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 de desarrollo colaborativo basada en Git, que permite a los programadores gestionar código, colaborar en proyectos y controlar versiones de software de manera eficiente. Es ampliamente utilizado para el desarrollo de software abierto y privado.
- ¿Cómo crear un repositorio en GitHub?
- Para crear un repositorio en GitHub debemos de:
    1. Iniciar sesión en GitHub.
    2. Ve a tu perfil y selecciona “Your Repositories”.
    3. Haz clic en “New” para crear un nuevo repositorio.
    4. Ingresa un nombre para tu repositorio
    5. Selecciona la visibilidad: Público o Privado.
    6. Opcionalmente, Inicializa con el archivo README
    7. Haz clic en “Create repository”

- ¿Cómo crear una rama en Git?
- Para crear una rama en Git debemos:
    1. Abrir la terminal y navegar a tu repositorio local.
    2. Ejecuta el comando “git Branch nombre-de-la-rama” para crear una nueva rama (Remplazar “nombre-de-la-rama” con el nombre que desees).
- ¿Cómo cambiar a una rama en Git?
- Para cambiar de rama debemos:
    1. Usamos git checkout “nombre-de-la-rama”.
    2. Verificamos las ramas disponibles con “git Branch”.
    3. La rama en la que estamos estaría resaltada.
- ¿Cómo fusionar ramas en Git?
- Para fusionar ramas en git debemos:
    1. Primero debemos de cambiar a la rama principal (“main” o “master”) Usando “git checkout main”.
    2. Ahora fusionamos la rama que debeseamos integrar con el comando: git merge “nombre-de-la-rama”(Cambiando “el-nombre-de-la-rama” con la rama que queremos fusionar).
- ¿Cómo crear un commit en Git?
- Para crear un commit debemos:
    1. Asegurarnos de estar en el repositorio correcto y abrir la terminal.
    2. Añadimos los archivos que deseamos incluir en el commit con el comando: git add “nombre-del-archivo” (Si se quiere agregar todos los archivos modificados usamos: “git add .”)
    3. Creamos el commit con un nombre descriptivo: “git commit -m “Descripcion de los cambios realizados”” (Es recomendable usar mensajes claros y concisos).
    4. Verificamos que el commit se haya registrado con el comando: “git log –online” (Esto muestra en pantalla la lista de commits recientes)
    5. Subimos el commit al repositorio remoto (si es necesario): “git push origin “nombre-de-la-rama”” (Remplazamos “nombre-de-la-rama” con la rama donde se quiera subir los cambios).
- ¿Cómo enviar un commit a GitHub?
- Para enviar un commit a GitHub debemos:

1. Asegurarnos de estar en la rama correcta de la terminal con el comando: “git checkout “nombre-de-la-rama”.
2. Agregamos los cambios al área de preparación: “git add .” (Esto incluye todos los archivos modificados o también se pueden agregar archivos específicos con: “git add nombre-del-archivo”).
3. Creamos el commit con un mensaje descriptivo: “git commit -m “Descripción de los cambios realizados””.
4. Enviamos el commit al repositorio remoto: “git push origin “nombre-de-la-rama”” (Se remplaza el “nombre-de-la-rama” con el nombre de la rama que queremos enviar).

- ¿Qué es un repositorio remoto?

- Un repositorio remoto es una versión de un repositorio Git que esta alojada en un servidor, como GitHub, GitLab o bitBucket. Su propósito es permitir la colaboración, el almacenamiento seguro del código y la sincronización entre varios desarrolladores.

- ¿Cómo agregar un repositorio remoto a Git?

- Para agregar un repositorio remoto a git debemos:

1. Abrir la terminal y asegurarnos de estar en nuestro repositorio local.
2. Debemos de usar el siguiente comando para agregar el repositorio remoto: “git remote add origin https://github.com/usuario/repositorio.git (Remplazando “usuario” con el nombre de usuario en GitHub y “repositorio.git” con el nombre de nuestro repositorio).
3. Verificamos que el repositorio remoto se haya agregado correctamente con el comando: “git remote -v” (Esto nos va a mostrar las URLs del repositorio remoto).
4. Para poder subir nuestro código al repositorio remoto ingresamos el comando: “git push -u origin man” (Cambiamos “main” por el nombre de la rama que deseamos enviar).

- ¿Cómo empujar cambios a un repositorio remoto?

