apartado de descripcion se encuentra el apartado para elegir si hacer el repositorio publico o privado. Seleccionar repositorio publico y luego crear el repositorio.



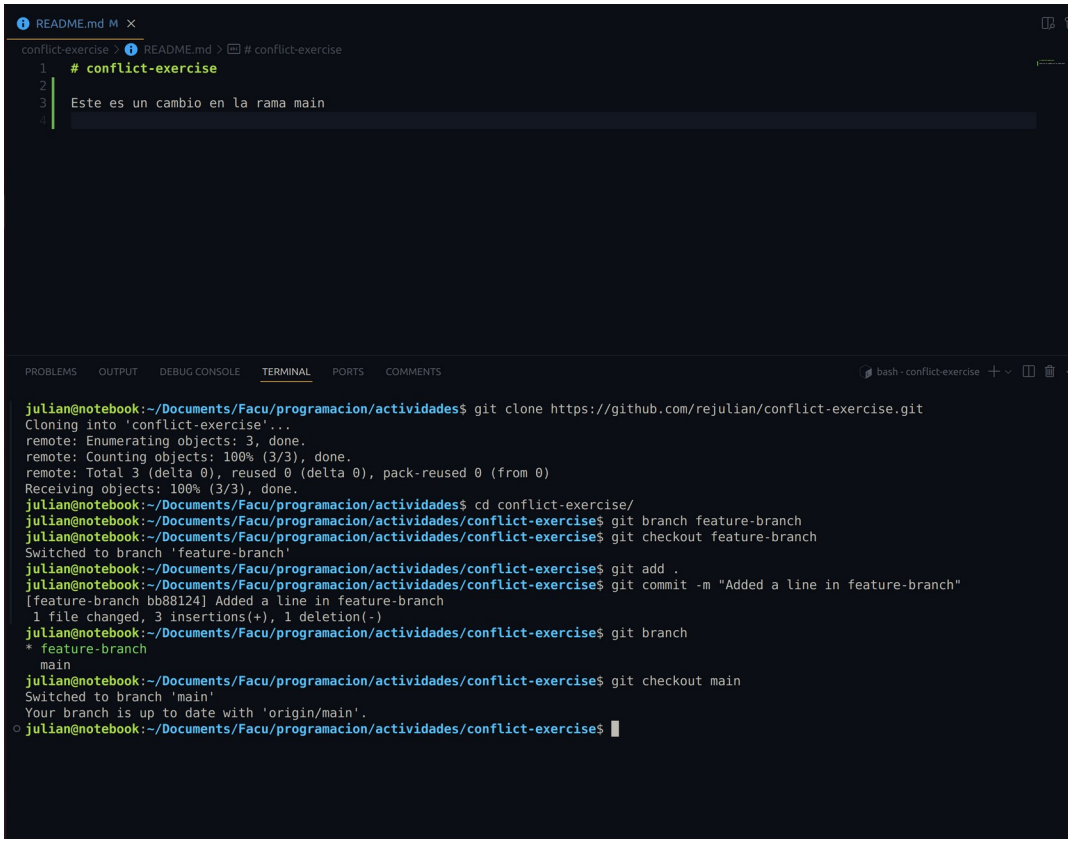
28. ¿Cómo compartir un repositorio público en GitHub?
La forma más sencilla de compartir tu repositorio es proporcionar el enlace directo al mismo.

