Práctico 2: Git y GitHub

Alumno: Rodriguez Miguel Andrés

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

• ¿Qué es GitHub?
GitHub es una plataforma en línea para alojar, gestionar y colaborar en proyectos de software usando Git. En ella podemos compartir nuestros repositorios de forma pública o privada. Un repositorio es un espacio donde se almacena el código fuente, historial de cambios y archivos de un proyecto.

• ¿Cómo crear un repositorio en GitHub?
Accede a GitHub e inicia sesión en GitHub.com.
En la parte superior derecha, haz clic en el botón "+" y selecciona "New repository".
Completa los campos:
Repository name: Nombre del repositorio (ejemplo: mi-proyecto).
Description (opcional): Breve descripción del proyecto.
Visibility:
Public: Cualquiera puede verlo.
Private: Solo tú y quienes invites podrán verlo.
(Opcional) Marca "Initialize this repository with a README" si quieres agregar un README.md desde el inicio.
Haz clic en "Create repository".

• ¿Cómo crear una rama en Git?
En Git, una rama (branch) permite desarrollar nuevas características sin afectar el código principal.
Para crear una nueva rama en Git, ingresa al terminal de git y escribe:
git Branch y seguido el nombre de la rama ej: git Branch rama1

• ¿Cómo cambiar a una rama en Git?
Para cambiar de rama escribe en el terminal git checkout y el nombre de la rama, ej: git checkout rama1

• ¿Cómo fusionar ramas en Git?
Para fusionar dos ramas en git se utiliza el código “merge” pero primero debes seguir una serie de pasos. Primero debes cambiar a la rama principal, ósea la rama donde quieres unir los cambios.
Ejemplo: Si quieres fusionar nueva rama en main, primero cambia a main, para esto ingresamos en la terminal git checkout main
Una vez posicionados en la rama principal ingresamos el código git merge y el nombre de la rama en este caso nueva_rama.

• ¿Cómo crear un commit en Git?
Un commit en Git es un registro de los cambios realizados en los archivos de un repositorio.

Antes de hacer un commit, revisa qué archivos han cambiado con el código:
git status
Esto mostrará los archivos modificados, nuevos o eliminados.
Luego debes agregar los archivos que quieres incluir:
Agregar un archivo específico:
git add nombre_del_archivo
Agregar todos los archivos modificados:
git add .
Una vez agregados los archivos, crea el commit con un mensaje descriptivo:
git commit -m "Mensaje explicativo del cambio realizado"

• ¿Cómo enviar un commit a GitHub?
Para Enviar los cambios al repositorio remoto de GitHub debes ingresar el siguiente código:
git push origin main

• ¿Qué es un repositorio remoto?
Un repositorio remoto es una versión de un repositorio de Git que está alojada en un servidor en línea, como GitHub

• ¿Cómo agregar un repositorio remoto a Git?
1. Crear un repositorio en GitHub (o en otro servicio)
Ve a GitHub y crea un nuevo repositorio.
No marques la opción "Initialize this repository with a README" si ya tienes archivos en tu repositorio local.
Copia la URL del repositorio (será algo como https://github.com/usuario/nombre-repo.git)
2. Vincular el repositorio local con el remoto
En la terminal, dentro de tu carpeta del proyecto, ejecuta:
git remote add origin URL_DEL_REPOSITORIO
Ejemplo:
git remote add origin https://github.com/usuario/nombre-repo.git
Esto conecta tu repositorio local con el remoto.

• ¿Cómo empujar cambios a un repositorio remoto?
"Empujar" cambios significa subir los cambios locales a un repositorio remoto en GitHub, GitLab o Bitbucket. Se ingresa por la terminal los siguientes códigos:
git status # Ver archivos modificados
git add . # Agregar todos los cambios
git commit -m "Corrige error en el formulario" # Crear commit
git push origin main # Subir el commit al repositorio remoto

• ¿Cómo tirar de cambios de un repositorio remoto?
Tirar cambios significa descargar y actualizar tu repositorio local con los últimos cambios del repositorio remoto.
Antes de tirar cambios, cambia a la rama en la que quieres actualizar los cambios:
git checkout main

Luego para descargar y fusionar automáticamente los cambios en tu código local:
git pull origin main

• ¿Qué es un fork de repositorio?
Un fork es una copia de un repositorio que se crea en tu cuenta de GitHub, desde otro repositorio público o privado.
Es una forma de copiar un proyecto para modificarlo sin afectar el repositorio original.

