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.

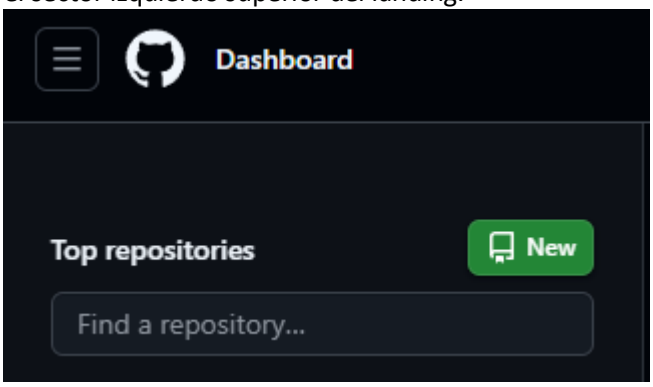
Actividades

1) Contestar las siguientes preguntas utilizando las guías y documentación proporcionada (Desarrollar las respuestas) :

- ¿Qué es GitHub?
  GitHub es una plataforma que permite a desarrolladores compartir su código en la nube, crear copias o clonar repositorios para proyectos propios.

- ¿Cómo crear un repositorio en GitHub?

  Para crear un repositorio en GitHub, primero se debe crear una cuenta en la plataforma. Una vez creada la cuenta, se debe ingresar en la sección **NEW**, ubicada en el sector izquierdo superior del landing.



Una vez en la sección https://github.com/new se podrá editar el nombre del nuevo repositorio, la privacidad del mismo, darle una descripción si así se lo desea, una licencia, entre otras configuraciones.

- ¿Cómo crear una rama en Git?

  Para crear una rama en Git, se debe de ingresar a la terminal desde el editor de código, e introducir el comando $git branch nuevaRama1

- ¿Cómo cambiar a una rama en Git?

  Para cambiar de rama dentro de un proyecto, se debe de ingresar el comando $git checkout main

  En este ejemplo estaríamos cambiando desde la rama nuevaRama1, recientemente creada, a la rama Main del proyecto.

- ¿Cómo fusionar ramas en Git?

  El comando para fusionar ramas en Git es el comando $git merge

  Para utilizarlo correctamente debemos estar ya posicionados en la rama a la cual queremos combinar los cambios o las modificaciones realizadas. Por ejemplo:

  Nos posicionamos en la rama Main ingresando el comando $git checkout main

  Luego ya en la rama main, ingresamos el comando $git merge nuevaRama1 para combinar ambas líneas de trabajo en una sola, la rama main.

- ¿Cómo crear un commit en Git?

  En primera instancia se deben de agregar todos los cambios realizados en el proyecto mediante el comando $git add .

  Una vez hecho esto, se crea un commit con el comando $git commit

- ¿Cómo enviar un commit a GitHub?

  Para ello debemos tener vinculada nuestra cuenta de GitHub a nuestra pc

  Trabajando en el proyecto, agregamos los cambios realizados mediante $git add .

  Luego realizamos el commit como vimos anteriormente con el comando $git commit –m “y dejamos un mensaje”

  Para enviar todos estos cambios a nuestro repositorio en gitHub se debe de ingresar el siguiente comando

  $git push origin master

  Donde origin es el nombre remoto y master es el nombre de la rama.

- ¿Qué es un repositorio remoto?

  Un repositorio remoto es un repositorio alojado en la nube, puede estarlo en GitHub o en alguna otra plataforma de control de versiones como Mercurial o Bitbucket.
  Este repositorio puede ser colaborativo y en el pueden estar trabajando muchas personas de manera remota.

- ¿Cómo agregar un repositorio remoto a Git?

  Para agregar un repositorio remoto a Git se debe de introducir desde el terminal el comando $git remote add [especificar el nombre]

- ¿Cómo empujar cambios a un repositorio remoto?

  Para empujar los cambios a un repositorio remoto, utilizamos el comando Push

  Ejemplificado quedaría: $git push origin [nombre de la rama]

- ¿Cómo tirar de cambios de un repositorio remoto?

  Para tirar los cambios de un repositorio remoto, utilizamos el comando Pull

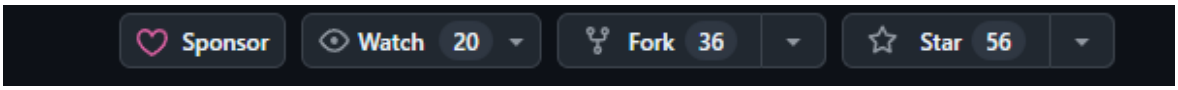
  $git pull origin [nombre de la rama]

- ¿Qué es un fork de repositorio?

  Un fork de repositorio en GitHub es una copia de un repositorio que se realiza de forma independiente a nuestro perfil, para poder manipularlo de forma aislada y con el, realizar un nuevo proyecto.

- ¿Cómo crear un fork de un repositorio?