- Para empujar o “push” cambios a un repositorio en Git debemos:

1. Asegurarnos de estar en la rama correcta: “git checkout “nombre-de-la-rama””
2. Agregamos los archivos modificados al área de preparación: “git add .” (Este comando incluye TODOS los archivos modificados, pero se pueden especificar cuales)
3. Creamos un commit con una descipcion de los cambios: “git commit -m “Descripcion clara de los cambios””

4. Empujamos los cambios al repositorio remoto: “git push origin “nombre-de-la-rama”” (Debemos de reemplazar “nombre-de-la-rama” con la rama donde deseemos subir los cambios, como “main”)

- ¿Cómo tirar de cambios de un repositorio remoto?
- Para poder tirar los cambios de un repositorio remoto podemos usar “git pull”:

1. Asegurarnos de estar en la rama donde queremos tirar los cambios: “git checkout “nombre-de-la-rama” (Reemplazamos “nombre-de-la-rama” con la rama deseamos actualizar, como, por ejemplo: “main”).
2. Descargamos y aplicamos los cambios remotos desde el repositorio remoto: “git pull origin “nombre-de-la-rama”” (Esto nos traerá los cambios de un área remota y lo fusionará con la rama local)

- ¿Qué es un fork de repositorio?
- Un fork de repositorio es una copia del código de otro repositorio que se crea en tu propia cuenta de GitHub. Permite modificar el código sin afectar el proyecto original y es muy útil para contribuir a proyectos de código abierto. Sirve para: Experimentar con cambios sin modificar el repositorio original, Proponer mejoras y luego enviar un pull request para contribuir, Usarlo como base para tu propio proyecto.
- ¿Cómo crear un fork de un repositorio?
- Para crear un fork en un repositorio debemos:

1. Iremos al repositorio que deseamos forkear en GiHub
2. Hacemos clic en el botón “fork”
3. Seleccionamos la cuenta o una organización donde queremos poner el fork
4. GitHub creara una copia del repositorio en nuestra cuenta

- ¿Cómo enviar una solicitud de extracción (pull request) a un repositorio?
- Para poder solicitar una “pull request” debemos:

1. Asegurarnos de haber realizado cambios en nuestra rama y haberlos guardado con un commit: “git add . git commit -m “Descripción de los cambios” git push origin “nombre-de-la-rama””(Reemplazamos “nombre-de-la-rama” con la rama donde hicimos los cambios).
2. Vamos al repositorio en GitHub y entramos en la pestaña “Pull request”.
3. Hacemos clic en “New pull request”
4. Seleccionamos la rama original en el menú “base” y tu rama en los cambios en “compare”.
5. Hacemos clic en “Create pull request”.

- ¿Cómo aceptar una solicitud de extracción?

    1. Iremos al repositorio en GitHub y entramos en la pestaña “Pull request”.
    2. Seleccionamos la solicitud de extracción que deseamos revisar.
    3. Revisamos los cambios realizados en la solicitud (Se pueden ver los archivos modificados y los comentarios de otros colaboradores).
    4. Si hay conflictos, GitHbub nos notificara. Podemos resolverlos manualmente antes de fusionar el Pull request.
    5. Hacemos clic en el botón “Merge pull request”.
    6. Confirmamos la fusión seleccionanlo la opción que sea de nuestra preferencia: Create a merge commit: Mantiene el historial de cambios, Squash and merge: Combina los commits en uno solo, Rebase and merge: Reestructura el historial sin crear un commit adicional.
    7. Hacemos clic en “Confirm merge” para finalizar.

- ¿Qué es una etiqueta en Git?

- Una etiqueta(tag) en git es una referencia especifica en un punto en la historia del repositorio, generalmente utilizada para marcar versiones importantes, como lanzamientos de software. Funciona como una marca que señala un commit especifico.

- ¿Cómo crear una etiqueta en Git?

- Para crear una etiqueta en Git debemos:

    1. Asegurarnos de estar en la rama donde queremos agregar la etiqueta: “git ckeckout nombre-de-la-rama”
    2. Creamos una etiqueta usando el comando: “git tag nombre-etiqueta”

- ¿Cómo enviar una etiqueta a GitHub?

- Para poder enviar una etiqueta a GitHub:

    1. Verificamos las etiquetas creadas en nuestro repositorio: “git tag” (Esto nos muestra las etiquetas disponibles)
    2. Subimos una etiqueta especifica al repositorio remoto: “git push origin nombre-etiqueta” (Reemplazamos el nombre de la etiqueta por el nombre que queremos enviar)
    3. Para poder subir todas las etiquetas existentes debemos de usar: “git push –tags” (Esto envia todas las etiquetas locales al repositorio remoto en GitHub)

- ¿Qué es un historial de Git?