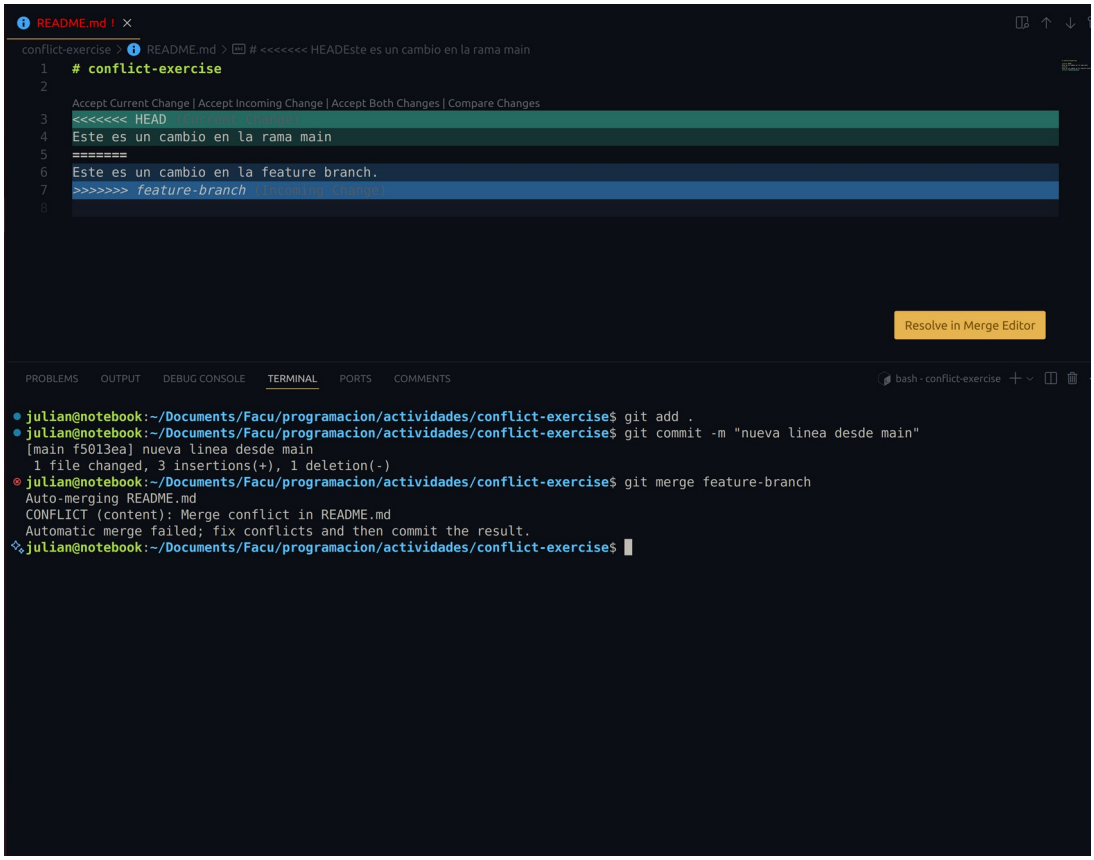
ACTIVIDAD 2

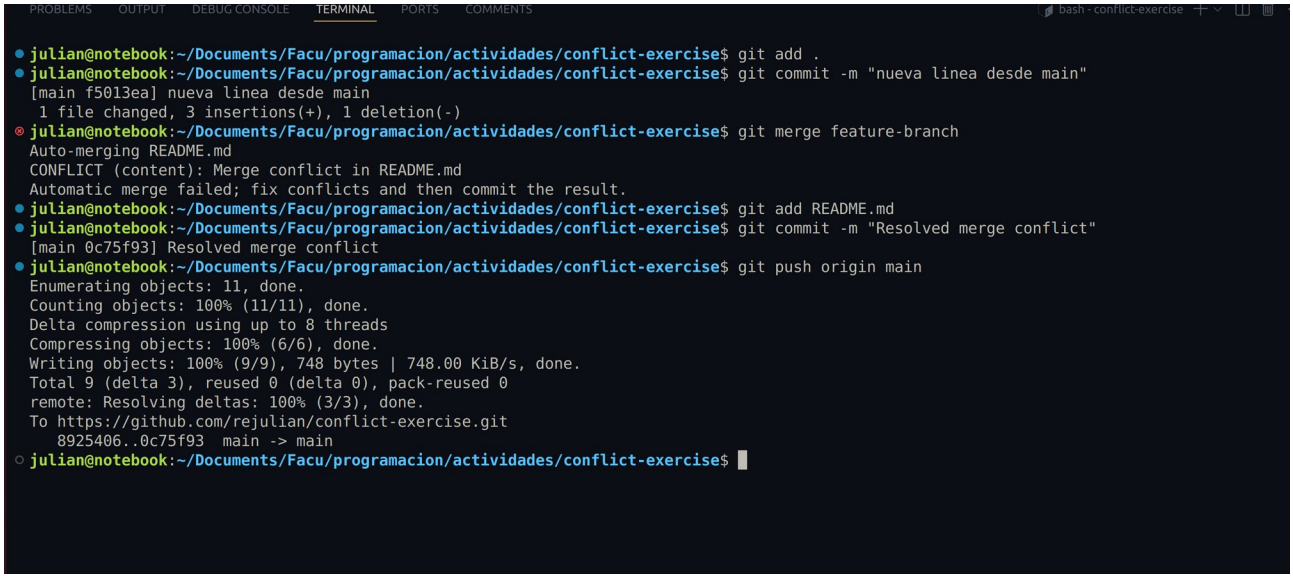

mi-archivo.txt
mi-archivo.txt
1 Commit para la tecnicatura
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS
bash
julian@notebook:~/Documents/Facu/programacion/actividades/ejercicio-2$ git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint: git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint: git branch -m <name>
Initialized empty Git repository in /home/julian/Documents/Facu/programacion/actividades/ejercicio-2/.git/
julian@notebook:~/Documents/Facu/programacion/actividades/ejercicio-2$ git remote add origin https://github.com/rejulian/ejercicio-2.git
julian@notebook:~/Documents/Facu/programacion/actividades/ejercicio-2$ git add .
julian@notebook:~/Documents/Facu/programacion/actividades/ejercicio-2$ git commit -m "agregando mi-archivo.txt"
[master (root-commit) 1735438] agregando mi-archivo.txt
1 file changed, 1 insertion(+)
create mode 100644 mi-archivo.txt
julian@notebook:~/Documents/Facu/programacion/actividades/ejercicio-2$ git push origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 254 bytes | 254.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/rejulian/ejercicio-2.git
* [new branch] master -> master
julian@notebook:~/Documents/Facu/programacion/actividades/ejercicio-2$

