Práctico 2: Git y GitHub

Objetivo:

El estudiante desarrollará competencias para trabajar con Git y GitHub, aplicando conceptos fundamentales de control de versiones, colaboración en proyectos y resolución de conflictos, en un entorno simulado y guiado.

Resultados de aprendizaje:

1. Comprender los conceptos básicos de Git y GitHub: Identificar y explicar los principales términos y procesos asociados con Git y GitHub, como repositorios, ramas, commits, forks, etiquetas y repositorios remotos.
2. Manejar comandos esenciales de Git: Ejecutar comandos básicos para crear, modificar, fusionar y gestionar ramas, commits y repositorios, tanto en local como en remoto.
3. Aplicar técnicas de colaboración en GitHub: Configurar y utilizar repositorios remotos, realizar forks, y gestionar pull requests para facilitar el trabajo colaborativo.
4. Resolver conflictos en un entorno de control de versiones: Identificar, analizar y solucionar conflictos de merge generados en un flujo de trabajo con múltiples ramas.

Actividades

1) **Contestar las siguientes preguntas utilizando las guías y documentación proporcionada (Desarrollar las respuestas) :**
   - ¿Qué es GitHub?

   GitHub es una comunidad digital a donde podemos subir nuestros repositorios/proyectos de manera pública o privada.

   - ¿Cómo crear un repositorio en GitHub?

   Para crear un repositorio debemos claramente encontrarnos en nuestro sitio de GitHub, y una vez allí presionamos el botón “Nuevo” o bien “Nuevo repositorio".Una vez ingresamos le colocamos un nombre y una descripción al mismo, aunque esta última es opcional. Luego seleccionamos si será público o privado. Y finalmente seleccionar “Crear repositorio”. Una vez finalizado estos pasos, ya contamos con nuestro repositorio en GitHub, pero el mismo se encuentra vacío. Para “cargarle” contenido a este repositorio debemos copiar la dirección del repositorio remoto y pegarla en nuestro programa de Git mediante la instrucción: “git remote add origin” + la url, y un vez ejecutado, la

instrucción “git push -u origin master” (esto sólo la primera, luego de creado sólo “git push”.

- ¿Cómo crear una rama en Git?

Par crear una rama en Git lo hacemos mediante el comando “git branch” mas el nombre que le asignaremos a la rama.

- ¿Cómo cambiar a una rama en Git?

Para cambiar de una rama desde la cual estamos hacia otra en Git, lo hacemos mediante el comando “git checkout” mas el nombre de la rama de destino.

- ¿Cómo fusionar ramas en Git?

Para fusionar ramas primero debemos posicionarnos (si no lo estamos) en la rama principal (“git checkout master o main”). Una vez allí ejecutamos el comando “git merge” más el nombre de la rama que queremos fusionar.

- ¿Cómo crear un commit en Git?

Para crear un commit ejecutamos el comando “git commit -m”. El mismo debe, según las buenas practicas, ir seguido de un mensaje encerrado entre comillas el cual detalla los cambios que se efectuaron.

- ¿Cómo enviar un commit a GitHub?

Para enviar un commit a GitHub lo hacemos mediante dos pasos: El primero crear el commit (mediante el comando “git commit -m + el mensaje”) para luego sí, realizar el envío propiamente dicho, el cual se realiza ejecutado el comando “git push origin” mas la rama a la que se enviarán los cambios (main, master, o la que sea).

- ¿Qué es un repositorio remoto?

Un repositorio remoto es una copia de un repositorio de Git que se almacena en un servidor remoto, como GitHub (aunque no es el servidor al que podemos acudir).

- ¿Cómo agregar un repositorio remoto a Git?

Para agregar un repositorio remoto a Git, y como ya se adelantó en el segundo punto, se ejecuta el comandos “git remote add origin” + la url.

- ¿Cómo empujar cambios a un repositorio remoto?

Para empujar cambios a un repositorio remoto ejecutamos el comando “git push”.

- ¿Cómo tirar de cambios de un repositorio remoto?

Para tirar de cambios de un repositorio remoto ejecutamos el comando “git pull”.

- ¿Qué es un fork de repositorio?

Un fork de repositorio es una copia de otro repositorio. Sólo que a diferencia de un repositorio clonado, este se encuentra en la nube (en la cuenta del usuario).

- ¿Cómo crear un fork de un repositorio?

