1) Contestar las siguientes preguntas utilizando las guías y documentación proporcionada (Desarrollar las respuestas):

• ¿Qué es GitHub?
GitHub es una herramienta en la nube basada en el sistema de control de versiones Git que permite la colaboración entre programadores para desarrollar, crear y compartir código fuente.

• ¿Cómo crear un repositorio en GitHub?
Para crear un repositorio en GitHub es necesario tener una cuenta registrada en la plataforma. Una vez loggeados en ella iremos a la esquina superior derecha en nuestro perfil y localizaremos la opción “create new” para luego ir a “new repository”. Le damos un nombre al repositorio, verificamos demás opciones que nos ofrece la página y finalmente le damos clic a “create repository”.

• ¿Cómo crear una rama en Git?
Crear una rama en Git es tan fácil como escribir el siguiente comando en la terminal: “git branch nueva_rama” (sin comillas), donde en nueva_rama pondremos el nombre que queramos.

• ¿Cómo cambiar a una rama en Git?
Para cambiar de rama en Git usamos “git checkout rama_a_moverse” (sin comillas y en rama_a_moverse ponemos el nombre de la rama a la que queramos ir).

• ¿Cómo fusionar ramas en Git?
Para fusionar ramas, debemos estar parados en la rama “principal” que es a la que le queremos fusionar la otra rama. Una vez estemos ahí debemos escribir en la terminal “git merge rama_secundaria” (sin comillas y en rama_secundaria se escribe el nombre de la otra rama).

• ¿Cómo crear un commit en Git?
Un commit en Git se crea después haber elegido los cambios que se quieren guardar con el comando “git commit –m ‘aca se incluye un breve texto que describa los cambios que se hacen’”.

• ¿Cómo enviar un commit a GitHub?
Para enviar un commit a GitHub primero que nada tenemos que tener asociado el repo local a uno remoto. Luego de esto se usa el comando “git push origin main (o el nombre de la rama que queremos enviar)”.

• ¿Qué es un repositorio remoto?
Un repositorio remoto es una copia del proyecto que está alojada en algún servidor en la nube.

• ¿Cómo agregar un repositorio remoto a Git?
Podemos agregar un repo remoto con el comando “git remote add origin url” donde en url escribimos la dirección del repositorio. También se puede clonar directamente desde el servidor y nos ahorraríamos este paso.

• ¿Cómo empujar cambios a un repositorio remoto?
Para empujar cambios se usa el comando “git push”.

• ¿Cómo tirar de cambios de un repositorio remoto?
Para traernos los cambios de un repositorio remoto usamos “git pull”.

• ¿Qué es un fork de repositorio?
Un fork es un procedimiento parecido a clonar solo que no nos descarga el proyecto en nuestro servidor local sino que crea uno idéntico en nuestra cuenta en la nube, teniendo obviamente una url distinta a la del original y a partir de ahí podemos trabajar libremente sobre él.

• ¿Cómo crear un fork de un repositorio?
Para crear un fork debemos ir al repositorio en cuestión y buscar la opción “fork” que suele estar arriba a la derecha. Una vez entramos ahí le agregamos el nombre que le queremos dar a nuestra copia, revisamos que las demás opciones que aparecen estén bien y finalizamos clickeando “create fork”.

• ¿Cómo enviar una solicitud de extracción (pull request) a un repositorio?
Para hacer eso debemos ir a la página del repositorio, buscar la rama que contenga los cambios y hacer click en “compare & pull request”, elegir la rama base y la de comparación, darle un título y una descripción y finalmente clickear en “crear solicitud de incorporación de cambios”.

• ¿Cómo aceptar una solicitud de extracción?
Para aceptar la solicitud de extracción vamos al repositorio, ubicamos la opción “pull request” y elegimos de una lista la solicitud a revisar. Clickeamos en “files changed”, revisamos los cambios hechos y podemos opcionalmente dejar algún comentario. Clickeamos en “review changes”, dejamos algún retroalimentació, aprobamos y enviamos.

• ¿Qué es una etiqueta en Git?
Una etiqueta es una marca que se aplica a una confirmación para identificar un punto específico en el historial del repositorio.