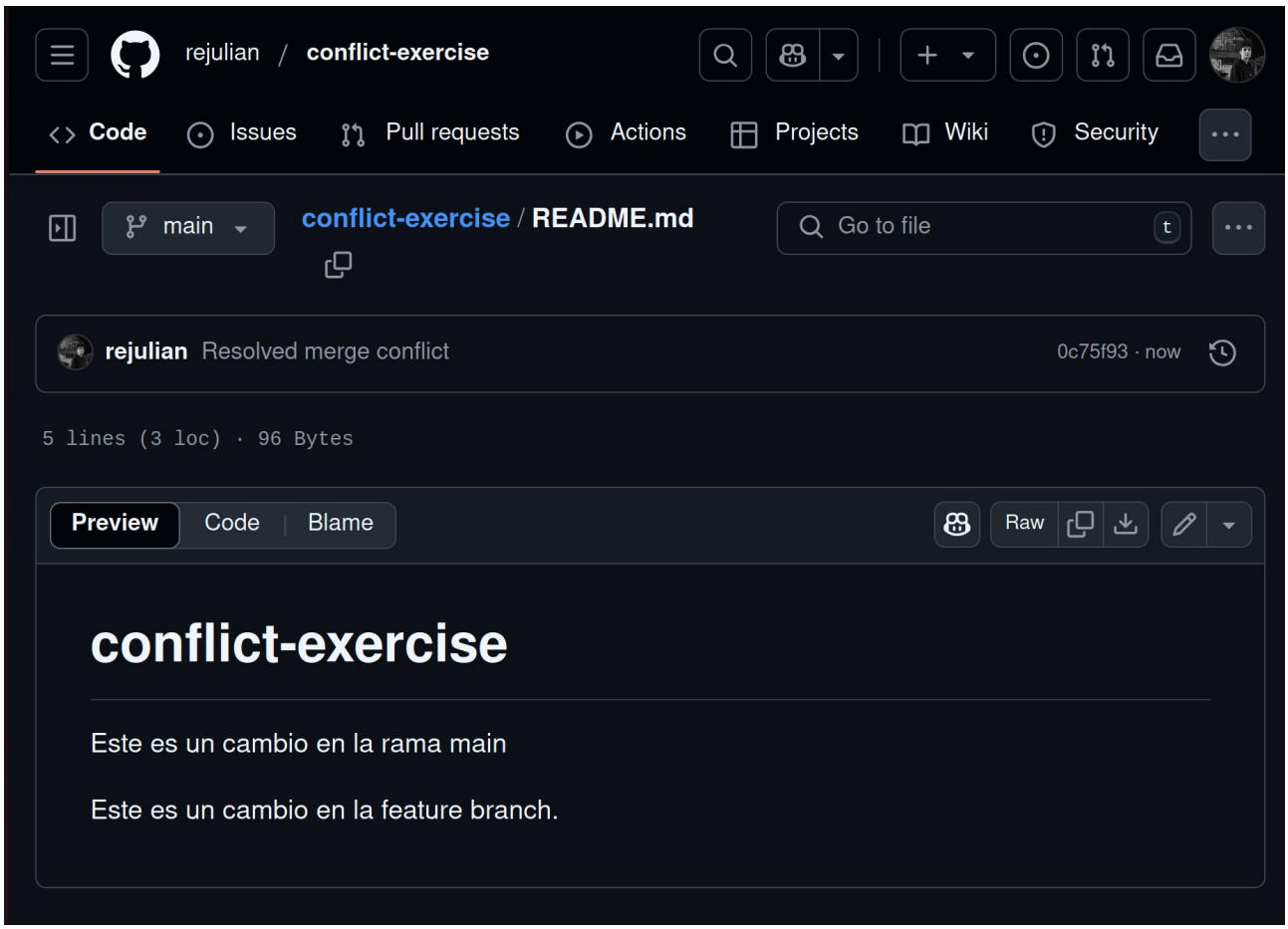

mi-archivo.txt
mi-archivo2.txt
mi-archivo2.txt
1 Nuevo archivo desde rama "cambios"
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS
bash
To https://github.com/rejulian/ejercicio-2.git
* [new branch] master -> master
julian@notebook:~/Documents/Facu/programacion/actividades/ejercicio-2$ git branch cambios
julian@notebook:~/Documents/Facu/programacion/actividades/ejercicio-2$ git checkout cambios
Switched to branch 'cambios'
julian@notebook:~/Documents/Facu/programacion/actividades/ejercicio-2$ git add .
julian@notebook:~/Documents/Facu/programacion/actividades/ejercicio-2$ git commit -m "agregando mi-archivo2.txt"
[cambios b6a9679] agregando mi-archivo2.txt
1 file changed, 1 insertion(+)
create mode 100644 mi-archivo2.txt
julian@notebook:~/Documents/Facu/programacion/actividades/ejercicio-2$ git push
fatal: The current branch cambios has no upstream branch.
To push the current branch and set the remote as upstream, use
git push --set-upstream origin cambios
julian@notebook:~/Documents/Facu/programacion/actividades/ejercicio-2$ git push --set-upstream origin cambios
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 318 bytes | 318.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'cambios' on GitHub by visiting:
remote: https://github.com/rejulian/ejercicio-2/pull/new/cambios
remote:
To https://github.com/rejulian/ejercicio-2.git
* [new branch] cambios -> cambios
Branch 'cambios' set up to track remote branch 'cambios' from 'origin'.
julian@notebook:~/Documents/Facu/programacion/actividades/ejercicio-2$