Para crear un fork nos posicionamos en el repositorio que queremos copiar y hacemos presionamos el botón fork. Hecho esto nos cambia de interfaz, allí podemos o no colocar una descripción, revisamos que todos los datos sean correctos y posteriormente presionamos el botón de “crear fork”.

• ¿Cómo enviar una solicitud de extracción (pull request) a un repositorio?

Para enviar un solicitud de extracción nos dirigimos a la página del repositorio original (de GitHub o del que sea) y presionamos el botón “New pull request”. Seleccionamos la rama que acabamos de pushear y la rama base del repositorio original (por ejemplo main). Luego realizamos un descripción de la extracción. Y finalmente enviamos la solicitud presionando el botón “Create pull request”. Llegará una solicitud al repositorio original cuyo propietario decidirá si acepta o no los cambios.

• ¿Cómo aceptar una solicitud de extracción?

Para aceptar una solicitud de extracción (previo análisis), simplemente presionamos el botón “Merge pull request”.

• ¿Qué es un etiqueta en Git?

Una etiqueta (tag) es un marcador que se utiliza para identificar un commit específico en el historial de un repositorio. Se utilizan por ejemplo para identificar versiones, marcar hitos o crear referencias.

• ¿Cómo crear una etiqueta en Git?

Para crear una etiqueta lo hacemos mediante el comando “git tag” mas el nombre que le asignaremos.

También se puede sumar un mensaje si posterior al nombre sumamos “-m” mas el mensaje.

• ¿Cómo enviar una etiqueta a GitHub?

Para enviar una etiqueta a GitHub lo hacemos mediante el comando “git push origin” mas el nombre de la etiqueta.

• ¿Qué es un historial de Git?

Un historial de Git es un registro de todos los cambios realizados en un repositorio.

• ¿Cómo ver el historial de Git?

Para ver un historial de Git se ejecuta el comando “git log”. El cuál muestra dicho historial en orden cronológico inverso (el commit más reciente aparece primero).

• ¿Cómo buscar en el historial de Git?

Para buscar en el historial de Git se puede hacer por fecha (“git log –since="desde fecha" –until="hasta fecha"”), por mensaje específico (git log –grep="palabra_clave"), por autor (git log –author = "nombre_autor"), por archivo (git log -- <nombre_del_archivo>), entre otros.

• ¿Cómo borrar el historial de Git?

Para borrar el historial de Git lo hacemos mediante el comando “git reset –hard”. Este borra todos los commits y restaura el repositorio a un estado limpio.

• ¿Qué es un repositorio privado en GitHub?

Un repositorio privado de GitHub es uno al cual sólo podrán acceder personas autorizadas. Es decir que sólo los colaboradores podrán verlo, no es visible para el público y que requiere autenticación.

• ¿Cómo crear un repositorio privado en GitHub?

Para crear un repositorio privado lo hacemos marcando la opción: “privado” en la sección “Nuevo repositorio”.

- ¿Cómo invitar a alguien a un repositorio privado en GitHub?

Para realizar la invitación nos posicionamos primero en el repositorio privado en cuestión. Luego ingresamos a la opción de “configuración”. Allí presionamos el botón de “colaboradores” (en la barra lateral izquierda). Luego presionamos el botón de “agregar personas” (add collaborator), agregamos el usuario o correo electrónico de la persona. Posteriormente el rol que le queremos asignar al mismo (colaborador, mantenedor, etc), y finalmente volvemos a presionar el botón “agregar personas”. L apersona invitada recibirá un correo electrónico con dicha invitación.

- ¿Qué es un repositorio público en GitHub?

Un repositorio público en GitHub es uno al cual puede acceder cualquier persona que tenga un cuenta en este servidor. Es decir que cualquiera puede ver el código, descargarlo o contribuir en él.

- ¿Cómo crear un repositorio público en GitHub?

Para crear un repositorio privado lo hacemos marcando la opción: “público” en la sección “Nuevo repositorio”.

- ¿Cómo compartir un repositorio público en GitHub?

Para compartir un repositorio público nos posicionamos primero dentro del mismo. Luego apretamos el botón de búsqueda en la parte superior derecha (“Share” o “To search”), copiamos la primer URL que muestra la ventana emergente y compartimos la dirección con otras personas, correo electrónico, redes sociales, mensaje de texto, etc.

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

Punto 2 resuelto en: https://github.com/Nacho1984-26/Punto-2-de-TP

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/conflictexercise.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 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 estés 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.

Punto 3 resuelto en: https://github.com/Nacho1984-26/Ejercicio-de-conflicto