• ¿Cómo crear una etiqueta en Git?
Para crear una etiqueta usamos el comando “git tag nombre_etiqueta”.

• ¿Cómo enviar una etiqueta a GitHub?

Para hacer eso se usa el comando “git push origin nombre_etiqueta”. Si queremos enviar varias a la vez podemos usar “git push origin –tags”.

• ¿Qué es un historial de Git?
El historial de Git es un registro de las confirmaciones del repositorio, es decir de los commits que se fueron haciendo.

• ¿Cómo ver el historial de Git?
Se puede ver el historial escribiendo en la terminal “git log” o “git log –oneline” para verlo un poco más resumido.

• ¿Cómo buscar en el historial de Git?
Se puede buscar de varias formas: por palabras claves en el commit “git log –grep = palabra_clave”
Por autor “git log –autor = autor”
Por fecha “git log –since= fecha_desde –until = fecha_hasta”
Por cambios en un archivo específico “git log –direccion_del_archivo”
Por el contenido modificado “git log –S ‘contenido_agregado_o_eliminado’”

• ¿Cómo borrar el historial de Git?
Se puede borrar el historial usando en la consola “rm –rf .git”

• ¿Qué es un repositorio privado en GitHub?
Es un espacio seguro en la plataforma donde se puede guardar archivos y código sin que sea visible al público.

• ¿Cómo crear un repositorio privado en GitHub?
Se crea igual que uno público pero marcando la opción “private” antes de crearlo.

• ¿Cómo invitar a alguien a un repositorio privado en GitHub?
Para hacer esto vamos a la configuración en la página del repo, hacer clic en “colaboradores y equipos”, “agregar personas” y escribir el usuario o correo que queremos invitar. Le damos un rol y finalmente lo agregamos.

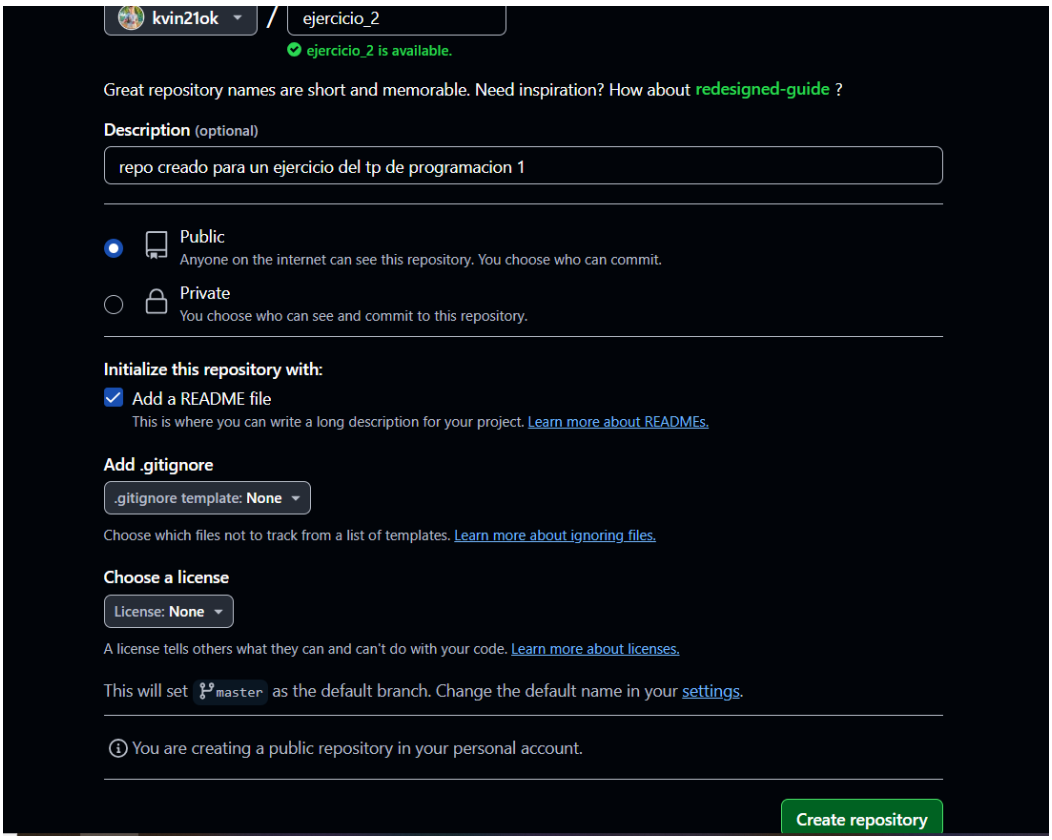
• ¿Qué es un repositorio público en GitHub?
Un repo público es un espacio para almacenar archivos y código y hacer revisiones de forma gratuita y accesible para todos.