ACTIVIDAD 3


README.md M
conflict-exercise > README.md > # conflict-exercise
conflict-exercise
Este es un cambio en la rama main
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS
bash - conflict-exercise
julian@notebook:~/Documents/Facu/programacion/actividades$ git clone https://github.com/rejulian/conflict-exercise.git
Cloning into 'conflict-exercise'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.
julian@notebook:~/Documents/Facu/programacion/actividades$ cd conflict-exercise/
julian@notebook:~/Documents/Facu/programacion/actividades/conflict-exercise$ git branch feature-branch
julian@notebook:~/Documents/Facu/programacion/actividades/conflict-exercise$ git checkout feature-branch
Switched to branch 'feature-branch'
julian@notebook:~/Documents/Facu/programacion/actividades/conflict-exercise$ git add .
julian@notebook:~/Documents/Facu/programacion/actividades/conflict-exercise$ git commit -m "Added a line in feature-branch"
[feature-branch bb88124] Added a line in feature-branch
1 file changed, 3 insertions(+), 1 deletion(-)
julian@notebook:~/Documents/Facu/programacion/actividades/conflict-exercise$ git branch
* feature-branch
main
julian@notebook:~/Documents/Facu/programacion/actividades/conflict-exercise$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
julian@notebook:~/Documents/Facu/programacion/actividades/conflict-exercise$


README.md !
conflict-exercise > README.md > # <<<<<<< HEADEste es un cambio en la rama main
conflict-exercise
Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
<<<<<<< HEAD (Current Change)
Este es un cambio en la rama main
=======
Este es un cambio en la feature branch.
>>>>>>> feature-branch (Incoming Change)
Resolve in Merge Editor
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS
bash - conflict-exercise
julian@notebook:~/Documents/Facu/programacion/actividades/conflict-exercise$ git add .
julian@notebook:~/Documents/Facu/programacion/actividades/conflict-exercise$ git commit -m "nueva linea desde main"
[main f5013ea] nueva linea desde main
1 file changed, 3 insertions(+), 1 deletion(-)
julian@notebook:~/Documents/Facu/programacion/actividades/conflict-exercise$ git merge feature-branch
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.
julian@notebook:~/Documents/Facu/programacion/actividades/conflict-exercise$


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
COMMENTS
bash - conflict-exercise
julian@notebook:~/Documents/Facu/programacion/actividades/conflict-exercise$ git add .
julian@notebook:~/Documents/Facu/programacion/actividades/conflict-exercise$ git commit -m "nueva linea desde main"
[main f5013ea] nueva linea desde main
1 file changed, 3 insertions(+), 1 deletion(-)
julian@notebook:~/Documents/Facu/programacion/actividades/conflict-exercise$ git merge feature-branch
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.
julian@notebook:~/Documents/Facu/programacion/actividades/conflict-exercise$ git add README.md
julian@notebook:~/Documents/Facu/programacion/actividades/conflict-exercise$ git commit -m "Resolved merge conflict"
[main 0c75f93] Resolved merge conflict
julian@notebook:~/Documents/Facu/programacion/actividades/conflict-exercise$ git push origin main
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 8 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (9/9), 748 bytes | 748.00 KiB/s, done.
Total 9 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), done.
To https://github.com/rejulian/conflict-exercise.git
8925406..0c75f93 main -> main
julian@notebook:~/Documents/Facu/programacion/actividades/conflict-exercise$


rejulian / conflict-exercise
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
main
conflict-exercise / README.md
Go to file
t
rejulian Resolved merge conflict
0c75f93 · now
5 lines (3 loc) · 96 Bytes
Preview
Code
Blame
Raw
conflict-exercise
Este es un cambio en la rama main
Este es un cambio en la feature branch.