Nombre: Santiago Gabriel Rodriguez

Trabajo practico 2: Git y GitHub

Actividades

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

1. **¿Qué es GitHub?**

GitHub es una comunidad donde podremos compartir nuestros repositorios de forma pública o privada. También podremos ver los repositorios de otras personas, clonarlos o guardarlos en favoritos.

2. **¿Cómo crear un repositorio en GitHub?**

Para crear un repositorio en GitHub primero tenemos que ingresar a la página de GitHub, ir a un botón con forma de + que se encuentra en la parte superior derecha y vamos a elegir la opción 'New repository', luego vamos a ver un apartado que dice 'Repository name', ahí vamos a ingresar el nombre del repositorio que queramos subir, luego van a darle al botón 'Create repository' y nos va a crear un repositorio vacío.

3. **¿Cómo crear una rama en Git?**

Para crear una rama en Git tenemos que usar el comando 'git branch [el nombre que queramos usar]'.

4. **¿Cómo cambiar a una rama en Git?**

Para cambiar de rama en Git tenemos que usar el comando 'git checkout [nombre de la rama a la que queremos ir]'.

5. **¿Cómo fusionar ramas en Git?**

Para fusionar ramas en Git primero debemos tener 2 ramas creadas, estar parados en una de esas ramas y usar el comando 'git marge [nombre de la rama en la que NO estamos parados]'. Ahora solo la rama en la que estamos parados tendría todo el trabajo de las dos ramas.

6. **¿Cómo crear un commit en Git?**

Para crear un commit en Git tenemos que usar el comando 'git commit', si queremos agregar un comentario de ese commit podemos usar el comando '-m', el comando completo quedaria 'git commit –m 'comentario''.

7. **¿Cómo enviar un commit a GitHub?**

Para enviar un commit a GitHub a la rama principal hay que usar el comando ‘git push origin [main o master], si es para otra rama pondríamos ‘git push origin [nombre de la rama].

8. **¿Qué es un repositorio remoto?**

Un repositorio remoto es un almacén de código que está alojado en un servidor en línea, en lugar de estar en tu máquina local. Se usa para colaborar con otros desarrolladores y mantener copias seguras del código.

9. **¿Cómo agregar un repositorio remoto a Git?**

Para agregar un repositorio remoto a Git vamos a usar el comando ‘git remote add origin [url del repositorio]’.

10. **¿Cómo empujar cambios a un repositorio remoto?**

Para empujar cambios a un repositorio remoto tenemos que usar el comando ‘git push origin [main] [o la rama que estés usando]’.

11. **¿Cómo tirar de cambios de un repositorio remoto?**

Para tirar de cambios de un repositorio remoto tenemos que usar el comando ‘git pull origin [main] [o la rama en la que estás trabajando]’.

12. **¿Qué es un fork de repositorio?**

En git un fork es una copia de un repositorio creada en una cuenta diferente permitiendo desarrollar cambios sin afectar el original. Se realiza generando una copia en la cuenta del usuario.

13. **¿Cómo crear un fork de un repositorio?**

Para crear un fork primero debemos ingresar a github y buscar donde está el repositorio, una vez estemos ahi tenemos que darle al botón ‘fork’ y cambiar la descripción del repositorio con la que quieras usar y luego le damos al botón de ‘create fork’ y se crearía una copia del repositorio original en nuestra cuenta de github.

14. ¿**Cómo enviar una solicitud de extracción (pull request) a un repositorio?**

Para enviar una solicitud de extracción (pull request) a un repositorio primero tenemos que ir a nuestro fork en GitHub, hacer clic en la pestaña ‘pull requests’, luego en ‘new pull request’, seleccionamos la rama de nuestro fork con los cambios, nos aseguramos de que el destino sea la rama principal del repositorio original (por lo general ‘main’), escribimos un título y descripción claros sobre los cambios realizados y hacemos clic en ‘create pull request’.

15. **¿Cómo aceptar una solicitud de extracción?**

Para aceptar una solicitud de extracción tenemos que ir a nuestro repositorio en GitHub, hacemos clic en la pestaña ‘pull requests’. Buscamos la PR que quieres revistar y hacemos clic en ella. Si todo está bien y queremos aceptar los cambios en la parte inferior de la PR hacemos clic en el botón ‘merge pull request’, luego confirmamos con ‘confirm merge’, con eso la PR ya ha sido aceptada.