  Para crear un Fork de repositorio debemos ingresar al proyecto deseado en la web de GitHub, y desde allí, en el sector superior derecho, buscar por el icono y la palabra Fork

  Una vez clickeando en Fork, se le podrán hacer algunas modificaciones al repositorio como el nombre, y la descripción.

- ¿Cómo enviar una solicitud de extracción (pull request) a un repositorio?
  Para hacer una pull request, se debe de ir a la pagina del repositorio donde quiera hacerse y hacer click en “New pull request”, una vez allí, se debe elegir la rama a la cual se quiera hacer la pull request. Luego se deben completar algunos datos como descripción y la sugerencia o modificaciones que querramos se implementen en el código.

- ¿Cómo aceptar una solicitud de extracción?

  El autor del repositorio tendrá la potestad de revisar las solicitudes que otros usuarios le sugieren, estas saldrán en la pestaña de Pull Request del repositorio, y allí podrá decidir si los cambios que le fueron sugeridos se tendrán en cuenta para la actualización del proyecto y así hacer merge de las propuestas con la rama main.

- ¿Qué es un etiqueta en Git?

  Las etiquetas en Git se asocian con las confirmaciones, por lo que se pueden usar para marcar un punto específico en el historial del repositorio. Por ejemplo, el número de versión de una publicación.

- ¿Cómo crear una etiqueta en Git?

  Para crear una etiqueta en un proyecto, utilizamos la etiqueta $git tag –a “versión” –m “mensaje”

  Donde –a se utiliza para especificar el número de versión del proyecto.

  Y –m es para dejar una anotación adicional al desarrollador que vea esta etiqueta.

- ¿Cómo enviar una etiqueta a GitHub?

  Para enviar una etiqueta desde Git hacia GitHub, debemos ingresar en la terminal el comando push origin [nombre de la etiqueta]

- ¿Qué es un historial de Git?

  El historial de git, accediendo mediante el comango $git log es un historial donde se muestran todos los cambios que se han realizado a lo largo del proyecto. En él se muestran principalmente las diferentes ramas que se han creado, su versión, el número de hash, los commits realizados y sus comentarios o mensajes.

- ¿Cómo ver el historial de Git?

  Para acceder al historial de registros de Git, lo hacemos mediante el comando $git log

- ¿Cómo buscar en el historial de Git?

  Al comando anterior, para buscar en Git podemos utilizar variantes como:

  git log --author="Nombre del Autor": Filtra los commits por autor.

  git log --since="hace una semana": Filtra los commits realizados en la última semana.

  git log --until="ayer": Filtra los commits realizados hasta ayer.

- ¿Cómo borrar el historial de Git?

  Existe en Git un comando que es capaz de eliminar una serie de commits realizados, especificando cuantos de ellos quieren ser borrados. Este comando es el

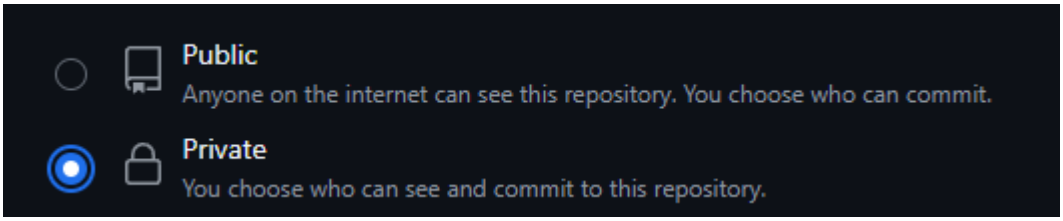
  $git rebase -i HEAD~N (Donde # N es el número de commits hacia atrás)

- ¿Qué es un repositorio privado en GitHub?

  Un repositorio privado en GitHub es un repositorio al que solo tendrá acceso su creador en primera instancia, pero que se le podrá brindar acceso a otros usuarios o desarrolladores mediante permisos exclusivos.

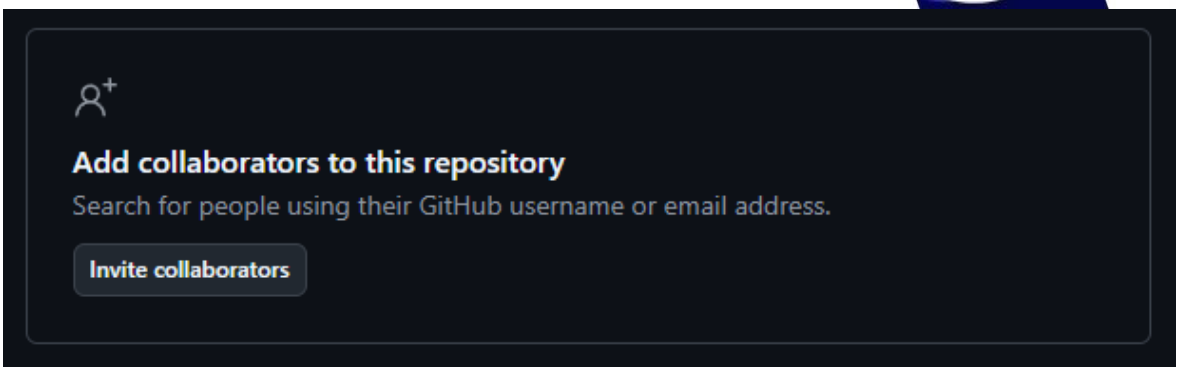
- ¿Cómo crear un repositorio privado en GitHub?