• ¿Cómo crear un repositorio público en GitHub?
Para crear un repositorio en GitHub es necesario tener una cuenta registrada en la plataforma. Una vez loggeados en ella iremos a la esquina superior derecha en nuestro perfil y localizaremos la opción “create new” para luego ir a “new repository”. Le damos un nombre al repositorio, verificamos demás opciones que nos ofrece la página, entre ellas si queremos que sea público y finalmente le damos clic a “create repository”.

• ¿Cómo compartir un repositorio público en GitHub
Para hacer esto vamos a la configuración en la página del repo, hacer clic en “colaboradores y equipos”, “agregar personas” y escribir el usuario o correo que queremos invitar. Le damos un rol y finalmente lo agregamos. Es necesario que la otra persona acepte la invitación.

---

2) Realizar la siguiente actividad:
   • Crear un repositorio.
      o Dale un nombre al repositorio.
      o Elije el repositorio sea público.

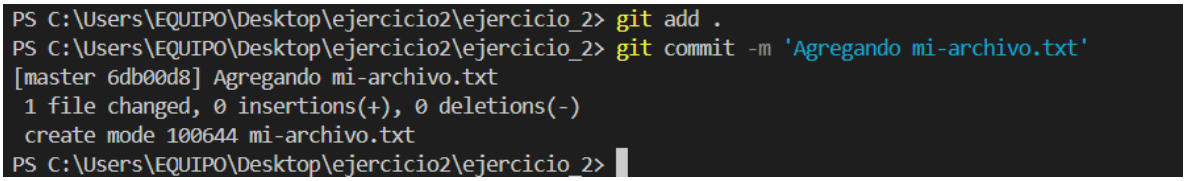


o Inicializa el repositorio con un archivo.