- En GitHub, el historial generalmente se refiere al registro de cambios realizados en un repositorio. A través de Git, se puede ver cada modificación realizada en los

archivos, los commits que han sido registrados, y los contribuyentes que participaron en el desarrollo.

- ¿Cómo ver el historial de Git?
- Para poder ver el historial debemos de:

    1. Abrir el repositorio en GitHub.
    2. Vamos a la pestaña “Commits” para ver una lista de los cambios realizados en la rama principal “main”
    3. Podemos inspeccionar cada commit para ver que archivos fueron modificados y quien los hizo.
    4. En la pestaña “History” dentro de cada archivo, puedes ver su evolución a lo largo del tiempo.

- ¿Cómo buscar en el historial de Git?
- Para buscar en el historial de Git debemos:

    1. Debemos de ver todos los commits en la historia del repositorio: “git log” (Esto demuestra todos los commits con detalles como autor, fecha y mensaje).
    2. Para buscar en commits debemos usar palabras claves: “git log –grep=”texto a buscar” (Remplazamos “texto a buscar” con la/las palabra clave que deseamos encontrar)
    3. Filtrando los commits por autor: “git log --author=”Nombre del autor”
    4. Viendo los cambios realizados en un archivo especifico: “git log – nombre-del-archivo
    5. Mostrando el historial compacto (Solo IDs de commits y mensajes breves): ”git log –online”
    6. Buscando en los cambios dentro de los archivos (como una palabra en el código): “git log -s “texto a buscar””

- ¿Cómo borrar el historial de Git?
- Para borrar todo el historial y comenzar de cero debemos introducir los siguientes comandos:

    Rm -rf .git

    Git init

    Git add .

    Git commit -m “Nuevo inicio sin historial”

- ¿Qué es un repositorio privado en GitHub?

- Un repositorio privado en GitHub es un repositorio que solo nosotros y los colaboradores autorizados pueden ver y modificar. A diferencia de los repositorios públicos, no esta disponible para otros usuarios a menos que se les de acceso.

- ¿Cómo crear un repositorio privado en GitHub?

    1. Entramos a GitHub e iniciamos sesión
    2. Haz clic en “New repository”
    3. Ingresamos un nombre para el repositorio.
    4. Seleccionamos la opción “Private” en visibilidad
    5. Creamos el repositorio y configuramos los permisos según la necesidad

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

- Para invitar a alguien a nuestro repositorio privado debemos de:

    1. Abrimos nuestro repositorio privado en GitHub.
    2. Vamos a la pestaña “Settings”.
    3. En el menú lateral, seleccionamos “collaborators” en la sección “Access”
    4. Hacemos clic en “Add people”.
    5. Introducimos el nombre de usuario o correo de la persona a la cual queremos invitar.
    6. Seleccionamos el nivel de acceso
        - Read: Solo puede ver el repositorio
        - Write: Puede realizar cambios y enviar commits.
        - Admin: Tiene permisos de administración total.
    7. Hacemos clic en “Invite” para enviar la invitacion

- ¿Qué es un repositorio público en GitHub?

- Un repositorio publico en GitHub es un repositorio que cualquier persona puede ver, clonar, y contribuir si tiene los permisos adecuados. Es ideal para compartir proyectos, colaborar con otros desarrolladores y poder trabajar en un código abierto

- ¿Cómo crear un repositorio público en GitHub?

- Para crear un repositorio publico debemos de:

    1. Iniciar sesión en GitHub.
    2. Hacemos clic en “New repository”.
    3. Elegimos un nombre para nuestro repositorio.
    4. Seleccionamos la opción “Public”.
    5. Opcionalmente, inicializarlo con un archivo README
    6. Hacemos clic en “Create repository”.

- ¿Cómo compartir un repositorio público en GitHub?

- Para compartir un repositorio en GitHub debemos:

    1. Asegurarnos de que nuestro repositorio sea publico en la configuración de GitHub.
    2. Copiamos la URL del repositorio desde la barra de direcciones o usando: “git remote get-url origin”
    3. Compartimos el enlace con otros usuarios mediante mensajes, correos o redes sociales.
    4. Si queremos que otros contribuyan, podemos habilitar las opciones de “issues” y “Pull requests” en la configuración del repositorio.
    5. Tambien podemos permitir que otros clonen usando: “git clone https://github.com/usuario/repositorio.git” (Reemplazamos “usuario/repositorio.git” con el nombre real del repositorio en Github).

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

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.