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?

Es una plataforma donde se puede alojar y gestionar proyectos basados en GIT, ya sean personales o grupales dadas sus características colaborativas.

- ¿Cómo crear un repositorio en GitHub?

Dentro de nuestra cuenta de GitHub desplegar el menú **+** (esquina superior derecha) y seleccionar **New repository**, darle un nombre, una descripción (opcional) y seleccionar la configuración de la visualización. Se pueden agregar adicionales, por ejemplo un archivo README para describir en forma más extensa el proyecto.

- ¿Cómo crear una rama en Git?

Para crear una rama hay que usar el comando git branch nombre-de-la-rama (o el que uno decida asignarle). Una aclaración importante es evitar espacios o caracteres especiales en los nombres, lo mismo con nombres reservados como main o master.

- ¿Cómo cambiar a una rama en Git?

Siguiendo el punto anterior cuando creamos una rama no nos lleva automáticamente a la misma, por eso para cambiar hay que usar el comando git checkout nombre-de-la-rama. En versiones más recientes también puede usarse git switch nombre-de-la-rama.

- ¿Cómo fusionar ramas en Git?

A través del comando *merge*. Supongamos que tenemos una rama principal (main por defecto) y queremos fusionar la del ejemplo anterior, primero debemos posicionarnos en la que queremos fusionar los cambios a través de *git checkout main.*
Luego fusionar a través de:
git merge nombre-de-la-rama

- ¿Cómo crear un commit en Git?

Una vez que estamos seguros de estar en la rama correcta donde queremos crear el commit hay que añadir el o los archivos modificados.
Si es uno en específico:
git add nombre-del-archivo
Si son todos:
git add .
Finalmente hacer el commit:
git commit -m “Se agregó la cifra faltante en la lista de gastos semanales de la casa”.
Indicando la -m que hay un mensaje específico del commit, el cual a su vez debe estar entre comillas y describir lo que se haya cambiado como referencia (de ahí la comparación al punto de guardado de un videojuego).

- ¿Cómo enviar un commit a GitHub?

Supongamos que el commit anterior lo hicimos en el branch *nombre-de-la-rama*, para que no sea tan confuso, sólo queda ejecutar el comando git push origin nombre-de-la-rama; donde origin es el nombre por defecto del repositorio remoto del proyecto.

- ¿Qué es un repositorio remoto?

Es una versión de nuestro repositorio que está almacenada en un servidor, para esta segunda instancia sería GitHub, y que tiene dos ventajas: permitir que otros colaboren en el proyecto y hacer copias de seguridad.

- ¿Cómo agregar un repositorio remoto a Git?

Primero hay que copiar la URL del repositorio, la misma se encuentra en el botón verde **<> Code** (arriba a la derecha).

Luego en la terminal de GitBash ir hasta el directorio donde se quiera clonar el repositorio y ejecutar:
git clone https://github.com/MaluLopez/Programacion1.git
A modo de ejemplo usé uno que armé para prueba.

- ¿Cómo empujar cambios a un repositorio remoto?

A través de git push origin nombre-de-la-rama, la rama indicada es en la que se hizo el cambio y el push indica justamente el empuje.

- ¿Cómo tirar de cambios de un repositorio remoto?

A través de git pull origin nombre-de-la-rama, la rama indicada es la que se quiere actualizar desde el remoto y pull marca la acción de tirar.
A veces puede presentarse un conflicto por modificaciones realizadas tanto en el repositorio remoto como en el local y que no fueron actualizadas a tiempo. Por éste motivo se recomienda hacer *git pull* con frecuencia para que los repositorios se mantengan sincronizados y tener diálogo constante con el equipo de trabajo.

- ¿Qué es un fork de repositorio?

Es una copia de un repositorio que permite hacer cambios en el proyecto sin afectar al repositorio original, una ventaja sobre todo para los trabajos colaborativos ya que se pueden hacer modificaciones de forma independiente.

- ¿Cómo crear un fork de un repositorio?

Una vez abierto el repositorio elegido clickear en la opción **Fork** (arriba a la derecha), esto creará una copia en nuestra cuenta. Hecho eso hay que clonarlo como se vio con anterioridad y de esa manera también tendremos una copia en nuestro repositorio local para poder hacer actualizaciones a futuro.

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

Ir al *fork* hecho en GitHub, clickear en la pestaña **Pull Requests**, luego en **New Pull Requests**. Chequear que GitHub esté preguntando por las ramas correspondientes de ambos repositorios, el forkeado y el original. Revisar los cambios agregando un título y descripción y finalmente seleccionar **Create Pull Request**.

- ¿Cómo aceptar una solicitud de extracción?

Una vez situados en la pestaña **Pull Requests** seleccionar la solicitud que se quiera revisar. Los cambios propuestos podrán verse en la pestaña **Files Changed**. A partir de acá pueden pasar dos cosas:
Que se necesiten modificaciones: en ese caso se pueden dejar comentarios en líneas específicas del código y luego solicitar los cambios a través de la opción **Request changes** en la pestaña **Review changes**.
Que esté correcto: ir a la parte inferior del *Pull Request* y clickear en el botón **Merge pull request**.
También da la opción de elegir entre tres tipos de fusión, siendo **Create a merge commit** la que queda por defecto y que mantiene el historial completo. Si se considera necesario se puede agregar un mensaje y finalmente se confirma a través de **Confirm merge**.