  Al igual que para crear un repositorio publico, y siguiendo los mismos pasos que en el primer punto de esta serie de preguntas, basta con cambiar en las opciones de privacidad el por defecto ‘Publico’ a ‘Privado’ como se muestra a continuación



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

  Una vez creado el repositorio y tildando la opción de repositorio privado. La siguiente landing nos dara la opción de invitar colaboradores a nuestro proyecto.

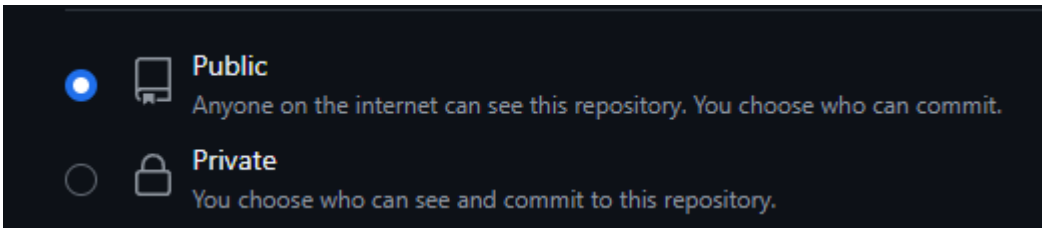


- ¿Qué es un repositorio público en GitHub?

  A diferencia de un repositorio Privado, en GitHub, los repositorios públicos son aquellos a los que todo el mundo tiene acceso, basta solo con dar con el link al repositorio para acceder a todo el código en el. Al ser publico, cualquier usuario de la plataforma podrá: clonar, hacer pull request, tendrá visibilidad del contenido.

- ¿Cómo crear un repositorio público en GitHub?

  Para crear un repositorio publico en GitHub, basta con modificar contrariamente como habíamos realizado en el ejemplo del repositorio privado, las opciones de privacidad del mismo, tildando esta vez la opción Public.

- ¿Cómo compartir un repositorio público en GitHub?

  Simplemente puede compartirse un repositorio publico copiando el link del repositorio como por ejemplo: https://github.com/gst-n/prueba1.git

  O desde un comando ssh: git@github.com:gst-n/prueba1.git

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
      Creamos la rama 'nueva-rama' para agregar modificaciones al archivo mi-archivo.txt

```
PS C:\Users\gasto\Desktop\TUPaD\Progra I\gitTest> git branch nueva-rama
PS C:\Users\gasto\Desktop\TUPaD\Progra I\gitTest> git checkout nueva-rama
Switched to branch 'nueva-rama'
PS C:\Users\gasto\Desktop\TUPaD\Progra I\gitTest> git add .
PS C:\Users\gasto\Desktop\TUPaD\Progra I\gitTest> git commit -m 'modificación desde nueva-rama'
[nueva-rama 8ba5222] modificación desde nueva-rama
 1 file changed, 1 insertion(+), 1 deletion(-)
```

    - Realizar cambios o agregar un archivo
      Modificamos algunos valores dentro del archivo
    - Subir la Branch

```
PS C:\Users\gasto\Desktop\TUPaD\Progra I\gitTest> git push origin nueva-rama
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 366 bytes | 183.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'nueva-rama' on GitHub by visiting:
remote:      https://github.com/gst-n/prueba1/pull/new/nueva-rama
remote:
To https://github.com/gst-n/prueba1.git
 * [new branch]      nueva-rama -> nueva-rama
```

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.

=======

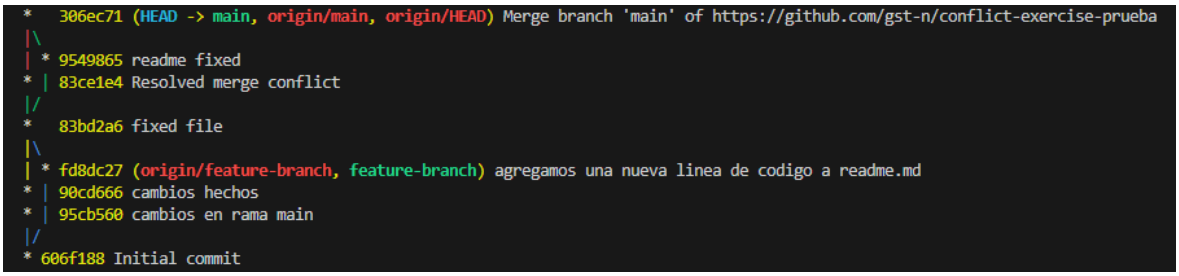
Este es un cambio en la feature branch.

>>>>>>> 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