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) :

docu oficial: https://docs.github.com/es/get-started/start-your-journey/about-github-and-git

- ¿Qué es GitHub?

Rta: GitHub es una plataforma en línea para almacenar y compartir código. Funciona como una red social para desarrolladores y está basada en **Git**, un sistema de control de versiones. En GitHub se puede:

- Guardar tus proyectos en la nube.
- Colaborar con otras personas en el mismo proyecto.
- Llevar un historial de cambios.
- Crear ramas, hacer revisiones, y más.

—-----—-----—-----—-----—-----—-----—-----—-----—-----—-----—-----—-----—-----—-----—---------------

- ¿Cómo crear un repositorio en GitHub?

Rta: Ir a https://github.com e iniciar sesión.

Hacer clic en el botón verde que dice **"New"** o **"New repository"** (generalmente en la parte superior derecha).

Llenar los campos:

- **Repository name**: nombre del proyecto.
- **Description** (opcional): una breve descripción.

- Elegir si se quiere que sea **público** o **privado**.

Se puede marcar la opción "Initialize this repository with a README" si se quiere que se cree un archivo README automáticamente.

Hacer clic en **"Create repository"**.

- ¿Cómo crear una rama en Git?

Rta: En la terminal, dentro de tu proyecto, usar:

git branch nombre-de-la-rama

- ¿Cómo cambiar a una rama en Git?

Rta: Usar el siguiente comando para cambiar de rama:

git checkout nombre-de-la-rama

O, en versiones modernas de Git, se puede usar:

git switch nombre-de-la-rama

- ¿Cómo fusionar ramas en Git?

Rta: Primero, cambiar a la rama en la que quieres aplicar los cambios (normalmente main o master):

git checkout main

Luego, fusionar la otra rama:

git merge nombre-de-la-rama

- **¿Cómo crear un commit en Git?**

Rta: Para crear un commit en Git, primero hay que tener un repositorio de Git iniciado y haber realizado algunos cambios en los archivos del proyecto. Una vez que los archivos del repositorio están preparados, ingresar el comando: **git commit -m "Mensaje descriptivo del commit"**

- **¿Cómo enviar un commit a GitHub?**

Para enviar un commit a GitHub, hay que asegurarse de tener el repositorio local vinculado a un repositorio remoto en GitHub, luego realizar un **commit** y finalmente enviar ese commit al repositorio remoto utilizando el comando **git push**.

- **¿Qué es un repositorio remoto?**

Rta: Un repositorio remoto es una versión del repositorio que está alojada en un servidor o en una plataforma en línea, como GitHub, GitLab, Bitbucket, etc. Este repositorio está disponible para ser accedido desde cualquier lugar, y suele ser utilizado para compartir código con otros usuarios o para tener una copia de respaldo de tu trabajo.

- **¿Cómo agregar un repositorio remoto a Git?**

Rta: Para agregar un repositorio remoto a Git, primero hay que tener creado un repositorio local y recién ahí, abrir uno remoto de la siguiente manera:

1. Iniciar Sesión en GitHub
2. Seleccionar “+” y luego “New repository”.
3. Ponerle un nombre al repositorio remoto y aceptar.

Una vez creado el repositorio remoto, se debe copiar la siguiente línea de comando que figura en él en la terminal de Git:

git remote add origin https://github.com/nombre-de-usuario/nombre-de-repositorio.git

Este comando agrega la dirección del repositorio remoto para vincularlo con el local.

- **¿Cómo empujar cambios a un repositorio remoto?**

Rta: Para empujar los cambios a un repositorio remoto se debe copiar el siguiente comando en Git desde GitHub: **git push -u origin main** cambiando “main” por “master” (ya que ésta es la rama/branch que crea Git por defecto)

- **¿Cómo tirar de cambios de un repositorio remoto?**

Rta: Para lograr esto debemos colocar, Git pull para poder descargar cambios y Git merge para poder fusionarlo con tu rama local (git pull origin < nombre_de_la_rama >).
En el caso de trabajar con una rama específica, como main, deberá colocarse de la siguiente forma: (git pull origin main).