• ¿Cómo crear un fork de un repositorio?
1. Ir al repositorio que deseas copiar
   - Accede a GitHub y busca el repositorio que deseas forkar.
2. Hacer clic en el botón "Fork"
   - En la parte superior derecha, haz clic en Fork.
   - GitHub creará una copia del repositorio en tu cuenta.
3. Clonar el fork en tu computadora
Después de hacer un fork, clónalo en tu máquina local:
git clone https://github.com/tu-usuario/nombre-repositorio.git

• ¿Cómo enviar una solicitud de extracción (pull request) a un repositorio?
Un Pull Request (PR) es una solicitud para que los mantenedores de un repositorio revisen y fusionen tus cambios en el código original. Se usa cuando has hecho un fork y quieres contribuir con mejoras o correcciones.

Ve a tu repositorio en GitHub.

Haz clic en Compare & pull request (aparece después de hacer push).

Verifica que la comparación sea entre:

- Base repository: el repositorio original.
- Head repository: tu fork con la nueva rama.

Escribe un título y una descripción clara sobre tus cambios.

Haz clic en Create pull request.

• ¿Cómo aceptar una solicitud de extracción?
Cuando alguien envía un Pull Request (PR) a tu repositorio, puedes revisarlo, aprobarlo y fusionarlo.
1. Ve a GitHub y abre tu repositorio.
2. Haz clic en la pestaña Pull Requests.
3. Selecciona el PR que deseas revisar.
4. Lee la descripción y verifica qué cambios propone.

Antes de aceptar, revisa los cambios:
- Usa la pestaña Files changed para ver los archivos modificados.
- Si encuentras errores o mejoras, deja comentarios.

Si todo está bien, puedes aprobarlo.
Cuando estés listo para aceptar los cambios:

1. Haz clic en Merge pull request.
2. Opcionalmente, agrega un mensaje de confirmación.
3. Presiona Confirm merge.
4. Una vez fusionado, puedes eliminar la rama con Delete branch.

• ¿Qué es una etiqueta en Git?
Una etiqueta (tag) en Git es una marca en un commit específico que se usa para identificar versiones importantes del código, como lanzamientos (v1.0, v2.0, etc.).
Las etiquetas son útiles para:

- Marcar versiones estables de un proyecto.
- Facilitar el acceso a ciertos puntos del historial de Git.
- Ayudar en la gestión de versiones y despliegues.

Hay dos tipos de etiquetas:

- Etiquetas anotadas: Guardan información extra como autor, fecha y mensaje.
- Etiquetas ligeras: Solo apuntan a un commit, sin metadatos adicionales.

• ¿Cómo crear una etiqueta en Git?
Para Crear una etiqueta ligera debemos ingresar:
git tag nombre_etiqueta
Ejemplo:
git tag v1.0
Esto marca la versión v1.0 en el último commit.
Crear una etiqueta anotada
git tag -a nombre_etiqueta -m "Mensaje descriptivo"
Ejemplo:
git tag -a v1.0 -m "Versión estable 1.0"
Ver las etiquetas creadas:
git tag

• ¿Cómo enviar una etiqueta a GitHub?
Las etiquetas no se suben automáticamente, debes hacerlo con:
git push origin nombre_etiqueta
O para subir todas las etiquetas:
git push origin --tags

• ¿Qué es un historial de Git?
El historial de Git es una copia completa y detallada de todos los cambios realizados en un repositorio a lo largo del tiempo. Cada vez que realizas un commit, Git guarda un registro del estado de los archivos en ese momento, permitiéndote ver cómo ha evolucionado el proyecto.
Este historial es esencial para:

- Rastrear cambios: Ver qué cambios se hicieron, cuándo y por quién.
- Deshacer errores: Volver a versiones anteriores si algo sale mal.
- Colaborar de forma efectiva: Mantener un registro claro de las contribuciones de cada miembro del equipo.

• ¿Cómo ver el historial de Git?
En Git, puedes ver el historial de cambios realizados en un repositorio con el comando git log. Existen varias opciones para personalizar la forma en que se muestra el historial.

• ¿Cómo buscar en el historial de Git?
Git ofrece varios comandos para buscar información dentro del historial de commits. Puedes buscar por autor, fecha, palabras clave en los mensajes de commit o incluso por cambios en el código.