- ¿Qué es un etiqueta en Git?

Es un referencia inmutable a un commit específico. Se usa por lo general para marcar aspectos importantes de un proyecto, por ejemplo versiones del software o identificación de puntos estables en el desarrollo.
Se clasifican en dos tipos:
Annotated Tags (etiqueta anotada): se almacenan en la base de datos como objetos completos, es decir que Git las reconoce como objetos independientes con información adicional (nombre del autor, fecha, etc)
Se recomienda para versiones oficiales del software.
Lightweight Tag (etiqueta ligera): no tiene datos adicionales y se usa para referencias temporales que no comprometen el historial del proyecto.

- ¿Cómo crear una etiqueta en Git?

Para Lightweight tag:
git tag nombre-de-la-etiqueta

Para Annotated tag:
git tag -a lologramos -m “Finalización del módulo de chequeo de datos”
Donde -a indica que es una etiqueta anotada, lologramos un nombre cualquiera para representar un acontecimiento importante, -m que hay un mensaje y entre comillas el propósito de la etiqueta.

- ¿Cómo enviar una etiqueta a GitHub?

Para subir todas las etiquetas que se crearon en el repositorio local ejecutar:
git push –tags
Para una etiqueta específica, supongamos que se llame casita, ejecutar:
git push origin casita

- ¿Qué es un historial de Git?

Es el registro completo de todos los cambios realizados en un repositorio, incluye información sobre cada commit hecho. Esto permite saber cómo evolucionó el código a lo largo del proyecto, quiénes hicieron cada cambio y cuándo.

- ¿Cómo ver el historial de Git?

Con el comando git log, que permite ver una lista de los commits realizados en el repositorio con detalles como el mensaje, la fecha y la hora, y demás datos de importancia.

- ¿Cómo buscar en el historial de Git?

Con el comando git log, al mismo se le puede agregar *--grep* para buscar textos o palabras, *--author* para autor o *--since/--until* para fechas, entre otros, para restringir la búsqueda.

- ¿Cómo borrar el historial de Git?

Existen varias opciones pero siempre giran alrededor del comando git reset.
git reset quita todos los archivos y carpetas del proyecto
git reset nombre-archivo quita el archivo indicado
git reset nombre-carpeta quita todos los archivos de esa carpeta
git reset nombre-carpeta/nombre-archivo quita ese archivo que está a su vez dentro de esa carpeta
git reset nombre-carpeta/*.extensión quita todos los archivos que cumplen con la condición indicada previamente dentro de esa carpeta
Siempre hay que tener presente que estos cambios se ejecutan en el repositorio local y que si se trabaja en equipo y hay que pushearlos es conveniente hablarlo previamente.

- ¿Qué es un repositorio privado en GitHub?

Es un repositorio cuyo acceso está restringido. Solo los usuarios con permiso, como el propietario o colaboradores invitados, pueden acceder. Las ventajas son el control del acceso (incluidas las modificaciones) y seguridad porque el proyecto no queda visible para cualquiera.

- ¿Cómo crear un repositorio privado en GitHub?

Son los mismos pasos que la creación de un repositorio, detallado con anterioridad, pero con la diferencia de que en el apartado de tipo de visibilidad hay que seleccionar **Private**.

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

Hecho el acceso al repositorio, en calidad de propietario o administrador, ir a **Settings** en la parte

superior derecha. En el menú que aparece a la izquierda seleccionar **Collaborators**, luego **Add people** y escribir el nombre de usuario o nombre completo o mail. Seleccionar en la lista el usuario buscado y clickear en **Add [usuario] to this repository**. A ese invitado a su vez se le puede asignar un nivel de acceso de acuerdo a su rol: *Read* sólo para ver el código, *Write* para hacer cambios y crear ramas o *Admin* para gestionar configuraciones y colaboradores. Finalizado todos estos pasos el colaborador recibirá una invitación y deberá aceptarla.

- ¿Qué es un repositorio público en GitHub?

Es un repositorio que puede ser visto y clonado por cualquier persona, además de ser gratuitos en su totalidad. El único punto que sigue teniendo en común con uno privado es que se requiere permiso para modificarlo.

- ¿Cómo crear un repositorio público en GitHub?

Son los mismos pasos detallados con antelación, pero con la diferencia de que en el apartado de tipo de visibilidad hay que seleccionar **Public**.

- ¿Cómo compartir un repositorio público en GitHub?

Como es de acceso ilimitado basta con copiar la URL y compartirla en el medio elegido. Si se quiere invitar a un colaborador, para que pueda ser intervenido, son los mismos pasos aclarados en la invitación para repositorio privado. La otra opción es a través de un *Fork*, de manera tal que el usuario que lo realice tenga una copia en su cuenta.

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

URL ejercicio realizado: https://github.com/MaluLopez/Ejercicio2-TP2-Programacion.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

URL ejercicio realizado: https://github.com/MaluLopez/conflict-exercise.git

Alumna: Lopez María Laura

Comisión 16

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.