- **¿Qué es un fork de repositorio?**

Rta: Cuando nos encontramos en Github muchas veces nos puede parecer interesante algún repositorio, pero no es de nuestro usuario, para esto Github tiene un botón llamado Fork el cual se encuentra en la parte superior derecha. Al hacer clic allí podremos generar una copia exacta en nuestro usuario de aquel repositorio que nos interesó, también podremos modificarlo a nuestro gusto sin problema, ya que no afectará al repositorio original, sino que irá cualquier modificación a nuestra copia local. Finalmente, vamos a copiar ese repositorio que se encuentra en nuestro canal para poder moverlo a una carpeta que elijamos y continuar trabajando desde nuestro

repositorio local.

—------—------—------—------—------—------—------—------—------—------—------—------—------—------—----------

- **¿Cómo crear un fork de un repositorio**?

Rta: Para crear un fork de un repositorio, debes ingresar a Github y encontrar aquel usuario que te interese su repositorio . Una vez allí, puedes hacer un Fork del repositorio (haciendo clic en el extremo derecho). Luego lo clonaremos en una carpeta y, si hay un archivo que creemos que puede ser mejorado, lo modificamos. Luego lo agregamos al stage y hacemos un commit y finalmente enviamos los cambios al fork que creamos utilizando **git push origin master**. Finalmente si deseamos que el autor del repositorio original vea la modificación que hicimos. Debemos avisarle, ya que los cambios que realizamos no se reflejarán en el repositorio original, se dirigen a nuestra copia local.

—------—------—------—------—------—------—------—------—------—------—------—------—------—------—----------

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

Rta: Para realizar la solicitud de Pull request, nos iremos a la pestaña de Solicitudes de pull request donde haremos clic en new pull request. Aparecerá una ventana que mostrará un resumen de las modificaciones que hemos realizado en comparación con el código original. Haremos clic en Crear solicitud de extracción, donde veremos el título, allí agregaremos un mensaje general y un poco más abajo hay suficiente espacio para explicar las razones por las cuales consideramos que el cambio que hemos hecho sería beneficioso para agregar al repositorio original.

—------—------—------—------—------—------—------—------—------—------—------—------—------—------—----------

- **¿Cómo aceptar una solicitud de extracción?**

Rta: El creador del repositorio podrá visualizar en sus solicitudes de extracción el aviso que le hemos enviado, lo que le permitirá revisarlo y, si lo considera adecuado, efectuar la modificación correspondiente, además de tener la opción de responder al usuario que sugirió dicho cambio. La ventaja de este proceso es que si el usuario original determina que esta alteración es beneficiosa y no ocasiona conflictos con la rama principal de su repositorio local remoto, podrá hacer clic en Merge pull request, integrando así en su repositorio las modificaciones realizadas por un usuario.

—------—------—------—------—------—------—------—------—------—------—------—----------

Bruno (Azul):

- **¿Qué es una etiqueta en Git?**

Rta: Como muchos VCS, Git tiene la posibilidad de etiquetar puntos específicos del historial como importantes. Esta funcionalidad se usa típicamente para marcar versiones de lanzamiento (v1.0, por ejemplo).

—------—------—------—------—------—------—------—------—------—------—------—------—------—------—----------

- **¿Cómo crear una etiqueta en Git?**

Rta: Git utiliza dos tipos principales de etiquetas: ligeras y anotadas. Una etiqueta ligera es muy parecida a una rama que no cambia - simplemente es un puntero a un commit específico. Sin embargo, las etiquetas anotadas se guardan en la base de datos de Git como objetos enteros. Tienen un checksum; contienen el nombre del etiquetador, correo electrónico y fecha; tienen un mensaje asociado; y pueden ser firmadas y verificadas con GNU Privacy Guard (GPG). Normalmente se recomienda que crees etiquetas anotadas, de manera que tengas toda esta información; pero si quieres una etiqueta temporal o por alguna razón no estás interesado en esa información, entonces puedes usar las etiquetas ligeras.