Buscar por palabra clave en mensajes:	git log --grep="palabra"
Buscar por autor:	git log --author="Nombre"
Buscar commits por fecha:	git log --since="AAAA-MM-DD" --until="AAAA-MM-DD"
Buscar commits que modificaron un archivo:	git log -- <archivo>
Buscar cambios en una línea de código:	git log -S "texto o código"
Ver diferencias entre commits:	git log -p
Ver historial simplificado con gráfico	git log --oneline --graph --all --decorate

• ¿Cómo borrar el historial de Git?
Borrar el historial de Git no es algo común, pero hay varias maneras de hacerlo dependiendo de tu necesidad
Si quieres borrar completamente el historial y empezar desde cero:
rm -rf .git # Borra la carpeta .git (elimina todo el historial)
git init # Reinicia el repositorio Git
git add . # Añade todos los archivos
git commit -m "Reinicio del repositorio" # Crea un nuevo commit
Si quieres eliminar el historial, pero conservar la última versión del código:
git checkout --orphan nueva-rama # Crea una nueva rama sin historial
git add . # Añade todos los archivos
git commit -m "Nuevo inicio" # Crea un nuevo commit
git branch -D main # Borra la rama anterior
git branch -m main # Renombra la nueva rama a "main"
Si solo necesitas eliminar un commit en particular, usa:

git rebase -i HEAD~n

(Reemplaza n por el número de commits que quieres revisar).

Si necesitas eliminar el historial en un repositorio remoto, usa:

git push --mirror --force

Esto reemplazará completamente el historial del repositorio remoto con el local

• ¿Qué es un repositorio privado en GitHub?
Un repositorio privado en GitHub es un repositorio que solo tú y las personas que autorices pueden ver y acceder. A diferencia de los repositorios públicos, no está disponible para el público en general.

• ¿Cómo crear un repositorio privado en GitHub?
Para crear un repositorio privado en GitHub debemos seguir los siguientes pasos:
1 -Inicia sesión en GitHub.
2 -Haz clic en el botón New (Nuevo) en la página de repositorios.
3 -Escribe un nombre para tu repositorio.
4 -En la sección Visibility (Visibilidad), selecciona Private (Privado).
5 -Opcionalmente, añade un README, .gitignore o una licencia.
6 -Haz clic en Create repository (Crear repositorio).

• ¿Cómo invitar a alguien a un repositorio privado en GitHub?
Si tienes un repositorio privado en GitHub y quieres que otra persona tenga acceso, debes agregarla como colaborador o miembro del equipo
Si el repositorio es tuyo y deseas invitar a alguien:

1 -Accede a GitHub y abre tu repositorio privado.
2 -Ve a la pestaña "Settings" (Configuración).
3 -En el menú lateral, selecciona "Collaborators" (Colaboradores) dentro de "Access" (Acceso).
4 -Haz clic en "Add people" (Agregar personas).
5 -Escribe el nombre de usuario o correo de la persona que quieres invitar.
6 -Selecciona el usuario y haz clic en "Add" (Agregar).
7 -GitHub enviará una invitación al usuario. Debe aceptarla para obtener acceso.

• ¿Qué es un repositorio público en GitHub?
Un repositorio público en GitHub es un repositorio que cualquier persona en internet puede ver y clonar. Es ideal para proyectos de código abierto, documentación o cualquier contenido que quieras compartir con la comunidad.

• ¿Cómo crear un repositorio público en GitHub?
Crear el repositorio en GitHub

1 -Inicia sesión en GitHub
2 -Haz clic en el botón "+" en la esquina superior derecha.
3 -Selecciona "New repository" (Nuevo repositorio).
4 -Escribe un nombre para el repositorio.
5 -Opcionalmente, agrega una descripción del proyecto.
6 -En la sección "Visibility" (Visibilidad), selecciona "Public" (Público).
7 -(Opcional) Marca la casilla "Add a README file" si quieres que tenga una descripción inicial.
8 -Haz clic en "Create repository" (Crear repositorio).

• ¿Cómo compartir un repositorio público en GitHub?
La forma más fácil de compartir un repositorio público es enviando su enlace.

1 -Ve a tu repositorio en GitHub.
2 -Copia la URL de la barra de direcciones (ejemplo: https://github.com/tu-usuario/tu-repositorio).
3 -Comparte el enlace con quien quieras.

Cualquier persona con el enlace puede ver el código y clonar el repositorio.

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
o Crear una Branch
o Realizar cambios o agregar un archivo
o Subir la Branch

Solución: https://github.com/andresro87/Primer-Repositorio.git

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 (usualmente algo como https://github.com/tuusuario/conflict-exercise.git).
• Abre la terminal o línea de comandos en tu máquina.
• Clona el repositorio usando el comando:
git clone https://github.com/tuusuario/conflict-exercise.git
• 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"
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.
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"
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.

Solución : https://github.com/andresro87/conflict-exercise.git