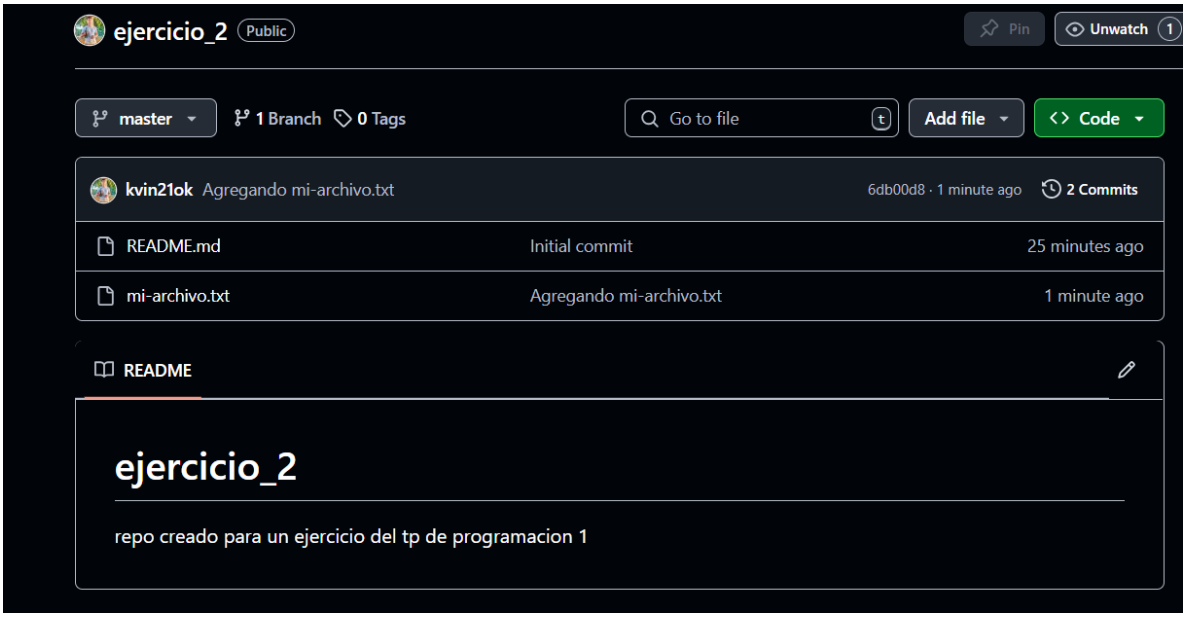
• Agregando un Archivo

o Crea un archivo simple, por ejemplo, "mi-archivo.txt".

o Realiza los comandos git add . y git commit -m "Agregando mi-archivo.txt" en la línea de comandos.



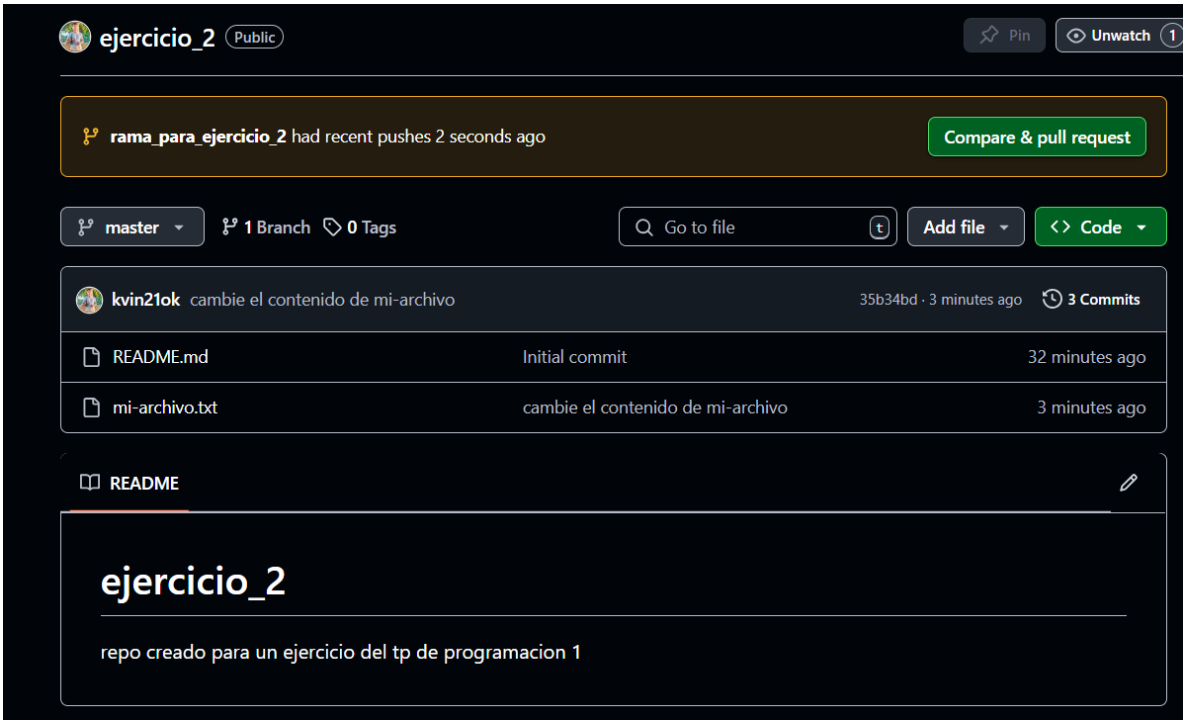
o Sube los cambios al repositorio en GitHub con git push origin main (o el nombre de la rama correspondiente).



- Creando Branchs
    - Crear una Branch
    - Realizar cambios o agregar un archivo

```
PS C:\Users\EQUIPO\Desktop\ejercicio2\ejercicio_2> git branch rama_para_ejercicio_2
PS C:\Users\EQUIPO\Desktop\ejercicio2\ejercicio_2> git branch
* master
  rama_para_ejercicio_2
PS C:\Users\EQUIPO\Desktop\ejercicio2\ejercicio_2>
```

o Subir la Branch



3) Realizar la siguiente actividad:

Paso 1: Crear un repositorio en GitHub

• Ve a GitHub e inicia sesión en tu cuenta.

