① Pasos Realizados:

Primero Luis creo el repositorio ToDoApp-Group1 en GitHub y agrego a rafael y a leisy como colaboradores. Luego cada uno clono en su computadora para trabajar, decidimos que cada quien tendria su propia rama: Luis, Rafael y Leisy. En nuestras ramas fuimos anadiendo las funciones asignadas. Luis marcaba tareas como completadas, Rafael las eliminaba y leisy se encargo de probocar un conflicto para resolverlo. Cuando terminamos fuimos fusionando todo en la rama Group-1 ahi fue donde aparecio el conflicto para aprender a resolverlo. Cuando terminamos fuimos fusionando todo en la rama group-1. finalmente hicimos el pull Request para unir el group-1 con main y limpiamos las ramas que ya no se necesitaban

② Comandos de GIT utilizados y para que sirvieron

Git clone <URL> → trae el repositorio a nuestra computadora.
Git checkout -b <rama> → Crea y entra en una nueva rama.
Git add <archivo> → Guardar cambios para prepararlos antes del commit.
Git commit -m "mensaje" → Confirmar los cambios con una descripción.
Git push origin <rama> → Subir la rama y sus cambios a github
Git merge <rama> → Juntar cambios de otras ramas con la que estamos trabajando
Git push origin --delete <rama> → Borrar ramas del Repositorio Remoto
Git log --oneline --graph --all → Ver de forma resumida como se ha movido el proyecto y sus ramas

③ Conflicto y Solución:

El conflicto principal fue en task_model.py, porque había dos formas de hacer lo mismo: is_completed contra is_done, y mark_as_complete contra set_done. En main.py también teníamos cambios en la misma parte.
Para resolverlo, juntamos lo mejor de cada uno y decidimos dejar is_completed como atributo principal, ajustando los metodos para que no se repitieran funciones.

④ Contribuciones de cada uno

Luis: Agrego la función mark_as_complete y actualizo main.py

Rafael: Agrego la función delete_task y actualizo el README.md

Leisy♥: Creo el conflicto con set_done y remove_task y luego lo resolvió

¿Como coordinamos el trabajo?

Nos pusimos de acuerdo que archivo haría cada uno y avisamos cuando íbamos hacer algo grande

¿Porque son importantes los Pull Requests?

Porque permite revisar lo que hizo antes que entre al codigo principal, evitando errores y mantenimiento de calidad.

¿Que aprendimos sobre resolver conflictos en Git?

¡Que es clave leer bien las marcas de conflicto y combinar los cambios de forma que nada se pierda.

¿Como mejoraríamos con CI/CD?

Configurando pruebas automaticas que validen el codigo antes de fusionarlo, así sabríamos si algo rompe la aplicación.