—-----—-----—-----—-----—-----—-----—-----—-----—-----—-----—-----—-----—-----—-----—-----—---------------

- **¿Cómo enviar una etiqueta a GitHub?**

Rta: Primero, debes crear una etiqueta en tu repositorio local. Puedes crear una etiqueta anotada (que incluye información adicional como el autor
Podes verificar las etiquetas que has creado localmente con:
git tag
Una vez que has creado la etiqueta en tu repositorio local, necesitas empujarla al repositorio remoto en GitHub. Puedes hacer esto con el siguiente comando: git push origin v1.0 (origin es el nombre del repositorio remoto (por defecto suele ser origin) y v1.0 es el nombre de la etiqueta.)
Para empujar todas las etiquetas creadas, usar: git push origin --tags

—-----—-----—-----—-----—-----—-----—-----—-----—-----—-----—-----—-----—-----—-----—-----—---------------

- **¿Qué es un historial de Git?**

Rta: El historial de Git es una secuencia de todos los cambios realizados en un repositorio de Git. Cada cambio en el repositorio se guarda como un commit, y cada commit contiene información sobre el estado del proyecto en un momento específico, incluyendo: Identificador del commit Autor Fecha de realización Mensaje enviado.

—-----—-----—-----—-----—-----—-----—-----—-----—-----—-----—-----—-----—-----—-----—-----—---------------

- **¿Cómo ver el historial de Git?**

Rta: Esto lo conseguimos con el comando de Git: git log Con tipear este comando en el bash de Git podremos apreciar el histórico de commits, estando situados en la carpeta de nuestro proyecto. El listado de commits estará invertido, es decir, los últimos realizados aparecen los primeros. El comando git log --oneline es una forma compacta de visualizar el historial de commits en un repositorio Git. Muestra un resumen conciso de los commits recientes, con cada commit representado en una sola línea. Si tu proyecto ya tiene muchos commits, quizás no quieras mostrarlos todos, ya que generalmente no querrás ver cosas tan antiguas como el origen del repositorio. Para ver un número de logs determinado introducimos ese número como opción, con el signo "-" delante (-1, -8, -12...). Por ejemplo, esto muestra los últimos tres commits: git log -3 Si queremos que el log también nos muestre los cambios en el código de cada commit podemos usar la opción -p. Esta opción genera una salida mucho más larga, por lo que seguramente nos tocará movernos en la salida con los cursores y usaremos CTRL + Z para salir. git log -2 -p

—-----—-----—-----—-----—-----—-----—-----—-----—-----—-----—-----—-----—-----—-----—-----—---------------

- **¿Cómo buscar en el historial de Git?**

Rta: Para buscar en el historial de commits de Git, puedes utilizar varios comandos y opciones que

te permiten filtrar y localizar commits específicos. Para buscar commits que contengan una palabra o frase específica en el mensaje de commit, usa **git log** con la opción **–grep**: **git log --grep="palabra clave".**
Para buscar commits que han modificado un archivo específico, usar **git log** seguido del nombre del archivo: **git log -- nombre_del_archivo.**
Para buscar commits en un rango de fechas específico, usar las opciones **--since** y **--until**: **git log --since="2024-01-01" --until="2024-01-31".**
Para encontrar commits hechos por un autor específico, usar **--author**: **git log --author="Nombre del Autor".**

—-----—-----—-----—-----—-----—-----—-----—-----—-----—-----—-----—-----—-----—-----—---------------