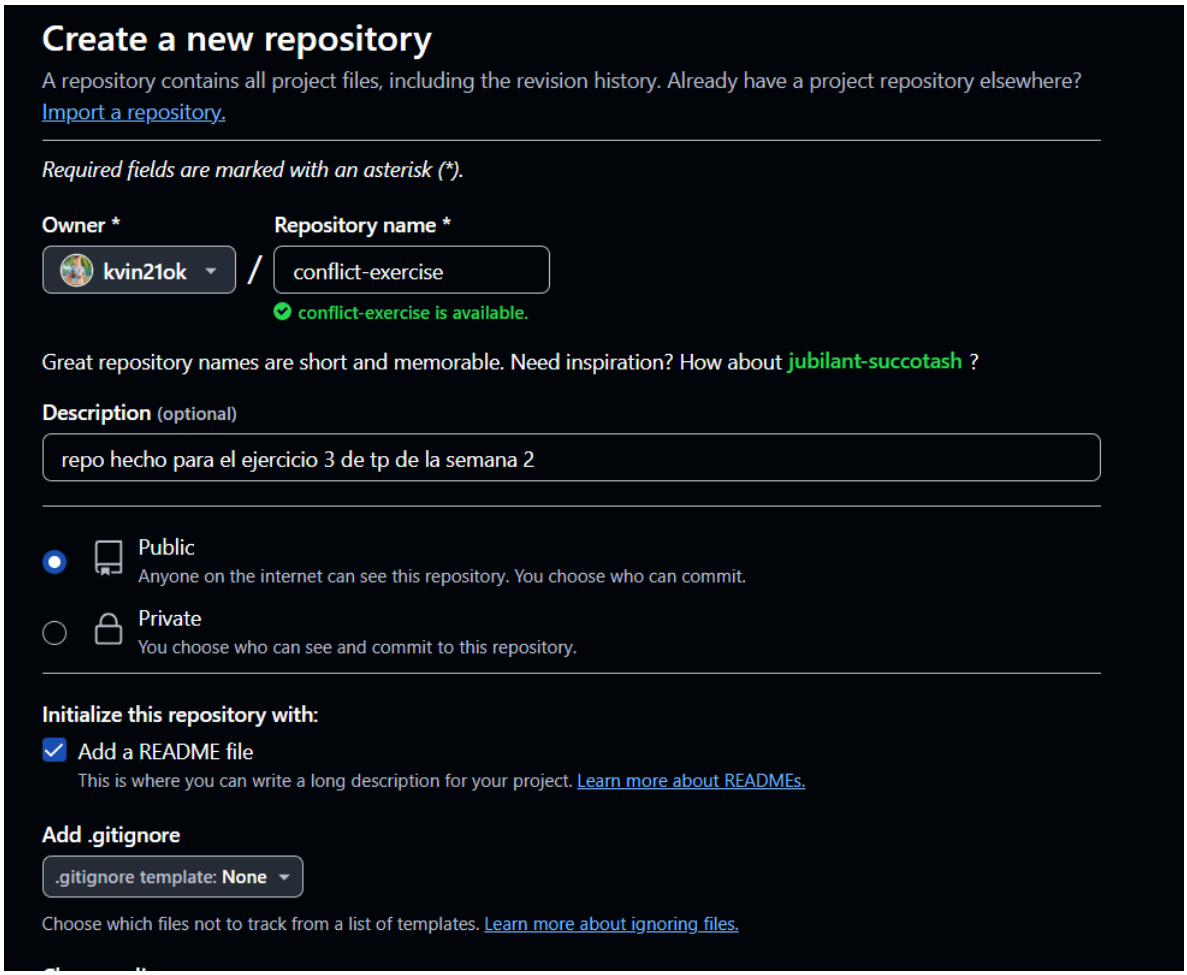
• Haz clic en el botón "New" o "Create repository" para crear un nuevo repositorio.

• Asigna un nombre al repositorio, por ejemplo, conflict-exercise.

• Opcionalmente, añade una descripción.

