Informe del grupo 10

Integrantes:

- Estudiante 1: Andres Felipe Sosa yepes
- Estudiante 2: Danellys Paola Iriarte Martinez
- Estudiante 3: Ana Milena Maury Palacios

1. Pasos realizados

- Se creo el repositorio toDoApp-Group10 en Github con el archivo README.md y se configuro el proyecto de la rama main.
- Cada integrante creo su propia rama (Estudiante1, estudiante2, estudiante3) y trabajo de forma independiente.
- Se realizaron modificaciones en task_model.py, main.py, y README.md, segun la rama
- Se creo la rama grupo10 desde main para integrar los aportes.
- Se fucionaron las ramas sin conflicto, Excepto estudiante3 que genero un conflicto en task_model.py y main.py; este fue resuelto unificando los cambios.
- Se subieron los cambios y se hizo un pull request desde group10 hasta main, el cual fue aprobado.
- Finalmente, se eliminaron todas las ramas individuales y de integracion.

2. Comandos Git utilizados

- git clone: Clonar el repositorio.
- git checkout -b : Crear nueva rama.
- git add / git commit : preparar y guardar cambios.
- git push / git pull : Subir o traer cambios del repo remoto.
- git fetch / git merge: Actualizar y unir ramas

- git branch -a: Ver ramas locales y remotas.
- git push --delete: Eliminar rama remota.
- git log --oneline --graph --all: Ver historial de commits.

3. Conflictos

- Se presento conflicto en el archivo task_model.py relacionado con varios metodos, entre ellos: is-done, remove_task, set_done, get_task-name, is-completed, delete_task y mark_as_complete.
- la solucion consistió en conservar y combinar los aportes de ambas ramas, segun lo indicado en el documento.

4. Contribuciones.

- Estudiante 1: Configuró el proyecto inicial, trabajo en su rama, resolvio conflictos, gestiono funciones priocipales y realizo pull request final.
- Estudiante 2: Desarrollo en su rama, subio sus cambios, redacto el archivo REPORT.txt y colaboro en la version manuscrita y la revision del pull request.
- Estudiante 3: trabajo en su rama y subio funcionalidades.

5. Reflexiones.

Aprendimos a usar git de forma colaborativa, aprovechando las ramas para trabajar sin interferencias.
Comprendimos como resolver conflictos mediante comunicacion efectiva y valoramos los pull request como herramienta clave para validar y evitar errores antes de fusionar cambios

6. Preguntas de reflexión

- ¿Como coordinaron el trabajo en equipo para evitar conflictos innecesarios?

Se asignaron tareas y se trabajo en ramas separadas para evitar conflictos.

- ¿Que aprendieron sobre la resolución de conflictos en git?

Aprendimos a identificar y resolver conflictos de manera efectiva, comprendiendo la importancia de la comunicación en equipo para entender su origen y solucionarlos adecuadamente

- ¿Porque es importante usar pull request en proyectos colaborativos?

porque permite revisar, comentar y validar los cambios antes de integrarlos, asegurando mayor control y calidad del codigo final.

- ¿Como podria mejorarse el flujo de trabajo con herramientas como CI/CD?

El uso de herramientas CI/CD (integracion y entrega continua) permiten automatizar pruebas, detectar errores tempranamente y mantener la calidad del proyecto.