16. **¿Qué es una etiqueta en Git?**

Una etiqueta (tag) en Git es un marcador que se usa para señalar puntos específicos en la historia del repositorio, generalmente para identificar versiones importantes. Las etiquetas son inmutables, lo que significa que no cambian con el tiempo, a diferencia de las ramas.

17. **¿Cómo crear una etiqueta en Git?**

Para crear una etiqueta en Git tenemos que usar el comando ‘git tag [nombre de la etiqueta]’.

18. **¿Cómo enviar una etiqueta a GitHub?**

Para enviar una etiqueta a GitHub tenemos que usar el comando ‘git push origin [nombre de la etiqueta]’, si queremos enviar todas las etiquetas usamos el comando ‘git push –tags'.

19. **¿Qué es un historial de Git?**

El historial de Git es el registro de todos los cambios realizados en un repositorio, organizados en commits. Permite ver quien hizo que, cuando y que rama, facilitando el seguimiento de la evolución del proyecto.

20. **¿Cómo ver el historial de Git?**

Para ver el historial de Git tenemos que usar el comando ‘git log’.

21. **¿Cómo buscar en el historial de Git?**

Hay muchas formas de buscar commits en el historial de Git, algunas de ellas pueden ser:

Por autor usando el comando ‘git log --author=’nombre del autor’’.

Por versiones anteriores usando el comando ‘git grep ‘palabra clave’’

22. **¿Cómo borrar el historial de Git?**

Para borrar el historial de Git tenemos que eliminar la carpeta oculta ‘.git’.

23. **¿Qué es un repositorio privado en GitHub?**

Un repositorio privado en GitHub es un repositorio al que solo las personas con permiso pueden acceder. No es visible para todos y no aparece en las búsquedas.

24. **¿Cómo crear un repositorio privado en GitHub?**

Para crear un repositorio privado en Github tenemos que ir a la página de GitHub y hacer clic en ‘new repository’ escribimos el nombre del repositorio, en el apartado ‘visibility’ marcamos la opción ‘private’, agregamos una descripción y hacemos clic en ‘create repository’.

25. **¿Cómo invitar a alguien a un repositorio privado en GitHub?**

Para invitar a alguien a un repositorio privado en GitHub vamos a la pestaña ‘settings’ del repositorio, en el menú lateral hacemos clic en ‘collaborators’, escribimos el nombre del usuario o correo de la persona que queremos invitar y hacemos clic en ‘add collaborator’ y esperamos a que acepte la invitación.

26. **¿Qué es un repositorio público en GitHub?**

Un repositorio público en GitHub es un repositorio al que cualquier persona puede acceder, ver y clonar. Es ideal para compartir código abierto, proyectos colaborativos o documentación a toda la comunidad.

27. **¿Cómo crear un repositorio público en GitHub?**

Para crear un repositorio publico en Github tenemos que ir a la página de GitHub y hacer clic en ‘new repository’ escribimos el nombre del repositorio, en el apartado ‘visibility’ marcamos la opción ‘public’, agregamos una descripción y hacemos clic en ‘create repository’.

28. **¿Cómo compartir un repositorio público en GitHub?**

Para compartir un repositorio público en GitHub tenemos que ir a nuestro repositorio en GitHub, copiar la URL en la barra de direcciones y compartir el enlace a quien queramos que vea nuestro repositorio.

2) Realizar la siguiente actividad:

- Crear un repositorio.
    - Dale un nombre al repositorio.
    - Elije el repositorio sea público.
    - Inicializa el repositorio con un archivo.
- Agregando un Archivo
    - Crea un archivo simple, por ejemplo, "mi-archivo.txt".
    - Realiza los comandos git add . y git commit -m "Agregando mi-archivo.txt" en la línea de comandos.
    - 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
    - Subir la Branch

Respuesta: https://github.com/Santi-R97/tp2-git-github

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.

Respuesta: https://github.com/Santi-R97/conflict-exercise