Documentación

1. Pasos realizados:

- Inicialización y Clonación del repositorio
- Creación de los archivos
- Commit y push de los archivos
- Creación de la rama #1 Kevin
- Modificación de los archivos (main.py y task_Model.py)
- Commit y push de los nuevos cambios en la rama estudiante 1- Kevin
- Creación de la rama estudiante 2- Javier
- Modificación de los archivos task_Model y README
- Creación de la rama estudiante 3- Mateo
- Modificación de los archivos task_model y main.py
- Commit y push de los archivos en la rama estudiante 2- Javier
- Creación de la rama group - Kjm
- Merge de estudiante1- Kevin con la rama group - Kjm
- Merge de la solución de Conflictos de la rama estudiante 2- Javier
- Merge y solución de Conflictos (Combinación de los cambios en task_model) con la rama estudiante 3- Mateo
- Pull request desde group - Kjm al main y fusión de los cambios
- Eliminación de todas las ramas creadas

2. Comandos GIT:

- git init: Inicializa un nuevo repositorio Git en la carpeta actual
- git clone <URL>: Clona un repositorio remoto en tu maquina local
- git add <archivos>: Añade archivos al area de preparación (staging area) para incluirlos en el proximo commit
- git add .: Añade todos los archivos modificados y nuevos al area de preparación

- git commit -m "mensaje": guarda los cambios en el historial de git con un mensaje descriptivo
- git push origin <Rama>: Envía los commits locales a la rama especificados en el repositorio remoto
- git checkout -b <nombre-rama>: Crea una nueva rama y cambia a ella
- git status: Muestra el estado actual del repositorio
- git switch <rama>: Cambia a otra rama existente
- git merge <rama>: Fusiona los cambios de otra rama en la rama actual
- git merge --abort: Cancela un merge en proceso y vuelve al estado anterior
- git fetch origin: descarga los cambios del repositorio remoto sin aplicarlos todavía
- git branch -D <nombre-rama>: Elimina una rama local de manera forzada
- git push origin --delete <nombre rama>: Elimina una rama en el repositorio remoto
- git log: Muestra el historial de commits con autor, fecha y mensaje
- git branch -r: lista todas las ramas remotas disponibles en el repositorio

3. Conflictos:

- Conflicto en task_model.py entre ls_mark y ls_delete en el 2do merge
- Conflicto en task_model.py entre ls_completed/ls_done y mark_as_completed/set_done
- Conflicto en el main.py, combinación de prints en el 3er merge

4. Contribuciones:

- Estudiante 1 (Kevin Carpio): Creo el repositorio, creo la rama group-KJM, añadió mark_as_completed y actualizó main.py
- Estudiante 2 (Javier Herazo): Añadió delete_task y actualizó README.md

-Estudiante 3 (Mateo Zambrano): Simulé conflicto con set_done/remove_task y lo resolvió

5. Reflexión:

Durante la creación y gestión de la rama surgieron pequeños conflictos como commits sin mensaje, errores al hacer push por no configurar el upstream y confusiones con los stashes. Estos inconvenientes muestran la importancia de seguir un flujo ordenado: Preparar bien los cambios, escribir mensajes claros y configurar la rama remota desde el inicio, son detalles simples, pero marcan la diferencia para evitar retrasos y mantener un manejo de git mas fluido y eficiente