- **¿Cómo borrar el historial de Git?**

Rta: El comando **git reset** se utiliza sobre todo para deshacer las cosas. Se mueve alrededor del puntero HEAD y opcionalmente se cambia el index o área de ensayo y también se puede cambiar opcionalmente el directorio de trabajo si se utiliza **- hard**. Esta última opción hace posible que este comando pueda perder tu trabajo si se usa incorrectamente, por lo que hay que asegurarse de entenderlo antes de usarlo. Existen distintas formas de utilizarlo:
• **git reset** -> Quita del stage todos los archivos y carpetas del proyecto.
• **git reset nombreArchivo** -> Quita del stage el archivo indicado.
• **git reset nombreCarpeta/** -> Quita del stage todos los archivos de esa carpeta.
• **git reset nombreCarpeta/nombreArchivo** -> Quita ese archivo del stage (que a la vez está dentro de una carpeta).
• **git reset nombreCarpeta/*.extensión** -> Quita todos los archivos que cumplan con la condición indicada previamente dentro de esa carpeta del stage.

—-----—-----—-----—-----—-----—-----—-----—-----—-----—-----—-----—-----—-----—-----—---------------

- **¿Qué es un repositorio privado en GitHub?**

Rta: Un repositorio privado en GitHub es un tipo de repositorio en el que el contenido sólo es accesible para usuarios específicos que han sido autorizados. A diferencia de los repositorios públicos, donde cualquier persona puede ver y clonar el contenido, un repositorio privado limita el acceso a los colaboradores que tú elijas. Esto es útil para proyectos que contienen información sensible o que aún están en desarrollo y no deseas que estén disponibles públicamente.

—-----—-----—-----—-----—-----—-----—-----—-----—-----—-----—-----—-----—-----—-----—---------------

- **¿Cómo crear un repositorio privado en GitHub?**

Rta: Pasos:

1. Iniciar sesión en GitHub
2. Ingresar a la página de creación de repositorios: En la esquina superior derecha de la página principal, hacer clic en el botón “+” y seleccionar “New Repository” o hacer clic en “New”:
3. Completar la información del repositorio (nombre del repositorio, descripción)
4. Seleccionar la configuración de privacidad: Esto asegura que el repositorio será privado y solo accesible para los colaboradores que tu elijas.

—-----—-----—-----—-----—-----—-----—-----—-----—-----—-----—-----—-----—-----—-----—---------------

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

Rta:

1. Acceder al repositorio
2. Hacer clic en la pestaña "Settings" del repositorio (está en la parte superior del repositorio, junto a las pestañas como "Code" y "Issues")
3. Seleccionar "Collaborators" en el menú de la izquierda. Esto te llevará a la página donde se puede administrar colaboradores.
4. En la sección "Collaborators", hacer clic en el botón "Add people" e ingresar el nombre de usuario de GitHub de la persona que se desee invitar.
5. Selecciona el nivel de acceso que se desea otorgar: Read, Triage, Write, Maintain, o Admin.
6. Hacer clic en el botón "Add" para enviar la invitación.

- **¿Qué es un repositorio público en GitHub?**

Rta: Un repositorio público en GitHub es un repositorio cuyo contenido es accesible a cualquier persona en Internet. A diferencia de un repositorio privado, que está restringido a un grupo específico de colaboradores, un repositorio público permite que cualquier persona pueda ver, clonar y, si tienen los permisos adecuados, contribuir al proyecto.

- **¿Cómo crear un repositorio público en GitHub?**

Rta: Pasos:

1. Iniciar sesión en GitHub.
2. Ingresar a la página de creación de repositorios: En la esquina superior derecha de la página principal, hacer clic en el botón “+” y seleccionar “New Repository” o hacer clic en “New”.
3. Completar la información del repositorio (nombre del repositorio, descripción)
4. Seleccionar la configuración de privacidad: Esto asegura que el repositorio será público.

- **¿Cómo compartir un repositorio público en GitHub?**

Rta: La forma más sencilla de compartir tu repositorio es proporcionar el enlace directo al mismo. Accede a tu repositorio, copia la URL de tu repositorio que se encuentra en un cuadro de texto que dice "**<> Code**”:

Se puede copiar la URL directamente haciendo clic en el botón de copiar a la derecha de la URL.

---

2) Realizar la siguiente actividad:

- Crear un repositorio.
    - Dale un nombre al repositorio.
    - Elige que 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 una Branch
  - 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.