• Marca la opción "Initialize this repository with a README".

• Haz clic en "Create repository".



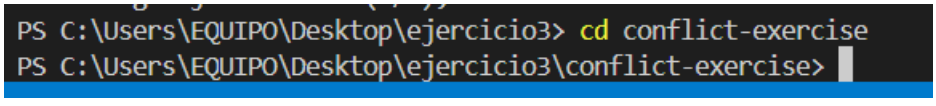
Paso 2: Clonar el repositorio a tu máquina local

• Copia la URL del repositorio

• Abre la terminal o línea de comandos en tu máquina.

• Clona el repositorio usando el comando: git clone

• Entra en el directorio del repositorio: cd conflict-exercise



Paso 3: Crear una nueva rama y editar un archivo

• Crea una nueva rama llamada feature-branch: git checkout -b feature-branch

• Abre el archivo README.md en un editor de texto y añade una línea nueva, por ejemplo: Este es un cambio en la feature branch.

• Guarda los cambios y haz un commit: git add README.md git commit -m "Added a line in feature-branch"

```
PS C:\Users\EQUIPO\Desktop\ejercicio3\conflict-exercise> git checkout -b feature-branch
Switched to a new branch 'feature-branch'
PS C:\Users\EQUIPO\Desktop\ejercicio3\conflict-exercise>
```

```
PS C:\Users\EQUIPO\Desktop\ejercicio3\conflict-exercise> git add README.md
PS C:\Users\EQUIPO\Desktop\ejercicio3\conflict-exercise> git commit -m 'escribi en el readme en la feature branch'
[feature-branch ef7c549] escribi en el readme en la feature branch
 1 file changed, 2 insertions(+)
PS C:\Users\EQUIPO\Desktop\ejercicio3\conflict-exercise>
```

Paso 4: Volver a la rama principal y editar el mismo archivo

• Cambia de vuelta a la rama principal (main): git checkout main

• Edita el archivo README.md de nuevo, añadiendo una línea diferente: Este es un cambio en la main branch.

• Guarda los cambios y haz un commit: git add README.md git commit -m "Added a line in main branch"

Paso 5: Hacer un merge y generar un conflicto

• Intenta hacer un merge de la feature-branch en la rama main: git merge feature-branch

• Se generará un conflicto porque ambos cambios afectan la misma línea del archivo README.md.

```
 1 file changed, 2 insertions(+)
PS C:\Users\EQUIPO\Desktop\ejercicio3\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.
PS C:\Users\EQUIPO\Desktop\ejercicio3\conflict-exercise> git add README.md
PS C:\Users\EQUIPO\Desktop\ejercicio3\conflict-exercise> git commit -m 'resolvi el conflicto'
```

Paso 6: Resolver el conflicto

• Abre el archivo README.md en tu editor de texto. Verás algo similar a esto: <<<<<<< HEAD Este es un cambio en la main branch. ======= Este es un cambio en la feature branch. >>>>>>> feature-branch

• Decide cómo resolver el conflicto. Puedes mantener ambos cambios, elegir uno de ellos, o fusionar los contenidos de alguna manera.

• Edita el archivo para resolver el conflicto y guarda los cambios(Se debe borrar lo marcado en verde en el archivo donde estes solucionando el conflicto. Y se debe borrar la parte del texto que no se quiera dejar).

• Añade el archivo resuelto y completa el merge: git add README.md git commit -m "Resolved merge conflict"

```
 1 file changed, 2 insertions(+)
PS C:\Users\EQUIPO\Desktop\ejercicio3\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.
PS C:\Users\EQUIPO\Desktop\ejercicio3\conflict-exercise> git add README.md
PS C:\Users\EQUIPO\Desktop\ejercicio3\conflict-exercise> git commit -m 'resolvi el conflicto'
```

Paso 7: Subir los cambios a GitHub

• Sube los cambios de la rama main al repositorio remoto en GitHub: git push origin main

• También sube la feature-branch si deseas: git push origin feature-branch

Paso 8: Verificar en GitHub

• Ve a tu repositorio en GitHub y revisa el archivo README.md para confirmar que los cambios se han subido correctamente.

• Puedes revisar el historial de commits para ver el conflicto y su resolución.