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 plataforma en línea que permite el alojamiento de repositorios.

- ¿Cómo crear un repositorio en GitHub?

  Para crear un repositorio en GitHub basta con seguir estos simples pasos:

  - Iniciar Sesión en GitHub.
  - Hacer clic en el botón (+) Create new… → New repository → Configurarlo → Hacer clic en el botón Create repository

- ¿Cómo crear una rama en Git?

  Para crear una nueva rama en Git utilizamos el comando “git Branch nombre-rama”

- ¿Cómo cambiar a una rama en Git?

  Para cambiar de una rama a otra en Git utilizamos el comando “git checkout nombre-rama”

- ¿Cómo fusionar ramas en Git?

  Para fusionar ramas en Git, primero debemos estar posicionados en la rama principal

(entendiendo por esta, la rama donde queremos copiar/traer los cambios), luego utilizamos el comando “git merge ‘nombre-rama’”

- ¿Cómo crear un commit en Git?

  Para crear un commit en Git, luego de realizar los cambios, debemos utilizar el comando “git add .”, luego con el comando “git commit -m ‘mensaje-commit’” creamos el commit.

- ¿Cómo enviar un commit a GitHub?

  Para enviar un commit a GitHub, luego de “commitear” los cambios, debemos utilizar el comando “git push -u origin main”

- ¿Qué es un repositorio remoto?

  Un repositorio remoto es una versión de mi proyecto, que se almacena en un servidor en línea o en la nube

- ¿Cómo agregar un repositorio remoto a Git?

  Para agregar un repositorio remoto a Git primeramente deberíamos inicializar git localmente con el comando “git init”.
  Luego, para agregar el repositorio remoto, utilizamos el comando “git remote add origin url-del-repositorio”

- ¿Cómo empujar cambios a un repositorio remoto?

  Para empujar los cambios al repositorio remoto, utilizamos el comando “git push -u origin nombre-rama
  Por lo general es main/master

- ¿Cómo tirar de cambios de un repositorio remoto?

  Para tirar los cambios de un repositorio remoto, utilizamos el comando “git pull origin nombre-rama
  Por lo general es main/master

- ¿Qué es un fork de repositorio?

  Un form de un repositorio es una copia independiente de un repositorio existente, que se crea en la cuenta de GitHub.

¿Cómo crear un fork de un repositorio?

Para crear un fork de un repositorio, debemos iniciar sesión en GitHub, encontrar el repositorio que queremos “forkar”, hacer clic en el botón “Fork” y esperamos que GitHub cree la copia, una vez terminado el proceso vamos a tener una copia exacta del repositorio en nuestra cuenta

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

  Para enviar una solicitud de extracción, primero tuvimos que haber forkeado un repositorio, haberlo clonado localmente, realizar cambios, subir los cambios a remoto y luego desde GitHub damos clic en “Compare & pull request”, seleccionamos la rama que contiene los cambios y la comparamos con la rama principal del repositorio original, escribimos una descripción y damos clic en el botón “Create pull request”

- ¿Cómo aceptar una solicitud de extracción?

  Para aceptar una Pull Resquest debemos revisar el contenido de diche PR, si todo esta en orden debemos hacer clic en el botón “Merge pull request”, elegimos una estrategia de integración y confirmamos la fusión, haciendo clic en el botón “Confirm merge”

- ¿Qué es una etiqueta en Git?

  Una etiqueta (tag) en Git es un marcador que se utiliza para identificar de manera especifica un punto en la historia de un repositorio.

- ¿Cómo crear una etiqueta en Git?

  Para crear una etiqueta en Git, utilizamos el comando “git tag nombre-etiqueta”. También podemos crear etiquetas anotadas, las cuales contienen metadatos, como un mensaje, fecha de creación, etc… Para ello, utilizamos el comando “git tag -a nombre-etiqueta -m ‘Descripción’ ”

- ¿Cómo enviar una etiqueta a GitHub?

  Para subir cambios incluyendo los tags, utilizamos el comando “git push --tags”

- ¿Qué es un historial de Git?

  El historial de Git es una representación de todos los cambios realizados en un repositorio a lo largo del tiempo.

- ¿Cómo ver el historial de Git?

  Para ver el historial de Git debemos utilizar el comando “git log” el mismo nos mostrara los diferentes commits de la rama actual, incluyendo hash, autor, fecha y mensaje

- ¿Cómo buscar en el historial de Git?

  Para buscar en el historial de Git tenemos diferentes maneras de hacerlo. Podríamos buscar el historial con graficas, historial de un archivo especifico, historial con filtros de tiempo, historial por palabras clave

- ¿Cómo borrar el historial de Git?

- ¿Qué es un repositorio privado en GitHub?

  Es un espacio donde se puede almacenar y gestionar el código de manera segura y restringida.
  A diferencia de los repositorios públicos, los privados están protegidos para que solo las personas que estén autorizadas puedan verlos o interactuar

- ¿Cómo crear un repositorio privado en GitHub?

  Para crear un repositorio privado en GitHub, debemos seguir estos pasos:

  - Iniciar sesión en GitHub
  - Hacer clic en el botón (+) Create new… → New repository → Configurarlo (Private) → Hacer clic en el botón Create repository

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

  Para invitar a alguien a un repositorio privado, debemos seguir estos pasos:

  - Iniciar sesión en GitHub
  - Acceder al repositorio privado
  - Dirigirse a la configuración del repositorio "Settings"
  - Acceder a la sección de colaboradores "Collaborators and teams"
  - En el campo de "Invite a collaborator" escribir el nombre del usuario de GitHub o correo electrónico, después hacer clic en el botón de "Add" o "Invite"

- ¿Qué es un repositorio público en GitHub?

  Es un espacio donde el código, archivos y documentación están accesibles para cualquier persona en la plataforma.

- ¿Cómo crear un repositorio público en GitHub?

  Para crear un repositorio público en GitHub basta con seguir estos simples pasos:

  - Iniciar Sesión en GitHub.
  - Hacer clic en el botón (+) Create new… → New repository → Configurarlo (Public) → Hacer clic en el botón Create repository

- ¿Cómo compartir un repositorio público en GitHub?

  Para compartir un repositorio público en GitHub, debemos ir al repositorio, y copiar la URL en la barra de direcciones del navegador.
  Ya que es público con el solo hecho de tener la URL de este, cualquier persona podría clonar o descargar el contenido.

2) Realizar la siguiente actividad: https://github.com/robendev/actividad-repositorio

- 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

3) Realizar la siguiente actividad: https://github.com/robendev/conflict-exercise/tree/main

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.