Informe Grupo 1

Integrantes: David Escamilla, Alejandro Cancino, Sebastián Arteta.

1. Pasos realizados

Nuestro equipo llevó a cabo el desarrollo de una aplicación de lista de tareas (ToDoApp) siguiendo un flujo de trabajo colaborativo en GitHub:

- Creamos el repositorio público ToDoApp en GitHub y añadimos a todos los integrantes como colaboradores.
- Cada uno creó su propia rama de trabajo (david-escamilla, Alejandro, sebastian) para desarrollar funcionalidades específicas
- Desarrollamos las funciones asignadas de forma independiente y luego fuimos integrando los cambios en una rama de grupo (group-1).
- Durante la fusión, surgieron conflictos en el archivo task_model.p y en main.py, los cuales resolvimos manteniendo la coherencia del código
- Finalmente, realizamos la integración de todos los cambios en la rama main mediante un Pull Request, asegurando la revisión y validación previa

2. Comandos Git utilizados y su propósito

- git clone <URL> → Clonar el repositorio desde GitHub.
- git checkout -b <rama> → Crear y cambiar a una nueva rama.
- git add <archivo> → Añadir cambios al área de preparación.
- git commit -m "mensaje" → Guardar los cambios en la rama actual
- git push origin <rama> → Subir los cambios al repositorio remoto.
- git merge <rama> → fusionar otra rama en la actual
- git push origin --delete <rama> → Eliminar ramas remotas que ya no se utilizan

3. Conflictos y su solución

Tuvimos un conflicto al fusionar la rama de sebastian con la rama del grupo, debido a que en task_model.py existían métodos y atributos distintos con nombres distintos (is_completed vs is_done, y mark as_complete vs set_done).

Optamos por mantener is_completed como atributo principal y unificamos las funcionalidades de ambos métodos para evitar duplicidad. En main.py combinamos los cambios de forma que se preservan todas las nuevas funciones sin sobrescribir código existente.

4. Contribuciones individuales

- David implementó el método mark as_complete en task_model.py y actualizó main.py para integrarlo
- Alejandro añadió la función delete_task en task_model.py y actualizó README.md con instrucciones de uso.
- Sebastian simuló el conflicto añadiendo set_done y remove_task, y se encargó de la resolución de conflictos durante la integración final.

5. Reflexiones y lecciones aprendidas

Trabajar en equipo usando GitHub nos permitió entender la importancia de la organización y comunicación para evitar problemas innecesarios. Los conflictos que surgieron fueron una oportunidad para aprender a resolver discrepancias de código de forma ordenada y consensuada.

- El trabajo en ramas independientes permite avanzar en paralelo sin bloquear a los demás.
- Resolver conflictos requiere atención, comunicación y respeto por el trabajo de los compañeros

En resumen, esta experiencia nos dejó más seguros en el manejo de GitHub, en la colaboración remota y resolución de conflictos.